КИЇВСЬКИЙ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Н. П. ТМЄНОВА

КОМП'ЮТЕРНА ГРАФІКА

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Н. П. ТМЄНОВА

КОМП'ЮТЕРНА ГРАФІКА

Навчально-методичний посібник

КИЇВСЬКИЙ ВГУ 2017 УНІВЕРСИТЕТ

УДК 004.92:51-35(075.8)
Т49

Рецензенти:
канд. техн. наук, доц. В. А. Бородін,
канд. фіз.-мат. наук, доц. О. С. Тригуб,
канд. техн. наук, доц. В. І. Вялкова

Рекомендовано вченою радою
факультету інформаційних технологій
(протокол № 19 від 23 січня 2017 року)

Ухвалено науково-методичною радою
Київського національного університету імені Тараса Шевченка
(протокол № 4-16/17 н. р. від 16 червня 2017 року)

Тмєнова Н. П.

Т49 Комп'ютерна графіка : навч.-метод. посіб. / Н. П. Тмєнова. – К. : ВПЦ "Київський університет", 2017. – 111 с.

Висвітлено основні поняття, історію розвитку і сфери застосування комп'ютерної графіки. Розглянуто види комп'ютерної графіки й основні типи колірних моделей. Викладено математичні основи двовимірної та тривимірної графіки. Наведено паралельне, центральне проектування та їхні види, базові растрові алгоритми, алгоритми зафарбування та алгоритм Коена – Сазерленда. Особлива увага приділяється форматам графічних файлів та алгоритмам стиснення.

Для студентів університету, які вивчають дисципліну "Комп'ютерна графіка".

УДК 004.92:51-35(075.8)

ВСТУП

Комп'ютерна графіка – це область комп'ютерних технологій, яка в наш час набуває стрімкого розвитку. Комп'ютерна графіка використовується майже в усіх наукових та інженерних дисциплінах для наочності, полегшення сприйняття й передачі інформації. Вона знайшла застосування в рекламному бізнесі, видавничій справі, телебаченні та кіноіндустрії, індустрії розваг тощо.

Комп'ютерна графіка займає важливе місце в мережі Internet: удосконалюються способи передачі візуальної інформації, розробляються більш досконалі графічні формати, активно використовується тривимірна графіка, анімація, мультимедіа.

Комп'ютерна графіка важлива при проектуванні та моделюванні різного роду складних систем у господарстві, економіці, військовій справі, медицині тощо.

Растрову графіку застосовують при розробці електронних (мультимедійних) і поліграфічних видань. Більшість графічних редакторів, призначених для роботи з растровими ілюстраціями, орієнтовані не стільки на створення зображень, скільки на їх обробку. Програмні засоби для роботи з векторною графікою навпаки призначені, у першу чергу, для створення ілюстрацій і меншою мірою для їх обробки. Такі засоби широко використовують в рекламних агентствах, дизайнерських бюро, редакціях і видавництвах. Тривимірна графіка широко застосовується в інженерному програмуванні, комп'ютерному моделюванні фізичних об'єктів і процесів, у мультиплікації, кінематографії та комп'ютерних іграх. Програмні засоби для роботи з фрактальною графікою призначені для автоматичної генерації зображень шляхом математичних розрахунків. Фрактальну графіку рідко застосовують для створення друкованих або електронних документів, але її часто використовують у розважальних програмах.

Навчально-методичний посібник призначений для вивчення дисципліни "Комп'ютерна графіка" та спрямований на розкриття базових концепцій, алгоритмів, математичних методів і фізичних принципів, закладених в основу комп'ютерної графіки.

Посібник містить сім розділів.

Розділ 1 присвячено розгляду базових уявлень про комп'ютерну графіку, тобто розгляду основних понять, історії розвитку, сфер застосування. Наведено основні види комп'ютерної графіки: растрова, векторна та фрактальна, висвітлено принципи побудови зображень у тривимірній графіці.

Розділ 2 присвячено видам колірних моделей. Розглянуто адитивну колірну модель RGB, субтрактивні колірні моделі (CMY, CMYK), перцепційні колірні моделі (HSB) та універсальні колірні моделі (Lab).

У розділі 3 подано математичні основи двовимірної графіки, а саме афінні (лінійні) перетворення на площині, однорідні координати та комбіновані перетворення.

У розділі 4 розглянуто математичні основи тривимірної графіки: афінні перетворення у просторі та комбіновані перетворення.

Розділ 5 присвячено проектуванню та його видам. Розглянуто паралельне та центральне проектування, види паралельних проекцій.

У розділі 6 наведено огляд базових растрових алгоритмів. Розглянуто алгоритми зафарбування та алгоритм Коена – Сазерленда.

У розділі 7 подано формати графічних файлів та алгоритми стиснення.

У кінці кожного розділу наведено контрольні запитання.

Розділ 1
БАЗОВІ УЯВЛЕННЯ ПРО КОМП'ЮТЕРНУ ГРАФІКУ

Комп'ютерна графіка разом із розвитком комп'ютерів стрімко увійшла в наше життя, а кількість сфер її застосування постійно збільшується.

Комп'ютерна графіка – це сукупність методів та засобів уведення, відображення, редагування, перетворення та документування на комп'ютері графічної та символьної інформації для розв'язування прикладних задач.

Під графічною інформацією розуміють моделі об'єктів (абстрактний опис графічних об'єктів) та їх зображення.

1.1. Обробка графічної інформації

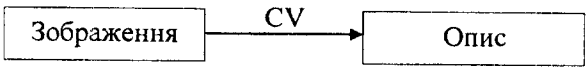
При обробці інформації, пов'язаної із зображенням на моніторі, виділяють три основні напрямки:

- розпізнавання образів (computer vision);
- обробка зображень (image processing);
- комп'ютерна (машинна) графіка (computer graphics).

Основне завдання *розпізнавання образів* полягає в перетворенні наявного зображення на формально зрозумілу мову символів.

Розпізнавання образів – це сукупність методів, що дозволяють отримати опис зображення, поданого на вході системи, чи віднести задане зображення до певного класу. При цьому зображення часто перетворюється на деякий абстрактний опис – набір чисел, ланцюжок символів або граф.

Символічне розпізнавання зображень або система технічного зору COMPUTER VISION (CV) можна описати таким чином: вхід – зображення; вихід – символ (текст). Як приклад розпізнавання образів наведемо систему космічної видової розвідки.

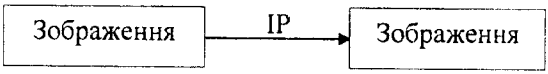


Обробка зображень пов'язана з розв'язанням технічних задач, в яких вхідними і вихідними даними є зображення.

Прикладами обробки зображень можуть служити:

- передача зображень (наприклад, із космічного апарата) разом із видаленням шумів і стисненням даних;
- перехід від одного вигляду зображення до іншого (наприклад, від півтонового до каркасного);
- контрастування різних знімків.

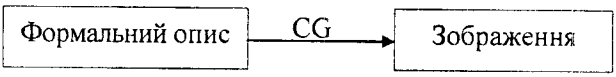
Тим самим система обробки зображень IMAGE PROCESSING (IP) має таку структуру: вхід – зображення; вихід – зображення.



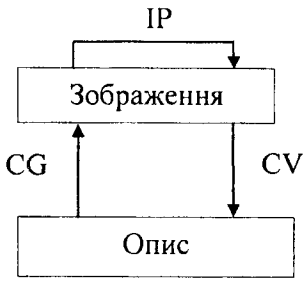
Комп'ютерна графіка (машинна графіка) відновлює зображення у випадку, коли вхідною є інформація необразотворчої природи. Наприклад, візуалізація експериментальних даних у вигляді графіків, гістограм, діаграм; виведення інформації на екран у комп'ютерних іграх.

Можна сказати, що комп'ютерна графіка малює, опираючись на формальний опис і наявний набір засобів.

Символічно систему комп'ютерної графіки COMPUTER GRAPHICS (CG) можна подати таким чином: вхід – формальний опис; вихід – зображення:



Взаємозв'язок процесів розпізнавання образів (CV), обробки зображень (IP) та комп'ютерної графіки (CG) можна зобразити у вигляді схеми:



Чіткі межі між цими трьома напрямками встановити досить складно.

Основним об'єктом вивчення буде саме комп'ютерна графіка.

Відмітимо, що у комп'ютерній графіці розглядають такі основні задачі:

- подання зображення;
- підготовка зображення до візуалізації (візуалізація виконується з урахуванням моделі того, що треба зобразити);
- створення зображення;
- дії над зображенням.

1.2. Історія розвитку комп'ютерної графіки

При всьому різноманітті зображень, отриманих за допомогою комп'ютера, і вражаючих ефектів, які ми можемо сьогодні спостерігати, усе починалося зовсім не так просто.

Точкою відліку початку розвитку комп'ютерної графіки вважається 1930 р., коли у США *Володимиром Зворикіним*, який працював у компанії Westinghouse, була винайдена електронно-променева трубка.

Початком ери комп'ютерної графіки вважають 1951 р., коли в Массачусетському технологічному інституті (МТІ) для системи протиповітряної оборони військово-морського флоту США

Джей Форрестер розробив перший дисплей для комп'ютера "Whirlwind" ("Вихор").

Першою системою комп'ютерної графіки була система SketchPad (Sketch – ескіз, Pad – блокнот), яку розробив *Айвен Сазерленд* у 1963 р. у MTI. Це була програма для малювання, вона використовувала світлове перо для малювання простих фігур на екрані.

1964 р. General motors (США) представила систему автоматизованого проектування (САПР) автомобіля DAC-1. САПР – система, призначена для автоматизації процесу проектування з використанням комп'ютерних технологій.

У 1965 р. IBM випустила перший комерційний графічний термінал – IBM-2250.

1968 р. під керівництвом *М. М. Константинова* створено комп'ютерну математичну модель руху кішки. Машина БЕСМ-4 малювала мультфільм "Кішечка", який для того часу був справжнім проривом.

У 70-х рр. стався стрімкий стрибок у розвитку обчислювальної техніки завдяки винаходу мікропроцесора. У цей час інтенсивно починає розвиватися індустрія комп'ютерних ігор.

У Київському політехнічному інституті в 1973 р. розроблено САПР електронних схем.

1977 р. – Commodore випустила персональний електронний діловод PET, а Apple створили "Apple-II".

У 80-х рр. комп'ютерна графіка впроваджується в кіноіндустрію.

Завдяки виникненню мережі Internet у 90-х рр. у комп'ютерній графіці з'являється ще одна велика сфера застосування.

Таким чином у розвитку комп'ютерної графіки можна виділити декілька етапів:

- 1960–1970-ті рр. – комп'ютерна графіка формується як наукова дисципліна. Розробляються основні методи та алгоритми;
- 1980-ті рр. – комп'ютерна графіка розвивається як прикладна дисципліна. Розробляються методи її використання в різних областях людської діяльності;
- 1990-ті рр. – методи комп'ютерної графіки стають основним засобом організації діалогу "людина – комп'ютер" і залишаються такими по теперішній час.

1.3. Галузі застосування комп'ютерної графіки

Сьогодні важко уявити комп'ютер, який не має графічних можливостей. Найпростіший приклад таких можливостей – графічний інтерфейс користувача, основними перевагами якого є наочність і простота використання (наприклад, Windows).

Комп'ютерна графіка використовується майже в усіх наукових та інженерних дисциплінах для наочності і сприйняття, передачі інформації. Застосовується вона також у медицині, рекламному бізнесі, індустрії розваг, web-дизайні, видавничій справі тощо.

Спробуємо виділити основні області застосування комп'ютерної графіки.

Наукова графіка. Перші комп'ютери використовувалися лише для вирішення наукових і виробничих завдань. Щоб краще зрозуміти отримані результати, виконували їх графічну обробку, будували діаграми, креслення конструкцій. Сучасна наукова комп'ютерна графіка дає можливість проводити обчислювальні експерименти з наочним зображенням їх результатів.

Ділова графіка – область комп'ютерної графіки, призначена для наочного зображення різних показників роботи установ. Планові показники, звітна документація, статистичні відомості – ось об'єкти, для яких за допомогою ділової графіки створюються ілюстративні матеріали.

Конструкторська графіка використовується в роботі інженерів-конструкторів, архітекторів, винахідників нової техніки. Цей вид комп'ютерної графіки є обов'язковим елементом САПР. Засобами конструкторської графіки можна отримувати як плоскі зображення (проекції, перерізи), так і просторові тривимірні зображення.

Ілюстративна графіка – це малювання і креслення на екрані комп'ютера. Прості програмні засоби ілюстративної графіки називаються графічними редакторами.

Художня і рекламна графіка – стала популярною багато в чому завдяки телебаченню. За допомогою комп'ютера створюються рекламні ролики, мультфільми, комп'ютерні ігри, відео-

уроки, відеопрезентації. Зазначимо, що графічні пакети для цих цілей вимагають великих ресурсів комп'ютера із швидкодії і пам'яті. За допомогою цих пакетів є можливість створення реалістичних зображень і "рухомих картинок".

Комп'ютерна анімація – отримання рухомих зображень на екрані дисплея. Художник створює на екрані малюнки початкового і кінцевого положень рухомих об'єктів, усі проміжні стани розраховує і малює комп'ютер на основі математичного опису даного виду руху. Отримані малюнки, що виводяться послідовно на екран із певною частотою, створюють ілюзію руху.

1.4. Види комп'ютерної графіки

Розрізняють три види комп'ютерної графіки:

- растрова графіка;
- векторна графіка;
- фрактальна графіка.

Вони відрізняються принципами формування зображення при відображенні на екрані монітора або під час друку на папері.

1.4.1. Растрова графіка

Растр – це умовна прямокутна сітка точок, що формують зображення на екрані комп'ютера (так званих пікселів).

Растрові зображення нагадують аркуш клітчастого паперу, на якому кожну клітинку (піксель) зафарбовано або білим, або чорним, або іншим кольором, що сукупно утворюють малюнок.

Піксель – мінімальний елемент растрових зображень. Із таких елементів і складається растрове зображення. Кожна точка растра характеризується двома параметрами: своїм положенням на екрані і своїм кольором. Колір будь-якого пікселя растрового зображення запам'ятовується в комп'ютері за допомогою комбінації бітів. Чим більше бітів для цього застосовується, тим більше відтінків кольорів можна одержати.

Кількість бітів, що використовується комп'ютером для будь-якого пікселя, називається *бітовою глибиною* пікселя. Кожному кольору відповідає певний двійковий код (тобто код із нулів та одиниць). Наприклад, якщо бітова глибина дорівнює 1, тобто під кожен піксель відводиться 1 біт, то 0 відповідає чорному кольору, 1 – білому, а зображення може бути тільки чорно-білим. Якщо бітова глибина дорівнює 2, то 00 відповідає чорному кольору, 01 – червоному, 10 – синьому, 11 – зеленому (тобто всього $2^2 = 4$ кольори). Кольори, що описуються 24 бітами, забезпечують більше 16 млн ($2^{24} \approx 16{,}7$ млн) доступних кольорів і їх часто називають природними кольорами. Зі збільшенням можливої кількості кольорів, збільшується об'єм пам'яті, необхідний для запам'ятовування зображення.

Для зображення кількості пікселів матриці малюнка по горизонталі і по вертикалі вводиться *коефіцієнт прямокутності зображення* (часто цей коефіцієнт називають *розміром зображення*). Наприклад, 320×240; 640×480; 800×600 та ін. Добуток цих двох чисел дає загальну кількість пікселів зображення. Щоб порахувати *інформаційний об'єм файлу*, треба кількість пікселів по горизонталі помножити на кількість пікселів по вертикалі, і це число помножити на бітову глибину.

Отже, *якість растрового зображення* залежить від розміру зображення і бітової глибини.

Растрове зображення дуже чутливе до масштабування (збільшення та зменшення). Після зменшення растрового зображення декілька сусідніх точок перетворюються на одну, тому втрачається помітність дрібних деталей зображення. Під час збільшення зображення збільшується розмір кожної крапки і з'являється ступінчастий ефект.

Растрові зображення мають безліч характеристик, які повинні бути організовані та фіксовані комп'ютером, але *дві основні характеристики* – розмір зображення і розташування пікселів у ньому – файл повинен зберегти, щоб створити картинку. Навіть якщо зіпсована інформація про колір будь-якого пікселя

чи про будь-яку іншу характеристику, комп'ютер все одно зможе відтворити версію малюнка, якщо знатиме, як розташовані всі його пікселі.

Оскільки розміри зображення зберігаються окремо, пікселі запам'ятовуються один за іншим, як звичайний блок даних. Комп'ютер усього лише створює сітку за розмірами зображення, а потім заповнює її піксель за пікселем.

Оскільки пікселі не мають своїх власних розмірів, а набувають їх при виведенні, наприклад, на монітор або принтер, то, щоб пам'ятати дійсні розміри растрового малюнка, файли растрової графіки іноді зберігають *роздільну здатність растра*. Роздільну здатність растра задають у пікселях на дюйм (dots per inch – dpi). Одиниця dpi визначає, на скільки пікселів перетвориться лінія завдовжки 1 дюйм. Як правило, використовують одиниці від 100 dpi до 2400 dpi. Лазерні принтери мають роздільну здатність від 300 dpi до 600 dpi (як і домашні сканери). Професійні сканери забезпечують якість приблизно 2400 dpi.

Приклад

Відскануємо аркуш А4, розміром 210×297 мм (тобто приблизно 8×11 дюймів). Нехай ми встановили на сканері максимальний дозвіл 600 dpi. Отже, розмір зображення буде $(8 \cdot 600) \times (11 \cdot 600)$ пікселів, тобто 4800×6600 пікселів. Кольоровий сканер видає зображення глибиною 24 біти, тобто 3 байти на піксель. Таким чином маємо $4800 \cdot 6600 \cdot 3 = 95040000$ байтів або 90,63 мегабайта. Такий об'єм пам'яті займає кольоровий аркуш формату А4 при скануванні з дозволом 600 dpi.

Файли растрової графіки займають велику кількість пам'яті комп'ютера. Найбільше впливають на кількість пам'яті, зайнятої растровим зображенням, три фактори:

- розмір зображення (кількість пікселів);
- бітова глибина пікселя;
- формат файлу, що використовується для зберігання зображення.

Зрозуміло, що чим більше в зображенні пікселів, тим більший розмір файлу. Чим більше є бітів у пікселі, тим більшим буде файл.

За інших рівних умов, таких як розміри зображення і бітова глибина, істотне значення має схема стиснення зображення. Наприклад, BMP-файл має, як правило, великі розміри порівняно з файлами PCX і GIF, які у свою чергу більші за JPEG-файли.

Переваги растрової графіки

- Растрова графіка ефективно представляє реальні образи. Реальний світ складається з мільярдів дрібних об'єктів, і людське око якраз пристосоване для сприйняття великого набору дискретних елементів. Зображення виглядають цілком реально.
- Пристрої виведення, такі як лазерні принтери, для створення зображень використовують набори точок. Растрові зображення легко друкувати, оскільки комп'ютерам легко керувати растровими пристроями виведення.

Недоліки растрової графіки

- Погане масштабування. При зменшенні зображення декілька сусідніх точок перетворюються на одну, тому втрачаються дрібні подробиці. При збільшенні масштабу відбувається збільшення розміру кожної точки, через що з'являється ступінчастий ефект.
- Файли растрової графіки займають велику кількість пам'яті комп'ютера, оскільки файл включає в себе дані про кожний піксель зображення.
- Споживання значних ресурсів комп'ютера при редагуванні растрових зображень.

Як приклади растрових редакторів наведемо такі: Adobe Photoshop, Corel Painter, GIMP, Microsoft Paint.

1.4.2. Векторна графіка

На відміну від растрової графіки у векторній графіці векторне зображення являє собою математичний опис об'єктів. При використанні векторного уявлення зображення є базою даних описів графічних примітивів, тобто геометричних об'єктів, що описані аналітично. Такими графічними примітивами є відрізки, кола, овали, трикутники, точки, криві Безьє тощо. Наприклад, для трикутника зберігаються такі атрибути: координати трьох вершин, колір (текстура), також може зберігатись колір кожного ребра трикутника. Із графічних примітивів будуються складні об'єкти. Оскільки векторне зображення є аналітично заданим, то в будь-який момент ми можемо змінити будь-який із параметрів будь-якої з його складових. Крім того, над векторним зображенням дуже зручно проводити математичні операції на зразок збільшення, повороту, нелінійних перетворень тощо.

