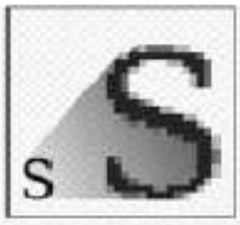
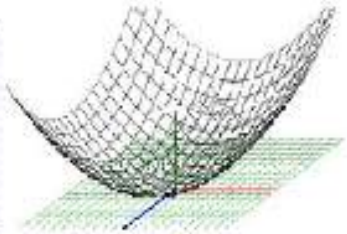
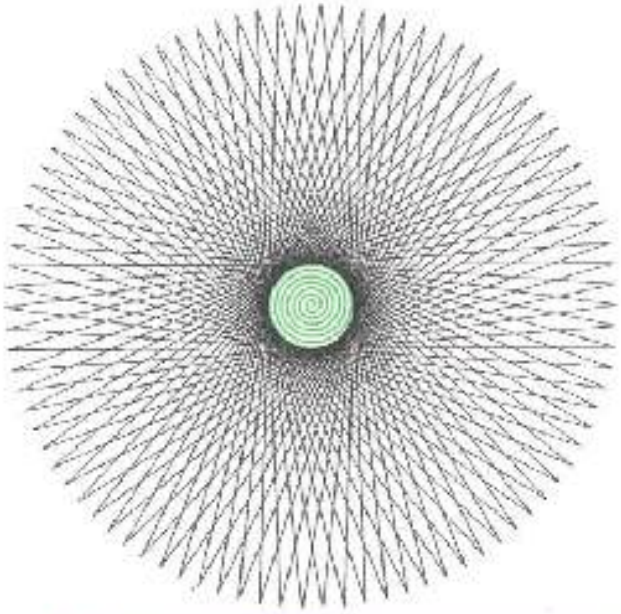
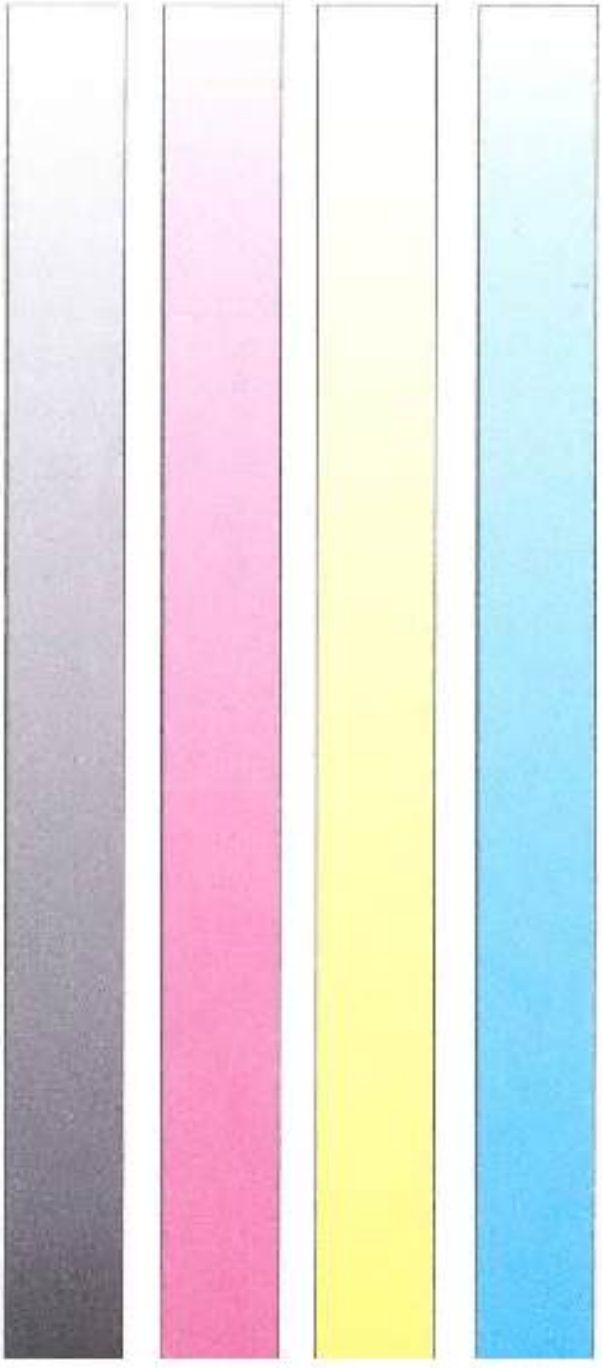


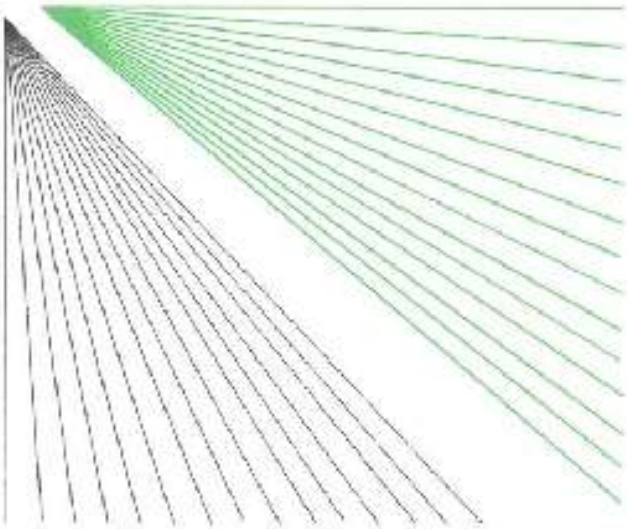


Федік Л. Ю.
Решетило О. М.
Смолянкін О. О.

Комп'ютерна графіка







Л. Ю. ФЕДІК, О. М. РЕШЕТИЛО, О. О. СМОЛЯНКІН

КОМП’ЮТЕРНА ГРАФІКА

Луцьк 2012

УДК 004.92(075)

ББК 32.973:85.15я73
Ф 32

Рецензенти: В.С. Михайленко – доктор технічних наук, професор КНУ будівництва і архітектури
В.П. Юрчук – доктор технічних наук, професор НТУУ «КПІ»
П.Д. Стухляк – доктор технічних наук, професор НТУ ім. Івана Пулюя

Федік Л.Ю., Решетило О.М., Смолянкін О.О.

Ф32 Комп'ютерна графіка: Навчальний посібник. – Луцьк: ЛНТУ, 2012. – 240 с.: іл.

Висвітлено основні види комп'ютерної графіки, колірних моделей і зображень. Розглянуті основні алгоритми для побудови фігур, а також алгоритми їх заповнення і приклади реалізації.

Подано контрольні завдання з описом середовищ програмування, прикладами складання програм, контрольними питаннями.

Для студентів вищих навчальних закладів.

УДК 004.92(075)
ББК 32.973:5.15я73

ЗМІСТ

РОЗДІЛ 1. ОСНОВНІ ВІДОМОСТІ ПРО КОМП'ЮТЕРНУ ГРАФІКУ

1.1. Історія розвитку комп'ютерної графіки

Комп'ютерна графіка – *це застосування обчислювальної техніки для створення графічних зображень, їх відображення різними засобами і маніпулювання ними.*

Отже, комп'ютерним (цифровим) може бути назване зображення, створене за допомогою комп'ютерної програми.

Спочатку програмісти навчились отримувати рисунки в режимі символьної печаті. На паперових листах за допомогою символів (зірочок, точок, хрестиків, букв і ін.) отримували рисунки, які нагадували мозаїку. Так друкувались графіки функцій, (рис. 1.1), зображення течій рідин і газів, електричних і магнітних полів і т.д. За допомогою символьної печаті програмісти отримувати навіть художні зображення. Потім з'явилися спеціальні пристосування для графічного виведення на папері – графопобудовувачі (плотери). За допомогою якого на лист паперу чорнильним пером наносяться графічні зображення: графіки, діаграми, технічні креслення і інше.

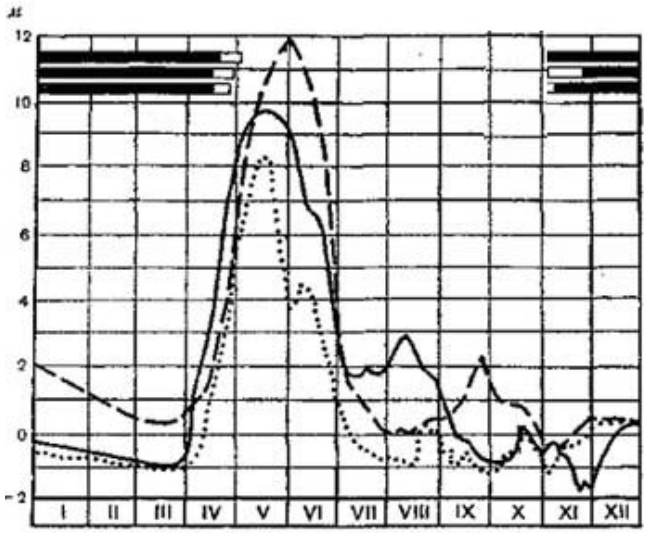


Рис. 1.1. Графіки функцій

Проте, справжня революція в комп'ютерній графіці відбулася з появою графічних дисплеїв. На екрані графічного дисплею (рис. 1.2.) стало можливим отримувати рисунки, креслення в такому вигляді, як на папері за допомогою олівців, фарб, креслярських інструментів.

Зв'язок традиційної і комп'ютерної графіки, з однієї сторони, визначає застосування розмножувальної техніки, з іншої – можна знайти ще одне пояснення

виникненню терміна «графіка», яке застосовується для роботи художника-комп’ютерника. Слово графіка означає зображення лініями, штрихами, точками.

Рис. 1.2. Екран графічного дисплея

Всі графічні комп’ютерні програми принципово розділяються на два типи: векторні (зображення будуються лініями) і растрові (зображення плямою із точок). Отже, яким би складним не видавалось зображення, створене в комп’ютері, за своєю сутністю, будь-яке з них відноситься до графіки.

1.2. Види комп’ютерної графіки

Для роботи з комп’ютерною графікою існує багато класів програмного забезпечення, проте розрізняють всього три види комп’ютерної графіки:

1) растрова,

2) векторна,

3) фрактальна.

Ці види відрізняються принципами формування зображення під час відображення на екрані монітора чи під час друкування на папері.

Правомірна й інша класифікація комп’ютерної графіки:

1) двовимірна,

2) тривимірна.

Двовимірна (2D) графіка *– це зображення, яке має два вимірювання, тобто яке лежить на площині* (рис. 1.3). *Цей вид графіки є основою комп’ютерної графіки, в тому числі і тривимірної.*

Тривимірна (3D) графіка *– це побудова на комп’ютері, за допомогою спеціальних програм, просторової моделі, яка складається з простих і складних*

геометричних форм, присвоєння цій моделі фактури (надання форми та оздоблення поверхні), кольору, ступені прозорості і матовості, надання їй і умовній камері руху у віртуальному (можливому) просторі, розміщення в цьому просторі джерел світла і, нарешті, прорахунок вибудованої сцени (рис. 1.3).

Цей вид комп'ютерної графіки застосовується під час створення комп'ютерних ігор, реклам і т.д.

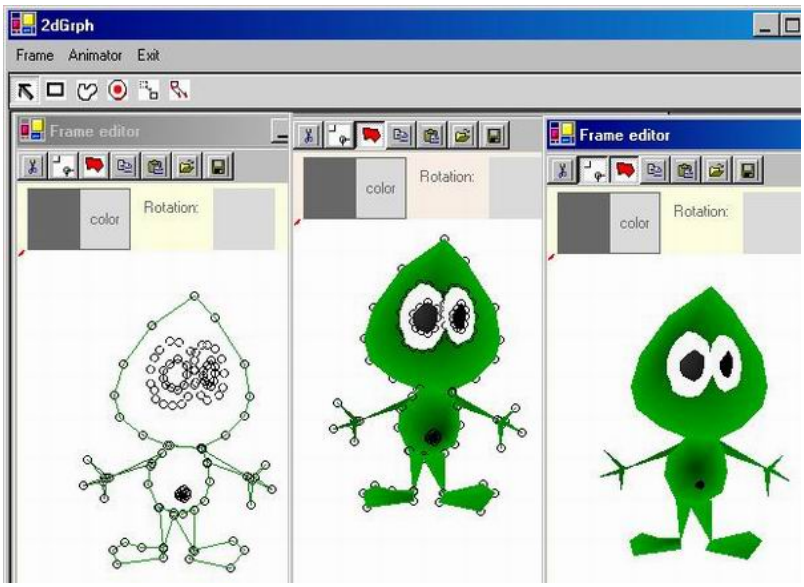


Рис.1.3. Двовимірна графіка

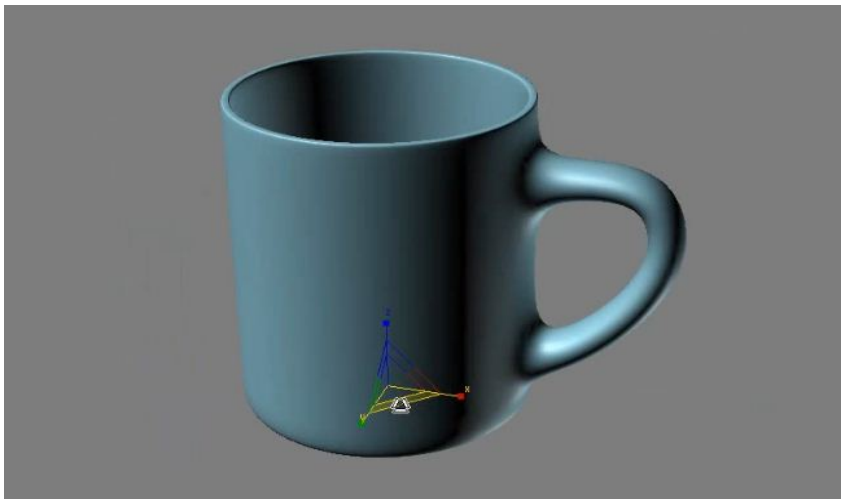


Рис.1.4. 3D графіка

1.3. Галузі застосування комп'ютерної графіки

Сучасне застосування комп'ютерної графіки дуже різноманітне. Основними галузями застосування комп'ютерної графіки є:

1) наукова,

2) ділова,
3) конструкторська,
4) поліграфічна,
5) Web-дизайн,
6) мультимедіа.

Напрям наукової графіки з'явився найпершим. Його призначенням є візуалізація (наочне зображення) об'єктів наукових досліджень, графічна обробка результатів розрахунку, проведення обчислювальних експериментів із наочним представленням їх результатів (рис. 1.5).

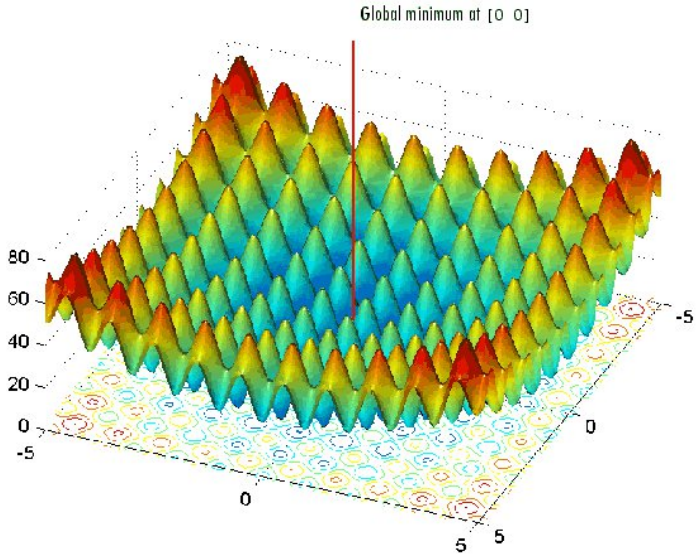


Рис. 1.5. Наукова графіка

Ділова графіка *– це галузь комп'ютерної графіки, призначена для створення ілюстрацій, які часто застосовуються у роботі різних установ (рис. 1.6). Планові показники, звітна документація, статичні зведення – ось об'єкти, для яких за допомогою ділової графіки створюються ілюстровані матеріали. Частіше всього це графіки, колові і стовпчикові діаграми.*

Конструкторська графіка *– застосовується в роботі інженерів-конструкторів, є обов'язковим елементом систем автоматизації проектування (САПР). Графіка в САПР застосовується для підготовки технічних креслень проектуючого пристрою. Графіка в поєднанні з розрахунками дозволяє проводити в наочній формі пошук оптимальної конструкції, найбільш вдалого компонування деталей, прогнозувати наслідки, до яких можуть привести зміни в конструкції. Засобами конструкторської графіки можна отримувати плоскі зображення (проекції, січення) і просторові, трьохмірні зображення (рис. 1.7).*

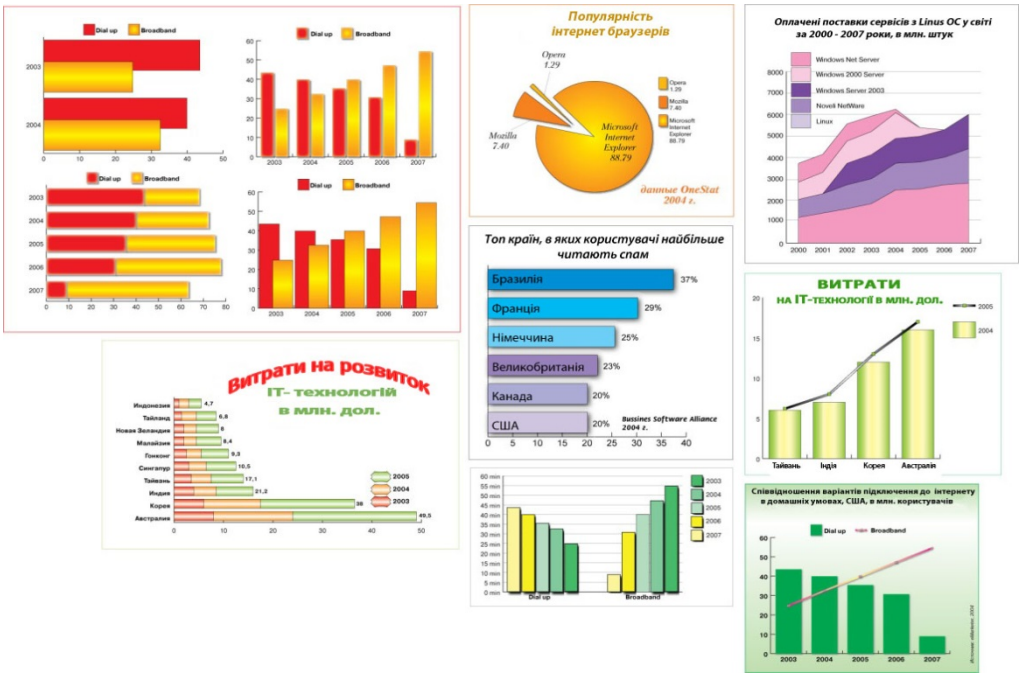


Рис. 1.6. Ділова графіка

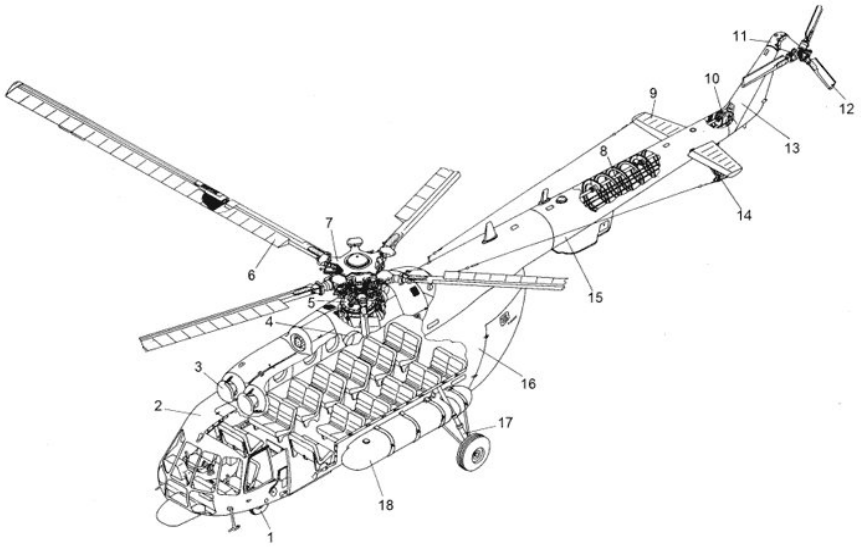


Рис. 1.7. Конструкторська графіка

Поліграфія *– це сукупність технічних засобів для множинного репродукування (відтворення, відновлення) текстового матеріалу і графічних зображень. Спеціаліст, який працює у цій галузі, повинен не тільки знати програми верстки і графічні редактори але й розбиратися в додрукарській підготовці видання (рис. 1.8, 1.9).*

Рис. 1.8. Плотер

Рис 1.9. Тиражувальна машина

Web-дизайн *– це оформлення web-сторінки. Ця галузь комп'ютерної графіки має таке саме значення для сайту, як і поліграфічний дизайн і верстка для паперового видання. Часто під web-дизайном розуміють не тільки створення графічних елементів для сайту, але й проектування його структури, навігації, тобто створення сайту повністю (рис. 1.10).*

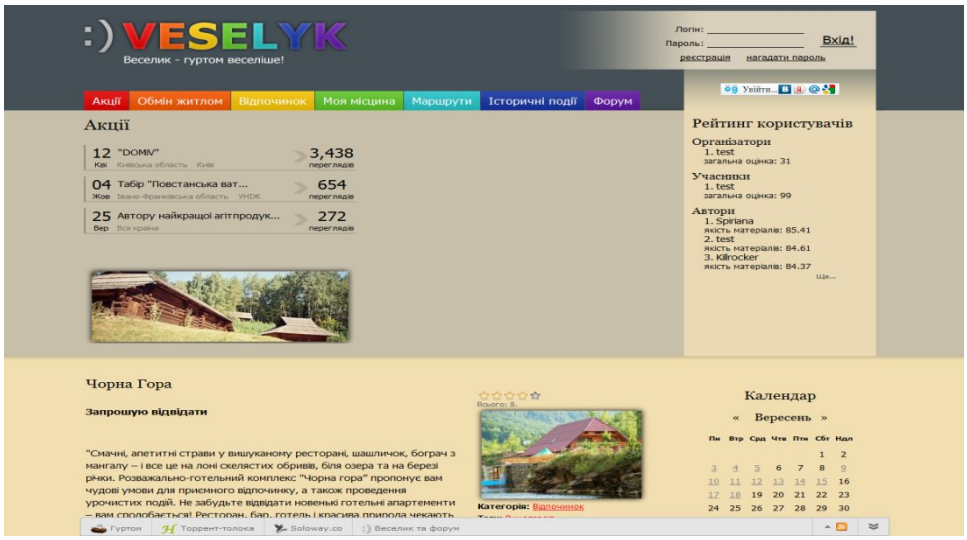


Рис 1.10. Web-дизайн

Мультимедіа *– це галузь комп'ютерної графіки, пов'язана зі створенням інтерактивних додатків, які дають можливість активно впливати на вміст і направленість енциклопедій, довідкових систем, навчальних програм і інтерфейсів до них (системи уніфікованих зв'язків (за видом передаючої інформації, параметрів сигналів, апаратури), призначених для обміну інформацією між пристроями обчислювальної системи, наприклад, між пристроями вводу даних і запам'ятовуючими пристроями (рис. 1.11).*

Рис 1.11. Мультимедіа

Питання для самоперевірки

1. Дайте визначення комп'ютерної графіки.
2. Які існують види комп'ютерної графіки?
3. Дайте визначення двовимірної і тривимірної комп'ютерної графіки.
4. Вкажіть галузі застосування комп'ютерної графіки
5. Який вид комп'ютерної графіки з'явився найпершим?
6. Що таке Web-дизайн?
7. Що таке мультимедіа?

Список рекомендованої літератури

1. Аппаратные средства IBM: Энциклопедия / Сост. М. Гук – СПб.: Питер, 2000. – 816 с.
2. Блінова Т.О., Порєв В.М. Комп'ютерна графіка / За ред. В.М. Горєва. – К.: Видавництво «Юніор», 2004.
3. Васильев В.Е. Компьютерная графика: Учебное пособие. – Санкт-Петербург, 2004. – 96 с.
4. Веселовська Г.В., Ходаков В.Є., Веселовський В.М. Комп'ютерна графіка: Навч. посібник для студентів вищих навчальних закладів / Під ред. В.Є. Ходакові. – Херсон: ОЛДІ-плюс, 2004. – С.112– 133.

5. Л.Ю.Федік, Л.О.Гуменюк Застосування комп’ютерної графіки та її класифікація // Наукові нотатки. Міжвузівський збірник (за напрямом «Інженерна механіка»), Випуск 20 (травень, 2007). – Луцьк, 2007. – С. 519– 521

6. Компьютерная графика: Учебник для вузов / Под ред. Петров М.Н., Молочков В.П. – СПб.: Питер, 2003. – 736 с.

7. Кукарин А.Б., Саган С.А., Шестаков В.Л. Термины по автоматизации производственных процессов. – К.: Высшая школа, 1992. – 156 с.

8. Основы компьютерной графики. Часть 1: Математический аппарат компьютерной графики (электронная версия) А.В. Казанцев. – Казань, 2001. – 62 с.

9. Пономаренко С.И. Пиксел и вектор. Принципы цифровой графики.– СПб.: БХВ-Петербург, 2002. – 496 с.

10. Сучасний словник іншомовних слів: Близько 20 тис. слів і словосполучень / Уклали: О.І. Скопненко, Т.В. Цимбалюк. – К.: Довіра, 2006. – 789 с. – (Словники України).

11. Шушан Р., Райт Д., Льюис Л. Дизайн и компьютер – М.: Издательский отдел «Русская Редакция», ТОО «Channel Trading Ltd», 1997. – 544 с.

РОЗДІЛ 2. РАСТРОВА (ПІКСЕЛЬНА) ГРАФІКА

2.1. Основні положення растрової графіки

Растрову графіку застосовують під час розробки електронних і поліграфічних видань (рис. 2.1). Ілюстрації, виконані засобами растрової графіки, рідко створюють вручну чи за допомогою комп'ютерних програм. Частіше всього застосовують відскановані ілюстрації, підготовлені художником на папері, чи фотографії. Останнім часом для введення растрових зображень у комп'ютер знайшли широке застосування цифрові фото- і відеокамери. Відповідно, більшість растрових графічних редакторів орієнтовані не стільки на створення зображень, скільки на їх обробку (рис. 2.2).

Рис. 2.1. Растрова графіка

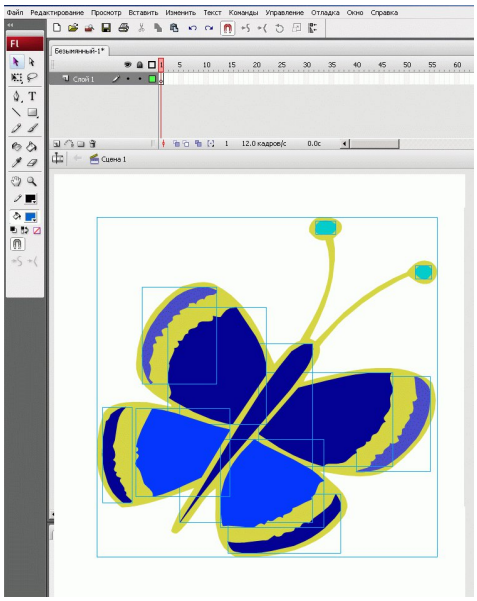


Рис. 2.2. Обробка растрової графіки

Растрова візуалізація грунтується на представленні зображення на екрані або папері у вигляді сукупності окремих точок (пікселів).

Піксель (pixel) *– це найменший елемент зображення, відтворений комп'ютером.*

Слово піксель походить від скорочення picture element (елемент зображення) з заміною букви С на Х.

Відмінними особливостями пікселя є його: однорідність (всі пікселі за розміром однакові) і неподільність (всередині пікселя не може бути ніяких більш дрібних елементів).

Якщо пікселі досить малі, то око сприймає «піксельну мозаїку» як одне ціле зображення.

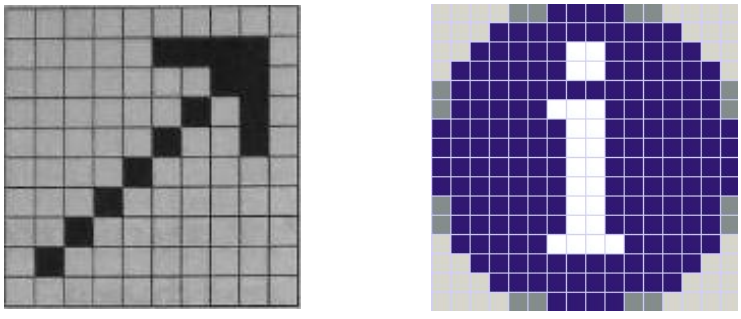
Разом пікселі утворюють растр.

Растр *– це прямокутна сітка пікселів, формуючих зображення на екрані монітора* (рис. 2.3).

Рис. 2.3. Приклади растру

Сама сітка отримала назву растрової карти (bitmap).

Кожна точка растру характеризується двома параметрами: своїм положенням на екрані і своїм кольором, якщо монітор кольоровий, чи ступенем яскравості, якщо чорно-білий (рис. 2.4).



а) б)

Рис. 2.4. Точка растру для моніторів: а) чорно-білого, б) кольорового

У залежності від розташування пікселів у просторі розрізняють такі основні типи растру:

1. квадратний, (рис. 2.5)

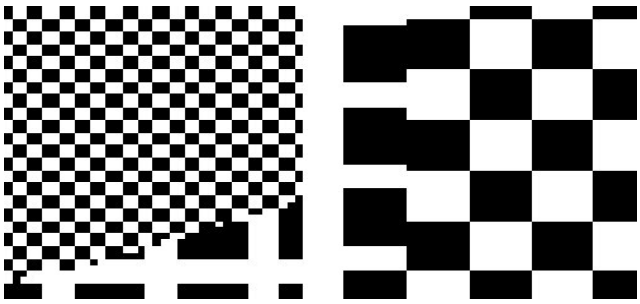


Рис. 2.5. Квадратний растр

2. прямокутний, (рис. 2.6)

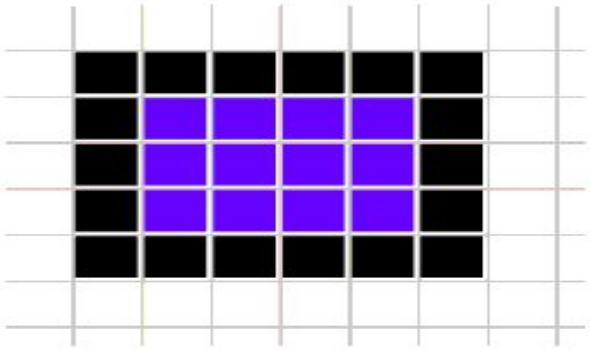


Рис. 2.6. Прямокутний растр

3. гексагональний, (рис. 2.7).

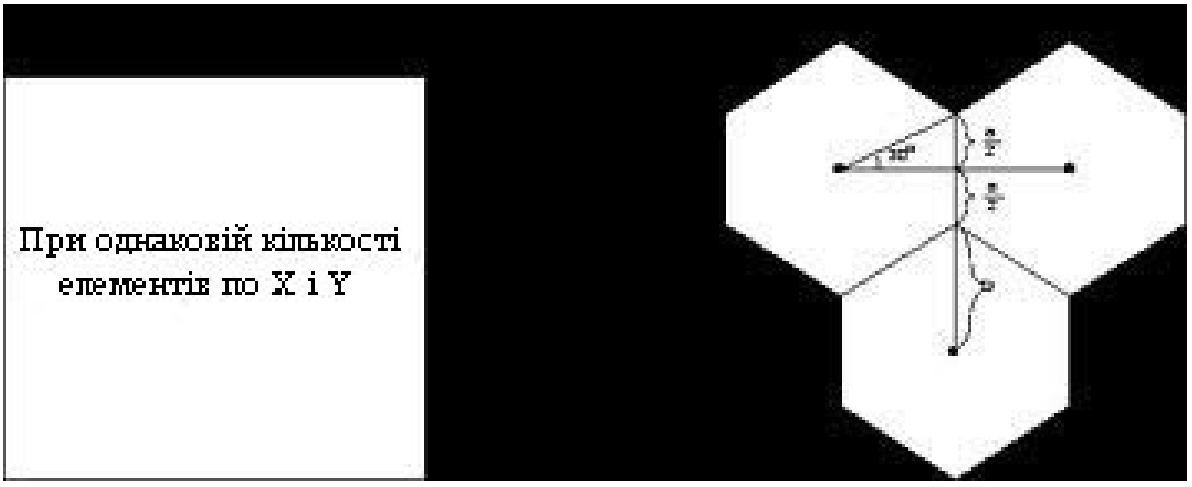


Рис. 2.7. Гексагональний растр

Для опису розташування пікселів використовують різноманітні системи координат. Спільним для всіх систем координат є те, що координати пікселів

утворюють дискретний ряд значень (необов'язково цілі числа). Часто використовується система цілих координат – номерів пікселів із (0,0) у лівому верхньому кутку.

Розмір растру зазвичай вимірюється кількістю пікселів по горизонталі та вертикалі (рис. 2.8).

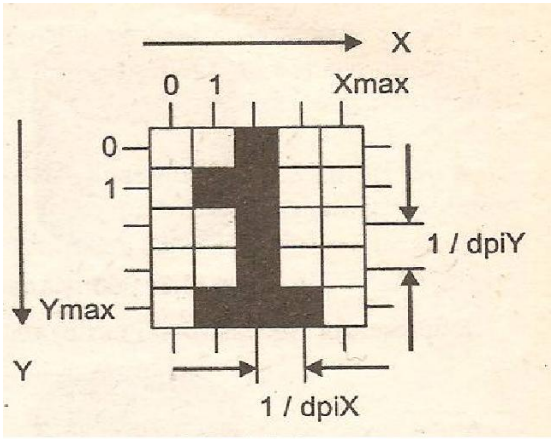


Рис. 2.8. Вимірювання розміру растру

Під час масштабування растрових зображень виникають спотворення (рис. 2.9) – східці (aliasing). Растрові редактори дозволяють частково прибрати ці спотворення за рахунок застосування спеціальних алгоритмів обробки. Ці операції називаються anti-aliasing.

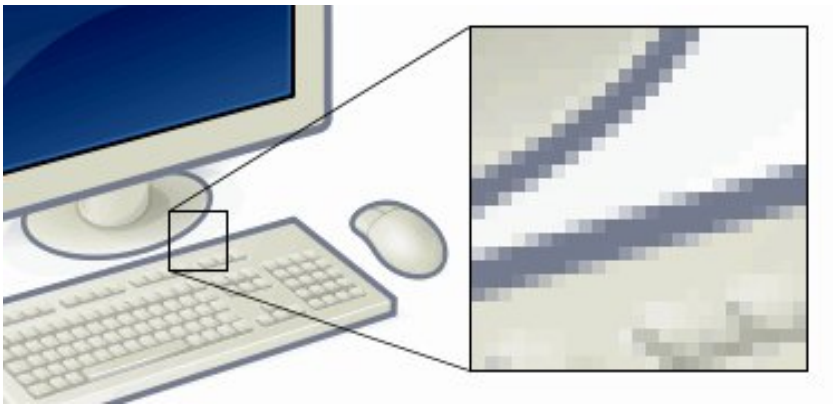


Рис. 2.8. Спотворення растру

2.2. Роздільна здатність растрової графіки

Якість цифрового зображення визначається багатьма параметрами. Одним із основних є поняття роздільна здатність. Вона характеризує відстань між сусідніми пікселями – крок дискретної сітки растра.

Роздільна здатність (resolution) *– це кількість дискретних елементів на одиницю довжини.*

За одиницю довжини був прийнятий дюйм (inch), рівний 25,4 мм. Отже, дискретний елемент це піксель. Таким чином роздільну здатність можна визначити як кількість пікселів у дюймі, як правило вона позначається як **ppi,** що є скороченням від словосполучення **pixel per inch** (пікселів у кожному дюймі).

Проте, в реальному житті пов'язана з роздільною здатністю термінологія не така однозначна. У залежності від пристосування, на якому виводиться зображення, можливе застосування таких одиниць вимірювання роздільної здатності:

1) spi (sample per inch) – елементів на дюйм, (рис. 2.10)

Рис. 2.10. Приклад роздільної здатності растру в spi

2) dpi (dot per inch) – точок на дюйм, (рис. 2.11),

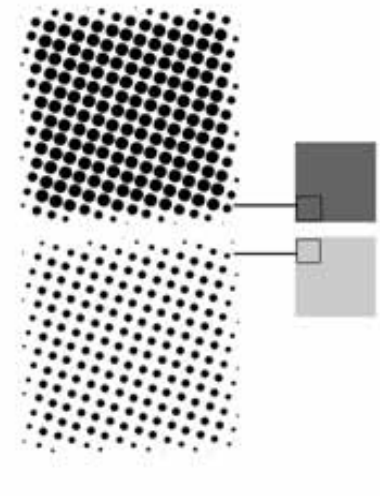


Рис. 2.11. Приклад роздільної здатності растру в dpi

3) ppi (pixel per inch) – пікселів на дюйм, (рис. 2.12),

Рис. 2.12. Приклад роздільної здатності растру в ppi

4) lpi (line per inch) – ліній на дюйм (рис. 2.13).

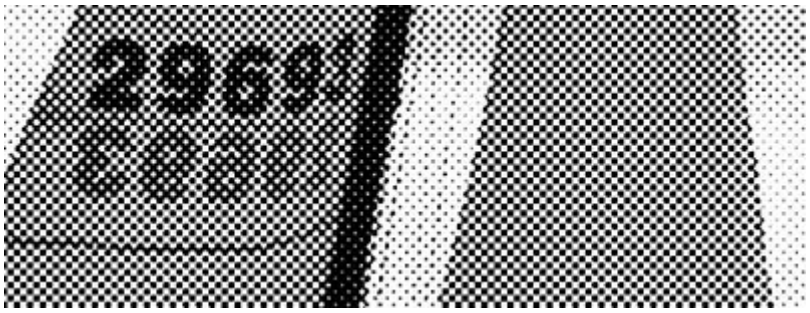


Рис. 2.13. Приклад роздільної здатності растру в lpi

Форма пікселів растра визначається особливостями пристрою графічного **виводу** (рис. 2.14). Наприклад, пікселі можуть мати форму прямокутника чи квадрата, які за розмірами дорівнюють кроку растра (дисплея на рідких кристалах); пікселі можуть мати круглу форму і за розміром можуть і не дорівнювати кроку растра (принтери).

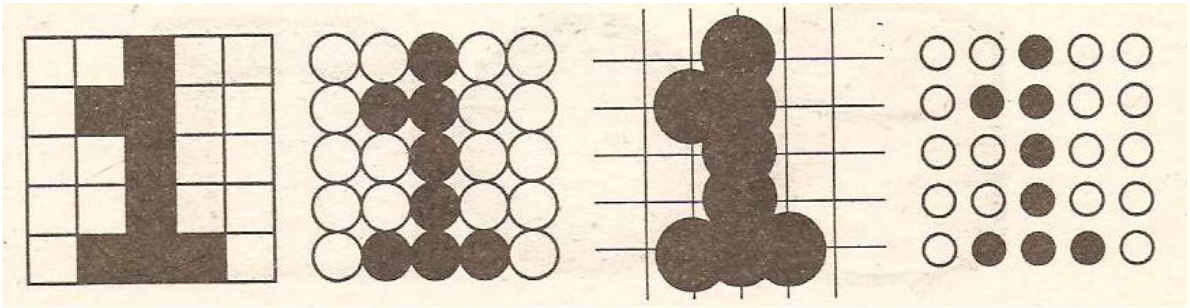


Рис. 2.14. Форми пікселів растра

2.3. Найбільш поширені редактори растрової графіки

Існує багато редакторів растрової графіки, які мають певні переваги і недоліки в можливостях і застосуванні. Велика кількість растрових графічних редакторів пояснюється тим, що оптимальної програми створення і обробки зображень не існує.

Програми обробки растрових зображень поділяють на професійні та любительські (аматорські).

Найбільш популярними системами растрової ілюстрації комп'ютерної графіки професійного класу є графічні пакети Adobe Photoshop і Meta Creations Painter.

Розповсюдженими програмами цього класу є: Adobe Image Ready, Corel Photo Paint, Micrografx Picture Publisher, Macromedia, Fractal Design Painter, Macromedia FreeHand, Fauve Matisse та інші. Такі професійні графічні пакети, як XRes, Photoshop, Painter, Paint Picture Publisher та інші, роблять можливим гнучкий монтаж. Це стосується і пакету Photo Impact, який не потрапив до класу професійних програм через відсутність повної підтримки кольорової моделі CMYK.

Фактичним стандартом серед програм для професійної роботи з растровою графікою, є пакет Adobe Photoshop. Він є лідером на платформах Macintosh і Windows серед програм ретушування растрових зображень і фотодизайну.

Adobe Photoshop – це потужний і гнучкий інструмент, задовольняючий вимоги майже будь-якого споживача – від новачка, який знайомиться з комп'ютерною графікою, до досвідченого професіонала. Тому користувачами Photoshop найчастіше стають дизайнери, фотодизайнери, працівники друкарні і видавництва.

Photoshop підтримує функції імпорту/експорту, створення (малювання), редагування, вклеювання, маскування, пересування змісту, обробки за допомогою численних спеціальних ефектів, підготовки для Web графіки, а також монтажу композицій цих зображень і створення фотореалістичних колажів за допомогою таких технологій як: імітація природних інструментів, матеріалів і технік малювання та корегування; виділення зображення та його окремих елементів; застосування звичайних і корегувальних шарів, багаторівневих масок, фільтрів, альфаканалів і шляхів, макросів, керування напливом; багаторазових відмін виконаних дій; корекції кольорових тонів і контрастів, системи гнучкого та точного керування кольором і роботи з плашечними кольоровими каналами зі зниженою ймовірністю помилок кольоропередавання в сервісних бюро; кешування зображень для підвищення швидкості їх перемальовування; роботи з текстом і текстово-графічними композиціями.

Photoshop має великий набір інструментарію для роботи з бітовими образами: традиційні інструменти (пензлі, олівці та інші), оригінальні інструменти (наприклад, штамп для копіювання пікселів із однієї вказаної області до іншої), Art-пензлі для імітування традиційних технік малювання тощо. Всі інструменти мають власні параметри, що дозволяють максимально точно настроювати їх для конкретної роботи.

Photoshop підтримує велику кількість графічних файлових форматів, включаючи формати Web-графіки. Він може імпортувати фотографії, отримані за допомогою цифрових камер, дозволяє виконувати функцію оптимізації зображення з урахуванням специфіки роботи з ним у Internet. Photoshop має можливість інтеграції (в процесі інсталяції) з іншим графічним пакетом фірми Adobe – Image Ready.

Adobe Image Ready включений у пакет Adobe Photoshop, починаючи з версії 5.5. Ця програма надає величезні можливості для обробки графіки під Web: оптимізація зображень, створення анімації, можливість «різати» картинки на більш дрібні та багато іншого.

Хоча інтерфейси цих програм однакові, а можливості редагування – одного рівня, з точки зору дизайнера, найоптимальніше використовувати Photoshop для виконання робіт із графічного редагування, а Image Ready – для виконання операцій, пов'язаних із використанням зображень у мережі, та імітації, що базується на шарах.

Проте Photoshop здається універсальною програмою до тих пір, поки здійснюється корегування та комбінування наявних фото.

Серед недорогих і розповсюджених растрових ілюстрацій них редакторів необхідно відмітити потужний редактор GIMP, який є практично повним аналогом професійного редактора Adobe Photoshop. Перша версія GIMP була випущена в лютому 1996 р. і мала великі можливості як перша професійна та безкоштовна програма обробки зображень, що конкурувала з великими комерційними графічними редакторами. Цей редактор має інструменти для створення десятків різновидностей форматів растрової графіки. З точки зору українських дизайнерів GIMP має дві значні переваги. По-перше, ця програма безкоштовна та її інтерфейс перекладено на українську мову. По-друге, цей графічний редактор має вбудовану підтримку скриптів на мові програмування. На жаль, GIMP поки що не має режиму повноцінної роботи з файлами, які не готуються до друку на поліграфічному обладнанні.

Photo Paint – не менш відомий графічний редактор із пакету Corel Draw, який призначений для обробки растрової графіки, що конкурує з Adobe Photoshop. Він є ще одним стандартом растрової графіки, що разом з Photoshop охоплює більшість споживачів працюючих із графічними пакетами. Photo Paint надає великі можливості не тільки в галузі створення зображень: оскільки містить всі інструменти, необхідні для обробки готових зображень, фотографій і ілюстрацій, за рахунок різноманітних фільтрів, текстур. Відмінною особливістю Photo Paint є зручність роботи, інтерфейс і швидкість накладання фільтрів. Проте ці операції відбуваються дещо повільніше, ніж у Adobe Photoshop.

Лідером серед графічних пакетів для художників є Meta Creation Painter. Він має ті ж базові можливості, що й Adobe Photoshop, але містить значно більше графічних пензлів, текстур, майстрів, спецефектів для імітування реальних інструментів, матеріалів і технік малювання професійного художника. Допомагають у роботі з цією

програмою графічні планшети. Meta Creation Painter підтримує кольорову модель, що зацікавлює користувачів, які працюють для поліграфії.

Традиційною складовою частиною графічного пакету Corel Draw є програма обробки зображень Corel Photo Paint. Вона приваблює творчих графіків чудовими природними зразками заповнень, що обладнані різноманітними переходами кольорів і художніми пензлями, точною роботою корекції контрастності. Професіоналам подобається така велика програма, бо має великий набір інструментів, складна багаторівнева технологія, зручність перегляду та роботи. Серед недоліків цієї програми необхідно відмітити, що початково робоча поверхня розсіює увагу, а численні кнопки та палітри вимагають довгого звикання.

Графічний пакет Picture Publisher, який раніше розповсюджувався як складова частина Micrografx Graphics Suite, є недорогою, гарно відпрацьованою програмою з великими можливостями, яку легко освоювати. Безперечними її перевагами є рекордер команд, багато макросів і майстрів, а одним із недоліків – відсутність автоматичного попереднього перегляду фільтрів.

На обробці великих файлів зображень спеціалізується графічний пакет Macromedia XRes. Його перевагою є зберігання монтажів із економією ємності пам’яті. Але він не є для професіонала навіть частковою заміною Photoshop і вимагає для процесу монтажу великого обсягу пам’яті.

Для створення двомірного комп’ютерного живопису часто застосовують графічні пакети Fractal Design Painter, Macromedia FreeHand, Fauve Matisse.

Ці графічні пакети мають широкий асортимент функціональних можливостей малювання, редагування зображень і тексту, роботи з кольором. Наприклад, для моделювання застосовують типові професійні інструменти художника вуглинки, олівці, пензлі, пір’їни, аерографи і інші матеріали для малювання (акварель, олія, туш, багатокольорову градієнтну заливку, ефекти природного середовища).

Зокрема програма редагування растрового живопису Painter фірми Fractal Design володіє достатньо широким спектром засобів малювання і роботи з кольором, моделює різноманітні пензлики (олівець, ручку, вугілля, аерограф та інші), дозволяє імітувати малюнки аквареллю та маслом, а також добитися ефекту натурального середовища. Недоліками роботи цього редактора є: складна робоча поверхня (через це його краще не застосовувати для «чистої» фотообробки), націленість тільки на тих, хто має художні навички, не найкращі можливості для створення ілюстрацій, насичених текстовими ефектами, вимогливість до пам'яті комп’ютера.

У свою чергу на платформі Macintosh популярними є пакет Pixel Paint Pro фірми Pixcel Resources, призначений для редагування растрового живопису та зображень.

На платформі графічних робочих станцій SGI популярним є пакет Studio Paint 3D виробництва Alias і Wavefront, призначенням якого є: малювання різними інструментами в режимі реального часу на 3D- моделях; робота з необмеженою кількістю шарів зображення; надання декількох десятків рівнів відміни попередніх

дій; виконання операцій корекції кольору; використання сплайнових пензлів (їх можна редагувати попіксельно як сплайнову криву); підтримання планшету з чутливим пером (це дозволяє художнику виконати традиційний ескіз від руки, перенести його в 3D–пакет для моделювання та анімації, та побудувати за цим ескізом 3D-модель).

Ринок програмного забезпечення пропонує більше десятка розповсюджених графічних пакетів різних цінових рівнів. До недорогих растрових ілюстраційних редакторів без підтримки CMYK для початківців з високою продуктивністю відносять RGB художники, що своєю стандартною роботою Windows–поверхнею полегшують створення користувачем перших витворів мистецтв, відносять такі пакети: Photo Impact, Paint Shop Pro, Micrografx Windows Draw, Photo Magic, Lview Pro, Microsoft Windows Paint.

Photo Impact дозволяє досить якісно покращувати фотографії, зроблені за поганого освітлення, непогано справлятися з автоматичною підгонкою кольорів об'єктів у відповідності з фоном, мати велику кількість вмонтованих фільтрів, ретельно розроблених із врахуванням балансу зручності та функціонування. Під час застосування незареєстрованих версій, програма успішно і повноцінно обробляє графічні файли всіх розповсюджених форматів.

За ліцензією Shareware розповсюджується добре відомий у всьому світі графічний редактор Paint Shop Pro, який пропонується компанією Jase Software. Він є умовно безплатним класичним піксельним редактором, нескладним у опануванні, що швидко та гарно розв'язує типові повсякденні задачі, але має високу ціну повної версії. Paint Shop Pro підтримує всі розповсюджені формати графічних файлів, у тому числі і PSD, який належить Adobe Photoshop, своєму головному конкуренту. У розповсюдженні споживача є об'ємна бібліотека попередньо розроблених ефектів і фільтрів (більше 70). Існує також можливість створювати власні ефекти і зберігати їх у спеціальному файлі для застосування в майбутньому. Для роботи з анімаційною графікою споживачу надається окрема програма Animation Shop, яка встановлюється з одного дистрибутива з самим редактором. Незаєрестрована копія дозволяє вільно редагувати графічні файли всіх форматів. Бажання компанії отримати зі споживача оплату виражається під час роботи з програмою не досить надоїдливими повідомленнями. Після закінчення тридцятиденного терміну, наданого на те, щоб увійти «в смак», споживачу, який не встиг переконати в усіх перевагах Paint Shop Pro, надається своєрідний бонус, який дозволяє безкоштовно попрацювати з програмою ще два місяці. Після цього програма остаточно перестає функціонувати.

Різноманітні функції ретушування має пакет Softkey Photo Finish, який є майстром у покращенні зображень. Але для монтажів – це не найкраща програма оскільки не підтримує багаторівневі технології.

Пакет Microsoft Windows Draw характеризується найкращим співвідношенням ціни та якості функціонування, надає сотні зразків для друкованих бланків усіх видів,

добре показує себе в створенні HTML-сторінок для WWW, містить потужну програму ретушування, програмні засоби для малювання, підтримує багаторівневу технологію та автоматичні ефекти. Однак суттєвим недоліком є використання власного формату.

У складі Windows Draw є програма обробки зображень Photo Magic. Яка підтримує численні пензлі, ефекти, здійснення фотомонтажу, набір напівавтоматичних послідовностей операцій для генерації цікавих ефектів.

У той же час пакет Lview Pro – це гарна програма для швидкого перегляду та редагування графіки, що не відрізняється унікальністю, але повністю працездатна та може бути заміною для Paintbrush.

Стандартним у наборі MS Windows є графічний пакет Paint, призначений для виконання найелементарніших графічних операцій. Він є найпростішим із перелічених растрових редакторів у середовищі MS Windows. Paint з'явився у версії Windows-95 і з тих пір не змінився. Paint являє собою простий одновіконний редактор растрової графіки, який, тим не менше, дозволяє створити досить складний малюнок. Програма включає засоби для побудови прямих і кривих ліній, еліпсів і кіл, прямокутників, квадратів і багатокутників (як контурних, так і зафарбованих), є інструменти для виділення фрагмента малюнка, заливки замкненої області кольором, а також інструменти, імітуючі малювання пензликом і пульверизатором, а також створення напису та задання товщини лінії. Крім цього Microsoft Painter надає чудові можливості для імітації роботи реальними інструментами малювання: графітом, крейдою, маслом і ін., дозволяє відтворювати фактуру поверхні різноманітних матеріалів і створювати анімацію. Цей редактор дуже зручно використовувати для розробки фонових малюнків або Web-сторінок у стилі живопису. Той, хто користується цією програмою, відчуває себе справжнім художником. Доступні і деякі оператори перетворення рисунка, а саме: дзеркальне відображення відносно горизонтальної і вертикальної вісі, інвертування (т.т. створення схеми з одним входом і одним виходом, у якій вихідний сигнал виникає лише тоді, коли немає сигналу на вході) і заміна кольорів, стиснення, розтяг і нахил. Проте в Paint зовсім відсутні різного типу ефекти і фільтри. Крім цього, цей редактор підтримує всього декілька форматів файлів. Редактор має набір найпростіших функцій (пензлик, олівець, гумка та ін.), які дозволяють створювати нехитрі картинки. На жаль, для обробки графічних зображень він не придатний.

Розповсюдженими графічними редакторами Microsoft є такі, як: Microsoft Photo Editor, Microsoft Photo DRAW, Paint Brush, Microsoft Picture It!, які дозволяють реалізувати найпростіші задачі, але не вважаються професійними.

Microsoft Photo Editor – це простий графічний редактор, який дозволяє переглядати файли малюнків різних форматів. Крім цього, цей редактор дає можливість застосувати різни ефекти, різкість і прозорість, змінювати розмір малюнка, повертати його, вирізувати, копіювати і вставляти його складові. Програма

Photo Editor пропонує вибір прямокутників та не дуже зручну «чарівну паличку», що містить (на основі подібних за кольором точок зображення) ті області об'єкта, які після вставки їх у документ Word, у сторінку Web або інші документи, повинні залишатися прозорими. Photo Editor може обрізувати зображення, обертати його, робити чіткішим, переобчислювати розмір файла та змінювати глибину кольору. Ця програма має також декілька простих можливостей корекції кольору та контрастності, чудовий фільтр ефектів (акварель, крейда, вугілля, папір для листів із ефектом ручного виділення).

Для створення і редагування малюнків і ділової графіки призначений графічний редактор Microsoft Photo DRAW. Його програмне забезпечення включає в себе засоби для створення ілюстрацій, редагування фотографій і растрової графіки.

Редактор Paint Brush також вважається одним із найпростіших графічних редакторів для Windows 3.1. Його програмний елемент знаходиться в програмній групі «Реквізити». Покращеним аналогом цього редактора є Paint для Windows-95. Microsoft Gif Animator для Windows-95 – слугує для створення рухомих зображень (анімацій) і для перетворення відеофайлів ***.** avi в анімаційні gif-файли. Окремі фрагменти анімаційного зображення (кадри чи фрейми) створюються, наприклад, в редакторі Photoshop або Paint, а потім через буфер обміну Windows вставляються в Gif Animator. Усі створені об'єкти можна редагувати: змінювати розміри, повертати, копіювати з одного місця і вставляти в інше, змінювати колір. Робота в цьому редакторі дає непогану підготовку для освоєння більш складних професійних пакетів.

Обробку зображень також можна здійснювати засобами офісних пакетів. Наприклад, у пакеті Microsoft Office зображення можна обробляти, застосовуючи наявну в комплекті програму Photo Editor і можливості Word, Excel і Power Point.

Багато вдалих інструментів пропонує графічний пакет Windows Draw. Проте тому, хто хоче зробити фотомонтаж із заготівок, потрібно звернутися до Photo Express чи Picture It!

У растровій графіці широко застосовуються програми Excel і Power Point. У них як і в Word можна керувати обробкою імпортованого зображення за допомогою інструментів малювання, які дозволяють регулювати яскравість, контрастність і робити прозорими подібні за кольором області тих об'єктів, які вставляють у зображення. Можна відміняти будь-яку кількість виконаних команд. Усі зміни, пов'язані з яскравістю, контрастністю чи виконанням вирізок, відміняють за допомогою команди повернення графіки в початковий стан, навіть після збереження зображення (оригінал завжди залишиться збереженим у фоні, а під час нашаровування елементів зображення одне на одне точки того зображення, що лежать нижче, не будуть стертими).

Навіть на швидких ПК і в разі простих пересувань об'єктів ці монтажі швидко викликають втому. Для користувачів, які багато монтують, потрібен професійний пакет обробки зображень.

До недорогих графічних пакетів, які є недорогими растровими графічними редакторами без підтримки CMYK для початківців із середньою продуктивністю відносять програми з нетрадиційним інтерфейсом і великою кількістю шаблонів зображень, що полегшують входження в процес обробки зображень, типових для будь-якої сфери діяльності та побуту (у першу чергу – різних друкованих бланків, календарів, вітальних листівок, афіш, фотоальбомів, рамок для фотографій тощо). Їх типовими прикладами є програми: Microsoft Picture It!, Live Pix, Kai's Photo Soap, Ulead Photo Express, Adobe Photo Deluxe.

Програма Microsoft Picture It! – це вдале рішення для дизайнерів-початківців, яке надає можливість інтуїтивного використання, гарні ефекти та зразки, але має не дуже вдалу робочу поверхню та неочевидне розташування потрібних пензлів і інструментів для вибору.