Векторні зображення, як правило займають менше пам'яті комп'ютера, ніж растрові. Проте растрове зображення має великі переваги при роботі з фотореалістичними об'єктами, наприклад, сценами природи або фотографіями людей, адже світ, можна вважати, є растровий, і його об'єкти важко представити у векторному, тобто математичному, по суті, уявленні.

Для різних векторних форматів характерні різні колірні можливості. Прості формати використовують колір за замовчуванням тих пристроїв, на які вони вводяться. Інші формати здатні зберігати дані про повний набір кольорів. Яку б колірну модель не застосував би векторний формат, на розмір файлу він не впливає (окрім тих випадків, коли файл містить растрові образи).

У звичайних векторних об'єктах значення кольору відносяться до всього об'єкту в цілому. Колір об'єкта зберігається у вигляді частки його векторного опису.

Переваги векторної графіки

- Векторна графіка використовує всі переваги роздільної здатності будь-якого пристрою виведення. Це дозволяє змінювати розміри векторного малюнка без утрати його якості.

- У векторній графіці можна редагувати окремі частини малюнка без впливу на інші. Об'єкти на малюнку можуть перекриватись без жодного впливу один на одного.
- Векторне зображення, що не містить растрових об'єктів, займає відносно невелике місце у пам'яті комп'ютера.

Недоліки векторної графіки

- Умовність отриманих зображень. Для побудови реалістичних зображень знадобилася б величезна кількість примітивів (але тоді б і дуже збільшився розмір файлу векторної графіки). Тому векторний формат застосовується для описування лінійних малюнків та ідеально підходить для креслень.
- Складнощі при суцільному заповненні фігур кольором: менша кількість кольорів, менша швидкість (порівняно з растровими пристроями).

Зазначимо окремі положення щодо перетворення зображення між двома форматами.

Перетворення *векторного зображення на растрове* особливих проблем не викликає, адже при роботі у векторному редакторі ви все одно бачите результат у растровому вигляді (монітор є растровим пристроєм). Тобто відбувається *растеризація* зображення – перетворення на піксельну форму. Цей процес є однозначним.

Перетворення *растрового зображення на векторне* є важчим завданням. У зв'язку з тим, що це процес неоднозначний, тут уже з'являється елемент емпірики, і нам у кожному конкретному випадку належить визначити, як краще представити вектором цей ланцюжок пікселів. Результати можуть бути різними. Потужні програми векторизації справляються із цим завданням добре.

Як приклади векторних редакторів наведемо такі: Adobe Illustrator, Corel Draw, Inkscape.

1.4.3. Фрактальна графіка

Фрактальна графіка як і векторна графіка – обчислювальна, але відрізняється від неї тим, що жодні об'єкти в пам'яті комп'ютера не зберігаються. Зображення будується за рівнянням (або за системою рівнянь), тому нічого, окрім формул, зберігати не треба. Змінивши коефіцієнт у рівнянні, можна отримати абсолютно іншу картинку. Фрактальна графіка базується на фрактальній геометрії.

Фрактал у широкому розумінні означає фігуру, малі частини якої за довільного збільшення є подібними до неї самої, тобто є самоподібною структурою.

Поняття фракталу ввів *Бенуа Мандельброт* у 1975 р.

Найвідомішими фрактальними об'єктами є дерева: від кожної гілки відходять менші, схожі на неї, від них – ще менші і т. д. За окремою гілкою математичними методами можна відслідкувати властивості всього дерева.

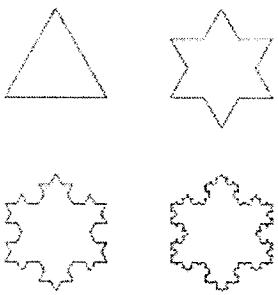
Фрактальні властивості мають такі природні об'єкти: сніжинка, корали, морські зірки, кристали, гірські хребти.

Із погляду комп'ютерної графіки фрактальна геометрія незамінна під час генерації штучних хмар, гір, поверхні моря.

Здатність фрактальної графіки моделювати образи живої природи обчислювальним шляхом часто використовується для автоматичної генерації незвичайних ілюстрацій.

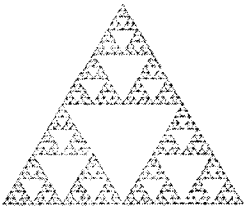
Приклади фракталів

1. Сніжинка Коха



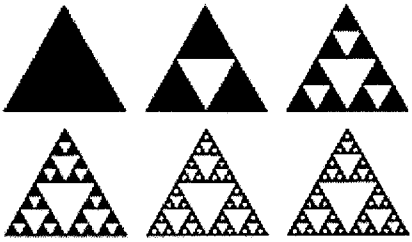
Сніжинка Коха є межею нескінченної конструкції, що починається з трикутника та доповнюється рекурсивною заміною кожного сегмента набором із чотирьох сегментів, які утворюють трикутний "виступ".

2. Трикутник Серпінського



Трикутник Серпінського – один із найбільш ранніх відомих прикладів фракталів. Існує кілька способів побудови цього фракталу. Трикутник Серпінського також називають серветкою або решетом Серпінського.

Запишемо послідовність *побудови трикутника Серпінського:*



Як приклади редакторів фрактальної графіки наведемо такі: Surfer, Grafer, Map Viewer, Ultra Fractal.

1.5. Тривимірна графіка

Окремим видом комп'ютерної графіки є тривимірна графіка, яка оперує об'єктами у тривимірному просторі. Зазвичай результатом є плоска картинка або проекція.

Тривимірна комп'ютерна графіка широко використовується в кіно, комп'ютерних іграх. У ній усі об'єкти зазвичай подаються як набір поверхонь.

Мінімальну поверхню називають *полігоном.* Як полігон зазвичай вибирають трикутники. Усіма візуальними перетвореннями у 3D-графіці керують матриці. Використовують матриці повороту, матриці перенесення та матриці масштабування.

Будь-який полігон можна подати у вигляді набору з координат його вершини. Так, у трикутника буде три вершини. Координати кожної вершини задають вектором (x, y, z). Помноживши вектор на відповідну матрицю, ми отримаємо новий вектор. Виконавши таке перетворення з усіма вершинами полігона, ми отримаємо новий полігон, а перетворивши всі полігони, отримаємо новий об'єкт, повернутий або зсунутий відносно початкового.

Усе різноманіття властивостей у комп'ютерному моделюванні зводиться до візуалізації поверхні, тобто до розрахунку коефіцієнта прозорості поверхні та кута заломлення променів світла на межі матеріалу й навколишнього простору.

Властивості поверхні описуються у створюваних масивах текстур, в яких містяться дані про міру прозорості матеріалу, коефіцієнт заломлення, колір у кожній точці, колір відблиску, його ширину й різкість тощо.

Як приклади програм для роботи з 3D-графікою можна виділити такі: Blender, K-3D, 3D-Canvas, SimLab Composer, Autodesk 3d(s) Max.

КОНТРОЛЬНІ ЗАПИТАННЯ ДО РОЗДІЛУ 1

1. Що таке "комп'ютерна графіка"?
2. Які основні напрямки виділяють при обробці графічної інформації?
3. Сформулюйте основні задачі комп'ютерної графіки.
4. Виділіть основні етапи в розвитку комп'ютерної графіки.
5. Які галузі застосування комп'ютерної графіки вам відомі?
6. Які види комп'ютерної графіки ви знаєте?
7. Що таке растр? Що таке піксель?
8. Що називається бітовою глибиною пікселя?
9. Як підраховується інформаційний об'єм файлу?
10. Від чого залежить якість растрового зображення?
11. Що розуміють під роздільною здатністю растра?

12. Сформулюйте сильні та слабкі сторони растрової графіки.
13. Які растрові редактори вам відомі?
14. Дайте характеристику векторної графіки.
15. Сформулюйте переваги векторної графіки.
16. Сформулюйте недоліки векторної графіки.
17. Які векторні редактори вам відомі?
18. Дайте характеристику фрактальної графіки.
19. Які об'єкти мають фрактальні властивості?
20. Наведіть приклади редакторів фрактальної графіки.
21. Що являє собою тривимірна графіка?
22. Наведіть приклади програм для роботи із 3D-графікою.

Розділ 2

КОЛІР. КОЛІРНІ МОДЕЛІ

2.1. Дослід Ньютона

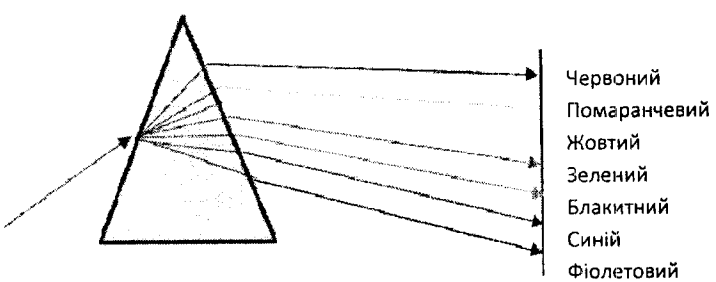
Для вивчення способів представлення кольору в комп'ютерних системах спочатку розглянемо деякі загальні аспекти.

Колір – це один із факторів нашого сприйняття світлового випромінювання.

Світлом і кольором дослідники зацікавились досить давно. Одними з перших видатних досягнень у цій галузі були досліди *Ісаака Ньютона в* 1666 р. із розкладання білого кольору на складові. Раніше вважалося, що білий колір є найпростішим. Ньютон скасував це твердження.

Дослід був таким. Через маленький круглий отвір у вікні в затемнену кімнату проходив білий промінь світла (використовувалося сонячне світло). Промінь направлявся на скляну трикутну призму. Пройшовши через призму, промінь заломлювався і після напрямлення на екран давав у результаті кольорову смугу – спектр. Спектр містив усі кольори райдуги, що плавно переходили один в одного.

Ньютон розбив увесь спектр на 7 ділянок, що відповідали яскраво вираженим різним кольорам. Він вважав ці 7 кольорів основними: червоний, помаранчевий, жовтий, зелений, блакитний, синій і фіолетовий.



Ньютон припустив, що деякий колір утворюється змішуванням основних кольорів, узятих у певній пропорції.

Друга частина дослідів Ньютона була така. Промені, що пройшли через призму, направлялися на двоопуклу лінзу, яка знову накладала різні кольори один на одного. Збігаючись, вони знову утворювали на екрані білу пляму.

Наступні дослідження кольору виконували *Томас Юнг*, *Джеймс Максвелл* та інші вчені. Вивчення людського кольорosприйняття було дуже важливою задачею, але основні зусилля вчених направлялися на вивчення властивостей світла.

Сьогодні фізики вважають, що світло має двоїстий характер. З одного боку, воно подається у вигляді потоку частинок (корпускулярна теорія Ньютона). З іншого боку, світло має хвильові властивості (хвильова теорія *Християна Гюйгенса*, 1678 р.).

2.2. Поняття колірної моделі

Людське око може сприйняти велику кількість кольорів, у той час як монітор та принтер у змозі відтворити лише обмежену частину цього діапазону.

У зв'язку з необхідністю опису різних фізичних процесів відтворення кольору розроблено різноманітні колірні моделі.

У сучасних комп'ютерних програмах маніпулювання з кольором здійснюється за допомогою колірних моделей та режимів.

Колірні моделі (Color Model) надають засоби для концептуального та кількісного опису кольору. *Колірний режим (Color Mode)* – це спосіб реалізувати певні колірні моделі в рамках конкретної графічної програми.

Колірні моделі використовують для математичного опису певних колірних областей спектра. Більшість комп'ютерних колірних моделей засновано на використанні трьох основних кольорів, що відповідають сприйняттю кольору людським оком. Кожному основному кольору присвоюють певне значення цифрового коду, після чого решта кольорів визначається як комбінація основних кольорів.

Незалежно від того, що лежить в основі колірної моделі, будь-яка модель повинна відповідати таким вимогам:

- реалізувати визначення кольору деяким стандартним способом, який не залежить від можливостей певного пристрою;
- точно задавати діапазон кольорів, що відтворюються, оскільки жодна множина кольорів не є безмежною;
- враховувати механізм сприйняття кольорів – випромінювання або відбиття.

2.3. Типи колірних моделей

Більшість графічних пакетів дозволяють оперувати великим набором колірних моделей.

За принципом дії колірні моделі можна умовно поділити на чотири класи:

1) адитивні (RGB), що основані на додаванні кольорів;
2) субрактивні (CMYK), основані на відніманні кольорів;
3) перцепційні (HSB, HSL), що базуються на сприйнятті;
4) універсальні (Lab, XYZ), що охоплюють весь спектр кольорів, який сприймається оком людини.

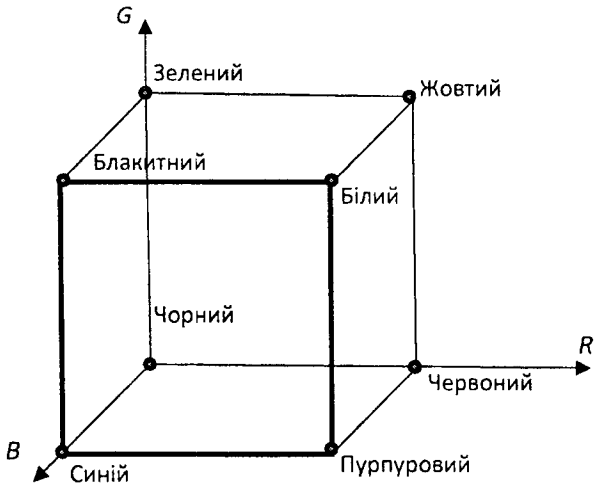
2.3.1. Адитивна колірна модель RGB

Адитивна модель відповідає сприйняттю кольорів людським оком. Ця модель використовується для опису кольорів, які отримують за допомогою пристроїв, заснованих на принципі випромінювання. За основні кольори вибрано червоний (Red), зелений (Green) і синій (Blue), їх названо *первинними кольорами* (основними кольорами).

Вибір основних кольорів обумовлено особливостями фізіології сприйняття кольору сітківкою людського ока. При парному змішуванні первинних кольорів створюються *вторинні кольори*: блакитний, пурпуровий та жовтий. Первинні та вторинні кольори належать до базових. *Базовими* кольорами називають кольори, за допомогою яких можна отримати практично весь спектр видимих кольорів.

Для одержання нових кольорів за допомогою адитивного синтезу можна використовувати й різноманітні комбінації із двох основних кольорів, зміни складу яких приводять до зміни результуючого кольору. Однак для отримання певних кольорів необхідно додати третій первинний колір. Так, білий колір отримують при змішуванні трьох основних кольорів: червоного, зеленого і синього.

Математично колірну модель RGB краще подавати у вигляді куба. Якщо на осі X відкладати червону складову, на осі Y – зелену, а на осі Z – синю, то кожному кольору можна поставити у відповідність точку всередині куба, тобто задати вектором у системі R, G, B.



Чорному кольору відповідає центр координат – точка (0,0,0). Білий колір виражає максимальне значення компонент. Нехай це максимальне значення вздовж кожної осі дорівнює 1. Тоді білий колір – це точка (1,1,1). Точки, що лежать на діагоналі куба від чорного до білого – це градації сірого. Їх можна вважати білим кольором різної яскравості.

Візуалізація за допомогою адитивної моделі

У графічних пакетах колірну RGB-модель використовують для створення кольорів зображення на екрані *монітора.* Основними елементами монітора на основі електронно-променевої трубки є три електронні прожектори (гармати) та екран із нанесеними на нього трьома різними люмінофорами. (Люмінофор – речовина, що може перетворювати енергію, яку поглинає, у світлове випромінювання.) Ці люмінофори мають різні спектральні характеристики. На відміну від ока вони не поглинають, а випромінюють світло. Один люмінофор під дією падаючого на нього електронного променя випромінює червоний колір, другий – зелений, а третій – синій.

Якщо роздивитись білий екран включеного монітора через лупу, то можна побачити, що він складається з великої кількості окремих точок червоного, зеленого і синього кольорів, об'єднаних у RGB-елементи у вигляді тріад основних пікселів. Під час розглядання зображення на екрані з деякої відстані ці кольорові складові RGB-елементів зливаються, створюючи ілюзію результуючого кольору.

Кожний із трьох компонентів RGB-тріади може набувати одне з 256 дискретних значень: від 0 до 255. Коли всі три кольорові компоненти мають максимальні значення (255, 255, 255), отриманий колір здається білим, а коли всі три компоненти мають нульові значення (0, 0, 0), отриманий колір – чисто чорний.

Обмеження RGB-моделі

При використанні RGB-моделі на практиці виникає дві проблеми.

1. Апаратна залежність.

Ця проблема пов'язана з тим, що колір, який виникає в результаті змішування кольорових складових RGB-елемента, залежить від типу люмінофора. При виробництві сучасних кінескопів застосовують різні типи люмінофорів і установка одних і тих самих інтенсивностей екранних променів у випадку різних люмінофорів призводить до синтезування різного кольору. Також перша проблема пов'язана з тим, що у процесі експлуатації

люмінофор старіє і змінюються характеристики електронних прожекторів. Для усунення (мінімізації) залежності RGB-моделі від апаратних засобів *використовують різні пристрої і програмні градуювання.*

2. Обмеження кольорової гами.

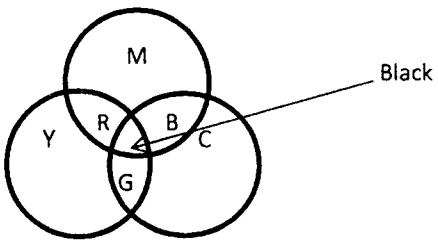
Кольорова гама – це діапазон кольорів, який може розпізнавати людина або відображати пристрій незалежно від механізму отримання кольору (випромінювання або відбиття). Обмеження кольорової гами пояснюється тим, що за допомогою адитивного синтезу неможливо отримати всі кольори видимого спектра. Хоча для відтворення фотореалістичних зображень на екрані монітора RGB-моделі достатньо.

2.3.2. Субтарктивні колірні моделі

На відміну від екрана монітора, в якому відтворення кольорів основане на випромінюванні світла, друкована сторінка може тільки відбивати колір. Субтрактивні моделі найточніше описують кольори при виведенні зображення на друк, тобто в поліграфії.

Субтрактивні кольори на відміну від адитивних отримують відніманням первинних кольорів із загального променя світла. Тобто, кольори, які використовують білий колір, віднімаючи від нього відповідні ділянки спектра, називаються субтрактивними.

Отже, якщо відняти від білого три первинних кольори Red, Green, Blue, ми отримаємо трійку додаткових (доповнюючих) кольорів CMY (C – Cyan – блакитний, M – Magenta – пурпуровий, Y – Yellow – жовтий):



У цій системі білий колір з'являється як результат відсутності всіх субтрактивних кольорів, а їх присутність дає чорний колір.

Наведемо співвідношення, які пов'язують адитивні і субтрактивні кольори:

- зелений + синій = блакитний;
- зелений + червоний = жовтий;
- червоний + синій = пурпуровий;
- зелений + синій + червоний = білий;
- блакитний + жовтий + пурпуровий = чорний.

Блакитний барвник поглинає (віднімає) червоний колір (із білого променя), пурпуровий – поглинає зелений, жовтий – поглинає синій. Кольори в цих парах називаються компліментарними.

Існує дві найбільш розповсюджені версії субтрактивної моделі: CMY і CMYK. Перша використовується у випадку, коли зображення виводиться на чорно-білому принтері, який дозволяє замінювати чорний картридж на кольоровий.

Теоретично при змішуванні трьох кольорів C, M, Y рівними пропорціями отримують чорний колір. Але в реальному технічному процесі отримання чорного кольору шляхом змішування трьох кольорів для паперу неефективне з ряду причин (виходить не чисто чорний колір, а брудно-коричневий або брудно-сірий; у випадку струменевого друку змішування трьох пігментів змочує папір; один чорний пігмент дешевше трьох кольорових пігментів).

Тому під час друку використовують додатковий чорний компонент кольору – Black. Тоді така модель називається CMYK (застосовують літеру "K" замість "B", щоб не було плутанини між Black і Blue).

Кожне з чисел, яке визначає колір у CMYK, являє собою відсоток фарби даного кольору, який входить до складу кольорової комбінації точки растра, що виводиться на пристрій друку.

Приклад

Для отримання темно-помаранчевого кольору потрібно змішати 30 % блакитної фарби, 45 % – пурпурової, 80 % – жовтої, 5 % – чорної. Це може позначатися як (30, 45, 80, 5) або C30M45Y80K5.