У той же час програма Live Pix – це ідеальний інструмент для виготовлення колажів, який містить зразки для багатьох чудових монтажів. Її безперечними перевагами є багато шаблонів і автоматичних тіней, недоліком – слабкі опції настроювання контрастності.

Продуктом для дизайнерів, який має елегантну робочу поверхню з фотореалістичними символами інструментів є пакет Kai's Photo Soap. Його перевагою є гнучкий монтаж, недоліками – обмежені графічні можливості використання робочої поверхні та відсутність складних інструментів. Програма надає зразки для календарів і вітальних листівок, але може бути застосована для оформлення без використання цього набору зразків. Вона є корисною для початкового опанування технологій цифрової обробки зображень.

Програма ж Ulead Photo Express пропонує опції для маніпуляцій з текстом і простий монтаж зображень для оформлення поздоровлень. Її перевагами є текстовий інструмент, який пропонує багато варіантів оформлення, та наявність багаторівневої технології для гнучких фотомонтажів, а недоліком те, що деякі корисні функції доводиться шукати.

У свою чергу пакет Adobe Photo Deluxe, призначений для початківців і тих, хто має деякий досвід роботи. Він пропонує два способи роботи: з попередньо штампованими проектами та самостійну обробку зображень. Ця програма підходить для швидкого виведення зразків із застосуванням їх. Перевагами Adobe Photo Deluxe є багаторівнева технологія та потужна функція тіней, а недоліками – те, що в разі вільного оформлення зображень заважають не дуже зручна робоча поверхня та мала кількість деяких функцій, а також незначний інструмент для плавних кольорових переходів.

Існують графічні редактори, які вважаються найбільш поширеними та більш спеціалізованими для роботи з тим або іншим форматом графіки. Серед них слід назвати такі, як: Xpaint – редактор растрової графіки, схожий на Paint, але містить багато додаткових можливостей; Kpaint – встановлений в KDE редактор растрової

графіки, менш функціональний ніж Xpaint; KDE Icon Editor – редактор растрової графіки, призначений для створення піктограм.

Пакет Photo Impact підтримує відмінну техніку для творчості, забезпечуючи, завдяки своїм багатим функціональним можливостям, швидке наведення лоску на зображення. Його недоліком є відсутність CMYK-функцій.

До розряду універсальних web-редакторів можна віднести Macromedia Fireworks (Ghjuhve Macromedia Fireworks), оскільки за допомогою Fireworks можна: створити графічні зображення, застосовуючи векторні і растрові інструменти; добавляти інтерактивні елементи і анімацію; зберігати в форматі HTML and Images, експортувати для роботи в інших програмах. Ця програма призначена для створення і редагування, як векторної, так і растрової графіки для web. З її допомогою можна створювати інтерактивні елементи інтерфейсу: різні кнопки і випадаючі меню, нарізувати картинки на частини (слайди/slices) для подальшого генерування HTML-коду. А також застосування програми разом із іншими продуктами Macromedia (Dreamweaver, Flash) забезпечує комплексність процесу приготування Web-продукції, хоча і використання Fireworks окремо дасть чудові результати.

2.4. Растрові формати графічних файлів

Формати графічних файлів можна розділити на такі класи:

1. ті, що зберігають зображення в растровому вигляді (BMP, GIF, JPEG, PNG, TIFF, PCX, RAW, IFF, MAC, PixelPaint, PIXAR, TGA, LWG і ін.);

2. ті, що зберігають зображення у векторному вигляді (EPS, Shockwave Flash, DXF, MIF-MID і ін.);

3. ті, що можуть сполучати растрові і векторні представлення (EPS, PICT, CDR, AI, FH7 і ін.);

4. метафайли (PDF, PSD, CT, WMF, EMF, CGM), які, крім інформації про растрові та/або векторні зображення, містять також інформацію про команди візуалізації.

Усі класи форматів мають якісь характерні особливості і можливості, роблячи їх незамінними в роботі.

Зокрема, растрові графічні формати слугують для опису растрової графічної інформації, яка в свою чергу застосовується, коли потрібна висока точність в передачі кольорів і напівтонів. Однак при цьому розміри файлів суттєво збільшуються з розміром роздільної здатності, що в свою чергу приводить до великих розмірів файлів.

BMP (Bitmap) – стандартний, давній та традиційний растровий формат, який став одним із основних у системі Windows і використовують його на DOS- та Windows-сумісних комп'ютерах. BMP-файли добре передаються між комп'ютерами

та обробляються. Оскільки дані не можна стиснути, то під час виведення на екран і на друк втрати якості немає. BMP-файл має просту структуру та зберігає єдине зображення з одним, чотирма, вісьмома та двадцатьма чотирма біт/піксель. Перші три подання відповідають індексованому кольоровому зображенню. Для цих зображень у заголовку BMP-файлу зберігається таблиця колірності. Зображення може бути стиснене з використанням кодування довжин серій для 4-ох та 8-бітових зображень.

Файли BMP мають найбільший розмір порівняно з файлами інших растрових форматів. Растр тут зберігається майже у тому вигляді (якщо не враховувати вирівнювання рядків на довжину, кратну 4 байтам), в якому він записується в оперативну пам'ять для відображення та обробки. Зазвичай програми так і записують файли BMP, хоча в цьому форматі передбачене ущільнення (компресія) растра методом RLE. BMP-файли з компресією можуть мати розширення DIB або RLE.

Різновидом формату BMP для операційної системи OS/2 є Bitmap 32. Від BMP він відрізняється тим, що дані про одну точку зберігаються не в 24, а в 32 бітах. Додаткові 8 бітів використовуються для Alpha-каналу. Отже, для розробника цей формат зручний, насамперед, за рахунок збереження додаткової складової прозорості, яка зберігається усередині файлу з текстурою.

Проте, якщо потрібно впакувати на жорсткому диску багато зображень, слід відмовитися від BMP-файлів: навіть невеликі кольорові зображення з роздільною здатністю 640´480 вимагають декількох Mb.

Формат GIF (Graphics Interchange Format, – формат взаємообміну графікою, розширення – gif) був запропонований найбільшою службою онлайна CompuServe спеціально для передачі растрових зображень у глобальних мережах. GIF використовують для подання індексованих кольорових зображень та HTML-документів у електронних мережах. Він є ущільненим форматом, розробленим для прискорення пересилання файлів телефонними лініями.

GIF дозволяє містити в одному файлі декілька самостійних зображень та здійснювати анімацію. Цей формат дозволяє зберегти растрові дані в пікселях (розміром до 64000) із глибиною кольору від 1 до 8 біт. Зображення записуються із застосуванням колірної моделі RGB і даних палітри. Але це не заважає зберігати зображення дворівневої та монохромної моделей, які реалізують шляхом настроювання таблиці колірності. А модель повнокольорового зображення зі зберіганням для кожного пікселя компонентів R, G, B не підтримує через обмеження у 8 біт/піксель. Цей формат вигідно застосовувати для зображень з малою кількістю кольорів і різкими межами (наприклад, для текстових зображень).

Безперечною перевагою формату є наявність стандартизованого протоколу передавання GIF-зображень. Донедавна формат був найважливішим файловим форматом для обміну кольоровими зображеннями модемним зв'язком. Крім цього

формат забезпечує швидке розпаковування під час перегляду та апаратну незалежність.

GIF використовує ефективний алгоритм стиснення LZW. Цей алгоритм стиснення майже не поступається програмам-архіваторам із ступенем стискання 3:1-5:1. GIF-файли не потрібно архівувати, оскільки це рідко дає відчутний виграш у об'ємі.

Проте використання одного методу кодування зображень (LZW) обмежує сферу застосування формату GIF. Оскільки LZW-стискання захищене патентом і під час його комерційного використання постають юридичні проблеми.

У 1987 році був запропонований фірмою CompuServe формат GIF 87a як незалежний від апаратного забезпечення засіб обміну растровими зображеннями у мережі Інтернет. Формат підтримує зображення, що містять до 256 кольорів.

У 1989 році опублікована переглянута специфікація формату, що одержала назву GIF 89a. Порівняно з GIF з'явилася можливість зберігати текстові та графічні дані в одному файлі і працювати з областями прозорості, що дозволяє створювати зображення непрямокутної форми.

Існує можливість запису зображень із чергуванням рядків (interlacing, черезрядкове розгорнення). Під час візуалізації черезрядкових відображень спочатку виводиться кожний восьмий рядок, потім – кожний четвертий і т.д. Користувач може перервати прийом зображення з мережі, не чекаючи виводу всіх рядків образа. У цій версії формату GIF можна призначити один або більше кольорів прозорими, вони стануть невидимими в браузерах Інтернету й у деяких інших програмах. Крім цього ще однією корисною властивістю цього формату є підтримка анімаційних зображень. Файл GIF може містити не одне, а кілька растрових зображень, які браузери можуть довантажувати одне за одним із зазначеною у файлі частотою. За рахунок цього досягається ілюзія руху (GIF-анімація).

До недоліків цього формату слід віднести підтримку не більше, ніж 256 кольорів і патентні обмеження. Недоліки призвели до того, що GIF поступово замінюють форматом JPEG.

JPEG (Joint Photographic Experts Group, розширення - jpg) розроблений групою експертів із фотографії (що видно з назви) під егідою ISO. Зараз він знайшов широке застосування для відображення фотографій та інших тонових зображень в електронних мережах.

Цей формат економить 50-70 % обсягу пам'яті за рахунок видалення з файлу тої його частини, яка не має значного впливу на якість зображення та дозволяє регулювати співвідношення між мірою стискання файлу та якістю зображення. На відміну від алгоритму стискання GIF, який аналізує файли по рядках, JPEG розбиває зображення на області низьких кольорів, оцінюючи які кольорові розбіжності між сусідніми блоками 8´8 зображення є невидимими для ока та відкидаючи інформацію про них.

Позитивними рисами алгоритму JPEG є те, що: досягається ущільнення в 10-30 разів без істотного погіршення зображення кольорового фото; користувачам надається можливість задавати необхідний ступінь ущільнення; алгоритм досить простий для реалізації на персональних комп'ютерах і автономних пристроях, таких як цифрові фотоапарати.

Негативними сторонами алгоритму є те, що під час підвищення ступеня ущільнення вище 20-30 зображення сильно спотворюється, розпадаючись на окремі великі квадрати 8×8. У зв'язку з великими втратами в низьких частотах під час квантування; для окремих зображень із високою деталізацією навіть за помірного ущільнення виникає ефект Гіббса – ореоли навколо контурів об'єктів. Несуттєвим є часткова втрата даних. Між якістю зображення та ступенем ущільнення існує зворотна залежність: чим вищу якість задати для підсумкового зображення, тим менш компактним буде впакований файл.

Крім цього JPEG дозволяє використовувати режим почерговості, забезпечуючи поступове підвищення якості зображення під час його завантаження по мережі, а також не допускає збереження в зображенні прозорих областей. Формат підтримує такі кольорові формати як: відтінки сірого, RGB і CMYK. Глибина кольору становить 24 біти.

У зв'язку з цим метод JPEG відкрив шлях масовому розповсюдженню й використання ефективних технологій роботи з повноколірними зображеннями. У 1991 році з'явилися офіційні стандарти для JPEG – 10918-1, 2.

Для уніфікації файлових форматів, що використовують метод JPEC, були розповсюджені рекомендації, названі JFIF (JPEG File Interchange Format). Файли цього формату мають розширення .JPEG, JPG, JFIF. Крім того, метод ущільнення JPEG використовується в інших форматах файлів, наприклад, TIFF, Kodak Photo CD, QuickTime та інших. Оскільки JPEG не дозволяє включати в файл більше одного зображення, то анімація JPEG не дуже поширена.

Формат PNG (Portable Network Graphics – мережева графіка, що переноситься, розширення – png), як видно з назви, призначений для передавання зображень по мережі. Він розроблений спеціальним комітетом, очолюваним Томасом Бутеллем. Затверджено стандарт PNG версії 1.0 і надруковано у 1996 році.

Цей формат не захищений патентами, не вимагає ліцензування й фінансових відрахувань і тому широко розповсюджується – саме через патентування й жорстке ліцензування алгоритму LZW і виник PNG. Він використовує алгоритм стискання Deflate (вдосконалений LZW – краще стискання, ніж GIF на 10–30 %), підтримує п'ять різних фільтрів стискання. Крім цього передбачені 254 рівня альфа-каналу та черезрядкова розгортка.

Це досить «молодий» формат, що конкурує з GIF. Всі останні версії браузерів підтримують його без спеціальних модулів, що підключаються.

До спільних властивостей GIF та PNG слід віднести: використання методів компресії без втрат; підтримка індексованих кольорів до 8 бітів на піксель; маска прозорості (Alpha-канал); забезпечення прогресуючого показу; окрім зображення, файл може також містити текст.

Відмінностями PNG від GIF є те, що: PNG ефективніше ущільнює 256 колірні індексовані зображення. Однак це спостерігається не для будь-якого зображення. Наприклад, GIF погано зображає малюнки з великою кількістю вертикальних ліній. А для формату PNG ущільнення зменшується за наявності горизонтальних ліній; 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 записується тільки одне зображення – немає підтримки анімації, як у GIF (у наступних версіях PNG планується це змінити).

Завдяки підтримці комп'ютерної інформаційної служби CompuServe, формат PNG у теперішній час став одним із стандартів в області представлення графічної інформації, хоча доки він використовується не надто широко. Фірма Macromedia, розроблювач програми Fireworks, зробила формат PNG внутрішнім форматом цього редактора. Усі останні версії найпоширеніших браузерів Internet (Internet Explorer, Netscape Navigator і Opera) підтримують даний спосіб збереження інформації.

Для подання зображень у електронних мережах формат зберігає всю кольорову інформацію та альфа-канали зображення. Специфікація формату PNG включає можливості автоматичної корекції кольорів під час перенесення зображень між апаратними платформами та ефектів змінної прозорості, у той час як GIF підтримує тільки перший формат, а JPEG – тільки два останніх; глибина кольору в 48 розрядів робить PNG-файли цікавими для DTP-професіоналів, що працюють із компактними файлами та платформово-незалежно. Формат файлів є вільним від ліцензування.

Для підтримки прогресуючого показу використовується двовимірний interlacing (не тільки рядків, але й стовпців). Черезрядковий режим виводу виконується за методом Adam7. Черезрядкова схема передачі зображень збільшує розміри графічного файлу.

На відміну від GIF (версії 87a), де прозорість або є, або ні, PNG підтримує також напівпрозорі пікселі за рахунок Alpha-каналу з багатьма (максимум 2^{16}) градаціями.

У стандарті PNG підтримується апаратна незалежність графіки. У файл формату PNG записується інформація про гамма-корекцію. Окрім цього, у файл можна записати значення кольорів R, G, B та білого в точних колориметричних координатах МКО XYZ. Це дозволяє професійній графічній системі найбільш точно відобразити кольорове зображення. Також PNG може зберігати співвідношення ширини та висоти зображення – це може бути використано під час виводу зображення на графічному пристрої, у якого різна роздільна здатність по горизонталі та вертикалі.

Разом із графічною інформацією у форматі PNG можна зберегти метадані чи інформацію про індексування зображення. Ці дані використовуються пошуковими машинами, що значно прискорює й полегшує пошук файлів PNG в Інтернеті.

У форматі PNG використано ефективний алгоритм ущільнення даних "без утрат" Deflate (це різновид словникового методу LZ77). Для програмістів, які бажають використовувати метод Deflate, пропонується бібліотека zlib. Для підвищення ефективності ущільнення передбачено фільтри, якими обробляють рядки зображення. Є чотири фільтри Sub, Up, Average та Paeth, які обробляють байти (не пікселі) кожного рядка.

Основним недоліком PNG-формату є те, що алгоритм стискання для PNG-файлів працює правильно тільки тоді, коли на диск заносять кольорові зображення з високою глибиною кольору. Формат PNG не підтримує багатоканальних зображень, колірних профілів і контурів обтравки.

Найпоширенішим растровим форматом, який допускається для використання практично всіма існуючими графічними середовищами та переноситься між платформами IBM PC та Apple Macintosh є файловий формат TIFF (Tagged Image File Format – формат файлу ознакових зображень, розширення – tif). Цей формат запропонований фірмою Aldus Corporation. Він був розроблений для зберігання сканованих зображень із високою роздільною здатністю. Виняткова гнучкість формату зробила його дійсно універсальним.

TIFF є відкритим форматом та дозволяє створювати будь-яку модель зображення, вибір якої залежить від конкретної задачі: дворівнева модель є найзручнішою для систем підготування документації; індексоване кольорове зображення придатне для формату зберігання графічної інформації в растрових графічних дисплеях тощо.

Формат TIFF містить не тільки інформацію про модель зображення, а й метричні характеристики зображення (його розміри, щільність у пікселях на одиницю довжини), важливі для систем підготування документації. Цей формат не накладає майже ніяких обмежень на параметри зображення. TIFF дозволяє зберігати в одному файлі будь-яку кількість зображень; декілька копій одного зображення з різними характеристиками (наприклад, варіанти зображення з різною щільністю доцільно мати при роботі з різними принтерами у видавничих системах).

У TIFF можуть використовуватися різноманітні колірні моделі. Підтримується багато методів ущільнення – LZW, Deflate, JPEG та інші. Цей формат дозволяє зберігати інформацію про шари, колірні профілі (ICC-профілі) і канали масок. Допускається збереження векторних контурів, а також інтеграція в файл колірного профілю. Крім цього TIFF підтримує всі колірні моделі, є апаратно незалежним. Формат містить метричні характеристики зображення.

Отже, цей формат можна вважати стандартом обміну графічними даними. У стандарті TIFF версії 6.0 визначено підмножину Baseline TIFF, яку повинні підтримувати всі програми.

Графічний файловий формат PCX (ZSoft) було розроблено для програми PC Paintbrush. Застосовувався графічним редактором ZSoft PC PaintBrush (однією з перших популярних графічних програм) для MS-DOS компанії Microsoft, текстових процесорів і настільних видавничих систем типу Microsoft Word і Ventura Publisher, розроблений компанією ZSoft Corporation (м.Маріетта, штат Джорджія, США). Оскільки PCX використовує RLE-компресію (вона дає низький коефіцієнт стискання, але швидке декодування), цей формат має переваги в інтерактивних системах із швидкою зміною зображень.

PCX – стандарт подання і зберігання зображень ділової графіки (креслення, діаграми, схеми тощо). Більшість растрових форматів файлів застосовують стандартну палітру кольорів, проте формат був розширений з розрахунку на зберігання 24-бітних зображень. PCX – апаратно-залежний формат для зберігання інформації у файлі в такому ж вигляді, як і у відеоплаті. Для сумісності зі старими програмами необхідна підтримка EGA-режиму відеоконтролером. Алгоритм такого стиснення дуже швидкий і займає великий об’єм пам’яті, проте не дуже ефективний, непрактичний для стиснення фотографій і більш детальної комп’ютерної графіки.

Негативними сторонами формату є: не підтримка зображення з відтінками сірого чи таблиці корекції шкали сірого, кольору CMYK або інших систем відмінних від RGB. Різноманітні варіанти, особливо під час роботи з кольорами, можуть здійснити роботу з файлом неможливою; незручна схема стиснення може збільшувати розміри деяких файлів.

До позитивних сторін формату слід віднести такі, як: створення обмеженої палітри кольорів (краще всього 16 чи 256), не можливість включення в себе зображення з поганим стисненням, наприклад, відскановане.

DjVu – це комплект технологій ущільнення формату файлу й програмної платформи, що розроблявся з 1996 р. у фірмі AT&T Labs спеціально з метою розміщення в Інтернеті сканованих зображень (у тому числі фотозображень), суміщених із текстом. Таких, як: книги, рукописи, географічні карти, художньооформлені меню ресторанів і інше. І згідно цього він призначений задовольняти такі вимоги: адекватне передавання повнокольорової графіки, збереження легкості тексту для читання, достатню компактність. Крім того, DjVu-формат оптимізований для передавання мережею таким чином, що сторінку можна проглядати ще до завершення викачування. Таким чином DjVu є унікальним інструментом для відкриття Інтернет-доступу до фундацій звичайних, паперових бібліотек. У березні 2000 р. формат був проданий фірмі LizardTech Inc.

Формат DjVu, є відкритим з дня свого створення. Це відносно новий формат, що використовує хвильовий (wavelet) алгоритм ущільнення. Кінцевий результат

ущільнення порівнюється за своєю якістю з оригінальною сканованою версією. Після багаторазового його збільшення не з'являється растрова мозаїка.

Чорно-білі документи 300 dpi звичайно стискаються до 5–30 Кб. Розмір якісно скануючих сторінок можна прирівняти зі HTML сторінками, які займають 50 Кб. Для чорно-білих сторінок DjVu файли в 10–20 разів менші JPEG і в 5 разів менші GIF. Також DjVu файли в 3–8 разів менші чорно-білих, отриманих із скануючих документів (сканувати кольорові документи в PDF непрактично).

Для кольорових документів, які містять одночасно текст і картинки, файли DjVu в 5–10 разів менші файлів JPEG тієї ж якості. Кольорові сторінки, що сканують у повному кольорі з роздільною здатністю 300 dpi можуть бути стиснутими з 25 Мб до 30–100 Кб.

Формат DjVu може використовувати ряд різних кодерів/декодерів, щоб ущільнити окремі частини чи все зображення сторінки. Фактично, DjVu – чотири методи ущільнення, що об'єднані в одному форматі DjVuPhoto (інакше IW44), DjVuBitonal (інакше JB2), DjVuDocument, BZZ.

DjVuPhoto є трохи старшим, подібний до формату JPEG-2000. Він порівняно з JPEG-2000 у термінах якості зображення й розміру файлу, але кращий у термінах часу рендерінга й вимог до пам'яті. DjVuBitonal кращий, ніж MMR/GroupIV (використовуючий форматами PDF, TIFF і більшістю факсимільних машин) приблизно в 3–10 разів. Він також кращий, ніж новий формат JBIG2 приблизно на 20%.

Однією з основних технологічних властивостей DjVu є здатність відокремити тіло зображення й передній план. Традиційні методи ущільнення зображення придатні для простих фотографій, але вони значно погіршують різкі переходи кольору між суміжними контрастними областями. Відокремлюючи текст від тла, DjVu може зберегти текст із високою роздільною здатністю (у такий спосіб зберігаючи гострі грані й максимізуючи чіткість), у той же час ущільнюючи тло з більш низькою роздільною здатністю (використовуючи методику на основі хвильового алгоритму).

Розмір документів настільки малий, що вони можуть відправлятися електронною поштою як вкладення. Типова журнальна сторінка без графіки займає у форматі DjVu 30-40 Кб, сторінка з одним-двома малюнками 60-100 Кб.

Використовуючи DjVu Browser plug-in, можна плавно змінювати розмір зображення від 5% до 1000%, а також розділяти текстовий і графічний рівні документа. Документи формату DjVu завантажуються швидко, можливий перегляд на всіх платформах без проблем сумісності через шрифти, кольори, й т.д. Крім сканування документів DjVu можна застосовувати до документів, створених іншими програмами, наприклад Adobe PostScript або PDF. У цьому випадку розмір файлу варіюється від 15 до 20 Кб за сторінку 300 dpi.

Відносно новим графічний файловим форматом є FPX (FlashPix), який набув великого успіху. FPX є неоціненним для тих, хто займається видавничими системами, скануванням, цифровою обробкою зображень, фотографією, HTML, пересиланням файлів зображень телефонною лінією. Файли FlashPix містять високоякісні фотографії з великою глибиною кольору та можуть бути швидко оброблені середнім мультимедіа-ПК або Macintosh.

Формат містить зображення відразу в декількох роздільних здатностях. По замовчуванню в браузер завантажується версія з найбільш низьким із них. Це надає користувачу можливість самому вибирати потрібне розширення.

FPX-формат підтримує напівтонові та повнокольорові RGB-зображення, але не підтримує альфа-канали, кольорові профілі та обтравочні контури.

Файл FlashPix побудований інакше, ніж інші файли зображень. Він зберігає не тільки зображення оригіналу, а й копії з нижчою роздільною здатністю. Завдяки цьому графічні редактори можуть у процесі обробки зображень швидко обирати найкращу роздільну здатність, не опитуючи оригінальне зображення. Якщо кольорове зображення оброблюють на екрані, в копіях із низькою роздільною здатністю здійснюються зміни.

Розширення імені файлу *MIX означає, що формати тексту та FlashPix перемішані.

RAW (у перекладі з англ. raw – сирий) призначений для передавання документів між програмами та комп'ютерними платформами тому він містить необроблені (чи оброблені в мінімальній ступені) дані, що дозволяє уникнути втрат інформації, і не має чіткої специфікації. У таких файлах міститься повна інформація про зберігаючий сигнал, і вона може бути нестисненою, стисненою без втрат, чи стисненою з втратами.

Цей формат знайшов широке застосування у таких сферах, як: цифрова фотографія – під форматом RAW розуміються дані, отримані напряму з матриці без обробки; 2. обробка звуку – у цьому випадку під RAW розуміються звукові дані без стиснення і заголовків; у форматі RAW подані дані з приладів, які не пройшли перетворення в фізичні величини, наприклад, мілівольти під час опиту електронного термометра.

Для роботи з Video Toaster та обміну файлами з комп'ютерами Commodore Amiga використовується формат IFF (Amiga Interchange File Format), який підтримується багатьма графічними програмами для IBM-сумісних ПК.

Графічний файловий формат MAC (MacPaint) є основним форматом у системах підготування документації та графічних редакторах на ПК фірми Apple (Macintosh, Lisa). Цей формат, як правило, використовують для передавання бітових зображень у програми системи Macintosh (розмірність зображень не повинна перевищувати 576´720 пікселів).

Формат файлів PixelPaint дозволяє відкривати документи в програмі PixelPaint на комп'ютерах Macintosh. Його використовують для зображень у градаціях сірого чи з індексованими кольорами.

Для обміну файлами з графічними робочими станціями було розроблено формат PIXAR, який призначений для професійних програм 3D-моделювання та анімації.

Популярне ім’я графічного формату «Targa» насправді відноситься до TGA-формату. Перша редакція TGA-формату має назву «Original TGA format» (Оригінальний TGA-формат) і друга – «New TGA Format» (новий TGA-формат). Цей формат був розроблений для підтримки своїх відеокарт фірмою Truevision. Кілька компаній час від часу перекуповували права на цей формат одна в одної. З файлами TGA працювали адаптери Targa, True Vista й інші. Формат дозволяє зберігати зображення з глибиною кольору до 32 біт і при цьому швидко читається й розпаковується, має спеціальну опцію Bottom-up orientation, тобто завантаження файлу не «зверху вниз», а «знизу нагору».

Разом із стандартними трьома RGB-каналами TGA-файл має додатковий альфа-канал для уявлення інформації про прозорість зображення. Формат використовується програмними продуктами багатьох відомих у світі комп’ютерної графіки фірм, серед яких Adobe і Autodesk.

У TGA використовується алгоритм ущільнення RLE, який відрізняється від варіанта алгоритму, використаного у форматі PCX. Другий варіант алгоритму RLE має більший максимальний ступінь компресії для деяких растрів і не так сильно збільшує в розмірах вихідний файл у найгіршому випадку.

Формат TGA 2.0 повторює первісний формат, доповнюючи його новими елементами, а саме: директорія і область розробника, область розширення, таблиця колірної корекції, етикетка (зменшене зображення), таблиця рядків розгорнення, кінцівка.

Формат TGA може зберігати не тільки стандартні чорно-білі та RGB-зображення, але й зображення, засновані на індексній палітрі. У форматі TGA записуються й Alpha-канали, через це формат TGA одержав поширення під час передачі телевізійних зображень.

Група розроблювачів, яка займається обробкою супутникових фотографій, створила у компанії LuraTech формат LWF (LuraTech Wavelet format). У ньому LWF використовується алгоритм ущільнення з втратами (на основі хвильового алгоритму), який часто дає кращі за JPEG результати. Формат LWF відрізняється тим, що під час ущільнення можна заздалегідь встановити розмір майбутнього файлу.

Програма LuraWave SmartCompress v2.2 (Shareware) може забезпечити високий ступінь ущільнення за досить високої якості. За ущільнення, наприклад, у 60 разів, втрати менші, ніж за такого ж ущільнення в JPEG. Зображення практично не втрачає деталізації, а набуває розмитого вигляду із підкресленими контурами, тобто якість сприйняття майже не погіршується. За збільшення коефіцієнта ущільнення не

проступає «мозаїка» квадратів, як у JPEG. Практично, з'являється можливість інтерактивної роботи із сервером. Програма існує для всіх ОС. Цей формат використовується нечасто, хоча LWF іноді ущільнює майже в 1,5 рази більше за інші формати.