Важливо відмітити, що цифрове значення фарби в CMYK не може само по собі описати колір. Цифри – лише набір апаратних даних, які використовують у друкарському процесі для формування зображення. На практиці реальний колір визначатиметься не тільки розміром точки растра на пристрої друку, існуючими цифрами в підготовленому до друку файлі, але й реаліями конкретного друкарського процесу, на які можуть впливати такі фактори: стан друкарської машини, якість паперу, вологість повітря в цеху тощо.

Обмеження CMYK-моделі

CMYK-модель, як і RGB-модель, має два типи обмежень.

1. Апаратна залежність

CMYK-модель є ще більш апаратнозалежною моделлю, ніж RGB. Це пов'язано з тим, що в ній є більша кількість дестабілізуючих факторів, ніж у RGB-моделі, а саме:

- варіація складу кольорових барвників, які використовують для створення друкованих кольорів;
- тип паперу;
- спосіб друку;
- зовнішнє освітлення.

2. Обмежений колірний діапазон

Кольорові барвники мають гірші характеристики порівняно з люмінофорами. Отже, CMYK-модель має вужчий кольоровий діапазон, ніж RGB-модель. Про екранні кольори, які неможливо точно відтворити під час друку, говорять, що вони лежать поза кольоровою гамою CMYK-моделі. Невідповідність кольорових діапазонів RGB і CMYK-моделей створює серйозну проблему – невідповідність друкованої картинки картинці на моніторі.

Для запобігання такій ситуації передбачено комплекс спеціальних засобів:

- редагування зображення у форматі CMYK; використання CMYK-орієнтованих палітр.
- застосування систем керування кольором – CMS (Color Management Systems).

2.3.3. Перцепційні колірні моделі

Для дизайнерів, художників, фотографів основним інструментом індикації і виведення кольору є око. Цей природний "інструмент" має кольорову гаму, яка набагато перевищує можливості будь-якого технічного пристрою (сканера, принтера). CMYK- і RGB-моделі є апаратнозалежними. Для усунення апаратної залежності розроблено ряд так званих *перцепційних* (інтуїтивних) колірних моделей. В їх основу закладено роздільне визначення яскравості й колірності. Цей підхід забезпечує ряд переваг:

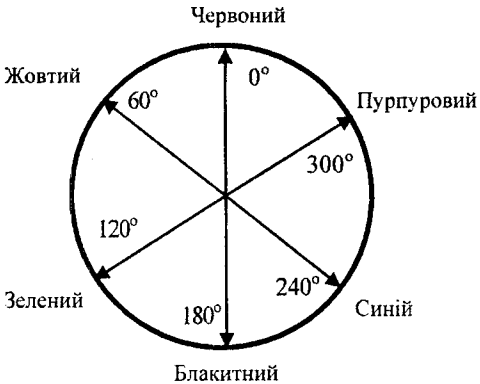
- дозволяє використовувати колір на інтуїтивно зрозумілому рівні;
- спрощує проблему узгодження кольорів, після встановлення значення яскравості можна зайнятися налагодженням кольору.

Прикладами перцепційних моделей є HSB-, HSV-, HSI-, HSL-моделі. Спільним у них є те, що колір задається не у вигляді суміші трьох основних кольорів, а визначається вказуванням двох компонентів: колірності (колірного тону й насиченості) та яскравості.

Колірна модель HSB

Модель HSB (Hue – колірний тон, Saturation – насиченість, Brightness – яскравість) або її наближений аналог HSL подано у більшості сучасних графічних пакетів. Тобто у HSB-моделі всі кольори визначають за допомогою комбінації трьох базових параметрів: колірного тону, насиченості та яскравості.

Колірний тон. Під колірним тоном розуміють світло з домінуючою довжиною хвилі. Кожний колірний тон займає визначене положення на периферії колірного круга і характеризується розміром кута в діапазоні від 0 до 360°:



Однак само по собі поняття колірного тону не несе повної інформації про кольори. Доповнювальними компонентами є насиченість та яскравість.

Насиченість. Насиченість характеризує чистоту (інтенсивність) кольору. Ця насиченість визначає співвідношення між основним, домінуючим компонентом кольору і всіма іншими довжинами хвиль (кількість сірого), які беруть участь у формуванні кольору.

Кількісне значення цього параметра виражається у відсотках від 0 (сірий) до 100 % (повністю насичений).

Чим вище значення насиченості, тим сильніше та ясніше відчувається колірний тон. Зниження насиченості приводить до того, що колір стає нейтральним, без чітко вираженого тону. (Якщо взяти кольорове фото і знизити насиченість до 0 %, то отримаємо чорно-біле фото.)

Природні кольори мають низьку насиченість.

Якщо рухатись упоперек колірного круга, то відбувається зменшення частки кольору, від якого ви віддаляєтесь, і зростання частки кольору, до якого ви наближаєтесь.У результаті це веде до зниження насиченості, яка має максимальне значення (100 %) на поверхні кола і мінімальне (0 %) – у центрі круга.

Яскравість. Яскравість кольору показує величину чорного відтінку, який добавлений до кольору. Тобто при нульовій яскравості ми нічого не бачимо, тому будь-який колір буде сприй-

матися як чорний. За відсутності чорного ми отримуємо чистий спектральний колір, а максимальна яскравість викликає відчуття сліпучого білого кольору.

Ахроматичні кольори (білі, сірі й чорні) характеризуються лише яскравістю. Це проявляється в тому, що одні кольори темніші, а інші світліші. Яскравість вимірюється у відсотках у діапазоні від 0 % (чорний) до 100 % (білий). Мірою зниження відсоткового вмісту яскравості колір стає темнішим.

Переваги HSB-моделі

- Відсутність апаратної залежності.
- Модель більше відповідає природі кольору, добре підходить для сприйняття людиною: колірний тон є еквівалентом довжини світла, насиченість – інтенсивності хвилі, а яскравість – кількості світла.
- Модель відрізняється більш простим та інтуїтивно зрозумілим механізмом керування кольором. Це пов'язано з тим, що колірний тон, насиченість та яскравість – незалежні характеристики кольору.

Недоліки HSB-моделі

- Модель HSB має той самий колірний простір, що і RGB-модель, а отже, і належний їй недолік – обмежений колірний простір.
- Модель носить абстрактний характер.

2.3.4. Універсальні колірні моделі (Lab, XYZ)

Модель Lab

Успішною спробою створення апаратнонезалежної моделі кольору, основаної на людському сприйнятті кольору, є модель Lab.

Будь-який колір у Lab визначається яскравістю (Lightness – L) та двома хроматичними компонентами: параметром a, який

змінюється від зеленого до червоного, та параметром b, що змінюється від синього до жовтого (визначення каналів Lab ґрунтується на тому факті, що точка не може бути одночасно чорною і білою, червоною і зеленою, синьою і жовтою).

L: від 0 % до 100 %, a : від –128 до 127, b : від –128 до 127.

Отже, модель Lab є триканальною, яскравість у Lab повністю відокремлена від кольору, її кольорова гама безмірно широка і відповідає видимій кольоровій гамі для стандартного спостерігача. Гама Lab включає гами всіх інших колірних моделей, які використовуються в поліграфічному процесі.

На відміну від моделей RGB і CMYK, заснованих на реальних процесах, Lab є чисто математичною моделлю. Важко знайти аналогічну їй у реальному світі.

Переваги моделі Lab:

- основана на сприйманні людиною кольору, її кольорова гама відповідає можливостям людського ока – вона включає в себе гами RGB і CMYK та перевищує їх;
- Lab є апаратнонезалежною моделлю.

Ці переваги зробили Lab *стандартом* при переведенні зображень з одного колірного простору в інший у процесі їх підготовки. Таку модель часто використовують для покращення якості зображення (наприклад, для підвищення різкості або контрастності, видалення колірного шуму).

КОНТРОЛЬНІ ЗАПИТАННЯ ДО РОЗДІЛУ 2

1. У чому полягає дослід Ньютона?
2. Що таке колірна модель?
3. Що називають колірним режимом?
4. Які типи колірних моделей вам відомі?
5. Опишіть адитивну колірну модель RGB.
6. Що таке первинні та вторинні кольори?
7. Як математично краще подати колірну модель RGB?

8. Які обмеження RGB-моделі вам відомі?
9. Які характеристики мають субтрактивні моделі?
10. Як утворюються субтрактивні кольори?
11. Назвіть відмінність між версіями субтрактивних моделей CMY і CMYK?
12. Охарактеризуйте обмеження CMYK-моделі?
13. Що закладено в основу перцепційних моделей?
14. Назвіть приклади перцепційних моделей.
15. Опишіть колірну модель HSB.
16. У чому полягає обмеження HSB-моделі?
17. Опишіть колірну модель Lab.

Розділ 3

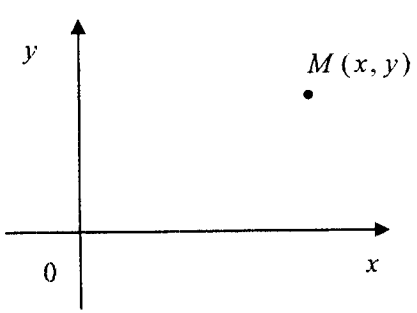
МАТЕМАТИЧНІ Й АЛГОРИТМІЧНІ ОСНОВИ ДВОВИМІРНОЇ ГРАФІКИ

Виведення зображення на екран монітора і різноманітні дії з ним, такі як відображення, розтягнення, перенесення, поворот, вимагають від користувача математичної грамотності. Геометричні поняття, формули і факти, що відносяться передусім до двовимірного і тривимірного випадків, відіграють у задачах комп'ютерної графіки особливу роль.

3.1. Афінні (лінійні) перетворення на площині

Комп'ютерна графіка на площині оперує об'єктами у двовимірному просторі, тому її прийнято називати 2D графікою (2-Dimension).

Нехай на площині введено прямолінійну координатну систему. Кожній точці M ставиться у відповідність пара чисел (x, y) її координат:



Увівши на площині ще одну прямолінійну систему координат, ми ставимо у відповідність тій самій точці M іншу пару чисел (x^*, y^*).

Перехід від однієї прямолінійної координатної системи на площині до другої описується такими співвідношеннями:

$$x^* = \alpha x + \beta y + \lambda,$$

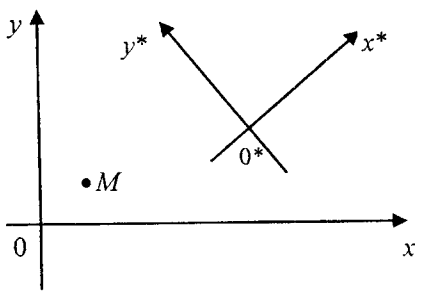
$$y^* = \gamma x + \delta y + \mu, \qquad (*)$$

де $\alpha, \beta, \gamma, \delta, \lambda, \mu$ – деякі числа, пов'язані нерівністю

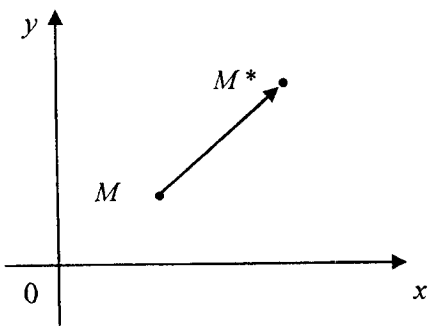
$$\begin{vmatrix} \alpha & \beta \\ \gamma & \delta \end{vmatrix} \neq 0.$$

Ці формули можна трактувати подвійно:

1) або зберігається точка і змінюється координатна система. У цьому випадку довільна точка M замінюється тією ж самою, змінюються лише її координати:



2) або змінюється точка і зберігається координатна система:



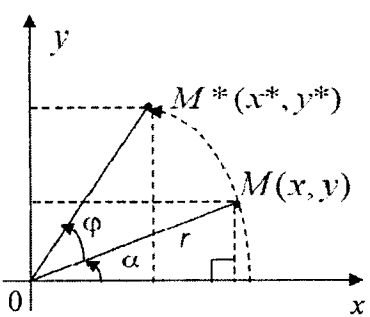
Надалі розглядатимемо формули (*) як правила, згідно з якими в заданій прямолінійній системі координат перетворюються точки площини (випадок 2).

В афінних (лінійних) перетвореннях площини особливу роль грають кілька важливих частинних випадків, що мають добрі геометричні характеристики. Вважатимемо, що задана система координат є прямокутною декартовою (тобто осі перпендикулярні одна одній). Опишемо ці випадки.

А. *Поворот на кут* φ (навколо початку координат) описується формулами:

$$x^* = x\cos\varphi - y\sin\varphi;$$
$$y^* = x\sin\varphi + y\cos\varphi.$$

Виведемо їх:



Нехай r – довжина вектора OM, α – кут між вектором OM і віссю Ox, тоді

$$M(x,y),\ \text{де}\ x = r\cos\alpha, y = r\sin\alpha,$$

$$M^*(x^*,y^*),\ \text{де}\ x^* = r\cos(\alpha+\varphi), y^* = r\sin(\alpha+\varphi).$$

Використовуючи формули для косинусу та синусу суми кутів, маємо

$$x^* = r(\cos\alpha\cos\varphi - \sin\alpha\sin\varphi);$$

$$y^* = r(\cos\alpha\sin\varphi + \sin\alpha\cos\varphi).$$

Оскільки $x = r\cos\alpha, y = r\sin\alpha$, то

$$x^* = x\cos\varphi - y\sin\varphi;$$
$$y^* = x\sin\varphi + y\cos\varphi.$$

Або в матричному вигляді

$$(x^*, y^*) = (x, y)\begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix}.$$

Таким чином, перетворення повороту на кут φ відносно початку координат задається матрицею

$$A_1 = \begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix}.$$

Повороти, що здійснюються проти напрямку руху годинникової стрілки, відповідають додатним кутам.

Визначник матриці повороту має вигляд

$$\det A_1 = \cos^2\varphi + \sin^2\varphi = 1.$$

Б. *Розтягнення (стиснення)* уздовж координатних осей (центр розтягнення збігається з початком координат) описується формулами:

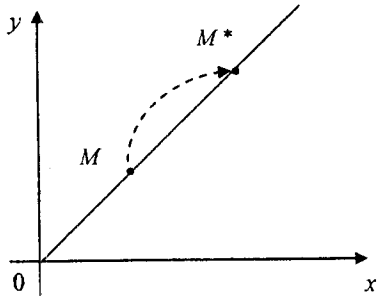
$$x^* = \alpha x,$$

$$y^* = \delta y,$$

де $\alpha > 0, \delta > 0$.

При $\alpha > 1\,(\delta > 1)$ відбувається розтягнення вздовж осей $Ox\,(Oy)$, а при $\alpha < 1\,(\delta < 1)$ – стиснення вздовж $Ox\,(Oy)$.

Якщо $\alpha = \delta$, то маємо пропорційне розтягнення (стиснення):



або в матричному вигляді:

$$(x^*, y^*) = (x, y)\begin{pmatrix} \alpha & 0 \\ 0 & \delta \end{pmatrix}.$$

Таким чином, розтягнення (стиснення) уздовж координатних осей задається матрицею

$$A_2 = \begin{pmatrix} \alpha & 0 \\ 0 & \delta \end{pmatrix}.$$

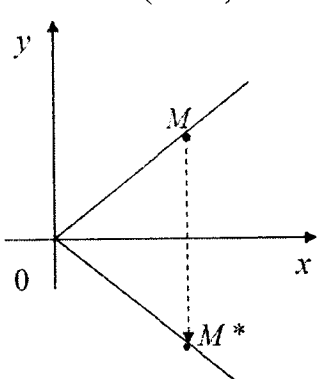
В. *Відображення*

а) відносно осі Ox :

$$x^* = x,$$

$$y^* = -y;$$

$$A_3' = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix};$$



б) відносно осі Oy :

$$x^* = -x,$$
$$y^* = y;$$
$$A_3'' = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}.$$

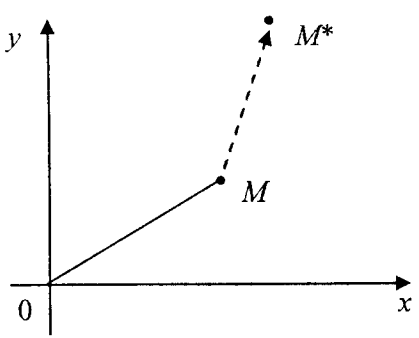
в) відносно прямої $x = y$:

$$x^* = y,$$
$$y^* = x;$$
$$A_3''' = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Визначник матриць відображень дорівнює –1.

Г. *Перенесення* описується формулами:

$$x^* = x + \lambda,$$
$$y^* = y + \mu;$$



Будь-яке перетворення типу (*) завжди можна подати як послідовне виконання перетворень на зразок **А, Б, В, Г.**

У задачах комп'ютерної графіки зручнішим є матричний запис. Однак побудувати матрицю перенесення складніше через наявність у формулах вільних членів.

Проте для розв'язання задач бажано охопити матричним підходом усі чотири найпростіші перетворення, а, отже, і загальне афінне перетворення. Цього можна досягти таким чином: перейти до опису довільної точки площини не впорядкованою парою чисел, а впорядкованою *трійкою* чисел.

3.2. Однорідні координати точки

Нехай M – довільна точка площини з координатами (x, y), підрахованими відносно заданої прямолінійної координатної системи. *Однорідними координатами* цієї точки називається будь-яка трійка одночасно не рівних нулю чисел (x_1, x_2, h), пов'язаних з величинами x і y такими співвідношеннями:

$$\frac{x_1}{h} = x; \qquad \frac{x_2}{h} = y;$$

де h – масштабний множник ($h \neq 0$).

Застосування однорідних координат виявляється зручним вже при розв'язанні найпростіших задач.

Розглянемо, наприклад, питання, пов'язане зі зміною масштабу. Якщо пристрій відображення працює тільки з цілими числами, то для точки з координатами (0,5; 0,1) можна ввести однорідні координати, вибравши $h = 10$, і тим самим звести координати точки до вигляду (5; 1; 10).

Розглянемо інший приклад. Щоб результати перетворення не призводили до арифметичного переповнення, для точки j з координатами (80000; 1000) можна взяти $h = 0{,}001$. У результаті отримаємо точку (80; 1; 0,001).

Наведені приклади показують користь від застосування однорідних координат при розрахунках. Однак основною метою введення однорідних координат у комп'ютерній графіці є їх зручність у застосуванні до геометричних перетворень.

За допомогою трійок однорідних координат і матриць третього порядку можна описати будь-яке афінне перетворення площини.

Однорідні координати в комп'ютерній графіці вводять так: будь-якій точці $M(x,y)$ площини ставиться у відповідність точка $M(x,y,1)$ у просторі (тобто $h=1$).

Тоді співвідношення (*) буде еквівалентне такому:

$$(x^*,y^*,1)=(x,y,1)\begin{pmatrix}\alpha & \gamma & 0\\ \beta & \delta & 0\\ \lambda & \mu & 1\end{pmatrix}. \qquad (**)$$

Після перемноження виразів у правій частині (**), ми отримаємо вираз (*) і правильну числову рівність 1=1.

Елементи довільної матриці афінного перетворення не несуть у собі явно вираженого геометричного змісту. Щоб знайти елементи матриці за заданим геометричним описом, побудову матриці розбивають на кілька етапів.

На кожному етапі шукають матрицю, що відповідає тому чи іншому з виділених вище випадків **А**, **Б**, **В** або **Г**, які мають добре виражені геометричні властивості.

Випишемо відповідні матриці третього порядку.

А. *Матриця обертання (rotation):*

$$R=\begin{pmatrix}\cos\varphi & \sin\varphi & 0\\ -\sin\varphi & \cos\varphi & 0\\ 0 & 0 & 1\end{pmatrix}.$$

Б. *Матриця розтягнення (стиснення) (dilatation):*

$$D=\begin{pmatrix}\alpha & 0 & 0\\ 0 & \delta & 0\\ 0 & 0 & 1\end{pmatrix}.$$

В. *Матриця відображення відносно осі Ox (reflection):*

$$M=\begin{pmatrix}1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & 1\end{pmatrix}.$$

Г. *Матриця перенесення (translation):*

$$T=\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ \lambda & \mu & 1\end{pmatrix}.$$

3.3. Комбіновані перетворення

1) Нехай $R_1=\begin{pmatrix}\cos\varphi_1 & \sin\varphi_1 & 0\\ -\sin\varphi_1 & \cos\varphi_1 & 0\\ 0 & 0 & 1\end{pmatrix}$ і $R_2=\begin{pmatrix}\cos\varphi_2 & \sin\varphi_2 & 0\\ -\sin\varphi_2 & \cos\varphi_2 & 0\\ 0 & 0 & 1\end{pmatrix}$ – матриці повороту на кут φ_1 і φ_2 відповідно. Розглянемо

$$R_1R_2=\begin{pmatrix}\cos\varphi_1\cos\varphi_2-\sin\varphi_1\sin\varphi_2 & \cos\varphi_1\sin\varphi_2+\sin\varphi_1\cos\varphi_2 & 0\\ -(\sin\varphi_1\cos\varphi_2+\sin\varphi_2\cos\varphi_1) & -\sin\varphi_1\sin\varphi_2+\cos\varphi_1\cos\varphi_2 & 0\\ 0 & 0 & 1\end{pmatrix}=$$

$$=\begin{pmatrix}\cos(\varphi_1+\varphi_2) & \sin(\varphi_1+\varphi_2) & 0\\ -\sin(\varphi_1+\varphi_2) & \cos(\varphi_1+\varphi_2) & 0\\ 0 & 0 & 1\end{pmatrix}.$$

Таким чином, два, а отже і будь-яку кількість послідовних поворотів можна записати у вигляді однієї матриці сумарного повороту.