Питання для самоперевірки

1. Наведіть приклади застосування растрової графіки
2. Що таке піксель?
3. Дайте визначення растру.
4. Яким чином вимірюється розмір растру?
5. Що таке роздільна здатність?
6. Вкажіть основні типи растрів.
7. Вкажіть найбільш поширені редактори растрової графіки.
8. Які існують класи форматів графічних файлів?
9. Призначення растрових графічних форматів.
10. Вкажіть формати файлів, що можуть зберігати зображення у растровому вигляді.

Список рекомендованої літератури

1. Блінова Т.О., Порєв В.М. Комп'ютерна графіка / За ред. В.М. Горєва. – К.: Видавництво «Юніор», 2004. – С. 186–228.
2. Васильев В.Е. Компьютерная графика: Учебное пособие. – Санкт-Петербург, 2004. – 96 с.
3. Веселовська Г.В., Ходаков В.Є., Веселовський В.М. Основи комп'ютерної графіки: Навчальний посібник / Під ред. В.Є.Ходакова. – Херсон: ОЛДІ-плюс, 2004. – С.112–123; 260–269
2. Горобець С.М. Основи комп'ютерної графіки: Навч.пос. /За ред. М.В.Лемківського. – К.: Центр навчальної літератури, 2006. – С.66–69
3. Федік Л.Ю. Основні формати графічних даних у мережі World Wide Web // Наукові нотатки. Міжвузівський збірник (за напрямом «Інженерна механіка»). – Луцьк, 2008 – С. 306–315.

РОЗДІЛ 3. ВЕКТОРНА ГРАФІКА

3.1. Основні відомості про векторну графіку

Програмні засоби для роботи з векторною графікою призначені, в першу чергу, для створення ілюстрацій і меншою ступінню для їх обробки.

Принципи векторної графіки базуються на відмінному від піксельної графіки математичному апараті і мають метою побудувати лінійні контури, складені з елементарних кривих, які описуються математичними рівняннями.

Векторна графіка *– це вид комп'ютерної графіки, в якому зображення подається у вигляді сукупності окремих об'єктів, описаних математично.*

У растровій графіці основним елементом зображення є точка, а у векторній графіці – лінія (при цьому не важливо, пряма це лінія чи крива).

Звичайно, в растровій графіці теж існують лінії, проте там вони розглядаються як комбінації точок. Для кожної точки лінії в растровій графіці відводиться одна чи декілька комірок пам'яті (чим більше кольорів можуть мати точки, тим більше комірок їм виділяється). Відповідно, чим довша растрова лінія, тим більше пам'яті вона займає. У векторній графіці об'єм пам'яті, який займає лінія, не залежить від розмірів лінії, оскільки лінія представляється у вигляді формули, а точніше, у вигляді декількох параметрів. Щоб ми не робили з цією лінією, змінюються тільки її параметри, які зберігаються в комірках пам'яті. Кількість же комірок залишається незмінною.

Отже, лінія – це елементарний об'єкт векторної графіки. Найпростіші об'єкти об'єднуються в більш складніші. Наприклад, об'єкт чотирикутник можна розглядати як чотири зв'язані лінії.

Під час редагування елементів векторної графіки змінюються параметри прямих і вигнутих ліній, які описують форму цих елементів. Можна переносити елементи, змінювати їх розмір, форму і колір, але це не відіб'ється на якості їх віртуального уявлення. Векторна графіка не залежить від роздільної здатності, тобто може бути показана в різноманітних вихідних пристосуваннях і з різною роздільною здатністю без втрати якості (рис.3.1).

Векторний формат більш компактний, проте він зовсім непридатний для збереження фотографічних зображень. У цьому форматі задавати їх математично було б дуже громіздко. А ось рисунки і креслення значно зручніше і практичніше зробити саме у векторному вигляді.

Основними перевагами векторної графіки є: зміна масштабу без втрати якості і практично без збільшення розмірів вихідного файлу; велика точність (до сотої частки мікрона); невеликий розмір файлу в порівнянні з растровими зображеннями; висока якість друку; відсутність проблем із експортом векторного зображення в растрове; можливість редагування кожного елемента зображення окремо.

Рис. 3.1. Приклади векторної графіки

Основними недоліками векторної графіки є такі: складність експорту з растрового формату у векторний; неможливість застосування численної бібліотеки ефектів, які застосовуються під час роботи з растровими зображеннями.

3.2. Математичні основи векторної графіки

В основі векторної графіки лежать математичні уявлення про властивості геометричних фігур. Оскільки, найпростішим об'єктом векторної графіки є лінія, то в основі векторної графіки лежить, перш за все, математичне уявлення лінії.

Точка на площині задається двома числами (x,y), які визначають її положення відносно початку координат.

Для задання прямої лінії достатньо двох параметрів. Звично графік прямої лінії описується рівнянням y = kx + b. Знаючи параметри k і b, завжди можна зобразити безкінечну пряму лінію у відомій системі координат.

До кривих другого порядку відносяться: параболи, гіперболи, еліпси, кола і інші лінії, рівняння яких не містять ступенів вище другого.

Відрізняються криві другого порядку тим, що не мають точок перегину. Загальна формула кривої другого порядку може виглядати, наприклад, так:

$$x^2 + a_1 y^2 + a_2 xy + a_3 x + a_4 y + a_5 = 0. \qquad (3.1)$$

П'яти параметрів достатньо для опису безкінечної кривої другого порядку.

Відмінною особливістю кривих третього порядку є те, що вони можуть мати точку перегину.

Криві третього порядку добре відповідають тим лініям, які ми можемо бачити в живій природі, наприклад, лініям вигину людського тіла, тому в якості основних об’єктів векторної графіки застосовують саме такі лінії. Всі прямі і криві другого порядку (наприклад, коло чи еліпс) є окремими випадками кривих третього порядку.

У загальному випадку рівняння кривої третього порядку можна записати так:

$$x^3 + a_1 y^3 + a_2 x^2 y + a_3 x y^2 + a_4 x^2 + a_5 y^2 + a_6 xy + a_7 x + a_8 y + a_9 = 0. \qquad (3.2)$$

Із рівняння видно, що для запису кривої третього порядку достатньо дев’яти параметрів і зображення кривої третього порядку здійснюється за заданими коефіцієнтами її рівняння. Для спрощення цієї процедури у векторних редакторах застосовують не будь-які криві третього порядку, а їх особливий вид, названий кривими Без’є.

Питання про побудову апроксимуючого багаточлена привернуло увагу багатьох математиків. Видатний вчений Сергій Натанович Бернштейн на початку XX століття запропонував доведення теореми Вейєрштрасса за допомогою теорії ймовірності. У цьому випадку необхідний поліном будується в явному вигляді (не параметрично). Саме даний поліном і став основою сплайнових кривих, зокрема NURBS-кривих (рис.3.2) і кривих Без’є (рис. 3.3-3.5).

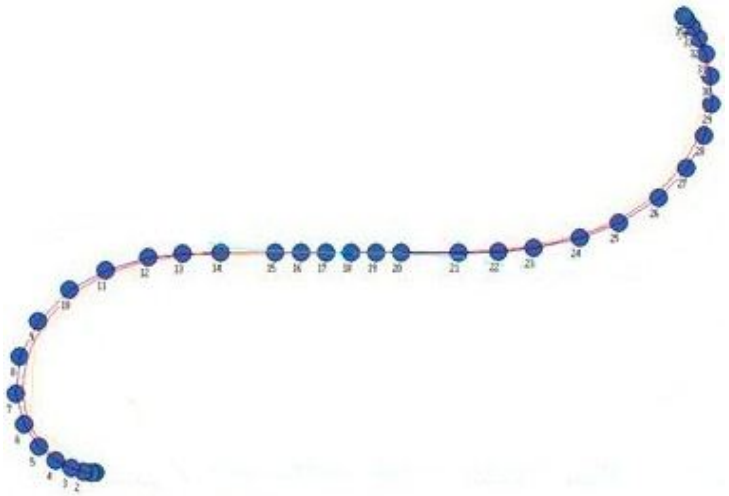


Рис. 3.2. Крива NURBS

Рис. 3.3. Крива першої ступені (пряма)

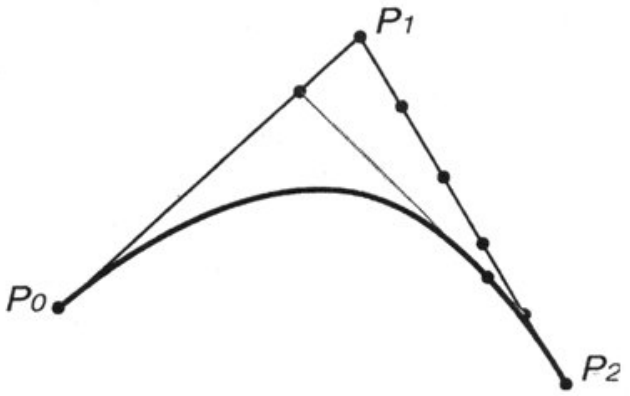


Рис. 3.4. Крива другого ступеня (квадратична крива)

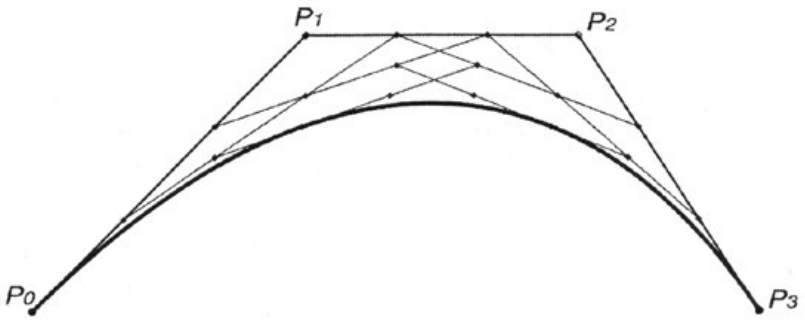


Рис. 3.5. Крива третього ступеня (кубічна крива)

3.3. Криві Без'є

Відрізки кривих Без'є – це окремий випадок відрізків кривих третього порядку. Вони описуються не одинадцятьма параметрами, як довільні відрізки кривих третього порядку, а лише вісьмома, і тому працювати з ними зручніше.

Метод побудови кривих Без'є базується на застосуванні пари дотичних проведених до лінії в точках її закінчення (рис. 3.6).

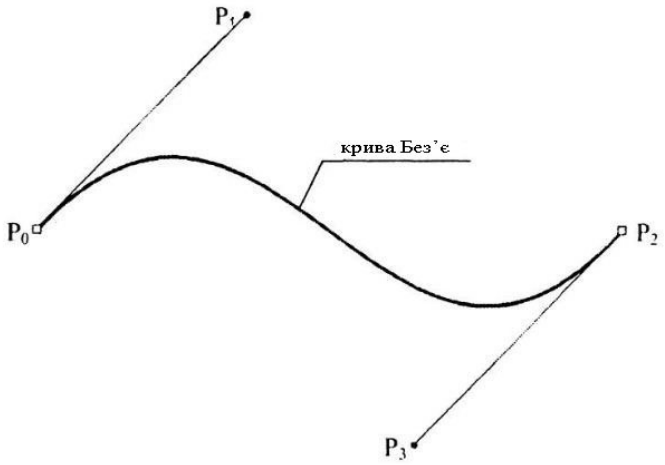


Рис. 3.6. Канонічний вид кривої Без'є

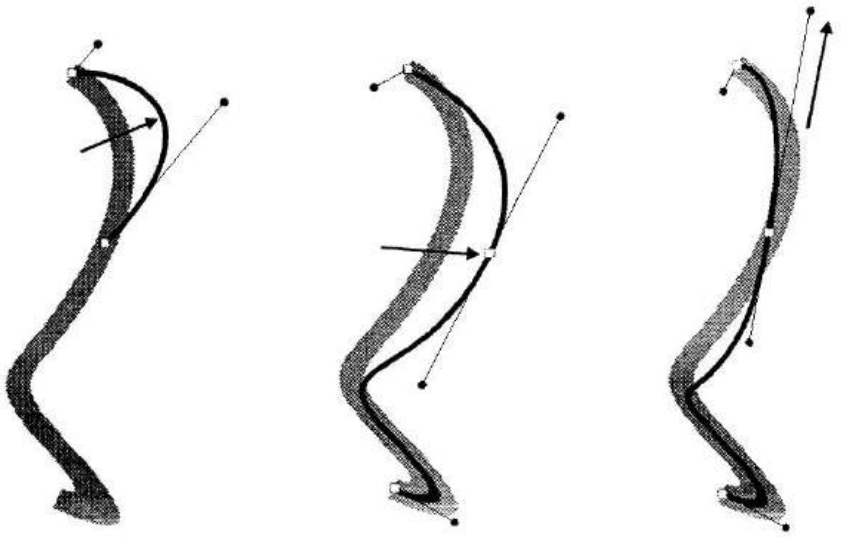
Для побудови кривої потрібно чотири контрольні точки. Проте крива фізично проходить тільки через дві з них, що отримали назву опорних. Одна з цих точок називається початковою (start point), а інша – кінцевою (end point). Дві точки залишаються в стороні, вони отримали назву керуючих (control point). Для того, щоб їх не «втратити», в програмах векторної графіки керуючі точки з'єднуються з опорними точками лінією.

Для того, щоб отримати величезну різноманітність форм, із яких можна скласти об'єкт будь-якої складності потрібно змінити форму канонічної кривої Без'є.

У програмах векторної графіки існує єдиний спосіб зміни форми кривих – це інтерактивне переміщення опорних і керуючих точок (рис. 3.7). Якщо переміщуються початкова чи кінцева точки, то крива стане відповідним чином змінюватися (витягуватися чи стискуватися як пружна резинка). Переміщення керуючих точок змінює кривизну відповідної частини кривої Без'є.

Таким чином, за допомогою переміщення цих чотирьох точок отримують необмежену кількість форм кривих Без'є, яка може бути всього-навсього одним окремим сегментом складного векторного контуру.

У кожному сегменті можна добавляти опорні точки, які теж дозволяють змінювати форму кривої. Добавляння нових опорних точок у межах одного сегмента кривої не протирічить тій умові, що окремі криві з'єднуються в ланцюг. Просто крива Без'є добавляється не до кінця контуру, а розміщується всередині вже існуючого контуру.



а) б) в)

Рис. 3.7. Способи зміни форми кривих: а) переміщення сегмента, б) переміщення опорної точки, в) переміщення керуючої точки

Взагалі будь-яка конструкція (векторний контур або векторна форма) створюється з векторних сегментів, кожен з яких ідентичний окремій елементарній кривій Без'є.

Звідси слідує, що між ними утворюються з'єднувальні точки, які деколи називаються вузлами (наприклад, nodes – у графічному редакторі Corel Draw).

Для підтримання співвідношення між елементарними сегментами існують різні типи опорних точок.

Існують такі типи опорних точок:

1) точки перегину,

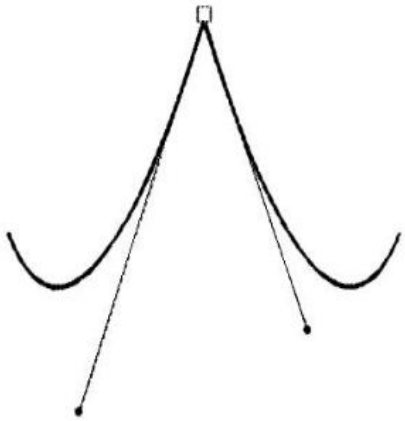
2) гладка опорна точка,

3) симетрична опорна точка,

4) тангенціальна опорна точка.

Точки перегину. Цей тип опорної точки, який з'єднує два сегменти, забезпечує незалежність керуючих точок у напрямку і довжині один від одного. Такий стан сегментів називається злам (рис. 3.8).

Гладка опорна точка. Кутове зчленування сегментів (вигин**)** далеко не завжди вигідне. Наприклад, для створення кола необхідно забезпечити з'єднання, яке в кресленні і в геометрії називають гладким спряженням, коли одна крива плавно переходить у іншу. Таке зчленування забезпечує точка перегину (smooth) (рис. 3.9).



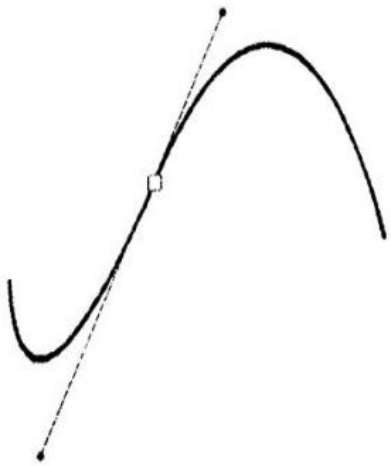


Рис. 3.8. Приклад точки зламу Рис. 3.9. Приклад точки перегину

Симетрична опорна точка. У програмах CorelDRAW передбачений підвид гладкого спряження, який називається симетричний вузол (symm від слова «symmetcal») (рис. 3.10). Сутність його в тому, що керуючі лінії фіксують не тільки по напрямку, але й по величині (довжина направляючих завжди однакова). Якщо одну з них збільшувати чи зменшувати, друга буде синхронно повторювати цю дію.

Тангенціальна опорна точка. У свою чергу, в програмі FreeHand у окремий вид опорних точок виділений випадок гладкого спряженням прямолінійного і

криволінійного сегментів (рис. 3.11). Така точка отримала назву тангенціальної (connecter point). Під час виділення така точка позначається трикутником.

Зміст цієї точки в тому, що для того, щоб криволінійний сегмент гладко з'єднувався з прямою лінією, дотична криволінійного сегмента повинна співпадати з продовженням прямого сегмента. Тому керуюча точка криволінійного сегмента спроможна рухатися тільки вздовж цієї дотичної 3.12–3.19.

Типи опорних точок у різних векторних програмах подано в табл. 3.1.

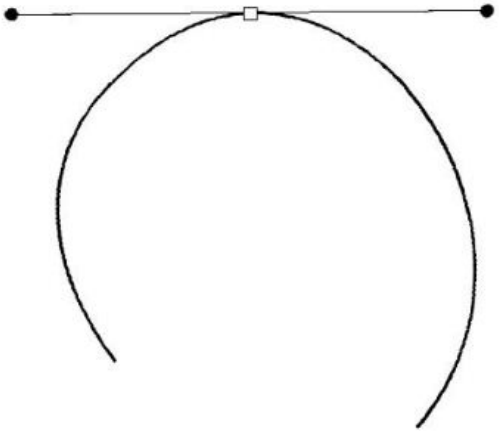


Рис. 3.10. Симетрична опорна точка

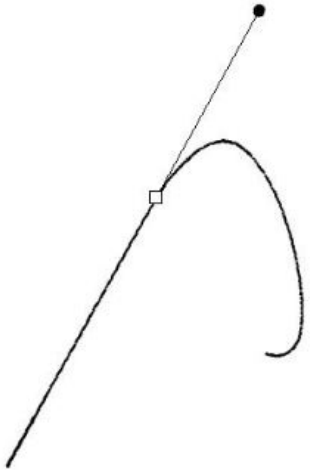


Рис. 3.11. Тангенціальна опорна точка

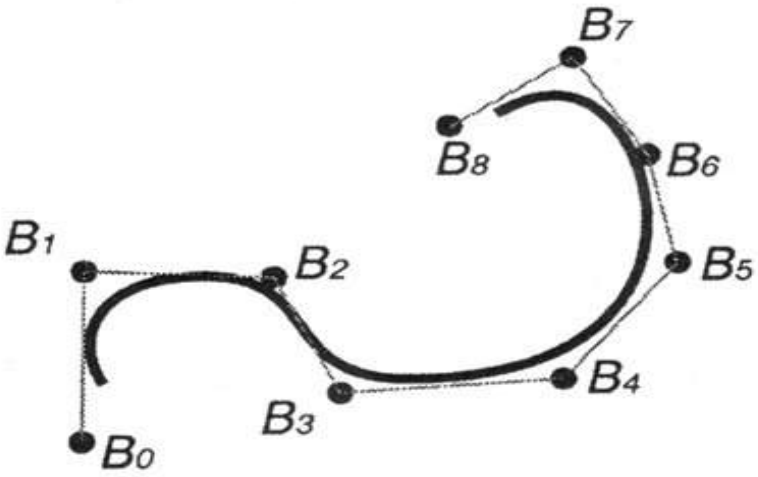


Рис. 3.12. NURBS–крива. Контрольні точки

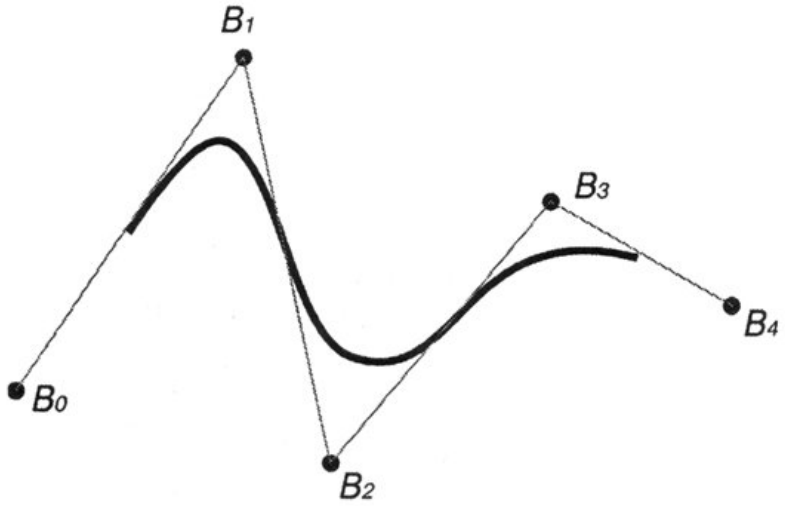


Рис. 3.13. NURBS–крива з однорідним вузловим вектором

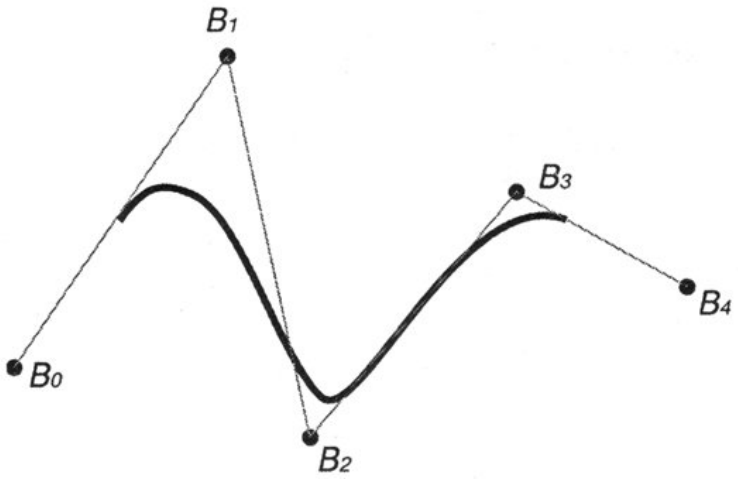


Рис. 3.14. NURBS–крива з неоднорідним вузловим вектором

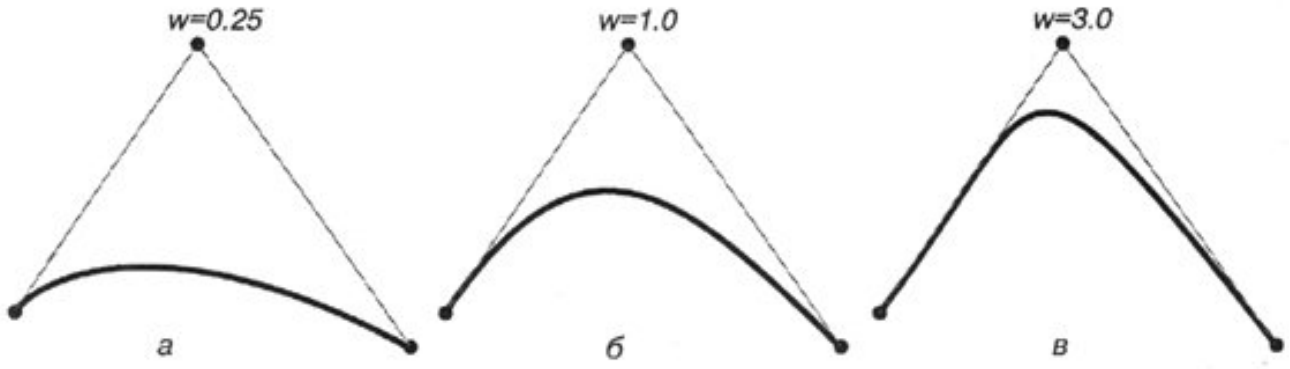


Рис. 3.15. Зміна форми кривої при зміні ваги контрольної точки

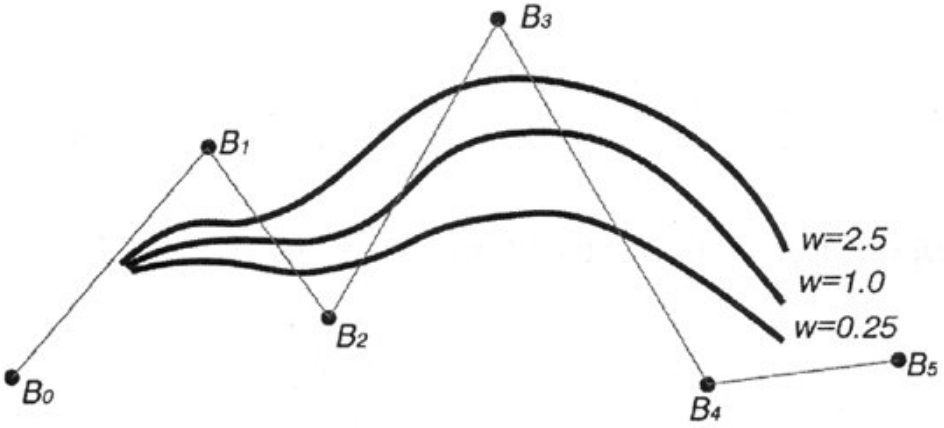


Рис. 3.16. NURBS–криві з різними вагами центральної контрольної точки

Таблиця 3.1. Типи опорних точок у різних векторних програмах

Тип опорної точки	Adobe Illustrator	Macromedia FreeHand	CorelDRAW
Кутова	Corner anchor point	Corner point	Cups node
Гладка	Smooth anchor point	Curve point	Smooth node
Тангенціальна	-	Connector point	-
Симетрична	-	-	Summ node

Види кривих Без'є

Крива Без'є була розроблена математиком *П*'єром Без'є. Криві та поверхні Без'є були використані у 60-х роках компанією Рено для комп'ютерного проектування форми кузовів автомобілів. У теперішній час вони широко використовуються в комп'ютерній графіці.

Криві Без'є описуються в параметричній формі таким чином: $x = P_x(t)$, $y = P_y(t)$. При цьому значення t виступає як параметр, якому відповідають координати окремої точки лінії. Параметрична форма опису може бути більш зручною для деяких кривих, ніж задання у вигляді функції f(x) тому, що функція f(x) може бути набагато складнішою, ніж та $P_x(t)$ та $P_x(t)$ крім того f(x) може бути неоднозначною.

Розглянемо криві Без'є, класифікуючи їх за значеннями *m*.

m = 1 (по двох точках)

Крива вироджується у відрізок прямої лінії, який визначається кінцевими точками P_0 і P_1 як показано на рис 3.17:

$$P(t) = (1 - t_0)P_0 + tP_1.$$

m = 2 (по трьох точках, рис. 3.18):

$$P(t) = (1 - t)^2 P_0 + 2t(1 - t)P_1 + t^2 P_2.$$

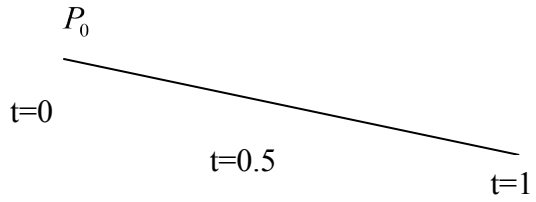


Рис. 3.17. Крива Без'є при m=1

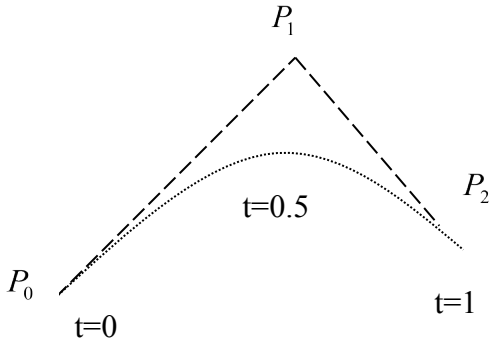


Рис. 3.18. Крива Без'є (m=2)

Крива Без'є при m=3 (по чотирьох точках, кубічна, (рис 3.19), використовується досить часто, особливо в сплайнових кривих: $P(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t(1-t)^2 P_2 + t^3 P_3$.

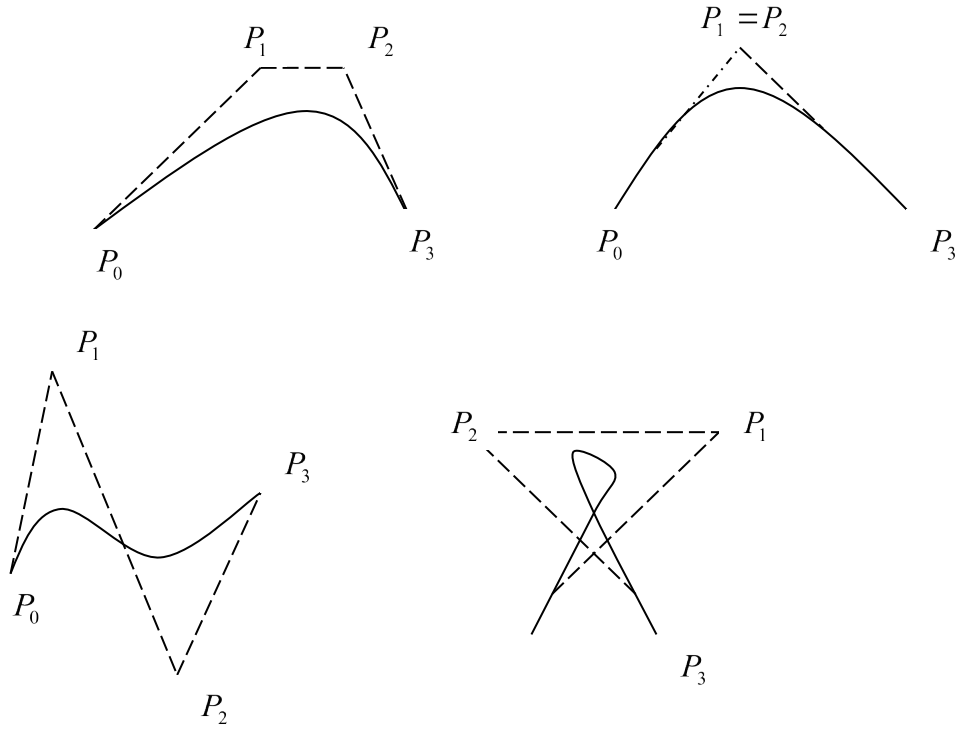


Рис. 3.19. Кубічні криві Без'є (m=3)

3.4. Редактори векторної графіки

Найбільш поширеними редакторами векторної графіки є: CorelDraw, Adobe Illustrator, Macromedia FreeHand, Xara Xtreme, Adobe Flash.

Безперечними фаворитами є редактори CorelDraw і Adobe Illustrator.

CorelDRAW – це векторний графічний редактор, розроблений канадською корпорацією Corel. Популярність CorelDraw пояснюється великим набором засобів створення і редагування графічних образів, зручним інтерфейсом і високою якістю одержуваних зображень. З його допомогою можна створювати як прості контурні

малюнки, так і ефективні ілюстрації з вражаючим уяву переливом фарб і приголомшливими ефектами.

Хоча CorelDraw призначений для обробки векторної графіки, він має в своєму розпорядженні потужні засоби для роботи з растровими малюнками, дозволяє вставляти растрові малюнки в документ. При цьому кожен растровий малюнок є окремим об'єктом і його можна редагувати незалежно від інших об'єктів.

Ця програма також має засоби для роботи з текстом. Завдяки безлічі видів форматування фігурного і простого тексту як об'єкта, в редакторі можна створювати малюнки з текстовим супроводом. При цьому фігурний текст дозволяє виконувати над ним операції, властиві векторним об'єктам. CorelDraw має стандартний інтерфейс, характерний для всіх програм, що працюють під керуванням MS Windows. У той же час на екрані є ряд елементів, характерних тільки для графічних програм. На екрані запуску програми відображаються два вікна: вікно самої програми і вікно документа.

Під рядком меню знаходяться дві основні панелі: стандартна панель і панель властивостей. Перша панель має дванадцять командних панелей, які характерні для багатьох програм, що працюють під керуванням MS Windows. Розміщені на ній значки забезпечують швидкий доступ до стандартних команд. Панель властивостей є контекстно-залежною панеллю. Це означає, що її значки і списки динамічно змінюються в залежності від режиму роботи, активного інструменту і типу виділеного об'єкта. Таким чином забезпечується доступ до найбільш важливих команд, пов'язаних з обраним об'єктом або інструментом.

Основними сферами застосування програми CorelDRAW є: поліграфія і веб-дизайн. У професійній поліграфічній діяльності редактор застосовується для створення листівок, плакатів, візиток, буклетів, календарів та іншої продукції, для веб-дизайну у програму включені спеціальні команди експорту графіки та інструмент "Керування кольором", який дозволяє вибирати вже готові налаштування для Web, і створювати свої.

Пакет Illustrator послужив моделлю, яка лягла в основу всіх представлених у цьому огляді програм векторної графіки. Macromedia FreeHand краще виконує імпортування файлів EPS і AI і при цьому забезпечує високу точність передачі кольору в форматі CMYK, якою завжди відрізнявся Illustrator. CorelDraw вже давно передбачила у своїх пакетах градієнтне зафарбовування, істинні шари, булеві операції та спеціальні ефекти.

Редактор ACanvas 5 має у своєму розпорядженні засоби редагування растрових зображень на рівні пікселів, з робочою площею майже 140 м^2, у той час як для Illustrator ця площа становить усього 0,2 м^2.

Micrografx Designer надає чудовий інструментарій для малювання, інтегрується з Windows і Microsoft Office та містить засоби для підготовки технічних ілюстрацій.

CorelXara забезпечує справжню прозорість для векторних об'єктів і можливість вбудовування растрових зображень. У свою чергу Fractal Design Expression за допомогою інструменту Skeletal Strokes дозволяє отримувати самі найнезвичайніши ефекти і видозмінювати зображення. Порівняно обмежений набір засобів пакету Illustrator не означає, що він простий у застосуванні. І хоча якість кольорового друку залишається найсильнішою стороною пакета Illustrator, можливості обробки кольорів CMYK у FreeHand також не гірші. Крім того, одна і та ж версія FreeHand може працювати в середовищі як для Windows, так і Mac. Проте швидкість промальовування низька і значно поступається швидкості Xara Xtreme та FreeHand. Напевно, саме тому картографи та інші фахівці, що працюють з об'ємними зображеннями, насиченими деталями і містять до 200 шарів, віддають перевагу іншим редакторам.

Програма Adobe Illustrator створювалася як редактор векторної графіки, проте основними споживачами програми є художники-дизайнери. Зображення, які створюються в програмі Adobe Illustrator, легко інтегруються в мультимедійні програми (наприклад, Adobe Premiere, Adobe After Effects та ін), тому вона зручна для використання фахівців з виробництва мультимедійних продуктів. Програма Adobe Illustrator CS4 володіє новими можливостями, зокрема: використання декількох монтажних областей дозволяє зберігати, експортувати та друкувати монтажні області разом або працювати тільки з окремими областями, а інструмент "Кисть-клякса" створювати одиночні чіткі векторні фігури. Крім того існує робота з градієнтами, виконання перевірки створеного графічного об'єкта. Illustrator CS4 підтримує перевірку для обох типів червоно-зеленої колірної сліпоти – протанопія і дейтеранопія.

Пакет Macromedia FreeHand вражає бездоганною якістю виведення на екран, чотириколірним друком CMYK і наявністю декількох форматів для Web. FreeHand завжди відображає кольори так само, як вони будуть виглядати під час друку. Програма дозволяє відбирати кольори з кількох бібліотек, у тому числі Pantone і Hexachrome для друку, і з палітри Web, оптимізованою як для Mac, так і PC. Інструментарій FreeHand для малювання і роботи з текстом відповідає необхідним вимогам, але трохи обмежений. У інтерфейсі FreeHand віддано перевагу редагуванню вузлів, а не редагування об'єкта в цілому. Кожна з операцій масштабування, повороту, дзеркального відображення і деформації, які виконуються в CorelDraw маніпуляціями в робочому вікні об'єкта - вимагає окремого інструменту з набору інструментарію FreeHand. Під час вибору об'єкта, його точки (вузли) завжди доступні для безпосереднього редагування, але це тільки вузли і траєкторії об'єкта, а не його "закінчений" вигляд.

Програма підтримує великий набір векторних ефектів і дозволяє підключати додаткові розширення (Xtras). Цікавою і важливою особливістю FreeHand під час

роботи над великими проектами є можливість глобальних змін документа. FreeHand дозволяє глобально змінити графічні об'єкти, виділити об'єкти із заданими властивостями для подальшого перетворення. Основними властивостями, за якими ведеться пошук, заміна чи виділення об'єктів, є: колір, товщина контуру чи число складових його точок, шрифт, кут повороту. Багато з цих характеристик допускають деталізовану, уточнену настройку. Областю пошуку може бути виділена область, сторінка чи весь документ. Крім того, редактор дуже невимогливий до ресурсів: для роботи достатньо 64 Мб оперативної пам'яті і 70-100 Мб на жорсткому диску. FreeHand працює в Windows 98, на відміну від останніх версій CorelDRAW і Adobe Illustrator, які не підтримують дану ОС. Особливості Macromedia FreeHand 11.0 дають можливість швидко спланувати та організувати багатосторінкові сайти та проекти, а також підготувати інтерактивні презентації, розкадровки та web-сторінки з навігацією, не виходячи з робочого середовища FreeHand.

До складу редактора входять живі ефекти, тобто широкий набір засобів, що включає як растрові, так і векторні ефекти, з можливістю їх застосування до всіх або деяких вибраних об'єктів.

Macromedia FreeHand дозволяє редагувати примітиви в режимі реального часу, використання нових видів градієнта (конусний і прямокутний), використання шаблонів сторінок і фону, а також пошуку і заміни в графічних зображеннях, що дозволяє знизити витрати часу на створення та редагування.

Графічний редактор Xara Xtreme за функціональними можливостями практично не поступається іншим професійним програмам. Крім того, після інсталяції Xara X займає лише 32 Мб дискового простору.

Слід зазначити невибагливість редактора до швидкодії системи, що дозволяє використовувати його навіть на комп'ютерах з 32 Мб оперативної пам'яті. В останніх версіях програми були зроблені спроби змінити редактор так, щоб він підходив розробникам сайтів у Інтернеті – в редактор були додані інструменти для створення різних ефектів таких, як: тінь і обсяг, сценарії Javascript для кнопок, створення і обробка анімації у форматі.

Xara Xtreme спочатку розроблялася для Microsoft Windows. У даний час комерційна версія продукту існує тільки для Windows. Однак крім комерційної версії, існує і вільна версія Xara Xtreme for Linux (або Xara Xtreme Linux Edition).

Останньою версією редактора Xara Xtreme для Microsoft Window є Xara Xtreme 4 – випущена в квітні 2008. За інформацією з офіційного сайту Xara Xtreme забезпечує до 10 разів більш високу швидкість векторного рендерингу, ніж Illustrator CS3. Основними поліпшеннями в порівнянні з попередніми версіями є розширені можливості під час роботи з текстом, поліпшена інтеграція з обробником растрових зображень, а також експорт веб-сторінок. Версія Xara Xtreme 4 відрізняється підвищеною в 10 разів швидкістю обробки растрових зображень, що привело до

скорочення розміру проміжних файлів у 10 разів у порівнянні з іншими графічними пакетами.

Інструменти верстки веб-сторінок у візуальному режимі WYSIWYG дозволяють створювати анімаційні ефекти, задавати обтікання фігур текстом, а також включати в сторінку спливаючі вікна і ін.

Xara Xtreme for Linux – це версія програми Xara Xtreme з відкритим вихідним кодом для GNU/Linux, FreeBSD і Mac OS X. Раніше ця версія називалася Xara Xtreme LX або просто Xara LX. Xara Xtreme for Linux у даний час розробляється. Зараз на офіційному сайті Xara Xtreme for Linux доступна версія 0.7, в якій реалізована частина функцій, присутніх у комерційній версії для Windows.

У версії 0.7 покращена робота інструменту введення текст, додана можливість через лінійку вказувати відступи, імпорт файлів перетягуванням з файлових менеджерів. Завдяки інтеграції з ImageMagick підтримується ще більше форматів: TIFF, BMP, PICT, PPM, PDF (як растрові зображення), PSD і XPM, доданий новий фільтр імпорту SVG, поки що неповнофункціональний – обмежена підтримка градієнтних заливок, немає підтримки тексту, можливий імпорт з файлів Adobe Illustrator раніших версій ніж версія Adobe Illustrator 8, покращено друк: малюнки розміщуються на сторінці, помилка експорту в Postscript виправлена.

Xara Xtreme має свої недоліки: не має встановленого редактора тексту та функції створення шрифтів, а також спотворення кольорів під час експорту растрового зображення в кольоровій моделі CMYK.

Сфера застосування Adobe Flash набагато ширша, ніж представлених раніше програм, оскільки в першу чергу, це середовище для створення додатків під Flash платформу, і отже її можливості багатші і універсальніші, ніж у звичайного редактора векторної графіки. Програма популярна серед розробників векторних карт, різних анімацій та ігор.

Flash-додатки створюються за допомогою ActionScript (остання версія 3.0) – мови програмування. Якщо програмний код не містить в собі інструкції мови, то під час компіляції в будь-якому випадку генерується деякий базовий код на ActionScript (це можна побачити переглянувши Flash-байткод). Далі відповідно існує базовий кліп на сцені, успадкований від MovieClip класу, який і починає відтворюватися. Середовище Adobe Flash орієнтоване в більшій мірі на дизайнерів, аніматорів. При створенні продукту можна використати медіа, звукові та графічні файли, створювати інтерактивні інтерфейси та повноцінні веб-програми з використанням PHP і XML.

В основі Flash лежить векторний морфінг, тобто плавне «перетікання» одного ключового кадру в інший. Це дозволяє робити досить складні мультиплікаційні сцени, задаючи лише кілька ключових кадрів для кожного персонажу. Flash використовує мову програмування ActionScript, засновану на ECMAScript.

Сфера застосування Adobe Flash дуже популярна серед дизайнерів і художників-ілюстраторів. Є набір легких у застосуванні інструментів редагування графіки, серед яких векторні лінії, можливість заповнити вибрані області текстурами растрового типу, легкий спосіб зміни кривих (за допомогою інструмента «direct selection»). AdobeFlash використовує для об'єктно-орієнтованого програмування (ООР) технологію Actionscript і повністю підтримує мову XML.

3.5. Формати файлів векторної графіки

Векторні графічні формати слугують для збереження зображень у вигляді сукупності геометричних примітивів – ліній, дуг, прямокутників, еліпсів тощо. Вони більш компактні у порівнянні з растровими, проте зовсім непридатні для збереження фотографічних зображень. Однак малюнки і креслення значно зручніше і практичніше зробити саме у векторних форматах.

Формат EPS (Encapsulated PostScript) є стандартом для векторного зображення, яке потрібно встановити в програму макетування сторінок. Його підтримують багато програм верстання та підготовки ілюстрацій: він є найбільш розповсюдженим. Файл EPS являє собою точний опис малюнка програми код PostScript, який буде застосовуватися під час друку. Крім цього, в EPS включене зображення низької роздільної здатності для попереднього перегляду (preview). Preview полегшує ідентифікацію зображень під час верстки, дозволяє візуально контролювати його положення, масштаб і поворот. Цей формат підтримує прозорий білий колір у бітовому режимі.

Формат файлів Shockwave Flash був розроблений фірмою Macromedia. Його використовують для створення цілої сторінки чи більшої її частини. Недоліком Flash є наявність додаткового модуля для його перегляду, який хоча і розповсюджується безкоштовно, проте під час завантаження через мережу виникає проблема в перегляді Flash-заставки, оскільки розмір його становить 0,24 Мб. Проте поява вбудованого модуля вже в 4.5 версії «Коммуникатора» і обіцянки Microsoft зробити аналогічне і в IE 5.0+ може цілком висунути Flash у лідери серед графічних форматів для Internet і дещо змінити саме уявлення про web-дизайн.

Цей формат володіє такими перевагами векторних форматів, як масштабування та відносно невеликий об'єм файлу. Проте він не є чисто графічним. Інструментарій для малювання в програмі Macromedia Flash Direct більше схожий до растрових редакторів, ніж до векторних, а здатність створювати анімації, озвучувати їх, примушувати реагувати на переміщення «миші» і здатність працювати з гіперпосиланнями роблять Flash схожим на мультимедійний формат.

Як окремі елементи сайту Flash-заставки використовуються в основному як інтерактивні банери в деяких рекламних мережах. Такого типу банерам пророкуватимуть велике майбутнє у зв'язку з їх більшою можливістю впливати на користувачів мережі, ніж класичних, виконаних у GIF.

Графічний файловий формат DXF (англ. Drawing eXchange Format) – відкритий формат файлів для обміну двомірної графічної інформації між додатками САПР (для обміну кресленнями та іншими графічними документами в середовищі AutoCAD). Він був представлений в грудні 1982 р. як частина AutoCAD 0,1 і повинен був надати точне представлення даних файлів формату AutoCAD. Незважаючи на вік цього формату та його недоліки, DXF зараз підтримується багатьма програмами як формат обміну даними. Підтримується практично всіма CAD-системами на платформі PC.

Головним недоліком формату DXF є великий об'єм файлів. У середовищі системи AutoCAD для роботи з документами використовується більш компактний формат – DWG через бібліотеки, які надаються некомерційною організацією Open Design Alliance. Однак він є внутрішнім форматом, його не розуміють інші програми. DXF це напевне один з найбільш підтримуючих векторних форматів у світі на сьогоднішній день. Формат має багато характеристик, включаючи: підтримку 3D об'єктів кривих, тексту, вимірювань і т.д. Хоча DXF був розроблений для подання даних у програмі CAD, він широко застосовується багатьма програмами.

А формат MIF-MID використовується в геоінформаційних системах (ГІС). Ці системи описують просторові об'єкти сукупністю метричних і атрибутивних (семантичних) даних. Крім цього 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 з двох файлів – *.MIF та *.MID. Файл із розширенням MIF містить загальний опис і координати вузлових точок об'єктів. Об'єкти можуть бути точковими, лінійними чи площинними. Графічні примітиви: Arc, Ellipse, Line, Pline, Rect, Region, Roundrect та Text. Кольори та стилі показу об'єктів позначаються записами Brush, Pen та Symbol.

Файли MIF та MID є текстовими файлами ASCII, в яких містяться графічні дані (об'єкти), а також може існувати опис таблиці даних, що має атрибутивну інформацію, пов'язану з об'єктами. Самі табличні дані розміщені в файлі з розширенням MID, ім'я якого повинно співпадати з ім'ям відповідного MIF-файла. Кожен запис таблиці займає одну стрічку файла. Поля запису відділяються одне від одного за допомогою символа-розділювача, заданого в MIF-файлі. Позитивною особливістю текстових форматів обміну є відкритість і ясність змісту – дані легко розпізнаються, навіть якщо вже не функціонують програми, які обробляли такі формати. Недолік текстового векторного формату – великий об'єм файлів.

Графічний файловий формат PICT широко використовує програми верстки та обробки графіки в системах Macintosh як загальний формат для обміну документами. Формат PICT був розроблений компанією Apple Computer у 1984 спочатку для програми MacDraw. Багато програм для PC також розуміють цей формат. Він є ефективним для ущільнення зображень з великими однокольоровими областями. Найбільш відчутний ефект отримують у разі ущільнення альфа-каналів, які містять великі області чорного та білого кольору. Під час зберігання RGB-зображень у форматі PICT можна задавати роздільну здатність 16 або 32 біт/піксель. Для зображень у градаціях сірого роздільна здатність може становити 2, 4 або 8 біт/піксель.

PICT–файли в об'єктно-орієнтованому форматі складаються з окремих графічних об'єктів (ліній, дуг, овалів або прямокутників), кожен з яких можна незалежно редагувати й масштабувати. У PICT–файлах можуть також зберігатися растрові зображення. Файли PICT кодуються командами QuickDraw.

Існують дві версії формату PICT. Формат PICT 1 – старий формат, що застосовує тільки 8 кольорів. Він був розроблений для перших чорно-білих Макінтошів. У форматі PICT 2 можуть зберігатись бітмапи будь-якого типу. Об'єктно-орієнтований формат може включати багатокутники, лінії, колірні установки, овали й інші примітиви, а також бітмапи. Файл PICT може включати 1-бітні, 4-бітні, 8-бітні, 16-бітні та 24-бітні кольорові об'єкти. 32-бітний колір також підтримується, але не використовується для CMYK. Замість цього останні 8 бітів використовуються для альфа-каналу. Формат PICT дозволяє зберігати тільки один альфа-канал, крім того, у режимі 16 бітів на піксель альфа-канал не зберігається. Стан опції глибини кольору береться з установок, що збережені під час останнього збереження PICT-файлу.

Для бітмапів використовується ущільнення RLE. Якщо встановлений QuickTime V2.0 чи більш пізній, файл PICT може містити бітмапи, ущільнені за стандартом JPEG. QuickTime постачається з програмами для ущільнення файлів PICT за алгоритмом JPEG чи будь-яким іншим компресором QuickTime. Будь-яка програма, що використовує PICT на Macintosh з інстальованим QuickTime, може декомпресувати такі файли.

Формат CDR використовується програмою CorelDraw, яка на сьогоднішній день є однією з найпопулярніших серед програм, що дозволяють малювати векторні малюнки. Формат CDR дозволяє записувати векторну й растрову графіку, текст. У процесі випуску нових версій формат отримав серйозні зміни (починаючи з четвертої версії, файл у форматі CDR може мати кілька сторінок, починаючи із сьомої версії, підтримується технологія OPI тощо). Через це виникло кілька прикрих незручностей, наприклад, погана сумісність файлів. Але, починаючи з восьмої версії, цей формат не викликає у користувачів будь-яких серйозних нарікань. У формату CDR є

незаперечна перевага — можливість читати ушкоджені файли. Програма, знайшовши ушкоджений об’єкт, просто пропускає його.

Отже, основними вимогами до графічних форматів є економність і інформативність. Векторні формати складаються або зі списку примітивів, або містять у собі набір інструкцій, команд для побудови примітивів. Векторні файли важко застосовувати для зберігання складних фотореалістичних зображень. Не виключається і комбінація цих способів.

Метафайлові формати Під час сумісної роботи декількох програм часто виникає потреба в обміні растровими і векторними графічними даними. Для підтримки обміну такими даними використовуються спеціальні графічні формати – метафайли. Метафайли можуть зберігати інформацію про растрові та/або векторні зображення й про команди візуалізації.

Широке застосування знайшли метафайлові формати даних PDF, AI, PSD, Scitex CT, WMF, EMF, CGM.

PDF (Portable Document Format – формат документів, що переносяться. Це універсальний формат, розроблений фірмою Adobe System. Графічний файловий формат PDF використовує програма Adobe Acrobat – основний засіб електронного розповсюдження документів фірми Adobe на платформах Macintosh, Windows, Unix і Dos. Формат PDF, розроблений на основі мови PostScript Level 2, можна використовувати для подання векторних і растрових (бітових) зображень. Під час роботи з комп’ютерною графікою в Web, не згадують про шрифти у графічних об'єктах. Пов’язано це з тим, що до недавнього часу в HTML-сторінках передбачалася вельми обмежена можливість керування параметрами шрифтів. Але вже в специфікації цієї мови вводиться поняття вбудовування шрифтів, що відкриває можливість використовувати їх не тільки стандартні, а й впроваджувати построковий текст, форми, мультимедіа-вставки, включати механізм електронних підписів для захисту і перевірки справжності документів, імпорту з безлічі сучасних форматів текстових документів, векторних і растрових графічних форматів. У цьому плані PDF- сторінки ідентичні PostScript-сторінкам.

PDF-формат у першу чергу призначений для подання в електронному вигляді поліграфічної продукції, значної кількості сучасного професійного друкарського обладнання. На даний момент це єдиний формат файлів який містить текст, графіку, електронні таблиці, змішані документи, що містять як текст, графіку та документ будь-якого іншого типу, що буде гарантовано прочитаний на будь-якому комп’ютері. Тому цей формат застосовується для створення електронної документації, презентацій, передачі верстки і графіки, розповсюдження більшості супутньої документації. Крім цього PDF дозволяє впроваджувати необхідні шрифти, підтримує RGB, CMYK.

Для досягнення мінімального розміру PDF-файлу застосовується компресія, декілька типів стиснення растрової інформації. Причому кожен вид об'єктів стискається за найбільш вигідним для нього алгоритмом. Односторонні файли PDF можуть створювати Photoshop і Illustrator.

PDF-файли створюються шляхом конвертації з PostScript-файлів або функцією експорту програм. Photoshop і Illustrator можуть створювати односторінкові файли PDF. Багатосторінкові PDF можуть створювати InDesign, FreeeHand 7-9, PDFWriter і Acrobat Distiller, які постачаються в пакеті Adobe Acrobat і разом із PageMare'ом – у пакеті Adobe Acrobat разом із PageMare'ом, деякі інші програми. PDFWriter працює як віртуальний принтер. Він не базується на PostScript і не може коректно обробляти графіку. PDFWriter призначений для швидкого застосування простих текстових документів. У нього є недолік із встановленням шрифтів, що присутнє і в Illustrator'а. FreeHand, також, не може впроваджувати шрифти. Для перегляду можна застосовувати офіційну безкоштовну програму Acrobat Reader, а також програми інших розробників. А для створення PDF-документів потрібні shareware-програми Adobe Systems, Acrobat Standard, Acrobat Professional, чи програми сторонніх розробників.

PDF має власний технічний формат для поліграфії: PDF/X-1, PDF/X-3.

Формат PDF сьогодні безумовно, став загальносвітовим стандартом обміну і збереження цифрових документів. Проте він не досконалий, а для роботи з PDF-файлами в Windows необхідно встановлювати додаткове пристосування.

AI (Adobe Illustrator, Adobe AI) – формат, розроблений фірмою Adobe для Macintosh Microsoft Windows і NeXT. Формат зберігає в собі інформацію про векторні форми, колір, шари, які можуть бути не тільки векторними, але й растровими (малюнки, креслення і декоративні надписи), тобто якщо потрібно створити зображення з фотографіями, то можна сміливо імпортувати фотографію в один із шарів Illustrator. Стиснення даних відсутнє. Характеризуються стабільністю і сумісністю з іншими форматами. AI підтримують майже всі програми так чи інакше зв'язані з векторною графікою, включаючи більшість настільних видавничих систем. Цей формат багатьма спеціалістами вважається найкращим посередником під час передачі векторів із одного програмного середовища в інше, наприклад, з PC на ПК Macintosh і назад. Одним із недоліків цього формату є те, що максимальний розмір зображення може бути не більше 3´3 метра.

Програма FreeHand кілька разів переходила з рук в руки і на сьогоднішній день права на неї належать фірмі Macromedia. Як і Adobe Illustrator та CorelDraw, FreeHand працює з векторними і растровими зображеннями. Вона має свій формат – FH7 (останній символ у розширенні файлу вказує на номер версії програми). Цей формат призначений для збереження графічних документів, а також для переносу цих документів у інші середовища. Він близький до векторного формату AI, що

використовується в Adobe Illustrator. Щоб передати готове зображення іншій програмі, звичайно доводиться записати його в більш сумісному форматі, наприклад, EPS.

Зображення можна також зберігати у файлах програм верстки. Наприклад, при роботі з пакетом Adobe PageMaker, коли вставляти зображення в документ, програма запитує чи хочете зберігати зображення цілком. При позитивній відповіді кінцевий файл збільшиться на розмір малюнка (у цьому випадку жоден малюнок не буде втрачено, але іноді це приводить до неякісного виводу на PostScript-пристрої). Якщо обрати другий варіант, то у файлі буде зберігатися тільки зображення для попереднього перегляду і адреса файлу із зображенням. При цьому варіанті макет буде складатися з декількох файлів: основного документа PageMaker – і файлів із зображеннями. Усі вони повинні бути доступні програмі під час якісного виводу на друк, але для попереднього друку можна залишити тільки основний документ. Інша популярна програма верстки – QuarkXPress – допускає тільки другий варіант збереження документів.

Формат PSD розроблений для програми Adobe Photoshop. Він підтримує всі типи зображень від монохромної графіки до кольорового CMYK, дозволяє записувати растрове зображення з багатьма шарами, додатковими колірними каналами й іншою інформацією. Однак цей формат невідомий програмам верстки. Для роботи з ними необхідно зробити спрощену копію файлу в іншому форматі. Починаючи з версії 3.0, Photoshop записує такі файли з компресією, що не позначається на якості зображення при помітному зменшенні розміру, а у Photoshop 4.0 файли стають ще меншими.

Формат PSD сприймається обмеженою кількістю графічних пакетів. Зазвичай, формат PSD використовується, якщо необхідно зберегти створені користувачем альфа-канали, чи зарезервувати шари, що можуть знадобитися пізніше під час редагування зображення.