2) Нехай $D_1=\begin{pmatrix}\alpha_1 & 0 & 0\\ 0 & \delta_1 & 0\\ 0 & 0 & 1\end{pmatrix}$ і $D_2=\begin{pmatrix}\alpha_2 & 0 & 0\\ 0 & \delta_2 & 0\\ 0 & 0 & 1\end{pmatrix}$ – матриці розтягнення. Розглянемо

$$D_1D_2=\begin{pmatrix}\alpha_1\alpha_2 & 0 & 0\\ 0 & \delta_1\delta_2 & 0\\ 0 & 0 & 1\end{pmatrix}.$$

Отже, послідовні розтягнення є мультиплікативними.

3) Нехай $D_1=\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ \lambda_1 & \mu_1 & 1\end{pmatrix}$ і $D_2=\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ \lambda_2 & \mu_2 & 1\end{pmatrix}$ – матриці перенесення. Розглянемо

$$D_1D_2=\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ \lambda_1+\lambda_2 & \mu_1+\mu_2 & 1\end{pmatrix}.$$

Отже, послідовні перенесення є адитивними.

Більше того, будь-яку послідовність операцій, що включає в себе поворот, розтягнення, відображення і перенесення в однорідних координатах, можна подати однією матрицею, що є добутком матриць цих операцій.

Приклад

Побудувати матрицю повороту на кут φ навколо точки $A(a,b)$.

1 крок. Перенесення на вектор $(-a,-b)$ для суміщення центра повороту з початком координат:

$$T_1=\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ -a & -b & 1\end{pmatrix}-$$

матриця відповідного перетворення.

2 крок. Поворот на кут φ :

$$R_{\phi} = \begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

3 крок. Перенесення на (a,b) для повернення центра повороту в початкове положення (a,b):

$$T_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ a & b & 1 \end{pmatrix}.$$

Перемножимо матриці в тому самому порядку, в якому вони були виписані:

$$T_1 \cdot R_{\varphi} \cdot T_2 .$$

У результаті отримаємо шукане перетворення в матричному вигляді

$$(x^*, y^*, 1) =$$

$$= (x, y, 1) \begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ -a\cos\varphi + b\sin\varphi + a & -a\sin\varphi - b\cos\varphi + b & 1 \end{pmatrix}.$$

Елементи отриманої матриці не так легко запам'ятати, тоді як кожна із трьох матриць, що перемножуються, за геометричним описом відповідного перетворення будується досить легко. Отже, розбиваючи перетворення на етапи, що підтримуються матрицями R, D, M, T, можна побудувати матрицю будь-якого афінного перетворення за його геометричним описом.

КОНТРОЛЬНІ ЗАПИТАННЯ ДО РОЗДІЛУ 3

1. Яким співвідношенням описується перехід від однієї прямолінійної координатної системи на площині до іншої?

2. Які афінні перетворення на площині вам відомі?

3. Випишіть відповідні матриці афінних перетворень на площині із запитання 2.

3. Що таке однорідні координати точки?

4. Випишіть матриці афінних перетворень на площині в однорідних координатах.

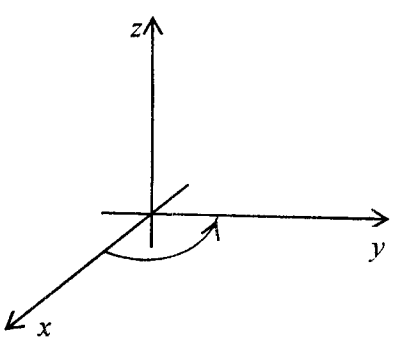
5. Яким чином здійснюються комбіновані перетворення на площині?

Розділ 4

МАТЕМАТИЧНІ ОСНОВИ ТРИВИМІРНОЇ ГРАФІКИ

У розділі 3 ми розглянули афінні перетворення на площині й побудували матриці повороту, розтягнення, відображення та перенесення на площині.

Перейдемо тепер до тривимірного випадку 3D (3-dimention). Одразу визначимося у такому: застосовуватимемо *праву систему декартових координат*. Тобто, якщо вісь Ox повернути на 90° проти напрямку руху годинникової стрілки, причому спостерігати такий поворот із боку додатного напрямку осі Oz, то додатний напрямок осі Ox збігається з додатним напрямком осі Oy:

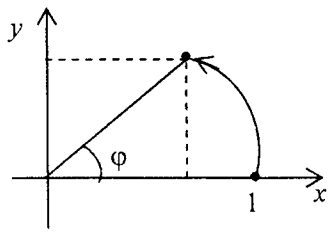


Нагадаємо, що додатним кутам відповідає поворот вектора у напрямку руху проти годинникової стрілки у правій системі координат.

4.1. Афінні перетворення у просторі

Почнемо з матриць повороту.

Випишемо лінійне перетворення, яке описує поворот площини, що перпендикулярна осі Oz, навколо цієї осі на кут φ в додатному напрямку.



$(z = \text{const})$

Маємо

$$x^* = x\cos\varphi - y\sin\varphi,$$

$$y^* = x\sin\varphi + y\cos\varphi,$$

$$z^* = z.$$

Або у матричному вигляді

$$(x^*, y^*, z^*) = (x, y, z)\begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Відповідно матрицю, що описує поворот навколо осі аплікат на кут φ, запишемо так:

$$A_\varphi = \begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Аналогічно подамо рівняння повороту навколо осі Oy на кут ψ:

$$x^* = x\cos\psi - z\sin\psi,$$

$$y^* = y,$$

$$z^* = x\sin\psi + z\cos\psi.$$

Це перетворення описує поворот у напрямку від осі Ox до осі Oz, який відбувається у напрямку за годинниковою стрілкою, якщо дивитися з додатного напрямку осі Oy.

Як уже зазначено, за додатний напрямок повороту беруть поворот проти годинникової стрілки. Отже замінимо ψ на $-\psi$, і, враховуючи, що $\cos(-\psi) = \cos\psi$ і $\sin(-\psi) = -\sin\psi$, отримаємо

$$x^* = x\cos\psi + z\sin\psi,$$

$$y^* = y,$$

$$z^* = -x\sin\psi + z\cos\psi.$$

Або в матричному вигляді

$$(x^*, y^*, z^*) = (x, y, z)\begin{pmatrix} \cos\psi & 0 & -\sin\psi \\ 0 & 1 & 0 \\ \sin\psi & 0 & \cos\psi \end{pmatrix}.$$

Тобто, матрицю повороту навколо осі ординат на кут ψ запишемо так:

$$A_\psi = \begin{pmatrix} \cos\psi & 0 & -\sin\psi \\ 0 & 1 & 0 \\ \sin\psi & 0 & \cos\psi \end{pmatrix}.$$

І нарешті, поворот навколо осі Ox на кут χ в додатному напрямку описується таким лінійним перетворенням:

$$x^* = x,$$

$$y^* = y\cos\chi - z\sin\chi,$$

$$z^* = y\sin\chi + z\cos\chi.$$

Тобто

$$(x^*, y^*, z^*) = (x, y, z)\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\chi & \sin\chi \\ 0 & -\sin\chi & \cos\chi \end{pmatrix}.$$

Отже, матриця повороту навколо осі абсцис на кут χ має вигляд

$$A_\chi = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\chi & \sin\chi \\ 0 & -\sin\chi & \cos\chi \end{pmatrix}.$$

Перейдемо тепер до однорідних координат.

Міркуючи аналогічно тому, як це було зроблено у випадку розмірності два, замінимо координатну трійку (x, y, z), що задає точку у просторі, на четвірку чисел $(x, y, z, 1)$ або, у більш загальному вигляді, на четвірку (hx, hy, hz, h), $h \neq 0$.

Кожна точка простору може бути задана четвіркою одночасно не рівних нулю чисел; ця четвірка чисел визначена однозначно з точністю до спільного множника.

Запропонований підхід до нового способу задання точок дає можливість скористатись матричним записом і у складніших, тривимірних задачах.

Будь-яке афінне перетворення у тривимірному просторі можна представити у вигляді суперпозиції (композиції) поворотів, розтягнень, відображень і перенесень. Тому детально опишемо матриці саме цих перетворень (зрозуміло, що в цьому випадку порядок матриць повинен дорівнювати чотирьом).

Аналогічно двовимірному випадку, за допомогою четвірок однорідних координат і матриці четвертого порядку, вважаючи, що $h = 1$, загальне афінне перетворення можна записати так:

$$(x^*, y^*, z^*, 1) = (x, y, z, 1) \cdot A,$$

де A – матриця деякого перетворення.

Випишемо тепер відповідні матриці четвертого порядку.

А. *Матриці повороту у просторі.*

1) Матриця повороту навколо осі Oz на кут φ:

$$R_z = \begin{pmatrix} \cos\varphi & \sin\varphi & 0 & 0 \\ -\sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

2) Матриця повороту навколо осі Oy ну кут ψ:

$$R_y = \begin{pmatrix} \cos\psi & 0 & -\sin\psi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\psi & 0 & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

3) Матриця повороту навколо осі Ox на кут χ:

$$R_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\chi & \sin\chi & 0 \\ 0 & -\sin\chi & \cos\chi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Б. *Матриці розтягнення (стиснення):*

$$D = \begin{pmatrix} \alpha & 0 & 0 & 0 \\ 0 & \beta & 0 & 0 \\ 0 & 0 & \gamma & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

де $\alpha > 0$ – коефіцієнт розтягнення (стиснення) уздовж осі Ox,

$\beta > 0$ – коефіцієнт розтягнення (стиснення) уздовж осі Oy,

$\gamma > 0$ – коефіцієнт розтягнення (стиснення) уздовж осі Oz.

В. *Матриці відображення.*

1) Матриця відображення відносно площини xOy:

$$M_z = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

2) Матриця відображення відносно площини yOz:

$$M_x = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

3) Матриця відображення відносно площини xOz:

$$M_y = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Г. *Матриці перенесення* (тут (λ,μ,ν) – вектор перенесення):

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \lambda & \mu & \nu & 1 \end{pmatrix}.$$

Зауважимо, що як і у двовимірному випадку, усі виписані матриці є невиродженими, тобто $\det A \neq 0$.

4.2. Комбіновані перетворення

Наведемо приклад побудови матриці складного перетворення за його геометричним описом.

Приклад 1

Побудувати матрицю повороту на кут φ навколо прямої (осі) L, яка проходить через точку $A(a,b,c)$ і має напрямлений вектор (l,m,n). (Напрямлений вектор – це вектор, що лежить на даній прямій або на паралельний до неї.) Можна вважати, що напрямлений вектор прямої є одиничним, тобто $l^2+m^2+n^2=1$.

Оскільки метод повороту навколо координатної осі відомий, то основна ідея полягає в тому, щоб сумістити довільну вісь обертання з однією з координатних осей.

Поворот навколо осі на деякий кут φ виконується за таким алгоритмом.

1) Виконати перенесення на вектор $(-a,-b,-c)$ так, щоб пряма (вісь) L проходила через початок координат, за допомогою матриці

$$T=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -a & -b & -c & 1 \end{pmatrix}.$$

2) Суміщення прямої (осі) L з віссю Oz (вибір осі довільний) двома поворотами: перший – навколо осі Ox на кут α (для переводу прямої L у площину xOz) і другий – навколо осі Oy на кут β (для суміщення прямої L із віссю Oz) за допомогою таких матриць:

$$R_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad R_y = \begin{pmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Ці кути (а точніше $\cos\alpha$, $\sin\alpha$, $\cos\beta$, $\sin\beta$) також знаходять після побудови проекції на площину yOz напрямленого одиничного вектора нашої осі:

$$\cos\alpha = \frac{n}{d}, \ \sin\alpha = \frac{m}{d}, \ \cos\beta = l, \ \sin\beta = -d,$$

де $d = \sqrt{m^2 + n^2}$.

1) Поворот навколо прямої (нашої осі) L на заданий кут φ за допомогою матриці

$$R_z = \begin{pmatrix} \cos\varphi & \sin\varphi & 0 & 0 \\ -\sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

2) Поворот на кут $(-\beta)$ навколо осі Oy за допомогою матриці R_y^{-1}.

3) Поворот на кут $(-\alpha)$ навколо осі Ox за допомогою матриці R_x^{-1}.

4) Поворот на кут (a,b,c) за допомогою матриці T^{-1}.

Перемноживши знайдені матриці в порядку їх побудови, отримаємо таку матрицю (пропонуємо її знайти читачеві самостійно):

$$T \cdot R_x \cdot R_y \cdot R_z \cdot R_y^{-1} \cdot R_x^{-1} \cdot T^{-1}.$$

Розглядаючи інші приклади на зразок прикладу 1, ми отримуватимемо в результаті невироджені матриці вигляду

$$A=\begin{pmatrix}\alpha_1 & \alpha_2 & \alpha_3 & 0\\ \beta_1 & \beta_2 & \beta_3 & 0\\ \gamma_1 & \gamma_2 & \gamma_3 & 0\\ \lambda & \mu & \nu & 1\end{pmatrix}.$$

За допомогою таких матриць ми можемо перетворювати будь-які просторові фігури.

Приклад 2

Застосувати афінне перетворення до опуклого багатогранника.

1) Спочатку за геометричним описом знаходимо матрицю A перетворення (це матриця четвертого порядку, невироджена).

2) Довільний опуклий багатогранник однозначно задаємо набором усіх своїх вершин:

$$V_i=(x_i,y_i,z_i),\ i=\overline{1,n}.$$

Тому будуємо матрицю

$$V=\begin{pmatrix}x_1 & y_1 & z_1 & 1\\ x_2 & y_2 & z_2 & 1\\ . & . & . & .\\ x_n & y_n & z_n & 1\end{pmatrix}.$$

3) $V\cdot A=V'$, де V' дає набір вершин нового опуклого багатогранника.

КОНТРОЛЬНІ ЗАПИТАННЯ ДО РОЗДІЛУ 4

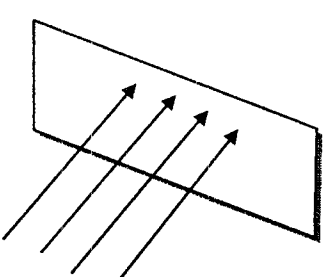
1. Які афінні перетворення у просторі вам відомі?
2. Випишіть відповідні матриці афінних перетворень у просторі із запитання 1.
3. Випишіть матриці афінних перетворень у просторі в однорідних координатах.
4. Яким чином здійснюються комбіновані перетворення у просторі?
5. За яким алгоритмом можна застосувати афінне перетворення до опуклого багатогранника?

Розділ 5
ПРОЕКТУВАННЯ ТА ЙОГО ВИДИ

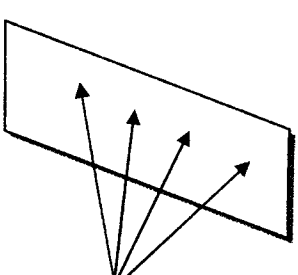
5.1. Види проектування: паралельне та центральне

Для зображення тривимірної сцени на двовимірному екрані застосовують операцію, яка називається *проектуванням* за допомогою пучка прямих. У комп'ютерній графіці на практиці використовують зазвичай два види проектування: паралельне і центральне (перспективне).

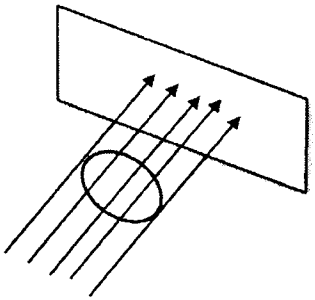
Для отримання проекції об'єкта на площину (її часто називають *картинною площиною*) необхідно провести через кожну його точку пряму із заданого *проектуючого пучка* (власного або невласного), а потім знайти координати точки перетину цієї прямої із площиною зображення. У випадку *центрального проектування* всі прямі виходять з однієї точки – центра *власного пучка*. При *паралельному проектуванні* центр *невласного пучка* вважається таким, що лежить у нескінченності:



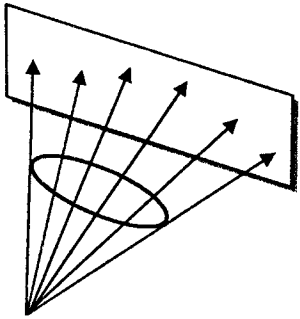
Паралельне проектування (невласний пучок)



Центральне проектування (невласний пучок)

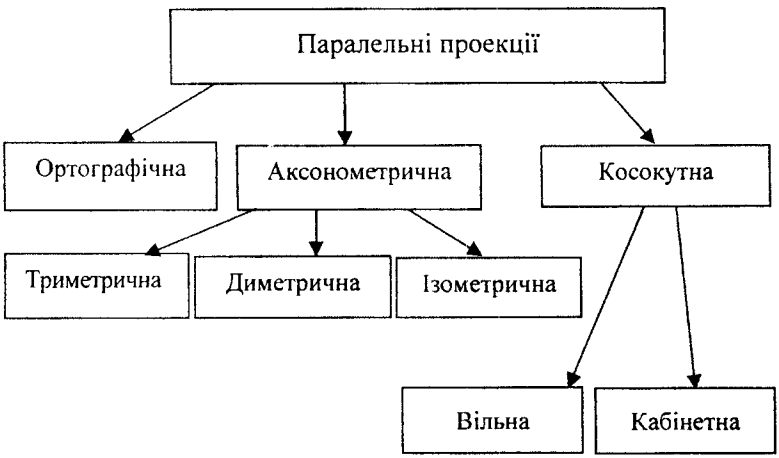


Паралельне проектування



Центральне проектування

Кожний із цих двох основних класів розбивається на кілька підкласів залежно від взаємного розташування картинної площини та координатних осей. Наведемо відповідні схеми паралельних і перспективних проекцій:



5.2. Види паралельних проекцій

При *ортографічній проекції* картинна площина збігається з однією із координатних площин або паралельна їй.

У випадку, коли картинна площина збігається з координатною площиною, для відображення тривимірного об'єкта на двовимірному моніторі відбувається "обнуління" однієї із трьох координат усіх точок.

Матриця проектування вздовж осі Ox на площину yOz має вигляд

$$P_x = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

тобто $x^* = 0, y^* = y, z^* = z$.

У випадку, якщо площина проектування паралельна координатній площині, а не збігається з нею, необхідно помножити матрицю P_x на матрицю зсуву. У результаті отримаємо

$$P_x \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & 0 & 0 & 1 \end{pmatrix} = P_x^p .$$

Аналогічно записують матриці проектування вздовж двох інших координатних осей:

$$P_y^q = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & q & 0 & 1 \end{pmatrix}; \quad P_z^r = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & r & 1 \end{pmatrix}.$$

При *аксонометричній проекції* проектуючі прямі перпендикулярні картинній площині, і залежно від взаємного розташування площини проектування та координатних осей розрізняють три види проекцій:

1) *триметрія* – коли нормальний вектор картинної площини (тобто вектор, перпендикулярний картинній площині) утворює з кожною координатною віссю різні кути;

2) *диметрія* – коли два кути між нормаллю координатної площини і координатними осями рівні;

3) *ізометрія* – коли всі три кути між нормаллю координатної площини і координатними осями рівні.

Кожний із трьох видів указаних проекцій отримують комбінацією поворотів, за якими слідує паралельне проектування (для того, щоб нормаль до площини стала паралельною або збігалася з координатною віссю).