Графічний файловий формат Scitex Continuous Tone (CT) можна використовувати для кольорових зображень у поданні RGB або CMYK; зображень у градаціях сірого. Його застосовують на комп'ютерах Scitex, які використовуються для високоякісної професійної обробки зображень.

Зображення в форматі Scitex CT є CMYK-файлами (часто великого обсягу), що утворюються під час введення даних за допомогою Scitex-сканерів. Виведення файлів у форматі Scitex CT здійснюється на плівку на растровому пристрої Scitex, який виконує кольороподіл за допомогою запатентованої напівтонової системи Scitex. Оскільки ймовірність появи муару є малою, цю систему часто використовують для виконання професійної роботи з кольором (наприклад, під час розробки рекламних оголошень у центральних журналах).

Файли формату PostScript містять у собі сам документ (те, що існує на сторінках), усі зв'язані файли (растрові й векторні), використані шрифти, а також іншу

інформацію вивідного пристрою: плати кольороподілу, додаткові плати, лінеатуру растра та форму растрової точки для кожної плати тощо. Дані в PostScript-файлі, як правило, записуються у двійковому кодуванні (Binery), яке приблизно вдвічі економніше за кодування ASCII. Але під час передачі файлів через мережі, при кросплатформному 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 робився чи які шрифти були використані. Однак навіть, якщо зроблені вірні установки у вікні друку, можуть виникнути проблеми, пов'язані з некоректним перекладом графічної мови програми, що використовується на мову PostScript (наприклад, з впровадженням інформації про невикористовуючі шрифти). Найкоректніші PS-файли створюють програми Adobe. Сказане відноситься до форматів, заснованих мовою PostScript, а саме: EPS і PDF.

Формат WMF (Windows Metafile) був розроблений компанією Microsoft для 16-розрядних версій Windows Ін. і можна сказати, є її рідним форматом. Він слугує для передачі векторів через буфер обміну (Clipboard). У цьому випадку ескіз також стає векторним і майже не відрізняється від основної копії. Такі файли можуть бути відредаговані в CorelDraw. Однак вони мають меншу сумісність. Формат розуміється практично всіма програмами Windows, так чи інакше пов’язаними з векторною графікою. Проте здається, що цей формат досить простий і універсальний, однак користуватися форматом WMF потрібно тільки в крайньому випадку для передачі «голих» векторів. WMF викривляє колір, не може зберігати ряд параметрів, які можуть бути присвоєні об’єктам у різних векторних редакторах, не може містити растрові об’єкти, не розуміється дуже багатьма програмами на Macintosh.

Можливості метафайла Windows значною мірою обмежені через залежність від пристроїв і програм. Метафайл не має інформації про розмір зображення, початкову роздільну здатність чи стан палітри пристрою, на якому записувалось це зображення, існують і інші обмеження, що накладаються GDI.

Формат файлів EMF (Microsoft Enhanced Metafile) являє собою оновлену версію формату WMF. Формат EMF підтримує 32-розрядну систему координат і 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 й підтримує нові функції GDI, отримує все більшу популярність. Оскільки, більш економно витрачає дисковий простір. Малюнок, збережений в форматі EMF, займає в середньому в два рази менше місця, ніж відповідно WMF-файлі.

Метафайл комп’ютерної графіки CGM – це стандарт, що визначає структуру метафайла: види і місця тих або інших його елементів. Формат CGM був розроблений

як формат обміну векторними даними між різними графічними пакетами і не був прив’язаний до будь-якого специфічного виробника. Сам формат є досить багатим і включає в себе опис ліній, кіл, секторів, багатокутників і тексту. У CGM існує три типи кодування даних: бінарний, який інтерпретує 99% додатків, і застосовує двійкові коди команд; буквенний (character), який містить опис графіки на мові CGM у стисненому вигляді і CLEAR TEXT, який ніяк не змінює команди і тому їх можна проглядати на екрані. Проте файли в такому форматі будуть значно більші за розмірами.

Для підтримки обміну растровими і векторними даними використовують метафайли. Головною ознакою метафайлу є те, що графічний опис об’єктів складається з команд для певного графічного пристрою.

Питання для самоперевірки

1. Дайте визначення векторній графіці.
2. Що є основним елементом у векторній графіці?
3. Який вид робіт можна здійснювати у векторній графіці?
4. Призначення кривих Без’є і їх особливості.
5. Вкажіть, які існують типи опорних точок.
6. Охарактеризуйте види кривих Без’є.
7. Що таке криві NURBS?
8. Назвіть основні редактори векторної графіки
9. Вкажіть призначення форматів файлів векторної графіки і назвіть їх.
10. Які особливості застосування метафайлових форматів? Назвіть їх.

Список рекомендованої літератури

1. Блінова Т.О., Порєв В.М. Комп’ютерна графіка / За ред. В.М. Горєва. – К.: Видавництво «Юніор», 2004. – С. 186–228.

2. Васильев В.Е. Компьютерная графика: Учебное пособие. – Санкт-Петербург, 2004. – 96 с.

3. Веселовська Г.В., Ходаков В.Є., Веселовський В.М. Комп’ютерна графіка: Навч. посібник для студентів вищих навчальних закладів / Під ред. В.Є. Ходакова. – Херсон: ОЛДІ-плюс, 2004. – С.112–133.

4. Горобець С.М. Основи комп'ютерної графіки: Навч. пос. /За ред М.В. Левківського. – К: Центр навчальної літератури, 2006. – 232 с.

5. Мельниченко В.В., Легейда В.В. Настоящий САМОУЧИТЕЛЬ компьютерной графики. – К.: Век+, СПб.: КОРОНА принт, К.: НТИ, 2005. – 560 с.

6. Петров М.Н., Молочков В.П. Компьютерная графика: Учебник для вузов. 2–е изд. – СПб.: Питер, 2005.

7. Федік Л.Ю. Основні формати графічних даних у мережі World Wide Web // Наукові нотатки. Міжвузівський збірник (за напрямом «Інженерна механіка»). – Луцьк, 2008. – С. 306–315.

8. http://bezobeda.net

9. http://wiki.auditory.ru

РОЗДІЛ 4. ФРАКТАЛЬНА ГРАФІКА

4.1. Основні відомості про фрактальну графіку

Поняття фрактал і фрактальна геометрія, які з’явилися в кінці 70-х, з середини 80-х міцно увійшли у вжиток математиків і програмістів. Слово фрактал утворене від латинського *fractus* і в перекладі означає – складатися з фрагментів. Воно було запропоноване Бенуа Мандельбротом в 1975 році для позначення нерегулярних, проте слабоподібних структур, якими він займався.

Народження фрактальної геометрії прийнято пов'язувати з виходом в 1977 році книги Мандельброта «The Fractal Geometry of Nature». У його роботах використані наукові результати інших учених, що працювали в 1875 – 1925 роках в тій же галузі (Пумнкаре, Фату, Жюліа, Кантор, Хаусдорф). Але тільки у наш час наші часи вдалося об'єднати їх роботи в єдину систему. Визначення фрактала, дане Мандельбротом: **фракталом** *називається структура, що складається з частин, які подібні до цілого.*

Самоподібність – одна з основних властивостей фракталів. Об'єкт називають самоподібним, коли збільшені частини об'єкта схожі на сам об'єкт і один до одного.

Фрактальна графіка базується на фрактальній геометрії.

Найвідомішими фрактальними об’єктами є дерева: від кожної гілки відходять менші, схожі на неї, від них – ще меньші. За окремою гілкою математичними методами можна відслідкувати властивості всього дерева. Фрактальні властивості мають такі природні об’єкти, як: сніжинка, що при збільшенні виявляється фракталом; за фрактальними алгоритмами ростуть кристали та рослини.

Поява нових елементів меншого розміру відбувається за певним алгоритмом. Очевидно, що описати подібні об’єкти можна всього лише декількома математичними рівняннями!

Значення фракталів у машинній графіці сьогодні досить значне. Вони приходять на допомогу, наприклад, коли потрібно, за допомогою декількох коефіцієнтів, задати лінії і поверхні дуже складної форми. З точки зору машинної графіки, фрактальна геометрія незамінна під час генерації штучних хмар, гір, поверхні моря. Фактично знайдений спосіб легкого уявлення неевклідових об’єктів, взірці яких дуже схожі на природні.

Фрактальна графіка, як і векторна, базується на математичних обчисленнях. Однак базовим елементом є математична формула, і ніяких об'єктів у пам'яті комп'ютера не зберігається і зображення будується виключно за рівняннями. Фрактальна графіка міститься у пакетах для наукової візуалізації для побудови, як найпростіших структур так і складних ілюстрацій, що імітують природні процеси та тривимірні об'єкти.

4.2. Класифікація фракталів

Існують такі види фракталів:

1) геометричні,

2) алгебраїчні,

3) стохастичні,

4) системи ітеруючих функцій.

Геометричні фрактали

З геометричних фракталів почалася їх історія. Цей тип фракталів отримують шляхом простих геометричних побудов. Зазвичай під час побудови цих фракталів береться «приманка»-аксіома – набір відрізків, на підставі яких будуватиметься фрактал. Далі до цієї «приманки» застосовують набір правил, який перетворить її у будь-яку геометричну фігуру.

Фрактали цього класу самі наочні. У двомірному випадку їх отримують за допомогою деякої ламаної (чи поверхні в трьохмірному випадку), яка називається генератором. За один крок алгоритму кожен із відрізків, які складають ломану, замінюється на ломану-генератор, у відповідному масштабі. У результаті безкінечного повторення цієї процедури, отримується геометричний фрактал.

Для побудови геометричних фрактальних кривих використовуються рекурсивні алгоритми. Рекурсія використовується при вирішенні завдань, які можуть бути розкладені на декілька підзадач. Таким чином, застосування рекурсії доцільне під час побудови фрактальних кривих, оскільки вони володіють такою властивістю як самоподібність.

Прикладом такого фрактального об’єкта є тріадна крива Кох (рис. 4.1).

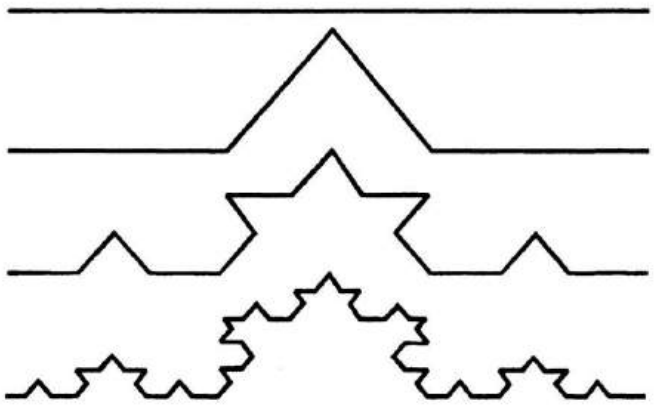


Рис. 4.1. Побудова тріадної кривої Кох

На ньому подано три покоління кривої. При n прямуючому до безкінечності крива Кох стає фрактальним об’єктом.

У машинній графіці застосування геометричних фракталів необхідне під час отримання зображень дерев, кущів, берегової лінії. Двомірні геометричні фрактали застосовуються для створення об’ємних текстур.

Прикладами геометричних фракталів слугують: сніжинка Коха, лист, криві і трикутник Серпинського, криві Гільберта (рис. 4.2-4.7), а також множина Кантора, крива Пєано, крива дракона, Т-Квадрат та губка Менгера.

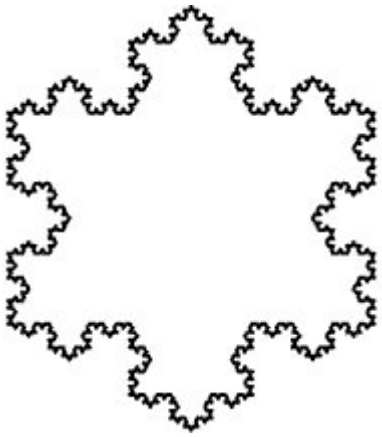


Рис. 4.2. Сніжинка Коха

Рис. 4.3. Лист

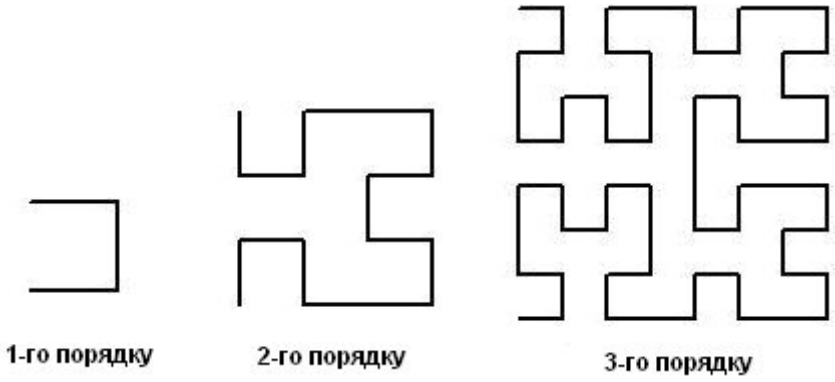


Рис. 4.4. Криві Гільберта

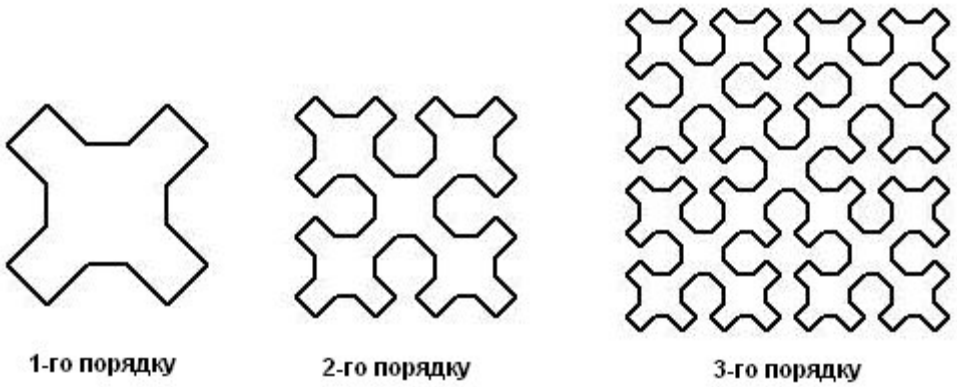


Рис. 4.5. Криві Серпинського

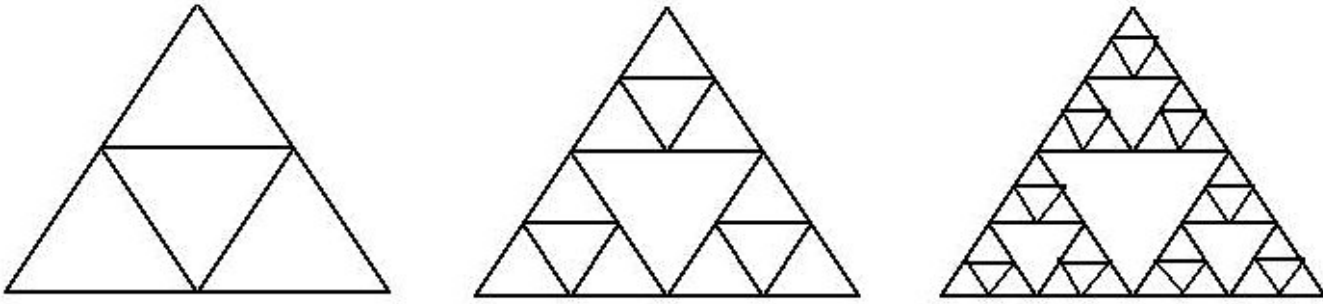


Рис. 4.6. Трикутник Серпинського

Алгебраїчні фрактали

Це найбільш значна група фракталів. Свою назву вони отримали за те, що їх будують, на основі алгебраїчних формул деколи досить простих.

Методів отримання алгебраїчних фракталів декілька. Один із методів являє собою багатократний (ітераційний) розрахунок функції $Zn+1=f(zn)$, де Z – комплексне число, а f – деяка функція. Розрахунок даної функції продовжується до виконання певної умови. І коли ця умова виконається – на екран виводиться точка. При цьому значення функції для різних точок комплексної площини може мати різну поведінку: з плином часу прагне до безкінечності; прагне до 0; приймає декілька фіксованих значень і не виходить за їх межі; поведінка хаотична, без будь-яких тенденцій.

Прикладами алгебраїчних фракталів є множина Мандельброта (рис. 4.7).

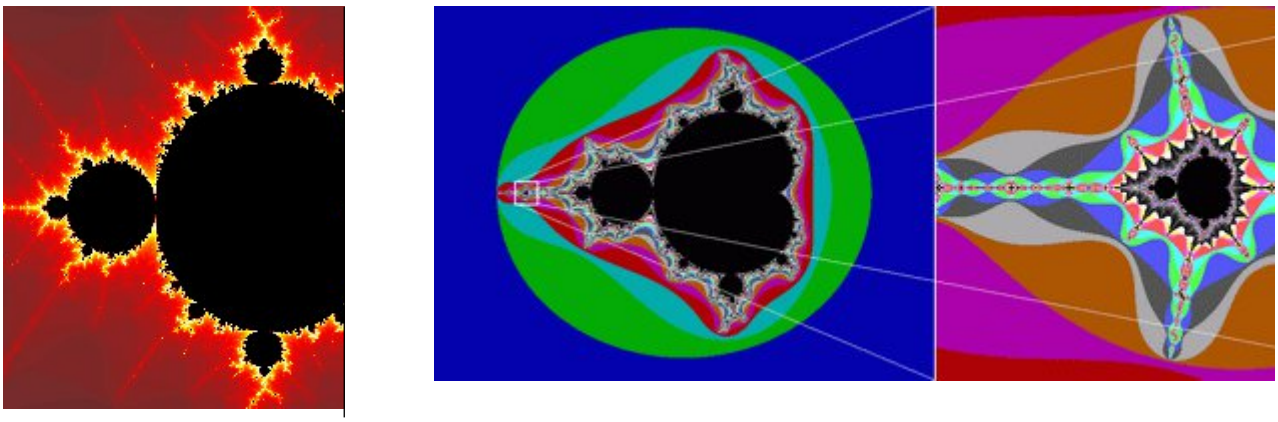


Рис. 4.7. Множина Мандельброта

Для побудови множини Мандельброта необхідні комплексні числа. Комплексне число – це число, що складається з двох частин, – дійсної і уявної і позначається ***a+bi.*** Дійсна частина ***a*** це звичайне число в нашому уявленні, а ***bi*** – уявна частина, ***i*** – називається уявною одиницею, оскільки під час піднесення її до квадрату, отримаємо – 1.

Комплексні числа можна додавати, віднімати, перемножувати, ділити, підносити до ступеня і отримувати корінь, не можна тільки їх порівнювати. Комплексне число можна зобразити як точку на площині, біля якої координата X дійсна частина, а Y – коефіцієнт при уявній частині b. Функціональна множина Мандельброта визначається як Zn+1=Zn·Zn+C.

Стохастичні фрактали.

Стохастичні фрактали, отримуються в тому випадку, коли в ітераційному процесі випадковим чином змінювати будь-які його параметри. При цьому отримуються об’єкти дуже схожі на природні – несиметричні дерева, врізані берегові лінії і т.д. (рис. 4.8). Моделювання рельєфу місцевості і поверхні моря. Двомірні стохастичні фрактали застосовуються під час моделювання рельєфу місцевості і поверхні моря.

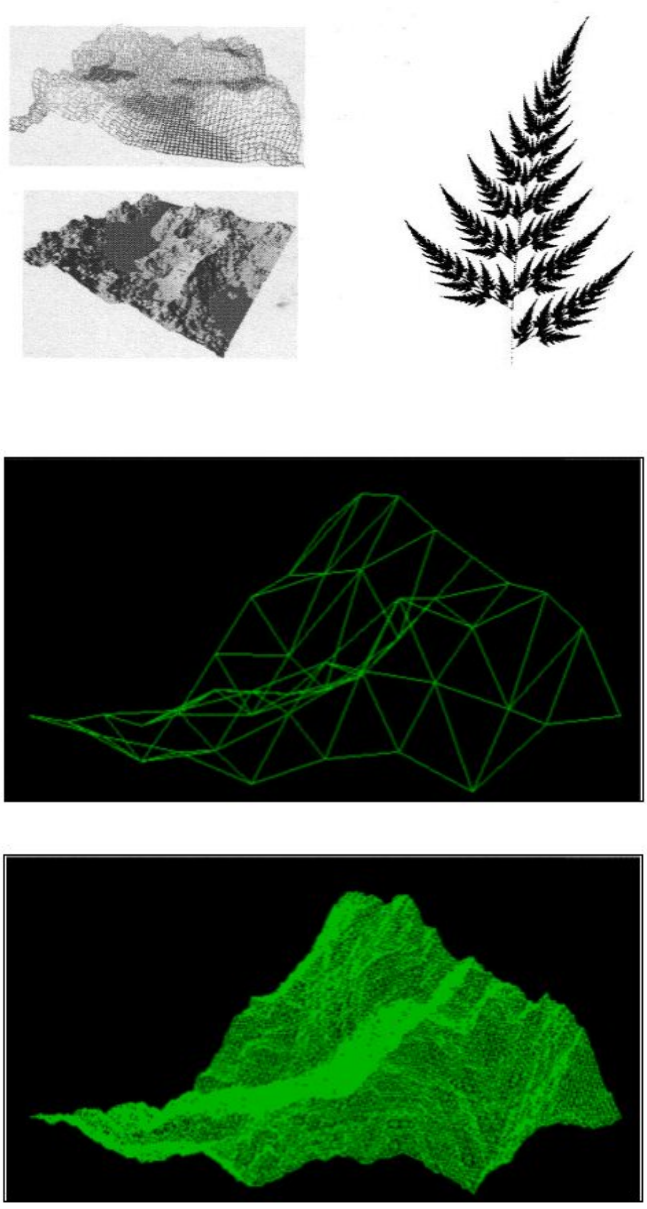


Рис. 4.8. Стохастичні фрактали

Спроможність фрактальної графіки моделювати образи живої природи обчислювальним шляхом часто застосовують для автоматичної генерації незвичайних ілюстрацій.

Типовий передставник даного класу фракталів «Плазма». Для її побудови візьмемо прямокутник і для шкіряного його кута визначимо колір. Далі знаходимо центральну точку прямокутника і розфарбовуємо її у колір рівний середньому арифметичному кольорів кутів прямокутника плюс деяке випадкове число. Чим більше випадкове число – тим більш «рваним» буде малюнок. Якщо, наприклад, колір крапки це висота над рівнем моря, то отримаємо замість плазми – гірський масив. Саме на цьому принципі моделюються гори в більшості програм. За допомогою алгоритму, схожого на плазму будується карта висот, до неї застосовуються різні фільтри, накладається текстура.

Системи ітеруючих функцій (IFS – Iterated Function Systems).

Ця група фракталів набула широкого поширення завдяки роботам Майкла Барнслі з технологічного інституту штату Джорджія. Він намагався кодувати зображення за допомогою фракталів. Запатентувавши декілька ідей з кодування зображень за допомогою фракталів, він заснував фірму «Iterated Systems», яка випустила перший продукт «Images Incorporated», за допомогою якого можна було переводити зображення з растрової форми у фрактальну (формат FIF). Це дозволяло добитися високих ступенів стиснення. Оскільки за низьких ступенів стиснення якість малюнків поступалася якості формату JPEG, а за високих – картинки виходили якіснішими. У будь-якому випадку цей формат не прижився, але роботи з його вдосконалення ведуться досі, бо він не залежить від роздільної здатності зображення. Оскільки зображення закодоване за допомогою формул, можна збільшити до будь-яких розмірів і при цьому з’являтимуться нові деталі, а не просто збільшиться розмір пікселів.

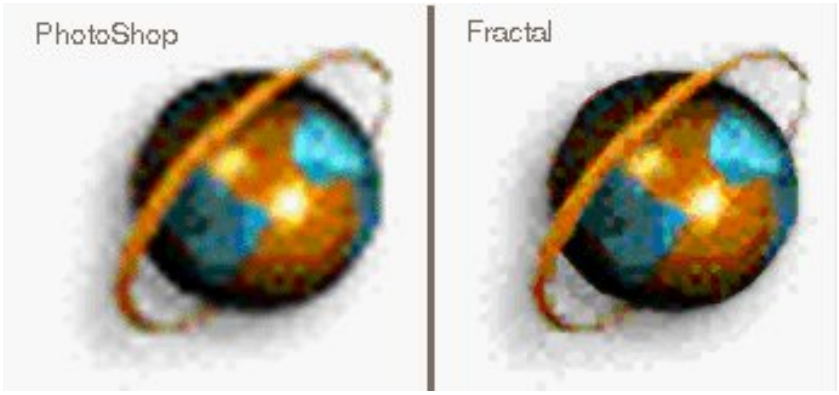


Рис. 4.9. Фрактали на базі ітеруючих функцій

Якщо в L-systems (алгебраїчних фракталах) йшлося про заміну прямої лінії якимсь полігоном, то в IFS у ході кожної ітерації замінюємо якийсь полігон (квадрат, трикутник, круг) на набір полігонів, кожен із яких піддадуть аффінним

перетворенням. За аффінних перетворень початкове зображення змінює масштаб, паралельно переноситься уздовж кожної з осей і обертається на деякий кут.

4.3. Редактори фрактальної графіки

Для наукової візуалізації, побудови, як найпростіших структур так і складних ілюстрацій, застосовуються редактори фрактальної графіки. Найбільш поширеними з них є Ultra Fractal, Fractal Explorer, ChaosPro, Apophysis, Mystica, Surfer, Iris Explorer, Art Dabbler, Earth Watch, Chart, Grapher, MapViewer.

Редактор Ultra Fractal відрізняється дружнім інтерфейсом, багато елементів якого нагадують інтерфейс Photoshop (що спрощує вивчення), і супроводжується неймовірно докладною і чудово ілюстрованою документацією з серією туторіалів, у яких поетапно розглядаються всі аспекти роботи з програмою. Ultra Fractal представлений двома редакціями: Standard Edition і розширеною Animation Edition, можливості якої дозволяють не тільки генерувати фрактальні зображення, але і створювати анімацію на їх основі. Створені зображення можна візуалізувати у високому розширенні, придатному для поліграфії, і зберігання у власному форматі програми чи в одному з популярних фрактальних форматів. Візуалізовані зображення також можуть бути експортовані в один з растрових графічних форматів (jpg, bmp, png і psd), а готові фрактальні анімації – у AVI-формат.

Принцип створення фрактальних зображень досить традиційний. Найпростішим методом є застосування формул (що розміщені у вбудованому браузері), а потім редагування параметрів формули. Останню дію легко скасувати. Готових фрактальних формул дуже багато, і їх кількість може бути розширена шляхом скачування нових формул з сайту програми. У пакеті Ultra Fractal є можливість створювати власні формули, для чого в редакторі існує вбудований текстовий редактор, який підтримує базові шаблони, що базуються на стандартних конструкціях мови програмування фрактальних формул.

Створювати зображення фракталів з досить вражаючими можливостями дозволяє редактор Fractal Explorer. Він має інтуїтивно зрозумілий класичний інтерфейс, який може бути налаштований у відповідності з користувацькими уподобаннями, і підтримує стандартні формати фрактальних зображень: *. frp; *. frs; *. fri; *. fro; *. fr3, *. fr4 та ін. Готові фрактальні зображення зберігаються у форматі *. frs і можуть бути експортовані в один з растрових графічних форматів jpg, bmp, png і gif, у той час як фрактальні анімації зберігаються як AVI-файли.

Генерація фракталів у цій програмі можлива двома способами – на основі базових фрактальних зображень, побудованих за вхідними формулами, чи з нуля. Перший варіант дозволяє отримати цікаві результати порівняно просто, оскільки вибрати відповідну формулу нескладно, тим більше що зручний файловий браузер

дозволяє оцінити якість фрактала з бази ще до створення на його основі фрактального зображення. У отриманого таким шляхом фрактального зображення можна змінити колірну палітру, додати до нього фонове зображення і визначити режим змішування фрактального і фонового шарів, а також ступінь прозорості фрактального шару. Потім можна буде піддати фрактальне зображення трансформації, за необхідності масштабувати, визначити розміри зображення і провести рендеринг. Створення зображення з нуля набагато складніше і передбачає вибір одного з двох способів. Використовуючи перший спосіб можна вибрати тип фрактала майже з 150 варіантів, а потім перейти до зміни різноманітних параметрів: налаштування палітри, фону.

Згідно другого способу можливе створення своєї формули, користуючись вбудованим компілятором. Перед рендерингом готового зображення може знадобитися проведення автоматичної корекції колірного балансу чи ручної корекції яскравості, контрастності і насиченості.