Приклад

При повороті на кут ψ відносно осі ординат, на кут φ навколо осі абсцис і наступного проектування вздовж осі аплікат виникає матриця

$$M = \begin{pmatrix} \cos\psi & \sin\varphi\sin\psi & 0 & 0 \\ 0 & \cos\psi & 0 & 0 \\ \sin\psi & -\sin\varphi\cos\psi & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} =$$

$$= \begin{pmatrix} \cos\psi & 0 & -\sin\psi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\psi & 0 & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi & 0 \\ 0 & -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Проекції, для отримання яких використовують пучок прямих, *не перпендикулярних* площині екрана, прийнято називати *косокутними*.

При косокутному проектуванні орта осі Oz на площину xOy вектор $(0,0,1,1)$ переходить у вектор $(\alpha,\beta,0,1)$.

Матриця відповідного перетворення має вигляд

$$D=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ \alpha & \beta & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix},$$

оскільки $(0,0,1,1)\cdot D=(\alpha,\beta,0,1)$.

Виділяють два типи косокутних проекцій: *вільну проекцію* (кут нахилу проектуючих прямих до площини екрана дорівнює половині прямого, тобто $\alpha=\beta=\cos\frac{\pi}{4}$); *кабінетну проекцію* (частинний випадок вільної проекції, коли $\alpha=\beta=\frac{1}{2}\cos\frac{\pi}{4}$).

Перспективні (центральні) проекції будуються значно складніше і нами детально не розглядатимуться. Зазначимо лише таке: залежно від того, скільки координатних осей перетинає проекційну площину, розрізняють *одно-, дво-, і триточкові проекції*.

Укажемо лише основні властивості центрального перетворення. У центральній проекції:

1) не зберігаються відношення довжин і площ;

2) прямі лінії зображують прямими лініями;

3) паралельні прямі зображують такими, що збігаються в одній точці (цю властивість широко використовують у нарисній геометрії для ручного малювання на папері).

КОНТРОЛЬНІ ЗАПИТАННЯ ДО РОЗДІЛУ 5

1. Які основні типи проектування вам відомі?
2. Як будуються паралельні та перспективні (центральні) проекції? Що таке власний та невласний пучки?
3. Які види паралельних проекцій вам відомі?
4. Як будується ортографічна проекція?
4. Які типи аксонометричних проекцій вам відомі?
5. Що таке косокутна проекція?
6. Які основні властивості центрального перетворення?

Розділ 6
БАЗОВІ РАСТРОВІ АЛГОРИТМИ (ОГЛЯД)

6.1. Поняття 4-зв'язності та 8-зв'язності

Сформувати растрове зображення можна по-різному.

Для того, щоб створити зображення на растровому дисплеї, можна просто скопіювати готовий растр у відеопам'ять. Цей растр можна отримати, наприклад, за допомогою сканера чи цифрового фотоапарата. А можна створити зображення об'єкта шляхом послідовного малювання окремих простих елементів.

Прості елементи, із яких складаються складні об'єкти, називаються графічними примітивами. Найпростішим і водночас найуніверсальнішим растровим графічним примітивом є піксель. Будь-яке растрове зображення можна намалювати по пікселях, однак це складно і довго. Необхідні більш складні елементи, для яких малюються зразу декілька пікселів. Розглянемо графічні примітиви, які використовуються найчастіше в сучасних графічних системах – це лінії та фігури.

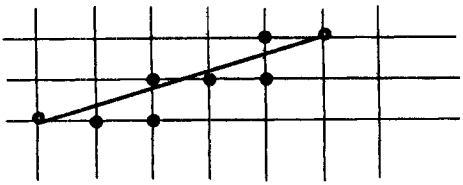
Достатньо важливим поняттям для растрової сітки є зв'язність – можливість з'єднання двох пікселів растровою лінією, тобто послідовним набором пікселів. Виникає питання, коли пікселі (x_1, y_1) і (x_2, y_2) можна вважати сусідніми?

Уведемо два поняття зв'язності.

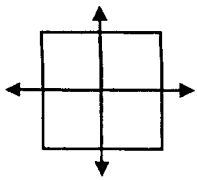
1) *4-зв'язність.*

Пікселі вважаються *сусідніми*, якщо або їх x-координати, або їх y-координати відрізняються на одиницю:

$$|x_1 - x_2| + |y_1 - y_2| \le 1;$$



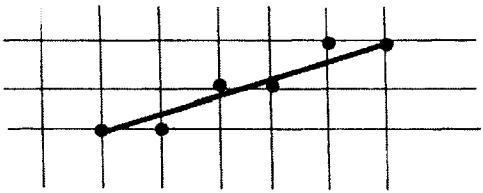
Тобто, при переході до сусіднього пікселя є 4 можливості:



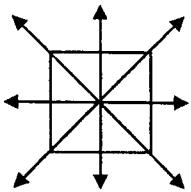
2) *8-зв'язність.*

Пікселі вважаються сусідніми, якщо їхні x- та y-координати відрізняються не більше ніж на одиницю:

$$|x_1 - x_2| \le 1, |y_1 - y_2| \le 1;$$



Тобто, при переході до сусіднього пікселя є 8 можливостей:



Поняття 4-зв'язності є більш сильним: будь-які 4-зв'язні пікселі є і 8-зв'язними, але не навпаки.

Як *лінію* на растровій сітці використовують набір пікселів $P_1, P_2, ...P_n$, де будь-які два пікселі P_i та P_{i+1} є сусідніми у сенсі заданої зв'язності.

Зауваження. Оскільки поняття лінії базується на понятті зв'язності, то природним чином виникає поняття 4- та 8-зв'язних ліній. Тому, коли ми говоримо про растрове представлення (наприклад, відрізка), то треба ясно розуміти, про яке саме представлення йдеться. У загальному випадку растрове представлення об'єкта не є єдиним і можливі різні способи його побудови.

6.2. Растрове представлення відрізка. Алгоритм Брезенхейма

Розглянемо задачу побудови растрового зображення відрізка, що з'єднує точки $A(x_1, y_1)$ і $B(x_2, y_2)$. Для простоти вважатимемо, що $0 \le y_2 - y_1 \le x_2 - x_1$. Запишемо рівняння прямої, що проходить через дві точки:

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1};$$

$$(y - y_1)(x_2 - x_1) = (y_2 - y_1)(x - x_1);$$

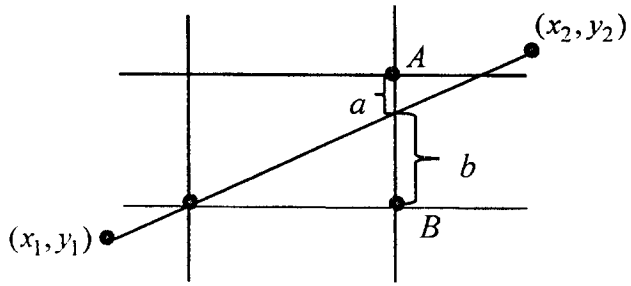
$$y = y_1 + \frac{(y_2 - y_1)}{(x_2 - x_1)}(x - x_1), \text{ де } x \in [x_1, x_2],$$

Або $$y = kx + b, \text{ де } b = y_1 - kx_1,\ k = \frac{y_2 - y_1}{x_2 - x_1}.$$

Отримані формули і дають найпростіший алгоритм растрового представлення відрізка. Проте цей алгоритм має великий недолік – використання дійсних чисел для роботи на цілочисельній решітці.

У 1965 р. *Джек Елтон Брезенхейм* запропонував підхід, що дозволив розробити так звані *інкрементні алгоритми растеризації.* Основною метою для розробки таких алгоритмів була побудова циклів обчислення координат на основі тільки цілочисельних операцій додавання і віднімання без використання множення і ділення. У результаті досягнуто підвищення швидкодії для обчислення кожного пікселя порівняно із прямим способом.

Ідея алгоритму така:



При побудові растрового зображення відрізка завжди вибирають найближчий по вертикалі піксель. При цьому із двох точок A і B (малюнок) беруть ту, яка ближче до вихідної прямої (у нашому випадку вибирають точку A, оскільки $a < b$). Це *метод серединної точки.*

Далі вводять число $d_i = (x_2 - x_1)(b - a) = \Delta x(b - a)$. У випадку $d_i > 0$ значення y від попередньої точки збільшується на 1, а d_i – на $2(\Delta y - \Delta x)$. У випадку $d_i < 0$ значення y не змінюється, а значення d_i змінюється на $2\Delta y$.

Відповідні програмні коди на C++ можна подивитись у [3].

6.3. Алгоритм Брезенхейма для генерацій кола

Для виведення контуру кола можна використати відношення між координатами X і Y для точок кола $X^2 + Y^2 = R^2$ і побудувати алгоритми прямого обчислення координат. Однак тоді слід обчислити квадратний корінь, що займає досить багато часу.

Один із найефективніших і простих для розуміння алгоритмів генерацій кола також належить Брезенхейму. У цьо-

му алгоритмі використано симетрію кола. Необхідно згенерувати тільки 1/8 частину кола (один октант). Інші її частини можна отримати послідовними відображеннями. Якщо згенеровано перший октант (від 0 до 45° у напрямку проти руху годинникової стрілки), то другий октант можна отримати дзеркальним відображенням відносно прямої $y = x$, що дає в сукупності перший квадрант. Перший квадрант відображається відносно прямої $x = 0$ для отримання відповідної частини кола у другому квадранті. Верхнє півколо відображається відносно прямої $y = 0$ для завершення побудови.

Двовимірні матриці відповідних перетворень відображають так:

1) $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ – відносно прямої $y = x$;

2) $\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$ – відносно осі $Oy\,(x = 0)$;

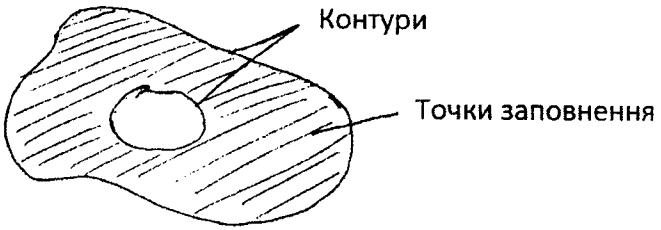
3) $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ – відносно осі $Ox\,(y = 0)$.

Для генерації одного октанта кола також використовують алгоритм серединної точки. Існують інкрементні алгоритми для генерації еліпса, гіперболa, параболи.

6.4. Алгоритми зафарбування

Вважатимемо фігурою плоский геометричний об'єкт, який складається з ліній контуру і точок всередині нього.

У загальному випадку ліній контуру може бути декілька – коли об'єкт має всередині порожнини. У цьому разі для опису таких фігур необхідно два и більше контурів:



Процес графічного виведення фігур розділяють на дві задачі: виведення контуру и виведення точок заповнення. Оскільки контур являє собою лінію, то виведення контуру виконують на основі алгоритмів виведення ліній. Залежно від складності контуру це можуть бути відрізки прямих, кривих або довільна послідовність сусідніх пікселів. Для виведення точок заповнення застосовують методи двох типів: алгоритми зафарбовування від внутрішньої точки до границь довільного контуру та алгоритми, в яких використано математичний опис контуру.

У загальному вигляді *алгоритм зафарбування* довільного контуру, який уже намальований у растрі, припускає виконання таких дій:

1) знаходимо піксель усередині контуру фігури;

2) колір цього пікселя змінюємо на потрібний колір заповнення;

3) аналізуємо сусідні пікселі: якщо колір деякого сусіднього пікселя не дорівнює кольору межі контуру або кольору заповнення, то колір цього пікселя змінюється на колір заповнення;

4) аналізуємо колір пікселів, сусідніх із попередніми; і так далі доти, поки всередині контуру всі пікселі не перефарбуються в колір заповнення.

Пікселі контуру – це границя, за яку не можна виходити у процесі послідовного перебору всіх сусідніх пікселів. Сусідніми можуть вважатися лише чотири пікселі (справа, зліва, зверху і знизу – у випадку 4-зв'язності) або всі вісім пікселів (у випадку 8-зв'язності). У цьому алгоритмі використовують рекурсію, але, як показує практика, цей алгоритм неприпустимий для фігур площею тисяча і більше пікселів через велику глибину рекурсії.

6.4.1. Хвильовий алгоритм зафарбування

Суть алгоритму полягає в тому, що для початкової точки (вершині на графі) знаходять сусідні точки (інші вершини), які відповідають двом умовам:

1) ці вершини зв'язані з початковою;

2) ці вершини ще не відмічені, тобто вони розглядаються вперше.

Сусідні вершини поточної ітерації зазначають у масиві опису вершин, і кожна з них стає поточною точкою для пошуку нових сусідніх вершин у наступній ітерації.

Якщо у спеціальному масиві відмічати кожну вершину номером ітерації та, коли буде досягнуто кінцевої точки, можна зробити зворотний цикл – від кінцевої точки до початкової за спаданням номерів ітерацій. У процесі зворотного циклу і знаходять усі найкоротші шляхи (якщо їх декілька) між двома заданими точками на графі. При зафарбуванні растрових фігур, вершинами графа є пікселі, а відмітка пройдених пікселів робиться прямо в растрі кольором зафарбування.

Як бачимо, це майже повністю повторює ідею попереднього найпростішого алгоритму, однак тут не використовується рекурсія. Це обумовлює зовсім іншу послідовність обробки пікселів при зафарбуванні.

Зауважимо, що цей алгоритм не є найшвидшим із відомих алгоритмів зафарбування. Більшу швидкість зафарбування забезпечують алгоритми, які обробляють не окремі пікселі, а відразу великі блоки з багатьох пікселів, які утворюють прямокутники або лінії.

6.4.2. Алгоритм зафарбування лініями

Алгоритм зафарбування лініями широко розповсюджений у комп'ютерній графіці. Від наведеного першого рекурсивного алгоритму він відрізняється тим, що на кожному кроці зафарбування малюється горизонтальна лінія, яка розміщуєть-

ся між пікселями контуру. Алгоритм рекурсивний, але оскільки виклик функції здійснюється для лінії, а не для кожного окремого пікселя, то кількість вкладених викликів зменшується пропорційно довжині лінії.

6.5. Відсікання відрізка. Алгоритм Коена – Сазерленда

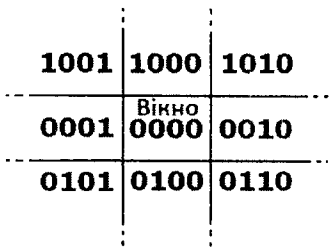
Розглянемо проблему, яка неявно присутня в більшості задач комп'ютерної графіки – це проблема відсікання зображення по деякій межі, наприклад, по межі екрана, або, у загальному випадку, – деякого прямокутного вікна.

Розглянемо задачу щодо відрізків прямих. Деякі з них повністю лежать усередині області екрана, інші – цілком поза нею, а деякі перетинають межу екрана. Правильне відображення відрізків означає знаходження точок перетину їх із межею екрана та малювання лише тих їхніх частин, які потрапляють на екран.

Розглянемо достатньо простий и ефективний алгоритм відсікання відрізків по межі довільного прямокутника – *алгоритм Коена – Сазерленда.*

У цьому алгоритмі вся площина розбивається на дев'ять областей прямими, що утворюють прямокутник (вікно). Для визначення тієї з дев'яти областей, якій належить кінець відрізка, вводять чотирибітовий код.

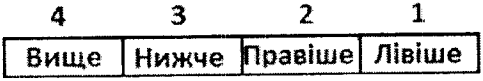
Коди цих областей показано на малюнку:



Крайній правий біт коду вважають *першим*.

У відповідний біт заноситься 1 за виконання таких умов:

- для першого біта – якщо точка перебуває лівіше вікна;
- для другого біта – якщо точка перебуває правіше вікна;
- для третього біта – якщо точка перебуває нижче вікна;
- для четвертого біта – якщо точка перебуває вище вікна.



В інших випадках у біт заноситься 0.

Звідси, очевидно, випливає таке: якщо коди обох кінців відрізка рівні нулю, то обидві ці точки лежать усередині вікна і відрізок видимий. Коди кінцевих точок можна використовувати і для тривіального відкидання повністю невидимих відрізків.

Нагадаємо таблицю істинності для логічного оператора "*і*" (1 – "Істина", 0 – "Хибність"):

x	*y*	"*і*"
1	0	0
0	1	0
0	0	0
1	1	1

Якщо побітовий логічний добуток кодів кінцевих точок відрізка не дорівнює нулю, то відрізок повністю невидимий і його можна тривіально відкинути.

Розглянемо *чотири ситуації*.

1) $1001 \times 1000 = 1000$ – відрізок невидимий цілком;
2) $0000 \times 0000 = 0000$ – відрізок цілком видимий;
3) $0000 \times 0010 = 0000$ – відрізок частково видимий;
4) $1000 \times 0010 = 0000$ – відрізок цілком невидимий.

Наведемо пояснення.

У ситуації 1) відрізок зразу відкидається.

У ситуації 2) відрізок приймається цілком.

У ситуації 3) відрізок не можна ані прийняти, ані відкинути, тому що він може перетинатися з вікном. У цьому випадку застосовується послідовний поділ відрізка так, що на кожному кроці кінцева точка відрізка з ненульовим кодом замінюється на точку, що лежить на стороні вікна або на прямій відрізка.

У ситуації 4) відрізок цілком невидимий.

Тобто, якщо логічний добуток дорівнює нулю, то відрізок може виявлятися цілком чи частково видимим або навіть цілком невидимим. Тому для визначення повної видимості необхідно перевіряти значення кодів обох кінців відрізка окремо.

Якщо передусім знайдено цілком видимі та тривіально невидимі відрізки, то далі обробляють лише ті відрізки, які, можливо, частково видимі, тобто такі, для яких результат логічного множення кодів їх кінцевих точок рівний нулю.

КОНТРОЛЬНІ ЗАПИТАННЯ ДО РОЗДІЛУ 6

1. Сформулюйте поняття 4-зв'язності та 8-зв'язності.
2. У чому полягає задача побудови растрового зображення відрізка?
3. Опишіть алгоритм Брезенхейма.
4. У чому полягає метод серединної точки?
5. Опишіть алгоритм Брезенхейма для генерацій кола.
6. Які алгоритми зафарбовування вам відомі?
7. Опишіть загальний алгоритм зафарбування.
8. У чому полягає хвильовий алгоритм зафарбування?
9. Опишіть алгоритм зафарбування лініями.
10. У чому полягає проблема відсікання зображення по деякій межі?
11. Опишіть алгоритм Коена – Сазерленда.

Розділ 7

ФОРМАТИ ГРАФІЧНИХ ФАЙЛІВ ТА АЛГОРИТМИ СТИСНЕННЯ

Практично кожна прикладна програма створює і зберігає деякі види графічних даних. Тому в рамках комп'ютерної графіки стрімко розвивається відносно нова область – алгоритми архівації зображень. Поява цієї області обумовлена тим, що зображення – це своєрідний тип даних, який характеризується трьома особливостями:

1. Зображення (як і відео) займає набагато більше місця в пам'яті, ніж текст. Ця особливість зображень визначає актуальність алгоритмів архівації графіки.

2. Другою особливістю є те, що людський зір під час аналізу зображень оперує контурами, загальним переходом кольорів та практично нечутливий до незначних змін у зображенні. Це дозволило створити спеціальні алгоритми стиснення, які орієнтовані лише на зображення.

3. У зображенні, як правило, сусідні точки як по горизонталі, так і по вертикалі близькі за кольором. Таким чином, при створенні алгоритму компресії графіки ми можемо використати особливості структури зображення.

Залежно від алгоритму, що застосовується для стиснення, існує велика кількість різноманітних форматів графічних файлів. Для ефективної роботи із графічним зображенням важливо зробити правильний вибір одного з численних графічних файлових форматів. Розмір графічного файлу істотно залежить від характеру зображення та вибраного формату.

Форматом файлу називають сукупність методів, правил подання й розміщення даних. Відповідно, формат графічних файлів – це набір методів, правил, призначених для подання, зберігання, обробки й розповсюдження зображень, наданих у цифровій формі.

7.1. Типи графічних форматів

Існує кілька різних типів графічних форматів, кожен з яких зберігає дані певним способом. Нині найбільш використовуваними є растровий, векторний і метафайловий формати. Існують, однак, й інші типи форматів – формати сцени, анімації, мультимедійні, гібридні, гіпертекстові, гіпермедійні, об'ємні, мова моделювання віртуальної реальності (VRML), аудіоформати, формати шрифтів, мова опису сторінки (PDL).

Розглянемо деякі з них.

Растрові формати

Растрові формати використовують для зберігання растрових даних. Файли цього типу особливо добре підходять для зберігання реальних зображень, наприклад фотографій. Растрові файли, по суті, містять точну попіксельну карту зображення. Програма візуалізації реконструює це зображення на поверхні відображення пристрою виведення.