Редактор ChaosPro вважається одним з кращих безкоштовних генераторів фрактальних зображень, за допомогою якого нескладно створити безліч дивовижних за красою фрактальних зображень. Програма має дуже простий і зручний інтерфейс і поряд з можливістю автоматичної побудови фракталів дозволяє повністю керувати даним процесом за рахунок зміни великої кількості налаштувань (кількості ітерацій, колірної палітри, ступеня розмиття, особливостей проектування, розміру зображення та ін.). Крім того, створюючі зображення можуть бути багатошаровими (режимом змішування шарів можна керувати) і до них можна застосувати серію фільтрів. Всі зміни, що накладаються на споруджуючі фрактали відразу ж відображаються у вікні перегляду. Створені фрактали можуть бути збережені у власному форматі програми, чи в одному з основних фрактальних типів завдяки наявності вбудованого компілятора. А також можуть бути експортовані в растрові зображення чи 3D-об'єкти (якщо попередньо було отримане тривимірне зображення фрактала).

Серед можливостей програми слід відзначити точне колірне настроювання, що забезпечує плавні градієнтні переходи кольорів один у одний, одночасну побудову кількох фракталів у різних вікнах, можливість створення анімації на основі фрактальних зображень з визначенням ключових анімаційних фаз, які можуть відрізнятися будь-яким параметром, що змінюється (кутами повороту і обертання, колірними параметрами і ін.), створення тривимірних зображень фракталів на основі звичайних двовимірних зображень, підтримку багатьох стандартних форматів фрактальних зображень, зображення в яких можуть бути імпортовані і відредаговані в середовищі ChaosPro.

Цікавим інструментом для генерації фракталів на основі базових фрактальних формул є редактор Apophysis. Він дозволяє редагувати і невпізнанно змінювати створені за готовими формулами фрактали, регулюючи різноманітні параметри. Так, наприклад, у редакторі їх можна трансформувати, змінювати, застосувавши метод перетворення, який сподобався, зокрема: хвилеподібне спотворення, перспектива,

розмиття за Гаусом та ін. Є можливість експериментувати з кольорами, вибравши один із базових варіантів градієнтної заливки. Список вбудованих заливок досить значний, і за необхідності можна автоматично підібрати найбільш прийнятну заливку до наявного растрового зображення, що доцільно, наприклад, під час створення фрактального фону в тому ж стилі, що й інші зображення якогось проекту. За необхідності нескладно підрегулювати гаму і яскравість, змінити фон, масштабувати фрактальний об'єкт і уточнити його розташування на фоні. Крім цього можна піддати результат різноманітним мутаціям у потрібному стилі і записати його візуалізований варіант у вигляді графічного файла (jpg, bmp, png).

Універсальним генератором унікальних фантастичних двовимірних і тривимірних зображень і текстур, які можна використовувати в різних проектах є редактор Mystica. Наприклад, у якості реальних текстур для Web-сторінок, фонів робочого столу чи фантастичних фонових зображень, які можуть бути задіяні, наприклад, під час оформлення дитячих книг. Пакет відрізняється нестандартним і досить складним інтерфейсом і може працювати в двох режимах: Sample (орієнтований на новачків і містить мінімум налаштувань) і Expert (призначений для професіоналів). Створюючі зображення можуть мати будь-який розмір і потім експортуватися в популярні графічні 2D-формати. Просто з вікна програми їх можна надіслати електронною поштою, опублікувати в Html-галереї чи створити на їх основі відеоролик у форматах divx, mpeg4 і ін.

Генерація зображень здійснюється на основі закладених у пакеті фрактальних формул, а система підготовки зображення багаторівнева та включає дуже докладну настройку кольорів, можливість найпростіших трансформацій, генерування елементів і масу інших перетворень, зокрема: застосування фільтрів, зміна освітлення, коректування колірної гами, яскравості і контрастності, зміна використаного під час генерації матеріалу, щоб додати до зображення "хаотичних" структур та ін.

Отже, редактори Ultra Fractal, Fractal Explorer, ChaosPro, Apophysis, Mystica, Surfer найбільш широко застосовуються у фрактальній графіці. Зокрема, для задання лінії і поверхні дуже складної форми за допомогою декількох коефіцієнтів.

Редактор фрактальної графіки Iris Explorer розроблений фірмою Graphics призначений для створення моделей погодних умов і океану.

Система IRIS DATA EXPLORER, або просто EXPLORER, – це продукт, що функціонує в середовищі UNIX (для станцій SGI) і Windows NT (для станцій на платформі Pentium). Основою системи є графічний інтерфейс OpenGL, що забезпечує реалізацію рендерингу.

Специфікація системи визначає її предметну галузь, зокрема як візуалізацію даних, так і обробку зображень і створення продукції мультимедіа. У складі системи EXPLORER є засоби, що підтримують повний цикл розробки додатків із обробки даних. Проте основним завданням є візуальне компонування. У зв’язку з цим система

відноситься до класу систем швидкої розробки додатків (RAD – Rapid Application Development) і прототипування.

У користувацькому та програмному інтерфейсі EXPLORER чітко витримується об'єктно-орієнтований підхід і модульність. У системі визначені всього п'ять базових типів даних, зокрема: Lattice, Pyramid, Geometry, Parameter і Pick. Lattice – це основний тип для представлення даних, що візуалізуються. Він містить числові масиви, що описують сітку, масиви значень і розмірності масивів. Pyramid – ієрархія даних типу Lattice разом з інформацією про те, як наступний рівень будується з попереднього. Цей тип використовується для компактного представлення кінцево-елементних і молекулярних даних. EXPLORER містить засоби для роботи з молекулярними даними в форматі Brookhaven Protein DataBase. Geometry – геометричне уявлення об'єктів, що складається з примітивів: полігонів, ліній і точок. Parameter – скалярні дані, які керують роботою програми, але можуть інтерактивно змінюватися. Pick – представляє інформацію, що виводяться на екрані. Редактор можна розглядати, як послідовність перетворювачів (модулів), які застосовуються до даних цих типів. На першому етапі вхідні масиви даних, які можуть бути введені, наприклад, з файлів довільної структури, перетворюються в дані типу Lattice або Pyramid. На останньому кроці проводиться перетворення в тип Geometry, який, власне, візуалізується. Згідно форм візуального представлення даних існують різні способи такого перетворення. У всьому ланцюжку перетворень беруть участь тільки дані п'яти вказаних типів, які параметризовані і мають підтипи.

Функціональною одиницею користувацького інтерфейсу є модуль обробки даних, тобто виконуюча програмна одиниця, реалізована на мові програмування Сі або Fortran. Вона має вхідні і вихідні параметри, які називаються в системі EXPLORER портами.

Візуально модуль представляється користувачу у вигляді керуючої панелі, яка залежно від необхідної деталізації може мати три стани: мікро, міні та максі.

Скороченим варіантом програми Painter є редактор Art Dabbler (створений фірмою Fractal Design, що тепер належить Corel). Його призначенням є навчання не тільки комп'ютерній графіці, але й азам малювання. Малий об'єм необхідної пам'яті (10 Мбайт), а також простий інтерфейс, дозволяє використовувати його у шкільній програмі.

Рядок меню редактора включає в себе шість пунктів, які є стандартними для більшості програм: File, Edit і Help, а також Effects, Options і Tutors, що присутні в більшості графічних програм і не потребують додаткових пояснень. Art Dabbler надає комплект ефектів (меню Effects), які можуть бути використані для зміни чи спотворення зображень. Наприклад, ефект Texturize створює текстури паперу, полотна тощо, розширюючи творчі можливості художника.

Редактор фрактальних зображень Earth Watch був розроблений фірмою Earth Watch, його призначенням є моделювання та демонстрація тривимірного зображення

метеоумов над Землею, побудова топологічних поверхонь за космічними знімками і прогнозування погоди на тиждень вперед.

Модуль Chart у стандартному пакеті MS Office дозволяє легко й наочно створити графіки на основі даних, що знаходяться у таблиці. Користувач може перетворити графіки у будь-яку з п’яти основних форм графіків: гістограму, лінії, площі, в полярних координатах, поверхні. Також, під час зміни даних у таблиці, змінюється відповідне значення на графіку.

Серед програмних засобів фрактальної графіки можна виділити продукти фірми Golden SoftWare: Surfer, Grapher, Map Viewer.

Пакет наукової графіки Surfer призначений для інтерполяції, апроксимації і візуалізації функцій двох змінних, а також полів, заданих у точках на плоскій області. Він має зручні засоби для відображення у вигляді ліній рівня (ізоліній) і у вигляді поверхонь. Surfer 7.0 вважається застарілим, проте його перевагою є те, що він встановлюється безпосередньою вставкою файлу виконуючої програми Surfer.exe у будь-яку папку, а не через процедуру встановлення, яка вимагає сповіщення серійного номеру. Запускати Surfer.exe треба з цієї папки. Surfer 8 зручніший в користуванні та має більш розвинені засоби.

Об’єктами відображення у свою чергу є: функція двох змінних; поле, що створюється окремими точками на площині. А відображеннями у Surfer 7.0 виступають: ізолінії (Contour); відтінки кольорів (Image); поверхні (Wireframe); винесені на площину точки, при необхідності у супроводженні значень z (Post). Функції в Surfer можуть вживатися у командах: Grid – Function, Grid – Math і деяких інших. Найбільш вживаними функціями є: sin(x), cos(x), tan(x), asin(x), acos(x), arctg(y/x), exp(x), sinh(x), cosh(x), tanh(x), ln(x), log10(x), pow(x,y), min(x,y), max(x,y), randn(x,y), randu(x), ceil(x), floor(x), sqrt(x), fabs(x), fmod(x,y), d2r(x), r2d(x). Також вживаються булівські чи логічні вирази – вирази, які можуть приймати значення логічних констант true або false. Логічний вираз складається з логічних операндів, з’єднаних знаками логічних операцій and, or, xor, not („і”, „або”, виключаюче „або”, „ні” – заперечення). Логічний операнд, у загальному випадку також логічний вираз, найчастіше є логічним відношенням. Логічне відношення – це два арифметичних вирази, з’єднані знаком логічного відношення: =, <>, <, >, <=, >=.

Редактор Grapher, призначений для математичної і графічної обробки даних, що описуються одновимірною функцією $y = f(x)$. Подібні можливості є в багатьох пакетах, пов’язаних з обробкою числової інформації (електронні таблиці, програми статистичного аналізу), але на практиці їх можливостей часто виявляється недостатньо для розв’язку спеціалізованих задач (обмеження на складність графіків, типи зображень і т.п.). Розглядаючи функціональні можливості Grapher 2.0, треба передусім відмітити, що в ньому немає обмежень ні у кількості графіків на одному малюнку, ні у кількості кривих на одному графіку, причому кожна крива може містити до 32 тис. точок (X, Y).

На одному графіку дозволяється розміщувати декілька осей з різними масштабами і одиницями вимірювання даних. Дуже зручний режим представлення даних можна реалізувати у вигляді зображення декількох осей Y з однією загальною віссю X. Самі графіки виводяться як з горизонтальним положенням осі X, так і з вертикальним.

На графіку дозволяється використовувати не тільки лінійні і логарифмічні, але і осі ймовірностей в різних комбінаціях. При цьому розмітка осей, виведення і легенди виконуються автоматично чи задаються користувачем. Для виводу застосовується декілька типів графіків: лінійні, символьні, діаграми.

Для всіх типів графіків реалізоване керування товщиною, кольором і стилем будь-яких ліній, використання готових стилів і створення своїх власних. Набір вбудованих інструментів малювання дозволяє наносити на графік спеціальні елементи малюнка (кола, прямокутники, лінії та інше). За допомогою стандартних маніпуляцій мишею можна легко змінювати розмір і положення зображення, масштабувати його для більш детального вивчення, додавати текстові пояснення, легенду і додаткові графічні зображення, в тому числі імпортовані з інших програмних продуктів.

Графіки можуть будуватися на основі наборів точкових (дискретних) значень X, Y (меню Graph: *New Graph/Line\Symbol*) і функцій типу, $y = f(x)$, що задаються користувачем, або параметричних рівнянь вигляду, $y = x(t)$, $x = x(t)$ (меню Graph: *New Graph/Function*). Пакет дозволяє імпортувати набори вихідних даних з файлів форматів DAT і SLK, а також з файлів електронних таблиць Excel. Відкоректовані значення запам'ятовуються в форматах DAT і SLK. Для апроксимації експериментальних даних кривими є такі методи апроксимації: лінійний, логарифмічний, степеневий, експоненціальний, поліноміальний. Для вибору методу апроксимації застосовується пункт Properties та вибирається закладка Fits.

Дані для різних кривих одного графіка дозволяється брати як з одного файлу (наприклад, з його різних рядків або стовпців), так і з декількох. Останній варіант дуже зручний під час обробки результатів серії експериментів, кожен з яких характеризується своїм набором даних.

Пакет має вбудовану електронну таблицю, що дозволяє коректувати, сортувати, перетворювати дані і обчислювати їх статистичні характеристики. Також на основі декількох колонок даних можна заповнювати інші, користуючись вбудованими функціями. Багатовіконний інтерфейс пакету забезпечує ефективну настройку робочого середовища. Наприклад, на екран можна одночасно вивести два вікна, в одному з яких буде електронна таблиця, а в іншому – графік, і під час редагування даних графік буде автоматично оновлюватися. Для переходу від графіка до електронної таблиці треба тільки натиснути кнопку миші. Окрім цього, вбудовані функції можна використовувати під час побудови заданих функціональних залежностей $y = f(x)$, про які говорилося вище. У пакет вбудовано текстовий редактор

для створення текстових блоків, що включає спеціальні засоби введення математичних формул. Сформовані зображення зберігаються у файлах формату GRF. Існує можливість оперативно відображати заданий регіон (ділянки) графіка, а також функцію трасування – можливості візуально переглянути процес побудови графіка переміщуючи повзунок у відповідному діалоговому вікні, а також відображається і значення функції у цій точці. Передбачена можливість обчислення визначеного інтеграла з функцією визначення як числового значення, так і отримання області, що інтегрується (за якою визначається площа).

Редактор MapViewer у свою чергу є високопродуктивним і точним інструментом для візуалізації та аналізу географічної числової інформації за допомогою побудови інформативних тематичних карт. Використовуючи його, консультанти в області ГІС-систем, вчені та картографи, керівники і аналітики підприємств можуть проводити геологічні, екологічні та епідеміологічні дослідження, виконувати аналіз ринку, готувати матеріали для презентацій і звітів та інше.

Основними функціями редактора є: представлення зображень в різних географічних проекціях, зокрема для перетворення сферичних координат земної кулі в прямокутну декартову систему координат карти, для більш точного уявлення кордонів. Такі перетворення особливо потрібні для накладання зображень, представлених у різних проекціях; карти вихідних зображень можна імпортувати з файлів різних форматів: DXF, LGS, STD, BLN, CLP, PLT, TIF, PCX, WPG, JPG, DCX, BNA, LGO, OPT, DLG, BND, WMF, BMP, GIF і PCT, а результуючі дані експортувати у формати GSB, DXF, CLP, BMP, PCX, WPG, PCT, WMF, BNA, CGM, TIF, TGA, GIF, JPG і DCX.

До складу пакету входить великий набір векторних карт у форматі GSB: карти США, її найбільших міст, окремих штатів, автомагістралей, карти Канади, Мексики та Росії, а також межі країн на картах материків. На Web-сайті фірми Golden Software розміщений архів карт з вільним доступом, який постійно доповнюється.

Для введення і перетворення числової інформації, її сортування і розрахунку статистичних характеристик у пакеті є повнофункціональна електронна таблиця.

Питання для самоперевірки

1. Дайте визначення фракталу.
2. Вкажіть найвідоміші фрактальні об’єкти.
3. Які види фракталів існують?
4. Проаналізуйте основні види фракталів.
5. Застосування редакторів фрактальної графіки.
6. Назвіть редактори, які застосовуються у фрактальній графіці.

Список рекомендованої літератури

1. Комп’ютерна графіка. Конспект лекцій для студентів спеціальності /6.092501/ «Автоматизоване управління технологічними процесами» денної та заочної форм навчання / Федік Л. Ю. – Луцьк: ЛДТУ, 2007. – С.25
2. http://www.osp.ru/os/1996/02/178854/
3. http://sergeyk.kiev.ua/conspect/comp-tech/AdvancedGrapher.shtml
4. http://www.geol.univ.kiev.ua/lib/zhukov_n_n/Tema_10_Surfer.pdf
5. http://allsoft.ru/program_page.php?grp=27286

РОЗДІЛ 5. ТРИВИМІРНА (3-D) ГРАФІКА

5.1. Поняття двовимірної і тривимірної графіки

Двовимірна (2D) графіка – *це зображення, які мають два вимірювання, тобто які лежать на площині.*

А тривимірна графіка (3D) (3 Dimensions, на укр. мові 3 виміри) – *розділ комп'ютерної графіки, сукупність засобів і інструментів (як програмних, так і апаратних) призначених для зображення об'ємних об'єктів* (рис. 5.1).

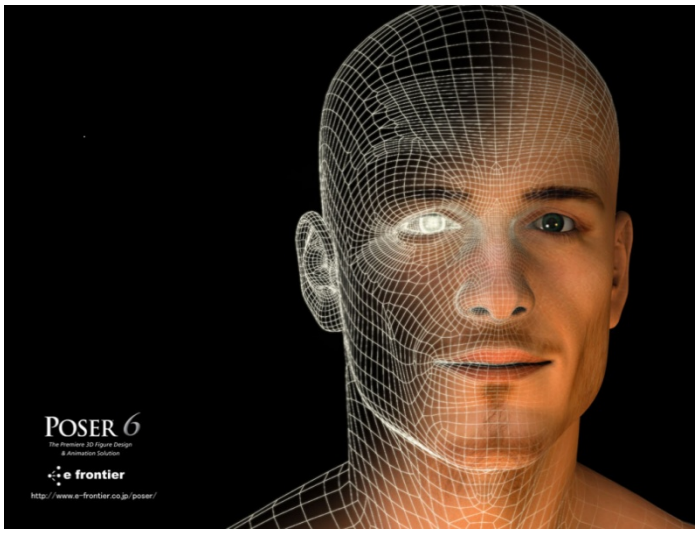


Рис. 5.1. Тривимірна графіка (3D)

На відміну від 2D графіки 3D використовує тривимірне представлення геометричної інформації (часто Декартову, тобто картезіанську), що зберігається в комп'ютері для подальшого обчислення і побудови 2D зображень. Тривимірне зображення на площині відрізняється від двовимірного тим, що включає побудову геометричної проекції спеціалізованих програм. При цьому модель може відповідати об'єктам реального світу (автомобілі, будинки, буревій, астероїд), чи бути цілковитою абстракцією (проекція чотирьохвимірного фракталу).

Цей вид графіки використовується для створення зображень на площині екрану чи листі друкованої продукції в архітектурній візуалізації, кінематографі, телебаченні, комп'ютерних іграх, друкованій продукції, під час створення сайтів, а також у науці та промисловості.

Поряд з програмами традиційної 2D-графіки в останнє десятиліття значне розповсюдження і популярність отримали програми 3D-моделювання, анімації і візуалізації. При цьому такі відомі програмні рішення, як 3D Studio MAX (рис. 5.2) чи Мауа є гібридними графічними пакетами. З однієї сторони, вони надають дизайнеру можливість маніпулювати 2D- і 3D-векторними об'єктами, з іншої, результатом роботи

(фінальної візуалізації) є піксельне (растрове) зображення – окремий кадр або відеоролик.

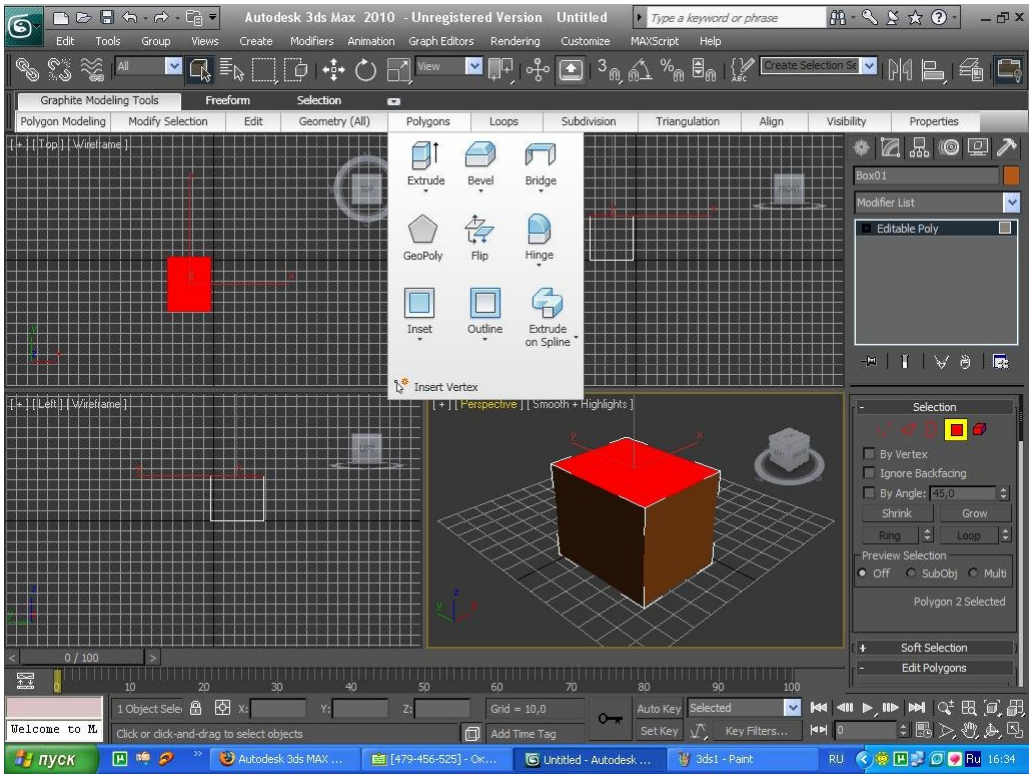


Рис. 5.2. 3D Studio MAX

У силу специфікації 3D-моделювання і можливості працювати з анімацією такі програми займають особливе місце серед графічних програм. Якщо спробувати позиціонувати пакети 3D-графіки з точки зору їх цільової функції, можна виділити такі: візуальні спецефекти для кіно- і відео індустрії; телевізійна реклама; інтерактивні ігри; промисловий та архітектурний дизайн; наукова, медична і судова візуалізація; комп'ютерні тренажери і навчальні програми.

Необхідно відмітити, що застосування пакетів тривимірної графіки пред'являють вимоги як до апаратно-програмних засобів застосовуючого комп'ютера, так і до рівня знань працюючого з ними дизайнера. Оскільки, всі програми 3D-графіки, перш за все, дозволяють застосовувати декартову (картезіанську) систему координат.

5.2. Типи просторів

У залежності від завдання і етапу роботи можна вибрати різні типи просторів і пов'язаних із ними координатних систем.

Найчастіше програми тривимірного моделювання надають такі варіанти просторів:

1. простір об’єкта (рис.5.4) – призначений для моделювання (опису) форми об’єкта в його власній (локальній) системі координат не відносно того, де він буде розміщений на сцені, як орієнтований чи масштабований. У кожного об’єкта існує своя власна система координат,

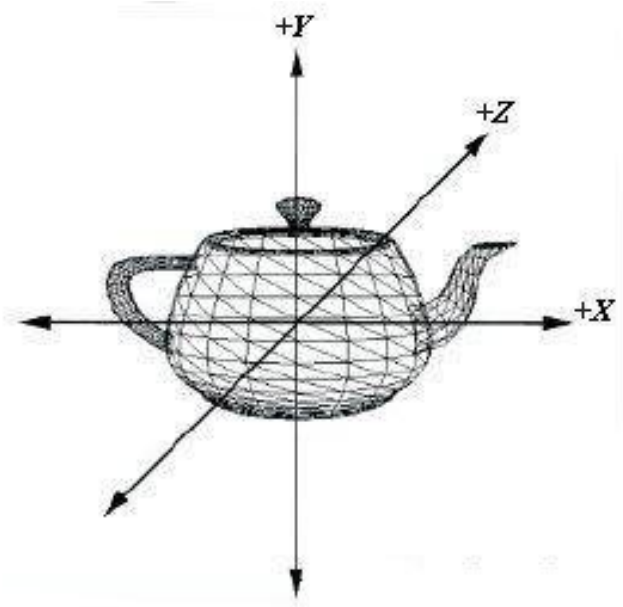


Рис.5.4. Простір об’єкта

2. світовий простір (рис.5.5) застосовується для розміщення об’єктів на сцені, здійснення афінних трансформацій (переміщення, повороту і масштабування об’єктів), опису освітлення сцени, розрахунок зіткнень між об’єктами під час моделювання динаміки їх руху і т.п. Це єдиний простір для всіх об’єктів сцени;

3. видовий простір (рис. 5.6) асоційований з віртуальним спостережником (звичайною камерою) чи певною проекцією сцени (наприклад, фронтальним виглядом) і описує ту частину сцени, яка доступна для перегляду і роботи у видовому вікні,

4. екранний простір (рис. 5.7) – це простір (площина), в якому відображаються аксонометричні чи перспективні проекції об’єктів на площину поверхні монітора,

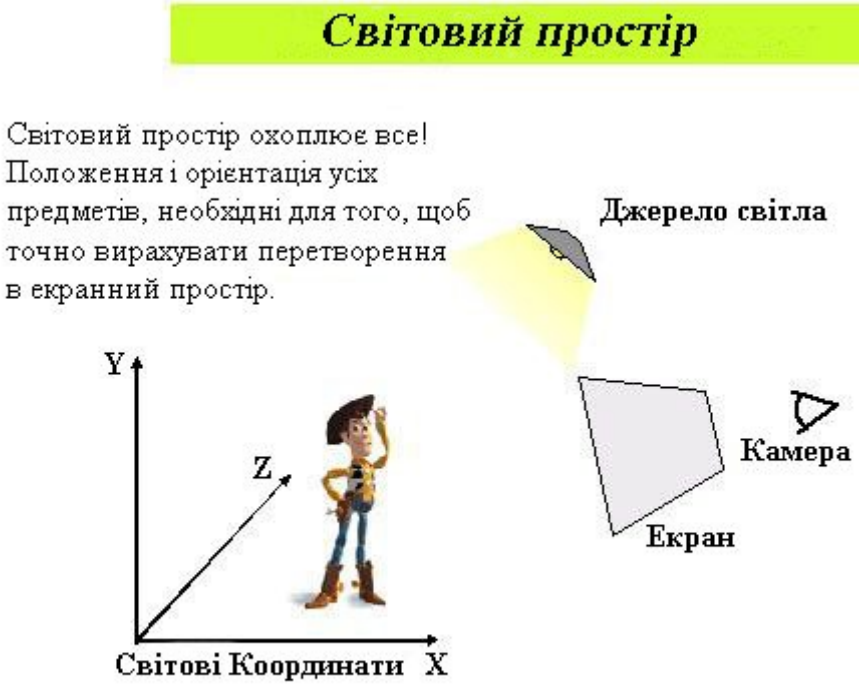


Рис. 5.5. Світовий простір

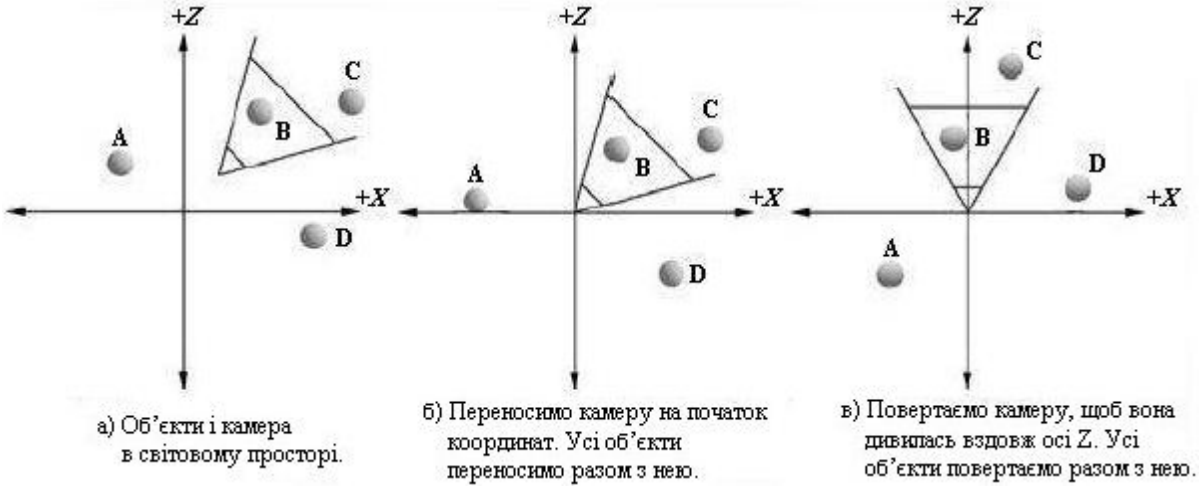


Рис. 5.6. Видовий простір

5. UVW-параметричний простір (рис. 5.8) застосовується під час математичного моделювання складних кривих і поверхонь (наприклад, NURBS-об'єктів) чи для задання UVW-координат тестування поверхонь.