Найпоширеніші растрові формати – це BMP, DIB, RLE, TIFF, PCX, PSD, GIF , JPEG, PNG, LWF, TGA, MrSID, RAW, IFF, CAM, CLP, CPT, CUR, DCM, DCX, IMG, FIF, KDC, LBM, KDC, PBM, PIC, PGM, RAS, RBS, PCD, PIC, РГХ, SUN, STING, XPM та ін.

Векторні формати

Файли векторного формату особливо корисні для зберігання лінійних елементів (ліній і багатокутників), а також елементів, які можна розкласти на прості геометричні об'єкти (наприклад, текст). Векторні файли містять не піксельні значення, а математичні описи елементів зображень. За математичними описами графічних форм (ліній, кривих, сплайнів) програма візуалізації будує зображення.

Векторні файли структурно простіші, ніж більшість растрових файлів, і зазвичай організовані у вигляді потоків даних.

Приклади найпоширеніших векторних форматів – AutoCAD DXF, AI, Microsoft SYLK, CDR (формат файлів векторного редактора CorelDRAW) та ін.

Метафайлові формати

Метафайли можуть зберігати як растрові, так і векторні дані. Найпростіші метафайли нагадують файли векторного формату; вони містять мову або синтаксис для визначення елементів векторних даних, але можуть включати і растрове подання зображення. Метафайли часто використовують для транспортування растрових і векторних даних між апаратними платформами, а також для переміщення зображень між програмними платформами.

Найпоширеніші метафайлові формати – CGM, PDF, EMF, WMF, графіка Excel, файли Adobe Table Editor, PLT- та HPGL-графіка, OLE-об'єкти, DXF-графіка, рисунки Lotus PIC та інші.

Формати сцени

Файли формату сцени (іноді їх називають файлами опису сцени) розроблено для зберігання стислого зображення (або сцени). Векторні файли містять описи частин зображення, а до файлів сцени належать інструкції, що дозволяють програмі візуалізації відновити зображення цілком. На практиці іноді важко визначити, чи йдеться про векторний формат чи про формат сцени.

Формати анімації

Формати анімації з'явилися порівняно нещодавно. Вони створюються за тим самим принципом, який ви використовували у своїх дитячих іграх із "рухомими" картинками. Якщо швидко відображати одне зображення за іншим, то створюється враження, що об'єкти цього зображення рухаються. Найпримітивніші з форматів анімації зберігають зображення в цілому, дозволяючи відображати їх просто в циклі одне за одним. Трохи більш ускладнені формати зберігають лише одне зображення і декілька колірних таблиць для цього зображення. Після завантаження нової колірної таблиці колір зображення змінюється і створюється ілюзія руху об'єктів. Ще складніші формати анімації зберігають лише різницю між двома послідовно відображуваними зображеннями (званими фреймами) і змінюють тільки ті пікселі, які змінюються при відображенні цього фрейму. Відображення зі швидкістю 10-15 кадрів за секунду типове для анімації мультиплікаційного ви-

ду. У відеоанімації для створення ілюзії плавного руху необхідно відображати 20 і більше фреймів за секунду. Прикладами форматів анімації можуть бути TDDD і TTDDD.

Мультимедійні формати

Мультимедійні формати відносно нові, але набувають усе більшого значення. Вони призначені для зберігання даних різних типів у одному файлі. Ці формати зазвичай дозволяють об'єднувати графічну, звукову і відеоінформацію. Прикладами можуть служити добре відомі формати RIFF фірми Microsoft, QuickTime фірми Apple, MPEG і FLI фірми Autodesk.

Гіпертекст і гіпермедіа

Гіпертекст – це система, що забезпечує нелінійний доступ до інформації. Більшість книг побудовано за лінійним принципом: вони мають початок, закінчення і певну схему розміщення тексту. Гіпертекст дозволяє створювати документи з одним або декількома початками, з одним і декількома закінченнями або взагалі без нього, а також із безліччю гіпертекстових зв'язків, які допомагають читачеві "перестрибувати" у будь-яке місце документа.

Гіпертекстові мови не є форматами графічних файлів, як GIF або DXF. Це, швидше, мови програмування на зразок PostScript або C. Вони спеціально призначені для послідовної передачі потоків даних, тобто потік гіпертекстової інформації можна декодувати мірою отримання даних. Щоб переглянути гіпертекстовий документ, не потрібно чекати його повного завантаження.

Термін гіпермедіа означає сплав гіпертексту і мультимедіа. Сучасні гіпертекстові мови і мережні протоколи підтримують найрізноманітніші засоби, включаючи текст і шрифти, нерухому і динамічну графіку, аудіо-, відео- і об'ємні дані. Гіпертекст забезпечує створення структури, яка дозволяє користувачеві комп'ютера організовувати мультимедійні дані, показувати їх та інтерактивно переміщатися ними.

Гіпертекстові й гіпермедійні системи, наприклад World Wide Web, зберігають великі інформаційні ресурси у вигляді файлів GIF, JPEG, PostScript, MPEG.

Тривимірні формати

У тривимірних файлах даних зберігається опис форми і кольору об'ємних моделей уявних і реальних об'єктів. Об'ємні моделі зазвичай конструюються з багатокутників і гладких поверхонь, об'єднаних з описами відповідних елементів: кольори, текстури, віддзеркалення тощо, за допомогою яких програма візуалізації реконструює об'єкт. Моделі перетворюються на сцени із джерелами світла й камерами, тому об'єкти у тривимірних файлах часто називають елементами сцени.

Програми візуалізації, в яких використовуються тривимірні дані, – це, як правило, програми моделювання й анімації (наприклад, Lightwave фірми NewTek і 3D Studio фірми Autodesk). Вони дозволяють коригувати зовнішній вигляд візуалізованого зображення, змінюючи і доповнюючи систему освітлення, текстуру елементів сцени та їх відносне розташування. Крім того, вони дають можливість користувачеві "оживляти" елементи сцени, тобто приписують їм рух. Після цього програма створює ряд растрових файлів (або кадрів). Якщо взяти їх по порядку,то вони збираються у фільм.

Формати аудіофайлів

Аудіоінформація зазвичай зберігається на магнітній стрічці у вигляді аналогових даних. Записам аудіоданих на такі носії, як компакт-диск (CD-ROM) і жорсткий диск, передує їх кодування за допомогою дискретизації, подібно до того, як це робиться під час запису цифрових відеоданих. Після кодування аудіодані можна записувати на диск як необроблений потік цифрових даних або, що зустрічається частіше, зберігати у форматі аудіофайлу.

Формати аудіофайлів за своєю концепцією ідентичні форматам графічних файлів, тільки зберігається в них інформація, яка призначена не для очей, а для вух. Більшість форматів містить простий заголовок, який описує збережені у файлі аудіодані. Найчастіше в заголовку вказано кількість відліків за секунду, кількість каналів та кількість бітів на відлік. Ця інформація в першому наближенні відповідає даним про кількість відліків

на піксель, кількість колірних площин і кількість бітів на відлік, що містяться в заголовках графічних файлів.

У форматах аудіофайлів застосовують різні методи стиснення даних. Для 8-бітових графічних і звукових аудіоданих зазвичай використовується кодування за алгоритмом *Девіда Хаффмана.* А ось для 16-бітових аудіоданих необхідні адаптовані спеціально для цих цілей алгоритми.

Формати шрифтів

Такі файли містять описи наборів буквенно-цифрових знаків і символів у компактному, зручному для доступу форматі. Із файлів шрифтів можна довільно вибирати дані, пов'язані з окремими знаками. У цьому сенсі вони являють собою бази даних про знаки і символи й тому іноді використовуються для зберігання графічних даних, хоч подібні дані за своєю природою не є буквенно-цифровими або символьними. Файли шрифтів можуть мати (а можуть і не мати) загальні заголовки, а деякі файли підтримують навіть підзаголовки для кожного знаку. У будь-якому випадку для того, щоб вибрати окремі знаки без читання й аналізу всього файлу, потрібно знати початок даних про знаки, обсяг даних по кожному знаку і порядок, у якому ці знаки зберігаються.

Деякі файли шрифтів підтримують стиснення, а багато – шифрування даних. Історично складено три основні типи файлів шрифтів: растрові, штрихові та сплайнові контурні.

Формати мов опису сторінки

Мови опису сторінки (PDL — Page Description Language) – це справжні машинні мови, які використовують для опису компонування, шрифтів і графіки друкованих та відображуваних сторінок. PDL призначені для передачі інформації на пристрої друку (наприклад, принтери) і на пристрої відображення (такі, як екрани графічного інтерфейсу користувача GUI (Graphical user interface – це різновид інтерфейсу користувача, в якому всі елементи (кнопки, меню, піктограми, списки),

представлені користувачу на дисплеї, виконано у вигляді картинок, графіки.). Особливість цих мов полягає в тому, що коди PDL залежать від апаратних засобів. Типовий файл PostScript містить детальну інформацію про пристрій виведення.

7.2. Елементи графічного файлу

Графічні файли складаються з послідовностей даних або структур даних, які називають файловими елементами чи елементами даних. Ці елементи належать до трьох категорій: поля, теги і потоки.

Поле – це структура даних у графічному файлі, що має фіксований розмір. Фіксоване поле може мати не лише фіксований розмір, а й фіксовану позицію у файлі. Для визначення місця розташування поля задають або абсолютний зсув від орієнтира у файлі, наприклад, від початку або кінця файлу, або відносне зміщення від будь-яких інших даних. Розмір поля можна зазначати у специфікації формату або визначати за іншою інформацією.

Теги – це структура даних, розмір і позиція якої змінюються від файлу до файлу. Аналогічно полю, позиція тегів задається або абсолютним зміщенням щодо відомого орієнтира у файлі, або відносним зміщенням від іншого файлового елемента. Теги можуть містити в собі інші теги або набори пов'язаних полів.

Поля і теги спроектовано таким чином, щоб допомогти програмі одержати швидкий доступ до потрібних даних. Якщо позиція у файлі відома, то програма може отримати доступ до неї безпосередньо, без попереднього читання проміжних даних.

Файл, в якому дані організовано у вигляді *потоку,* не дає таких можливостей, і має читатися послідовно. Ми припускаємо, що потік дозволяє підтримувати блоки даних змінної довжини, які є елементами потоку і повинні враховуватися програмою при читанні файлу. Хоч початок і кінець потоку можуть бути відомі й задані, місце розташування блоків даних (за винятком першого) невідоме і визначається у процесі читання.

Файли, що складаються з елементів одного типу, – велика рідкість. Найчастіше застосовують комбінації двох і більше елементів даних. Наприклад, формати TIFF і TGA використовують і теги, і фіксовані поля, а файли формату GIF – фіксовані поля й потоки.

Організація даних у вигляді фіксованих полів дозволяє читати їх значно швидше, ніж дані, організовані у вигляді тегів або потоків. Однак файли, що містять в основному фіксовані поля, менш зручні при додаванні або видаленні даних. Формати, що складаються з фіксованих полів, складно розширити. Однак файли, що містять в основному потокові дані, не підтримують довільний доступ і, отже, не можуть використовуватися для швидкого пошуку.

7.3. Растрові формати

Растрові файли, відрізняючись один від одного деталями, мають загальну структуру. Растрові файли містять заголовок, растрові дані та іншу інформацію, у тому числі і про колірну палітру. Через незрозумілу причину тривають розробки програм, що використовують так звані неструктуровані формати. Такі файли містять лише дані зображення, не залишаючи нам жодних шансів розібратися в їх структурі. Проте, як організовані дані в цих файлах, мають уявлення тільки їх розробник і програма візуалізації, яка їх використовує. Основними компонентами простого растрового файлу є:

Заголовок
Растрові дані

Якщо файл не містить даних зображення, то має подаватися лише заголовок. Додаткову інформацію, яка не поміщається в заголовку, розміщують у кінцівці файлу:

Заголовок
Растрові дані
Кінцівка

Якщо застосовується палітра, то її можна зберегти в заголовку файлу, але зручніше розмістити її в середині файлу, після заголовка:

Заголовок
Палітра
Растрові дані
Кінцівка

Таблиці рядків розгортки і таблиці кольорової корекції можна розташувати після заголовка як перед даними зображення, так і після них:

Заголовок
Палітра
Таблиця рядків розгортки
Таблиця колірної корекції
Растрові дані
Таблиця колірної корекції
Кінцівка

Заголовок – це розділ двійкових або символьних (у форматі ASCII) даних. Тут ASCII (*American standard code for information interchange*) – назва таблиці (кодування, набору), в якій деяким розповсюдженим друкованим і недрукованим символам поставлено у відповідність числові коди. Зазвичай заголовок розташовується на початку файлу і зберігає загальну інформацію про растрові дані, які в цьому файлі містяться. Усі структуровані растрові файли мають заголовки, структура і вміст яких визначаються конкретним форматом. Звичайно заголовок растрового

файлу складається з фіксованих полів. Жодне із цих полів не є обов'язковим, але певний набір полів типовий для більшості популярних нині форматів. Нижче наведено інформацію, що зазвичай міститься в заголовку:

Ідентифікатор файлу
Версія файлу
Кількість рядків у зображенні
Кількість пікселів у рядку
Кількість бітів у пікселі
Кількість колірних площин
Тип стиснення
x-координата початку зображення
y-координата початку зображення
Текстовий опис
Простір, що не використовується

Звичайно заголовок починається з певного унікального значення, що називається *ідентифікатором формату файлу,* файловий ID або ID-значення. Ідентифікатор дозволяє програмі визначити формат графічного файлу, з яким вона працює.

Як ідентифікатор може використовуватися послідовність символів ASCII (наприклад, BM або GIF) або дво-, або чотирибайтове слово (наприклад, 4242h або 596aa695h), або довільна послідовність даних, зрозуміла лише розробнику формату. Передбачено, що ідентифікатор має бути унікальним навіть *для форматів,* що використовуються на різних платформах, але, як ви побачите далі, цієї умови далеко не завжди дотримуються. Як правило, якщо значення, прочитане з певного місця у файлі, збігається з очікуваним ідентифікаційним значенням, то програма, що читає заголовок файлу, припускає, що їй відомий даний формат.

Після ідентифікатора в заголовку файлу зазвичай розташовується поле *версії файлу.* Версії одного і того самого формату можуть мати різні характеристики (розмір заголовка, підтримувані дані тощо). Тому після ідентифікації формату за допомогою ID-значення програма зазвичай перевіряє номер версії, щоб визначити, чи зможе вона обробити дані, що містяться в цьому файлі.

Після полів ідентифікатора і версії файлу розташовано декілька полів, що описують саме зображення. Як правило, растри фізично або логічно організовані в рядки пікселів. Поле *кількість рядків* у растровому зображенні також називають довжиною зображення, висотою зображення або кількістю рядків розгортки, воно містить значення, що визначає кількість рядків у реальних растрових даних. Кількість пікселів у рядку, що також називається шириною зображення або шириною рядка розгортки, визначає кількість пікселів, збережених у кожному рядку.

Кількість бітів на піксель визначає розмір даних, необхідних для опису кожного пікселя в кольоровій площині. Іноді в цьому полі насправді зберігається інформація про кількість байтів на піксель. Це поле характеризує піксельну глибину.

Якщо з метою зменшення обсягу файлу формат підтримує який-небудь вид кодування, то в заголовок слід внести поле *тип стиснення* (тип компресії). Деякі формати підтримують декілька алгоритмів компресії, включаючи необроблені і не стиснені дані. Часто модифікації форматів – це головним чином доповнені або змінені схеми стиснення. Проблема стиснення даних постійно привертає увагу розробників, тому досить часто з'являються нові, більш досконалі алгоритми компресії.

Поля *x-координата початку зображення* й *y-координата початку зображення* визначають координати точки початку зображення на пристрої виведення. Найчастіше вони мають значення (0, 0), що дозволяє поєднувати початок зображення з точкою відліку системи координат пристрою. Якщо ж застосовано інші координати, то при візуалізації зображення почне відтворюватися з іншої точки.

Іноді в заголовок включають поле *текстового опису растра.* Це поле являє собою коментар, який містить довільні символьні (у форматі ASCII) дані, наприклад назву зображення, ім'я файлу, ім'я автора зображення та/або ім'я програми, що використовується для його створення. Щоб забезпечити можливість переносити інформацію заголовка на інші платформи, це поле містить тільки 7-бітові дані у форматі ASCII.

У кінці заголовка можуть розташовуватися *невикористовувані поля,* які іноді називають заповнювачами зарезервованого простору або зарезервованими полями. Зарезервовані поля не містять даних, не описуються і не структуруються. Їхні розміри і місце розташування в заголовку відомі. Якщо виникне необхідність розширити формат, то відомості про нові дані заносяться в зарезервований простір. Таким чином зберігається сумісність із програмами, що підтримують старі версії цього формату, а проблеми, пов'язані з появою нових версій, зводяться до мінімуму. Заголовок формату має фіксовану довжину і структуру. У процесі модифікації формату його структура ускладнюється, при цьому нові поля додаються в зарезервовані області заголовка без зміни його розмірів. Часто заголовок спеціально розширюється зарезервованими полями до загальної довжини 128, 256 або 512 байтів. Це пов'язано з необхідністю враховувати розміри буферів уведення/виведення для підвищення продуктивності системи.

Растрові дані зазвичай займають більшу частину растрового файлу. У більшості форматів растрових файлів растрові дані розташовуються безпосередньо після заголовка, але можуть розміщуватися і в будь-якому іншому місці файлу. Разом із ними можуть зберігатися палітра й інші дані. У такому випадку в заголовку в полі зсуву даних зображення (або в документації) вказується місце розташування початку даних зображення у файлі.

Структура растрових даних у більшості форматів досить проста. Растрові дані складаються з піксельних значень. На пристрої виведення пікселі зазвичай виводяться у вигляді рядків

розгортки по всій ширині поверхні відображення, і цей факт, як правило, визначає порядок розташування даних у файлі. Тому інформація про те, на який пристрій виведення орієнтувався розробник формату, можливо, допоможе вам "вирахувати" точний порядок розташування даних.

Рядки розгортки об'єднують піксельні дані у двовимірну сітку, що дозволяє нам розглядати розташування кожного пікселя растра в заданих логічних координатах. Растр можна подати й у вигляді послідовності значень, які логічно відображають у файлі растрові дані, що відповідають картинці, на поверхні відображення пристрою виведення. Реальні растрові дані зазвичай складають більшу частину будь-якого растрового файлу.

Кінцівка, що іноді називається хвостом, являє собою структуру даних, яка часто доповнює основний заголовок, але розташовується в кінці файлу. Зазвичай кінцівку додають у тих випадках, коли файловий формат модифікувався (у нього внесено нові типи даних), а розширити або змінити структуру заголовка неможливо. Як правило, кінцівка додається для того, щоб зберегти сумісність формату з його попередніми версіями. Назва цього елемента структури однозначно вказує на те, що він розташовується після даних зображення змінної довжини. Отже, кінцівка ніколи не має сталого зміщення від початку файлу (виключаючи зображення незмінного розміру). Тому зміщення кінцівки, як правило, задається щодо кінця файлу.

Зсув кінцівки можна зазначити в інформації заголовка за умови, що в ньому є вільний простір. Кінцівка, як і заголовок, може містити поле ідентифікатора або спеціальне число, використовуване програмою візуалізації для того, щоб відрізнити її від інших структурних елементів растрового файлу.

Крім заголовків, кінцівок і палітр, растрові файли можуть містити додаткові структури даних, які використовуються програмою візуалізації при різних маніпуляціях із даними зображення.

Растрові формати відрізняються один від одного такими властивостями: колірними моделями, методами ущільнення, максимальним розміром зображення, який вони можуть забезпечи-

ти, шарами різних типів, наявністю Alpha-каналу (складова прозорості), можливістю здійснювати анімацію, наявністю черезрядкового розгорнення тощо.

Формат BMP

Формат BMP (скорочення від Bit Map Picture) – це один з перших растрових форматів. Формат відрізняє дуже великий обсяг файлу, оскільки дані записуються по кожному пікселю окремо. Цей формат є надзвичайно простою структурою і служить для опису та візуалізації невеликих зображень-піктограм (icons), широко застосовується у графічних інтерфейсах Windows, а також використовується в мультимедійних презентаціях. Існує кілька різновидів цього формату. Найвідоміший варіант – із розширенням *. bmp.

Формат Bitmap32

Це порівняно новий формат, створений на базі формату BMP, від якого відрізняється тим, що дані про одну точку зберігаються не у 24, а у 32 бітах. Додаткові 8 бітів використовують для Alpha-каналу. Формат поки що не одержав поширення, але після появи Windows XP, де Alpha-канал узаконено на рівні ядра системи, формат отримав добрі перспективи на майбутнє.

Отже, цей формат зручний насамперед через збереження додаткової складової прозорості, яка зберігається всередині файлу з текстурою.

Формат PCX

Формат PCX запропонований компанією Z-Soft у програмі Paintbrush, він може використовуватися на платформі Macintosh, хоча був написаний для PC. Цей формат застосовувався багатьма компаніями, що спеціалізуються в області програмного забезпечення. Він зручний для зберігання зображень типу ділової графіки (креслення, діаграми, схеми тощо). Підтримуються колірні формати 1, 4, 8 та 24 біти на піксель. До недоліків PCX

належать непристосованість до запису фотографій, а також наявність численних версій.

У форматі PCX використано один із варіантів алгоритму ущільнення RLE (від англійського Run Lenght Encoding – групове кодування). RLE – один із найдавніших і найпростіших алгоритмів ущільнення графіки, що базується на такій ідеї: якщо в деяких растрах трапляються ланцюжки з однакових пікселів, то у файлі замість цих ланцюжків зберігають пари чисел – лічильник повторень та саме значення. Чим довші ланцюжки, тим більше ущільнення.

Формат TGA

Для підтримки своїх відеокарт фірмою Truevision розроблено формат Target Image File (TGA). Кілька компаній час від часу перекупили права на цей формат одна в одної. Із файлами TGA працювали адаптери Targa, True Vista та ін. Формат дозволяє зберігати зображення із глибиною кольору до 32 бітів і при цьому швидко читається й розпаковується. Має спеціальну опцію "Bottom-up orientation", тобто завантаження файлу не "зверху вниз", а "знизу нагору".

У TGA використовується алгоритм ущільнення RLE, який відрізняється від варіанта алгоритму, що використовується у форматі PCX. Інший варіант алгоритму RLE має більший максимальний ступінь компресії для деяких растрів і не так сильно збільшує розмір вихідного файлу в найгіршому випадку.

Формат GIF

Формат GIF (Graphics Interchange Format) – популярний растровий формат, запропонований як незалежний від апаратного забезпечення засіб обміну растровими зображеннями в мережі Internet.

Основна перевага цього формату – високий ступінь стискування без особливих втрат, що досягається застосуванням алгоритму ущільнення, який належить до класу LZW-алгоритмів. В алгоритмах класу LZW використовують словниковий метод ущільнення. Створюють словник, що містить повторювані по-

слідовності символів (фрази), які зустрічаються в масиві, що кодується. Кожна фраза отримує код (індекс) у словнику. Кодування масиву символів виконується заміною фраз відповідними індексами зі словника.

У файлах формату GIF близько розміщені однакові за кольором точки групуються в горизонтальні лінії. Це дозволяє істотно зменшити об'єм графічного файлу. GIF-формат ефективно стискує графічні малюнки з великими фрагментами однорідної заливки, але погано стискує фотографії, оскільки фотографії містять багато відтінків. Обмеження GIF полягає ще і в тому, що кольорові зображення не можна записати в режимі більше ніж 256 кольорів, однак у багатьох випадках цього достатньо, наприклад для передачі графічних зображень в Internet.

Ще одна корисна якість цього формату – підтримка анімаційних зображень. Файл GIF може містити не одне, а декілька растрових зображень, які браузери можуть довантажувати одне за одним із зазначеною у файлі частотою. Так досягають ілюзії руху (GIF-анімація). Формат GIF добре підходить для розробки невеликих і досить простих анімаційних фрагментів. GIF-файли не потребують значного місця для їхнього зберігання. Щоб створити анімацію, слід спочатку створити кожний окремий файл, наприклад, за допомогою пакетів Adobe або Corel. Потім необхідно компілювати ряд окремих фреймів у єдиний GIF-файл.

Формат PNG

Формат PNG – це новий графічний стандарт, створений для заміни формату GIF. Він розроблений спеціальним комітетом, очолюваним *Томасом Бутеллем*. Абревіатура PNG (вимовляється як "пінг") означає Portable Network Graphics. Як видно з назви, цей формат призначений спеціально для передачі зображень мережами. Затверджено стандарт PNG і надруковано у 1996 р.

Цей формат не захищений патентами, не вимагає ліцензування й фінансових відрахувань і тому має широко розповсюджуватися – саме через патентування й жорстке ліцензування алгоритму LZW і виник PNG.

Спільні риси форматів GIF та PNG:

- використання методів компресії без утрат;
- підтримка індексованих кольорів до 8 бітів на піксель;
- маска прозорості (Alpha-канал);
- окрім зображення, файл може містити також і текст.

Відмінності форматів PNG та GIF:

- більша максимальна глибина кольору – до 48 бітів на піксель для зображень типу TrueColor, а для градацій сірого – до 16 бітів на піксель;
- повний Alpha-канал до 16 бітів на піксель;
- запис у файл гама-корекції;
- ефективне розпізнавання пошкоджень даних;
- у файл PNG базового формату записують тільки одне зображення – немає підтримки анімації, як у GIF.

Формат JPEG або JPG

Розвиток технічної бази машинної графіки й засобів мультимедіа викликали швидке зростання розмірів графічних файлів. Для пошуку кращого способу запису великих обсягів графічної інформації при Міжнародному комітеті зі стандартизації (ISO) організовано дослідницьку групу Joint Photographix Experts Group (JPEG). Робота групи завершилася пропозиціями зі створення файлового формату з високим ступенем ущільнення даних на основі алгоритму JPEG.

Офіційно ім'я "JPEG" указує на алгоритм, метод ущільнення, а не на формат файлу. Для уніфікації файлових форматів, що використовують JPEG, розповсюджено рекомендації, які названо як JFIF (JPEG File Interchnge Format). Отже, можливо, більшість файлів JPEG правильніше називати JFIF. Крім того, метод ущільнення JPEG використовується в інших форматах файлів, наприклад, TIFF, Kodak Photo CD, Quick Time тощо.

JPEG-формат застосовують для відображення фотографій та інших тонових зображень в електронних мережах. Він викорис-

товує ефективні алгоритми ущільнення, що сприяє значному скороченню обсягу файлу (економить від 50 до 70 % обсягу пам'яті), однак дає втрату інформації. У форматі JPG можна одержати файл у 500 разів менший за розміром, ніж у BMP. Це найменші за обсягом графічні файли. Реалізовано цілу групу алгоритмів стискування, зокрема алгоритм стискування, що збільшує розміри пікселів зображення, тобто утворює блоки з 8 × 8 пікселів і для кожного блока формує набір чисел. Перші кілька чисел представляють колір блока, а наступні числа відображають різницю між пікселями. Так зменшується розмір графічного файлу, але при цьому губиться інформація, яка майже не відчувається оком. Оскільки під час стискування втрачається частина інформації про колір, то в JPG-форматі не бажано зберігати зображення, для яких важливі всі особливості передачі кольорів. JPEG краще стискує растрові фотографічні зображення, ніж логотипи чи схеми. Із меншими втратами стискуються зображення з високою роздільною здатністю (200-300 dpi). Більшість зображень в Internet подано форматом JPG.

Отже можна сказати, що метод JPEG відкрив шлях масовому розповсюдженню й використанню ефективних технологій роботи з повнокольоровими зображеннями. Це пояснюється не тільки позитивними рисами запропонованого алгоритму і споживчими якостями відповідних технологій, але й відкритістю стандарту, відсутністю патентних обмежень та підтримкою для розробників програм.

Формат TIFF

Формат TIFF (Tagged Image File Format) запропоновано фірмою Aldus Corporation і розроблено для зберігання сканованих зображень із високою роздільною здатністю. Висока роздільна здатність обумовлює великий об'єм файлів, що ускладнює їхню обробку. Основна ідея формату TIFF – підтримка швидкого доступу до окремих фрагментів зображення. Це дозволяє редагувати зображення окремими частинами.

Позитивною рисою формату TIFF є його гнучкість. Він може зберігати декілька зображень. У TIFF використовують різноманітні колірні моделі. Цей формат підтримує багато методів ущільнення – LZW, Deflate, JPEG та інші.

Отже, формат TIFF можна вважати стандартом обміну графічними даними. Однак насиченість можливостей обумовлює проблеми для розробників програм – важко врахувати всі можливості. Трапляється, що файл, записаний однією програмою, не читається іншими. Для вирішення цієї проблеми у стандарті TIFF версії 6.0 визначено підмножину Baseline TIFF, яку повинні підтримувати всі програми.

Формат DjVu

DjVu (вимовляється "дежавю") – це комплект технологій ущільнення, формату файлу та програмної платформи, що розроблявся з 1996 р. у фірмі AT&T Labs спеціально з метою розміщення в Інтернеті сканованих документів (книг, журналів, документації, зображень із високою роздільністю тощо). Це відносно новий формат, що використовує хвильовий (wavelet) алгоритм ущільнення. Кінцевий результат ущільнення порівнянний за своєю якістю з оригінальною сканованою версією. При багаторазовому його збільшенні не з'являється растрова мозаїка.

DjVu зберігає зображення, використовуючи три шари:

- перший шар містить маску (бітовий масив), що вказує, яка точка зображення відповідає передньому плану (текст, малюнки, підписи) і яка – фону (текстура паперу, фотографії). Цей шар кодується алгоритмом без втрат;
- другий шар містить інформацію про колір фону, використовуючи кодування, засноване на хвильовому ущільненні;
- третій рівень містить інформацію про передній план.

Одна з основних технологій DjVu – здатність відокремити фон зображення й передній план. Традиційні методи ущільнення зображення придатні для простих фотографій, але вони значно погіршують різкі переходи кольору між суміжними контрас-

тними областями. Відокремлюючи текст від фону, DiVu може зберегти текст із високою роздільною здатністю (у такий спосіб зберігаючи гострі грані й максимізуючи чіткість), водночас ущільнюючи фон із нижчою роздільною здатністю (використовують методику на основі хвильового алгоритму). Розмір документів настільки малий, що вони можуть відправлятися електронною поштою як вкладення.

Формат LWF

Група розробників, яка займається обробкою супутникових фотографій, створила у компанії LuraTech формат LWF (Lura Tech Wavelet format). У форматі LWF використано алгоритм ущільнення з утратами (на основі хвильового алгоритму), який часто дає кращі за JPEG результати. Формат LWF відрізняється тим, що при ущільненні можна заздалегідь установити розмір майбутнього файлу.

Програма Lura Wave Smart Compress v2.2 (Shareware) може забезпечити високий ступінь ущільнення при досить високій якості. При ущільненні, наприклад у 60 разів, утрати менші, ніж при такому самому ущільненні у JPEG. Зображення практично не втрачає деталізації, а набуває дещо розмитого вигляду із підкресленими контурами, тобто якість сприйняття майже не погіршується. При збільшенні коефіцієнта ущільнення не проступає "мозаїка" квадратів, як у JPEG. Практично з'являється можливість інтерактивної роботи із сервером. Програма існує для всіх. Цей формат використовують рідко, хоча LWF іноді ущільнює майже в 1,5 рази більше за інші формати.

Формат PSD

Формат PSD – це власний формат програми Adobe Photoshop, один із найпотужніших форматів збереження растрової графічної інформації. Підтримує 48-розрядне кодування кольору, різні колірні моделі. Проте відсутність ефективного алгоритму стискування приводить до великого обсягу файлів.

7.4. Векторні формати

Растровий файл містить точну попіксельну карту зображення, яке відтворюється програмою візуалізації на поверхні відображення пристрою виведення. Програми візуалізації рідко беруть до уваги будь-які структурні елементи растрових форматів, крім точок, рядків розгортки, смуг і фрагментів – частин зображення, створених без урахування його змісту.

Векторні файли, навпаки, містять математичні описи всіх елементів зображення, що використовуються програмою візуалізації для конструювання кінцевого зображення. Таким чином, можна сказати, що векторні файли будують не з піксельних значень, а з описів елементів зображення, або об'єктів.

Вектор – це відрізок прямої, заданий початковою точкою, напрямком і довжиною. Однак визначення вектора може бути складнішим і містити дані про тип лінії, кривої або сплайну. Прямі та криві лінії застосовують для побудови геометричних фігур, таких як кола та прямокутники, які, у свою чергу, можуть використовуватися для створення більш складних об'ємних фігур – сфер, кубів і багатогранників. Векторні формати відрізняються один від одного більшою мірою, ніж растрові, тому що кожен із них проектувався для конкретних цілей. Якщо концептуально формати, що підтримують 1-бітові та 24-бітові растрові дані, відрізняються незначно, то розходження між векторними форматами, які використовуються програмами САПР, та форматами, які використовуються для обміну загальними даними, будуть досить істотними. Отже, узагальнити векторні формати тим самим способом, що й растрові, – завдання непросте.

З іншого боку, більшість пристроїв виведення підтримують сітку з пікселів, кожен з яких адресується окремо, так, начебто поверхня відображення являла б собою папір у клітинку. Завдяки цьому програма завжди може знайти спосіб намалювати елементи зображення у векторному форматі.

Незважаючи на те, що векторні файли значно відрізняються один від одного, більшість із них, подібно до растрових, має

певну базову структуру: заголовок, розділ даних і маркер кінця файлу. Така структура дозволяє коректно зберігати векторні дані та інтерпретувати їх при візуалізації. Загальна інформація, що описує структуру файлу, звичайно міститься в заголовку, хоча іноді для цих цілей можна використовувати і кінцівку.

Структура векторних файлів, як правило, простіша, ніж у більшості растрових; часто вони організовані у вигляді потоків даних. Велика частина інформації, що міститься у файлі, – це дані про зображення.

Формат DXF

Формат DXF (Drawing Exchange Format) розроблено фірмою AutoDesk у 1982 р. для обміну кресленнями та іншими графічними документами в середовищі AutoCAD. Незважаючи на вік цього формату та його недоліки, DXF зараз підтримується багатьма програмами.

Головним недоліком формату DXF можна вважати великий обсяг файлів. У середовищі системи AutoCAD для роботи з документами використовують компактніший формат – DWG, однак він є внутрішнім форматом, його не "розуміють" інші програми.

Формат MIF-MID

Тут ми дещо відхилимося від тематики "класичної комп'ютерної графіки" й розглянемо приклад векторного формату, який використовують у геоінформаційних системах (ГІС). Ці системи описують просторові об'єкти сукупністю метричних та атрибутивних (семантичних) даних. Формат MIF-MID є найпопулярнішим векторним форматом обміну даними для ГІС. Він розроблений фірмою MapInfo для власної ГІС, однак зараз використовується майже усіма ГІС як формат експорту-імпорту. Опис просторових об'єктів у цьому форматі складається із двох файлів – *.MIF та *.MID. Файл із розширенням MIF містить загальний опис та координати вузлових точок об'єктів. Об'єкти можуть бути точковими, лінійними або площинними. Графічні примітиви: Arc, Ellipse, Line, Pline, Point, Rect, Region, Roundrect та Text.

Формат CDR

Формат CDR використовується програмою Corel Draw, яка нині є однією з найпопулярніших серед програм, що дозволяють створювати векторні зображення. CDR дозволяє записувати векторну й растрову графіку, а також текст. У формату CDR є незаперечна перевага – можливість читати ушкоджені файли. Програма, знайшовши ушкоджений об'єкт, просто пропускає його. Одна з властивостей векторних форматів – відтворення масштабованих зображень без погіршення роздільної здатності під час подальшого друку або виведення на інші пристрої. Файли Corel Draw мають робоче поле до 45×45 метрів.

Формат AI

Формат AI – це власний векторний формат програми Adobe Illustrator. Цей формат підтримують практично всі програми векторної графіки.

7.5. Метафайли й інші формати

Під час сумісної роботи декількох програм часто виникає потреба в обміні різними графічними даними, як растровими, так і векторними. Для підтримки обміну цими даними використовують спеціальні графічні формати – метафайли. Метафайли можуть зберігати інформацію про растрові та/або векторні зображення і про команди візуалізації. Як приклади метафайлів можна назвати файли CGM, EPS, PICT, WMF/EMF, графіка Excel, файли Adobe Table Editor, PLT- і HPGL-графіка, OLE-об'єкти тощо. Серед прикладів використання метафайлів в Інтернеті в першу чергу назвемо формат PDF, який запропонований та активно просувається фірмою Adobe.

Зазначимо, що важко відокремлювати метафайли та векторні формати. Вважаємо головною ознакою метафайлу те, що графічний опис об'єктів складається з *команд* для певного графічного пристрою (принтера, драйвера GDI тощо). Тут GDI (Graphics Device Interface) – інтерфейс Microsoft Windows для представлення графічних об'єктів та передачі їх на пристрої відображен-

ня – монітори і принтери. Можна сказати, що, мабуть, будь-який векторний формат – це метафайл. Наприклад, формат DXF містить команди (хоча команди не для пристрою, а для програми, що інтерпретує ці команди). Крім того, оскільки векторний формат містить опис зображення як список графічних примітивів, то важливою є послідовність накладання цих примітивів – знову можна сказати, що векторний формат – це не список примітивів, а послідовність команд. Будь-яка спроба догматично класифікувати реальні формати, які зазвичай містять багато різноманітних можливостей, вважається сумнівною. Для класифікації можна скористатися й таким принципом – якщо розробник формату називає це метафайлом, то напевне так воно й є.

Формат WMF

Формат WMF (Windows Metafile) розроблено компанією Microsoft для 16-розрядних версій Windows. Метафайл являє собою послідовність команд GDI з усіх основних категорій графічних примітивів 16-розрядного інтерфейсу Windows GDI. У цей формат конвертують векторні зображення при перенесенні з програми у програму через clipboard (буфер обміну).

Формат WMF має велику сумісність для PC, його розуміють і деякі програми для Macintosh. Можливості метафайлу Windows значною мірою обмежені через залежність від пристроїв і програм. Метафайл не має інформації про розмір зображення, початкову роздільність чи стан палітри пристрою, на якому записувалось це зображення. Існують й інші обмеження, що накладаються GDI.

Формат EMF

У 32-розрядних версіях Windows, починаючи з Windows NT 3/5.1, компанія Microsoft використовує 32-розрядний формат метафайлів, що називається розширеним форматом метафайлів (Enhanced Metafile, EMF). Порівняно з WMF, формат EMF підтримує 32-розрядну систему координат та нові 32-розрядні функції GDI, містить заголовок із геометричними даними та палітрою і навіть забезпечує деяку підтримку OpenGL. Формат EMF,

що порівняно з форматом WMF менше залежить від пристрою і підтримує нові функції GDI, стає все популярнішим.

Формат PICT

На платформі Macintosh аналогічну формату WMF роль відіграє формат PICT. Формат PICT розроблено компанією Apple Computer у 1984 р. спочатку для програми MacDraw. Багато програм для PC також розуміють цей формат. PICT-файли в об'єктно-орієнтованому форматі складаються з окремих графічних об'єктів (ліній, дуг, овалів чи прямокутників), кожний із яких можна незалежно редагувати й масштабувати. У вказаних PICT-файлах можуть також зберігатися растрові зображення. Файли PICT кодуються командами QuickDraw. Файли PICT в основному використовують для обміну графікою між програмами на Macintosh, вони також підтримуються на платформі Windows за допомогою QuickTime for Windows.

Формат PostScript

PostScript – мова опису сторінок (мова керування лазерними принтерами) фірми Adobe. PostScript створено у 80-х рр. для реалізації принципу WYSIWYG (What You See is What You Get). Файли цього формату мають розширення .PS чи .PRN. Вони утворюються за допомогою функції PrintoFile графічних програм при використанні драйвера PostScript-принтера, тобто є по суті програмою з командами на виконання для вивідного пристрою.

Файли формату PostScript містять у собі сам документ (те, що ви бачите на сторінках), усі зв'язані файли (растрові й векторні), використані шрифти, а також іншу інформацію для пристрою: плати кольороподілу, додаткової плати, форму растрової точки для кожної плати тощо. Дані в PostScript-файлі, як правило, записують у двійковому кодуванні (Binery), яке приблизно вдвічі економніше за кодування ASCII. Проте під час передачі файлів через мережі, при крос-платформному обміні, під час друку через послідовні кабелі використовується й ASCII. Це викликано можливістю спотворення двійкового кодування у старих комп'ютерах.

Отже, якщо файл створений правильно, не має значення, на якій платформі він виконувався і які шрифти були використані. Найкоректніші PS-файли створюють програми Adobe. Сказане відноситься до форматів, заснованих мовою PostScript, а саме: EPS і PDF.

3D формати

VRML (мова моделювання віртуальної реальності – Virtual Reality Modeling Language) – графічний формат, що призначений для опису тривимірних зображень та обміну ними в мережі World Wide Web.

Мова VRML, яку розроблено GavinBell, RickCarey, MarkPesce і TonyParisi, стала першою мовою тривимірного моделювання для Web. У 1995 р. створено групу VAG (VRML Authoring Group) і з'явилася остаточна редакція специфікації VRML 1.0. У 1997 р. технологію підтримали у своїх браузерах як Microsoft, так і Netscape. ISO схвалила другу версією специфікації як міжнародний стандарт VRML 97 (або ISO/IEC 14772-1).

VRML-файл має розширення WRL. Він використовує формат ASCII і являє собою звичайний текстовий файл із списком об'єктів, які названі вузлами (nodes). До вузлів VRML 2.0, зокрема, належать 3D-геометрія, властивості світла, що створюється за допомогою VRML, файли зображень формату JPEG, відеофайли формату MPEG, звукові файли формату MIDI, текстові документи формату HTML.

Основні ідеї мови VRML активно використовують у сучасних засобах візуалізації тривимірних сцен.

Формат 3DS

Це один із найпоширеніших форматів для 3D-графіки. Файли формату 3DS були стандартними файлами програми 3D Studio, ще коли вона працювала під DOS. У 3D Studio MAX з'явився інший формат збереження – MAX, але для розробки ігор цей новий формат виявився незручним. Натомість формат 3DS виявився придатним для цієї мети: крім самих тривимірних моделей (які являють собою каркасні сітки), він зберігає їхнє поло-

ження у світових координатах, координати текстур, кольори вершин, ключові кадри анімації, дані про властивості матеріалів і навіть атмосферні ефекти. Це практично готовий формат для збереження моделей і цілих карт (тільки скриптові команди доводиться зберігати окремо). Підкреслимо, що під час збереження 3DS-файлів можна вказати, щоб координати текстур зберігалися разом із моделлю. Після цієї операції накладення текстури відбувається якісніше. Формат 3DS є зручним і практичним для будь-яких видів ігрових моделей. Він широко використовується для обміну даними між системами тривимірного моделювання.

Формати MDL, MD2 і MD3

Ці формати призначені для збереження анімаційних моделей, особливо анімації людей. Усі три формати відкриті, ними може скористатися будь-хто. У форматах MDL, MD2 і MD3 зберігаються моделі й дані про немов би "кістякову" анімацію. Анімації моделей зберігаються в цих форматах у тривимірних кадрах, причому кожному кадру відповідає своя повноцінна модель. Розмір файлів при цьому зростає пропорційно кількості кадрів анімації й може виявитися завеликим для серйозних ігор, утім, формати MDL, MD2 і MD3 можна досить сильно ущільнити.

Формати мультимедіа

Через особливості відеоінформації у цифровому відеозаписі ущільнення без утрат само по собі майже не застосовується. Використання методів ущільнення, подібних до методів, що застосовуються в архіваторах типу WinZIP, дозволяє зменшити розмір файлу не більше, ніж на 2/3, хоча, звісно, при цьому якість зображення не погіршується.

Ущільнення з утратою якості є основним методом зменшення розміру відеофайлів. Такі алгоритми дозволяють визначити ту частину інформації, яку глядач, найімовірніше, не помітить під час перегляду фільму, і видалити її з файлу. Основними форматами цифрового відео, в яких використовують ущільнення з утратами, на сьогоднішній день є Apple QuickTime, AVI, IntelIndeo, MJPED, MPEG-1, MPEG-2, і MPEG-4.

Формат AVI

AVI (Audio Video Interleaved – чергування аудіо й відео) – формат файлів, уведений фірмою Microsoft для використання систем роботи з відеозображеннями в середовищі Windows. У цьому форматі сектори відеоданих чергуються із секторами звукових даних таким чином, щоб відеоплеєр міг підтримувати мінімальну буферизацію даних.

AVI є спеціальним випадком формату RIFF (Resource Interchange File Format). RIFF – універсальний формат для обміну даними мультимедіа. За структурою файл AVI-формату являє собою варіант файлу формату RIFF. Файл цього формату складається із блоків (chunks), що у свою чергу можуть містити інші вкладені блоки. Якщо найбільш "поверхневий" блок містить ідентифікатор формату "avi", то це означає, що ми працюємо з *.avi-файлом.

Запис у форматі AVI може здійснюватися без або з ущільненням. Зазвичай використовується Motion JPEG. Також підтримуються формати компресії: Microsoft RLE (тільки 8-бітний колір), Microsoft Video 1 (лише 8- і 16-бітні кольори), Indeo, Cinepak Editable MPEG (використовує тільки I-кадри). Останнім часом набуває популярності формат компресії за алгоритмом Div.

Дані у форматі AVI можна експортувати в різні формати.

Незважаючи на значне поширення форматів MPEG, формат AVI є популярним для аудіо- відеоданих на PC. Значний відсоток існуючих систем захоплення кадрів і нелінійного монтажу використовують формат AVI.

У зв'язку з великою кількістю обмежень базового стандарту AVI, консорціумом Open Digital Media розроблено розширення формату AVI – Open DML AVI, з урахуванням особливостей, необхідних для професійного виробництва відео. Це розширення включає підтримку полів (не тільки кадрів), має часовий код, розміри його файлів можуть бути більшими за 1 Гб. Розширення також містить багато інших особливостей. Microsoft внесено підтримку Open DML AVI у DirectShow 5.1. Це розширення також використовується в різних професійних програмах для виробництва відео на PC, зокрема Digi Suite(Matrox).

Формат MPEG

Стандарт MPEG визначає ряд різних форматів. Концепція ущільнення відеозображень у форматі MPEG така:

1) визначити, яка саме інформація в потоці повторюється хоча б у плині якогось відрізка часу, та вжити заходів щодо запобігання дублюванню цієї інформації;

2) використати особливості людського зору при виборі відповідних методів ущільнення та їхньої точності.

Відомо, що пікова зорова роздільна здатність досягається в ділянці сітківки, так званій центральній ямці, що охоплює всього лише два градуси кута зору. Мірою віддалення від центральної ямки щільність фоторецепторів знижується майже за експонентним законом. Це означає, що для глядача, який дивиться у центр кіноекрана, роздільність у периферичних зонах зображення майже цілком утрачає значення (із другого боку, периферійний зір характеризується високою сприйнятливістю до коливань яскравості). Крім того, здатність людини розрізняти дрібні колірні елементи в зображенні, що рухається, значно програє його здатності розрізняти зміни яскравості – тому зміни кольорів можуть кодуватися значно грубіше, ніж зміни яскравості.

Пояснимо, як може реалізуватися концепція ущільнення відеозображень на прикладі такої сцени: переміщення автомобіля на фоні лісу. Фон при цьому можна вважати незмінним, оскільки основна увага глядача зазвичай звернена на рухомий предмет на передньому плані, навіть якщо є якісь дрібні зміни – глядач навряд чи зверне на них увагу, чи встигне їх помітити.

Отже, можна розбити кадр на дві складові частини – фон, що зберігається один раз, а потім підставляється при відтворенні всіх кадрів, і область, де рухається автомобіль, – її записують окремо для кожного кадру. Переміщення автомобіля може задаватися двома параметрами – вектором руху й кутом повороту навколо своєї осі. Завдяки реалізації цієї концепції MPEG-компресор ущільнює відеопотік у десятки й навіть сотні разів без значної утрати якості зображення.

Зауважимо, що однозначно оцінити якість кодування якимись приладами неможливо. Головним критерієм є те, як людина

сприйматиме ущільнену інформацію. Тому правила ущільнення відеоданих при MPEG-кодуванні здійснюється на основі моделі сприйняття людиною відеозображень (HVS – HumanVisual Sense).

Надмірність зображення згідно з HVS поділяють на три основні види:

1) невидимі людським оком ділянки зображення. Це, наприклад, місця гасіння по вертикалі й горизонталі. Видалення цієї інформації ніяк не позначається на зображенні;

2) статистично надмірні ділянки зображення. Розрізняють просторову (ділянки зображення, на яких суміжні пікселі практично однакові) і часову (незмінювані в часі ділянки зображення) надмірність;

3) надмірні за кольором і яскравістю ділянки зображення. Враховується обмежена чутливість людини до невеликих змін яскравості й особливо кольору фрагментів зображення.

Редагувати MPEG-відео дуже важко через неможливість точної до кадру нарізки фрагментів.

Формат MPEG-4

Популярним нині форматом є MPEG-4, який служить для передачі всіх видів мультимедійних даних: графіки, відео, анімації, звуку тощо. Основна його особливість – гнучкість, що дозволяє працювати в цьому форматі MPEG-4 навіть малопотужним клієнтам (таким, як кишенькові ПК).

Ще раз зазначимо, що MPEG-4 є стандартом, формат файлів MPEG-4 має назву MP4, його розробка ґрунтувалася на алгоритмі Apple QuickTime. Ряд структур цього алгоритму зустрічається й у MPEG-4. Це, наприклад, треки для задання варіанта передачі потокових медіаданих мережею, причому в MPEG-4 для одного файлу можуть бути задані декілька варіантів передачі в різні програмні засоби. На відміну від QuickTime, MPEG-4 забезпечує стійке відтворення всіх типів потокових медіаданих, тобто не тільки аудіо- й відеоданих, але й змішаного контенту (контент – цифрове подання медіаданих).

Стандарт MPEG-4 задає принципи роботи з контентом для трьох областей: інтерактивного мультимедіа (зокрема, це опти-

чні диски й інформація, розповсюджувана через мережу), графічних додатків (синтетичного контенту) і цифрового телебачення – DTV; фактично цей формат задає правила організації середовища, причому середовища об'єктно-орієнтованого. Він працює не просто з потоками й масивами медіаданих, а з медіаоб'єктами (ключове поняття стандарту).

Як медіаоб'єкти можуть виступати тексти, графіка, синтезовані мовні фрагменти, тривимірні моделі (у тому числі анімація обличчя), потокові аудіо та відео тощо. Ці медіаоб'єкти допускають відображення на екрані зв'язаної з ними у сценаріях медіаінформації будь-якого виду. MPEG-4 представляє користувачам гнучкі засоби роботи з мультимедійним контентом. Формат дозволяє виконувати прив'язку взаємного розташування аудіо- і відеоданих, природних і синтезованих комп'ютером 2D І 3D-об'єктів, їх взаємну синхронізацію, а також указувати їх інтерактивну взаємодію з користувачем.

Наявність у форматі MPEG-4 протоколу для підтримки фронтальної анімації у медіафайлах дозволяє анімувати в реальному часі тривимірні моделі обличчя, їх сполучення з аудіошаблонами чи мовним відтворенням тексту, отриманим за допомогою синтезатора, робить можливим абсолютно синхронне озвучування. Проте зазначимо, що в MPEG-4 пропонується тільки протокол для керування тривимірними моделями.

Значне нововведення при компресії відеоданих у стандарті MPEG-4 таке. Якщо попередні формати поділяли зображення на прямокутники, то кодер MPEG-4 при обробці зображень оперує об'єктами довільної форми. Наприклад, людина, що рухається кімнатою, буде сприйнята як окремий об'єкт, що переміщується щодо нерухомого об'єкта – тильного плану.

Завдяки підтримці альфа-каналів, закладеній у кодер MPEG-4, можна в реальному часі накладати відео на тильний план. Такий прийом є корисним, наприклад, при сегментації (виділенні на базовому зображенні елементів переднього й тильного планів). Сегментація значно підвищує якість зображення при заданій швидкості передачі. Наприклад, у випадку сюжету, в якому людина періодично з'являється на якомусь фоні, звичайний кодер зображення

тильного плану повинен передавати декілька разів. А якщо створити кодер відповідно до MPEG-4, то можна сегментувати і зберегти в пам'яті кодера інформацію про зображення тильного плану, тоді повторне пересилання даних не потрібне.

Щодо використання MPEG-4 в Інтернеті зазначимо таке: відбувається поліпшення якості передачі відеоданих порівняно з MPEG-1 при швидкостях обміну від 20 кбайт/с до 1000 кбайт/с. Крім того, MPEG-4 забезпечує повну підтримку контенту з черезрядковим розгорненням і роздільністю до 4096× 4096, а також швидкість передачі даних у діапазоні від 5 кбайт/с до 10 Мбайт/с (версія 1).

На відміну від інших форматів, для яких необхідна повна підтримка всіх умов, MPEG-4 доступний у різних варіантах. Більшість інструментів, створених з урахуванням MPEG-4, підтримують тільки функції версії 1.

Підсистеми стандарту розбито на кілька профілів. Кожен профіль підтримує ряд функціональних пакетів, орієнтованих лише на визначену область застосування. Для забезпечення сумісності у другій версії подано не тільки додаткові профілі, але і збережено всі старі.

У першу версію MPEG-4 включено дев'ять відео- і чотири аудіопрофілі, а у другу додано ще сім відео- і чотири аудіопрофілі. Пропоновані профілі охоплюють увесь спектр функцій відтворення: від простого відео з єдиним аудіопотоком (Simple Visual Profile) до повного розгортання медіафайлів, які містять різноманітні елементи, що забезпечується розширеним кодеком другої версії (Advanced Coding Profile). Стандартним для настільних комп'ютерів є визнаний Main Visual Profile.

Подальший розвиток форматів подання відеоінформації можна пов'язати з використанням психофізіологічних моделей візуального сприйняття, які досліджуються нині не тільки в наукових установах, але й у компаніях-розробниках відеоустаткування. Ці дослідження засновано на ідеї про те, що ущільнене відеозображення повинно не стільки апроксимувати первинний

матеріал, скільки найбільш адекватним способом збуджувати специфічні нервові вузли зорового тракту.

Ефективність алгоритмів відеокомпресії можна підвищити також за рахунок подальшого дослідження й урахування особливостей людського зору, зокрема, нерівномірної щільності розподілу фоторецепторів тощо.

7.6. Алгоритми стиснення даних

Графічна інформація, як і будь-які дані, що зберігаються або передаються по каналах зв'язку, має значну надмірність. Використання алгоритмів стиснення інформації дозволяє підвищити в кілька разів швидкість обміну по каналах зв'язку та значно зекономити обсяги пам'яті.

Нині існує багато підходів до стиснення даних і алгоритмів, які їх реалізують.

Стиснення – це процес зменшення фізичного розміру блока даних. Існує декілька способів стиснення. Розрізняють фізичне і логічне стиснення, симетричне й асиметричне, стиснення з утратами і без утрат.

Наведемо найпоширеніші методи (або алгоритми) стиснення.

- Упакування пікселів фактично не є методом стиснення даних, але дозволяє ефективно записувати їх у послідовно розташовані байти пам'яті. Цей метод застосовують у форматі Macintosh PICT та інших, що дають можливість записувати декілька одно-, дво- або чотирибітових пікселів у один байт пам'яті або дискового простору.
- Групове кодування (RLE) є алгоритмом стиснення, що застосовується в таких растрових форматах, як BMP, TIFF і PCX для зменшення обсягу надмірних графічних даних.
- Алгоритм Lempel-Ziv-Welch (LZW) використовують у форматах GIF і TIFF, а також застосовують як стандарт стиснення даних для деяких модемів.

- Кодування CCITT – форма стиснення даних, застосовувана для факсимільної передачі та стандартизована Міжнародним консультативним комітетом з телеграфії і телефонії (CCITT). Стандарт базується на схемі ключового стиснення, запропонованій Хаффманом, і є широковідомим як кодування за алгоритмом Хаффмана.
- Алгоритм, який розроблено об'єднаною експертною групою із фотографії (JPEG), – це набір методів стиснення, що використовуються в основному для обробки зображень із плавним переходом тону і для мультимедіа. Базова реалізація JPEG застосовує схему кодування за алгоритмом дискретних косинусних перетворень (DCT).
- Алгоритм, розроблений об'єднаною експертною групою із дворівневим зображенням (JBIG), – метод стиснення даних дворівневих (двоколірних) зображень, який покликаний замінити алгоритми стиснення MR (Modified READ) і MMR (Modified Modified READ), використовувані CCITT Group 3 і Group 4.
- ART – патентований алгоритм стиснення, розроблений фірмою Johnson-Grace, який у майбутньому можна буде адаптувати для підтримки аудіо, анімації і повномасштабного відео.
- Фрактальне стиснення – математичний процес, використовуваний для кодування растрів, що містять реальне зображення, у сукупність математичних даних, які описують фрактальні (тобто схожі, повторювані) властивості зображення.

Зазначимо, що один і той самий алгоритм часто можна реалізувати різними способами. Багато відомих алгоритмів, таких як RLE, LZW або JPEG, мають десятки різних реалізацій.

Крім того, алгоритми іноді мають кілька явних параметрів, варіюючи які, можна змінювати характеристики процесів архівації та розархівації. У разі конкретної реалізації ці параметри фіксують, враховуючи найімовірніші характеристики вхідних зображень, вимоги щодо економії пам'яті, вимоги до часу архівації тощо.

КОНТРОЛЬНІ ЗАПИТАННЯ ДО РОЗДІЛУ 7

1. Які типи графічних форматів існують?
2. Які елементи має графічний файл?
3. Яка загальна структура растрових форматів?
4. Наведіть приклади растрових форматів.
5. Дайте загальну характеристику векторних форматів та наведіть приклади цих форматів.
6. Опишіть метафайли.
7. Які алгоритми стиснення даних ви знаєте?

СПИСОК ЛІТЕРАТУРИ

Основна:

1. *Порев В. Н.* Компьютерная графика. – СПб : БХВ-Петербург, 2002. – 432 с.: ил.

2. *Шикин Е. В.* Начала компьютерной графики / Е. В. Шикин, А. В. Боресков, А. А. Зайцев. – М. : ДИАЛОГ–МИФИ, 1993. – 138 с.

3. *Шикин Е. В.* Компьютерная графика. Полигональные модели / Е. В. Шикин, А. В. Боресков – М. : ДИАЛОГ–МИФИ, 2001. – 461 с.

4. *Пічугін М.Ф.* Комп'ютерна графіка: навч. посіб. / М. Ф. Пічугін, І. О. Канкін, В. В. Воротніков. – К. : Центр учбової літератури, 2013. – 346 с.

5. *Аммерал Л.* Принципы программирования в машинной графике / Л. Аммерал. – М. : Сол Систем, 1992. – 224 с. : ил.

6. *Роджерс Д.* Алгоритмические основы машинной графики / Д. Роджерс ; пер. с англ. – М. : Мир, 1989. – 512 с.

7. *Артюхин В. В.* Методическое пособие по дисциплине "Компьютерная графика": математическое обеспечение и администрирование информационных систем / В. В. Артюхин. – М. : Моск. гос. ун-т экономики, статистики и информатики. – 2001. – 56 с.

8. *Вельтмандер П. В.* Машинная графика : учеб. пособие. В 3 кн. Книга 2. Основные алгоритмы.– Новосибирск : Новосибирский ун-т, 1997. – 193 с. : ил.

Додаткова:

9. Adobe Illustrator : учебник.– К. : Диасофт, 2002. – 368 с.

10. *Залогова Л. А.* Компьютерная графика. Элективный курс: Практикум / Л. А. Залогова. – М. : Бином. Лаборатория знаний, 2005. – 245 с. : ил.

11. *Тихомиров Ю. В.* OpenGL. Программирование трехмерной графики / Ю. В. Тихомиров. – 2 изд. – СПб. : Питер, 2002. – 1070 с.

ЗМІСТ

Навчальне видання

ТМЄНОВА Наталія Пилипівна

КОМП'ЮТЕРНА ГРАФІКА

Навчально-методичний посібник

Редактор *Л. Магда*

Оригінал-макет виготовлено ВПЦ "Київський університет"

Формат 60x84$^{1/16}$. Ум. друк. арк. 6,5. Наклад 100. Зам. № 217-8427. Вид. № І т4.
Гарнітура Times New Roman. Папір офсетний. Друк офсетний.
Підписано до друку 25.10.17

Видавець і виготовлювач
ВПЦ"Київський університет",
б-р Т. Шевченка, 14, м. Київ, 01601
☎ (38044) 239 32 22; (38044) 239 31 72; тел./факс (38044) 239 31 28
e-mail: vpc_div.chief@univ.net.ua; redaktor@univ.net.ua
http: vpc.univ.kiev.ua
Свідоцтво суб'єкта видавничої справи ДК № 1103 від 31.10.02