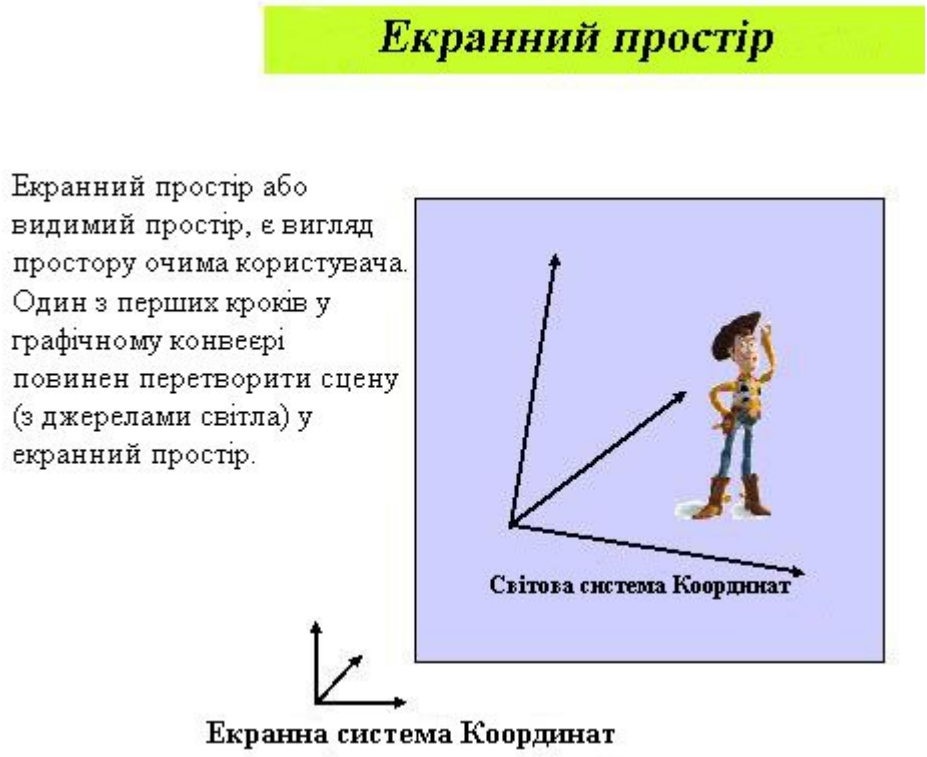


Рис. 5.7. Екранний простір

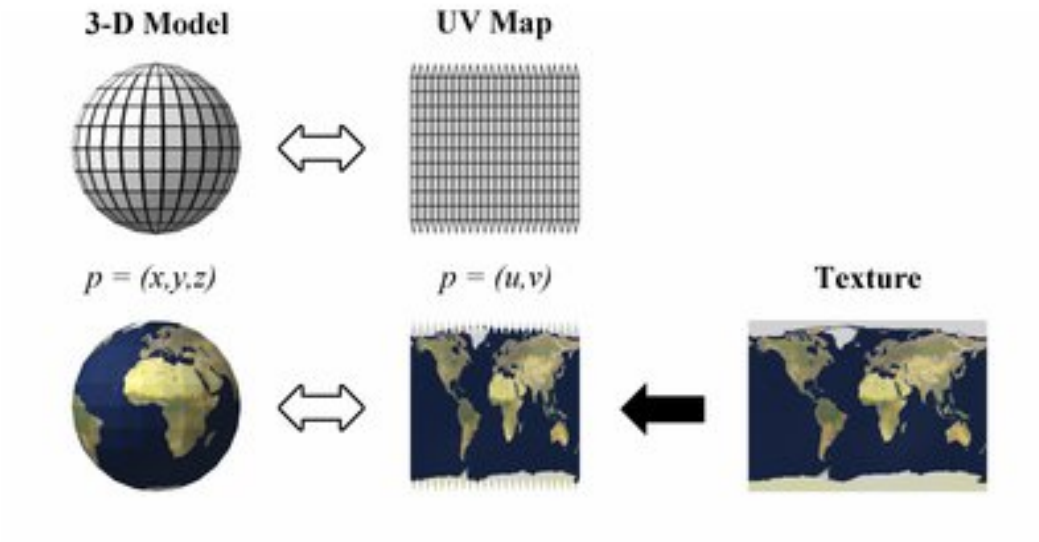


Рис. 5.8. UV- параметричний простір

5.3. Моделювання об’єктів

Класифікація створюючих тривимірних об’єктів

Всі створюючі тривимірні об’єкти можна розділити на:

1) геометричні,

2) негеометричні.

Геометричні об'єкти (рис. 5.9) застосовуються для моделювання об'єктів природного світу (персонажів, предметів).

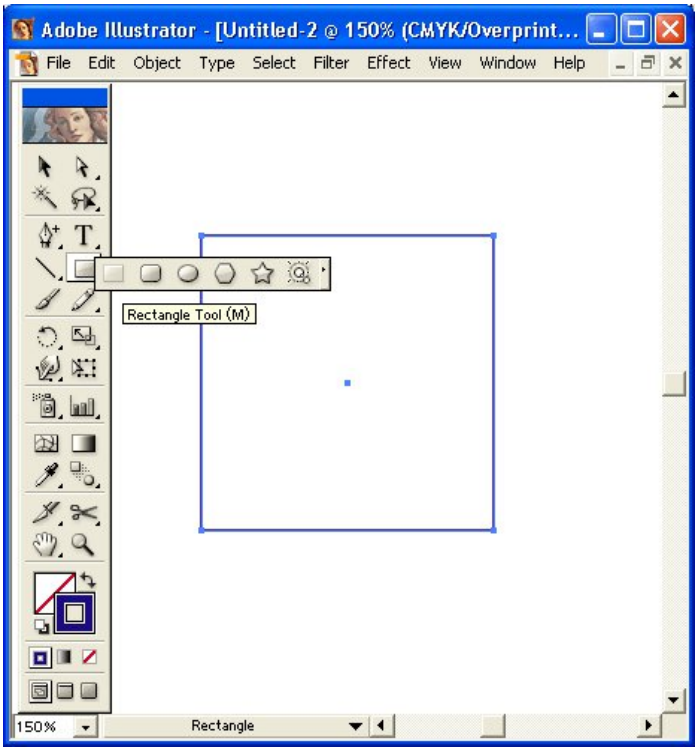


Рис. 5.9. Інструмент Rectangle в групі геометричних об'єктів

Негеометричні об'єкти (рис. 5.10) застосовуються для надання сцені реалістичності (наприклад, правильного освітлення), для моделювання фізичних сил, діючих на об'єкти (наприклад, гравітації чи поривів вітру).

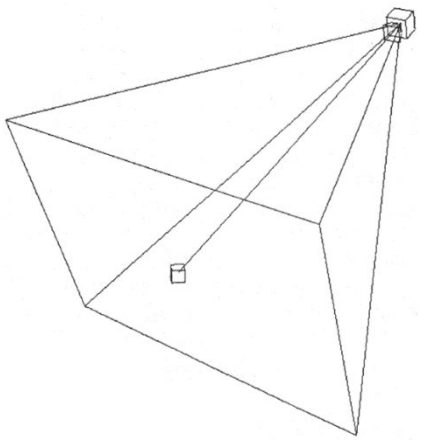


Рис. 5.10. Приклад негеометричного об'єкта (камера) в 3D редакторі

Іншими словами, геометричні об’єкти будуть присутні у візуалізованому кадрі явно (як криві чи поверхні), а негеометричні – опосередковано (у вигляді бліків, тіней, прискорень).

Основними типами геометричних об’єктів є: сплайн, полігональні об’єкти, поверхні Без’є, NURBS-поверхні, складові об’єкти, системи частинок, динамічні об’єкти.

Сплайн *– це гладка крива, яка проходить через дві чи більше контрольні точки, керуючі формою сплайну.*

Два з найбільш загальних типів сплайнів – криві Без’є (Bezier curves) і B-сплайни (B-spline curves). Типовим прикладом сплайнів є також неоднорідні раціональні B-сплайни (Non-Uniform Rational B-Spline) – NURBS (рис. 5.11).

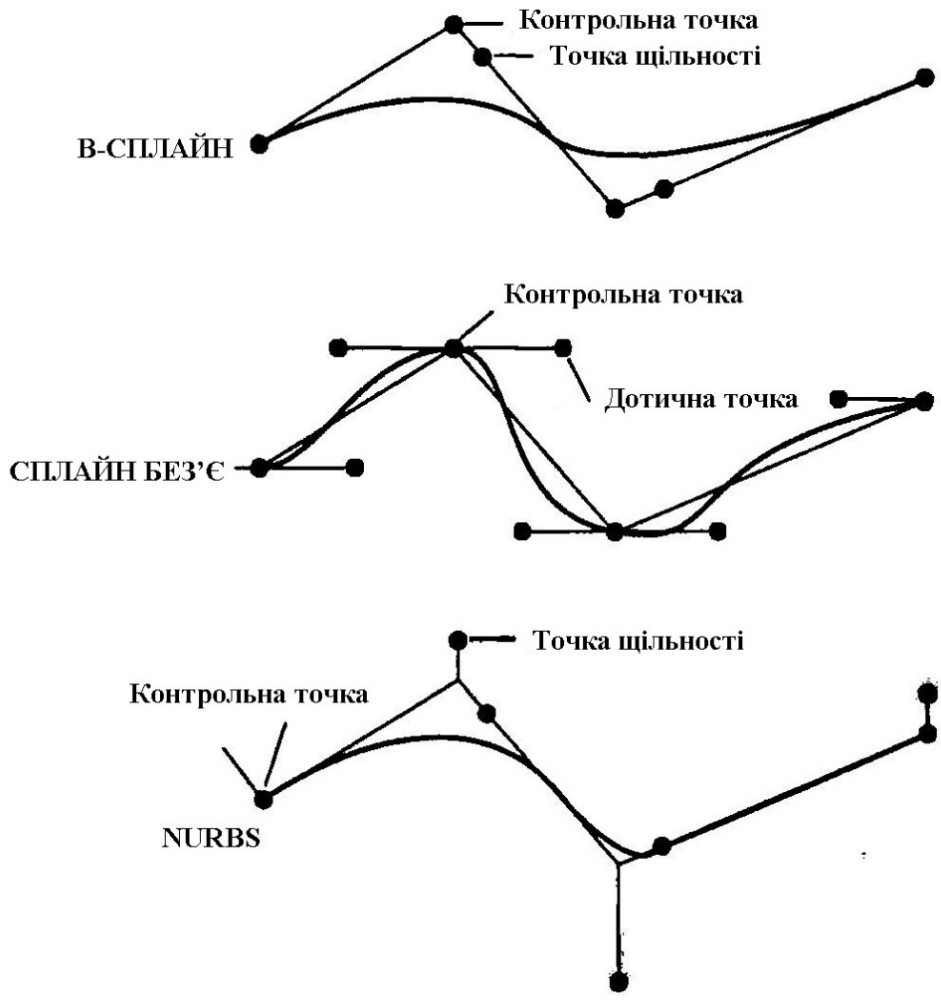


Рис. 5.11. Види сплайнів

Сплайни складаються з вершин (vertices) і сегментів (segments) (рис. 5.12). Кожна вершина сплайну має дотичні вектори (tangents), які містяться на кінцях керуючої точки, чи маркера (handels). Маркери дотичних векторів керують кривизною сегментів сплайну під час входження у вершину, які належать дотичним векторам, і під час виходу з неї.

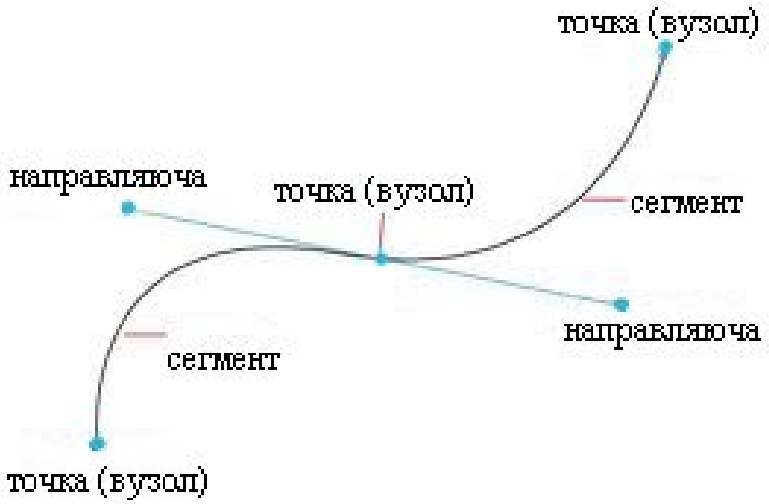


Рис. 5.12. Основні елементи сплайну

У залежності від властивостей дотичних векторів розрізняють такі типи вершин сплайнів (рис. 5.13): зі зламом (Corner); згладжена (Smoos); Без'є (Bezier); Без'є зі зламом (Bezier Corner).

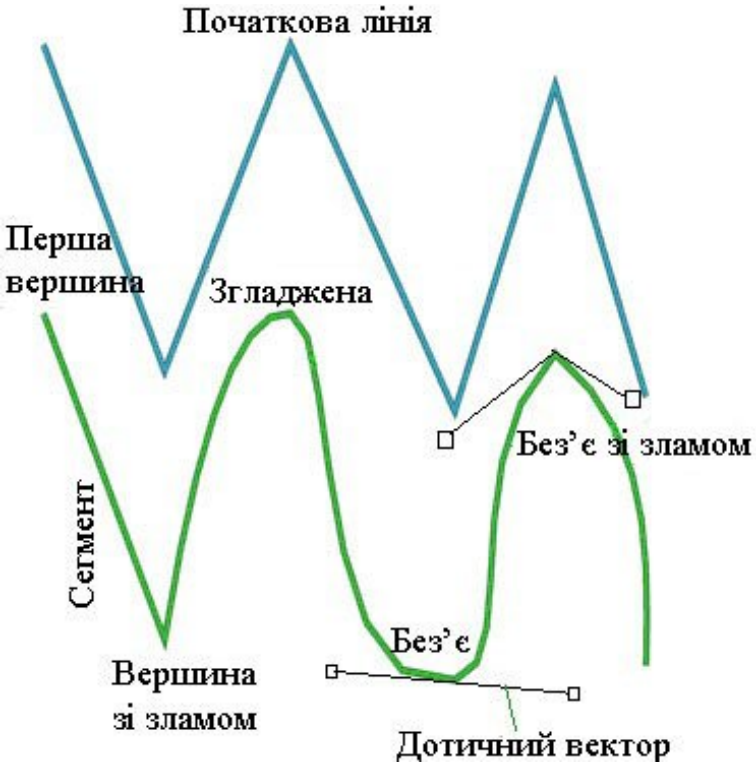


Рис. 5.13. Вершини сплайну

Сплайни можуть слугувати заготовками для побудови поверхонь або їх можна застосовувати в якості траєкторії руху.

Полігональні об'єкти *– це полігональні примітиви, які описуються набором параметрів що динамічно змінюються (наприклад, довжин, радіусів) або полігональними сітками, які визначаються як набір граней, обмежених ребрами, попарно з'єднуючими вершини.*

Polygon (багатокутник) – плоска фігура, обмежена з усіх сторін ломаною лінією. Трикутники, то б то прості тристоронні багатокутники формують основу, каркас об'єктів у тривимірному середовищі (рис. 5.14).

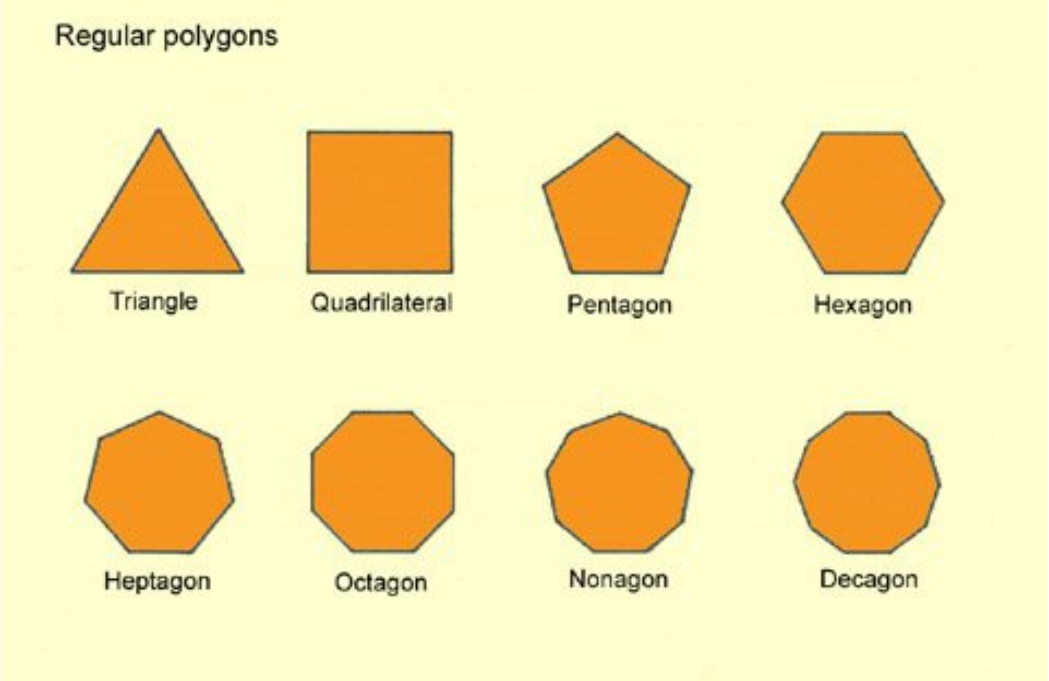


Рис. 5.14. Приклади полігонів

Поверхні Без'є *(рис. 5.15) – це математично гладкі поверхні, що описуються розміщенням вершин Без'є. Ці вершини визначають їх кривизну за допомогою додаткових керуючих точок на кінцях дотичних до поверхні векторів.*

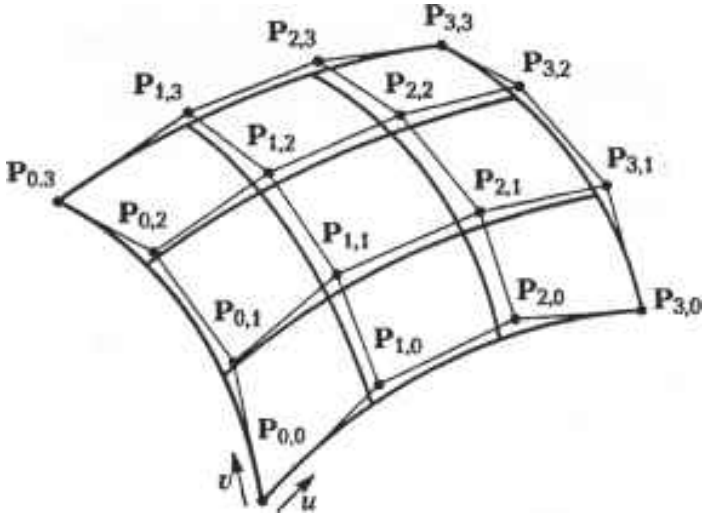


Рис. 5.15. Приклад поверхні Без'є

NURBS-поверхні *(рис. 5.16) – це найбільш універсальний і ефективний спосіб моделювання неоднорідних криволінійних поверхонь. Такі поверхні описуються в особливому чотирьохмірному гомогенному (однорідному) просторі, в якому кожна керуюча вершина, крім трьох координат x, y і z, має ще і додаткову вагову характеристику.*

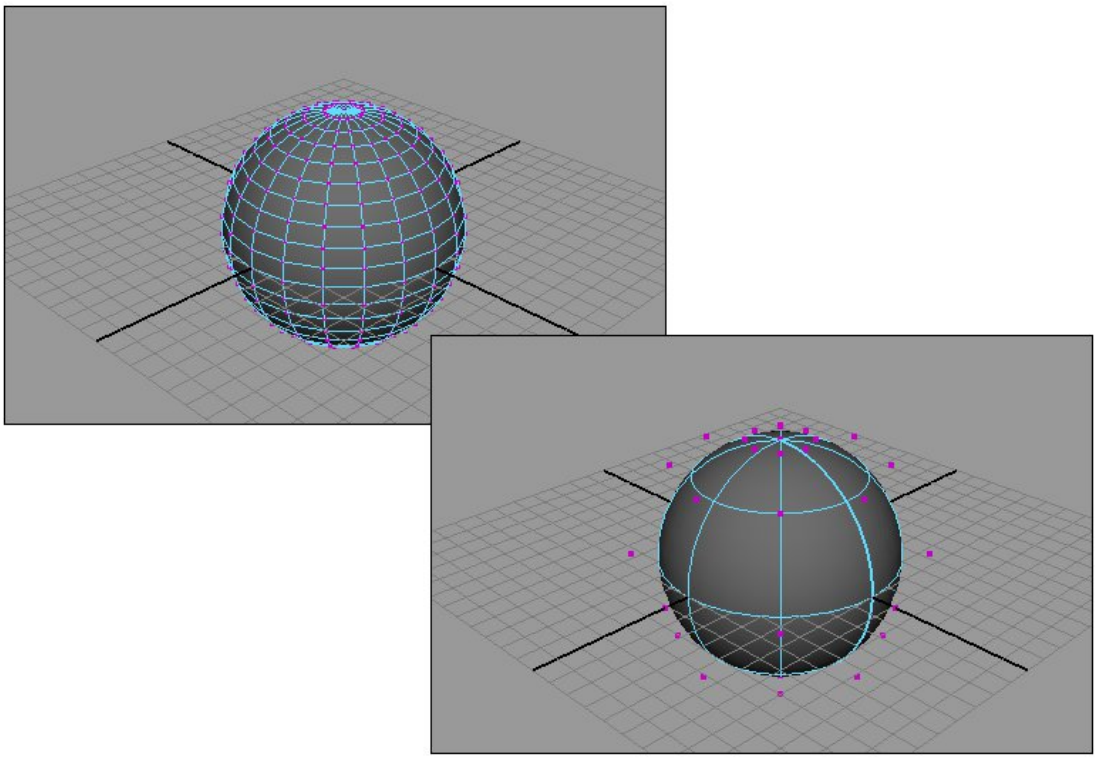


Рис. 5.16. Рівень деталізації полігональної сфери (зліва) і NURBS-сфери(справа) — видно, що число вершин в першої сфери на порядок вищий

Складові об'єкти *– являють собою комбінацію двох або більш змодельованих попередньо заготовок. У залежності від того, яке саме складове тіло створюється, заготовками можуть слугувати криві чи об'єми (поверхні).*

Системи частинок *– це об'єкти, генеруючі за заданим алгоритмом частинки з певною формою, початковою швидкістю, терміном життя і іншими характеристиками. Такі анімаційні об'єкти застосовуються для моделювання дощу, бульбашок газу в рідині, осколків снарядів, які вибухають і тому подібних взірців об'єктивної реальності.*

Динамічні об'єкти *– дозволяють моделювати об'єкти, реагуючі на прикладені до них внутрішні сили: пружини і амортизатори. Застосовуються під час моделювання динаміки руху об'єктів.*

До негеометричних об'єктів відносяться: джерела світла (рис. 5.17), камери, системи зчленування (рис. 5.18), викривлювачі простору (рис. 5.19).

Джерела світла – застосовуються під час моделювання зовнішнього і інтер'єрного освітлення. Різні типи джерел реалізують різні алгоритми розповсюдження світла.

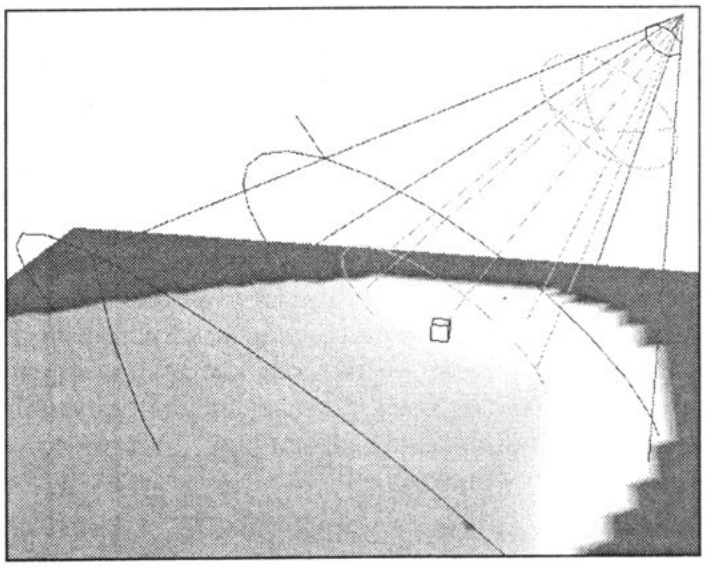


Рис. 5.17. Джерело світла

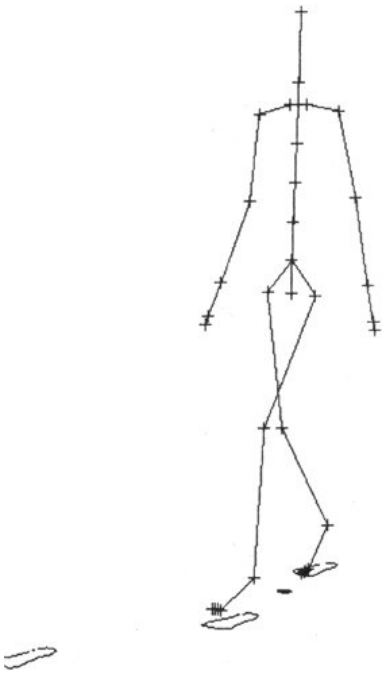


Рис. 5.18. Система зчленування

Матеріали і карти. Матеріали визначають візуальні властивості поверхонь, та описують те, як поверхня об'єкта взаємодіє з освітленням сцени.

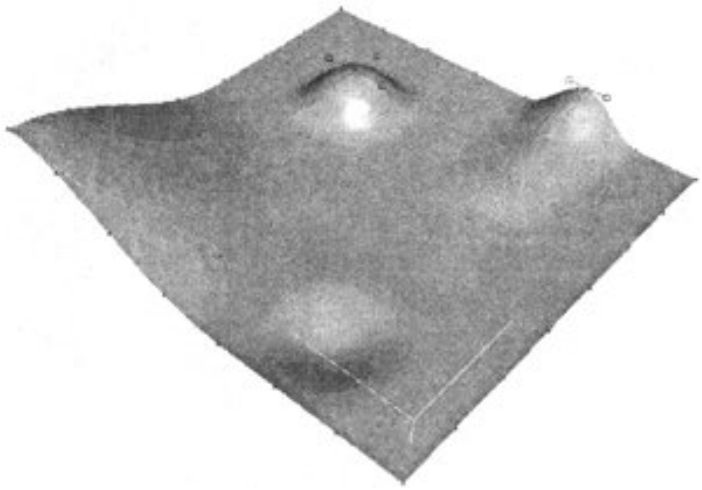


Рис. 5.19. Викривлювачі простору

Камери *– дозволяють повністю контролювати відображення об'єктів у площині кадру. Найважливішою характеристикою є фокусна відстань об'єктива камери, яка визначає її поле зору. Обидва ці параметри взаємопов'язані і вимірюються в міліметрах і градусах відповідно. Ще однією важливою характеристикою камери є площини відсіку****,*** *обмежуючі видиму за глибиною (відстані від спостерерігача частину сцени.*

Системи зчленування – *це структури****,*** *які складаються з ієрархічно зв'язаних «кісток», описуючих складну кінематику руху моделюючого об'єкта (наприклад, людину).*

Викривлювачі простору *– реалізують динамічні впливи зовнішніх сил на об'єкти, це – своєрідні силові поля, які впливають на певні об'єкти. Прикладами можуть слугувати хвильова деформація чи ударна хвиля, розкидаюча фрагменти об'єкта в просторі.*

Наступні властивості поверхонь визначають взаємодію матеріалу з світлом: колір, прозорість, глянцевість, коефіцієнт заломлення.

Тестування матеріалів. ***Тестування*** *– це основний метод моделювання поверхонь накладанням на них зображень, які називаються текстурою (рис. 5.20, рис.5.22).*

Текстура *(рис. 5.21) – це побітове відображення поверхонь, відскановане чи намальоване, надаюче поверхні реалістичний вид.*

Застосування текстурних карт (декоративних узорів, maps) дозволяє надати матеріалам додаткову реалістичність (наприклад, вигляд мощеною плиткою бруківки чи портмоне з крокодилової шкіри).

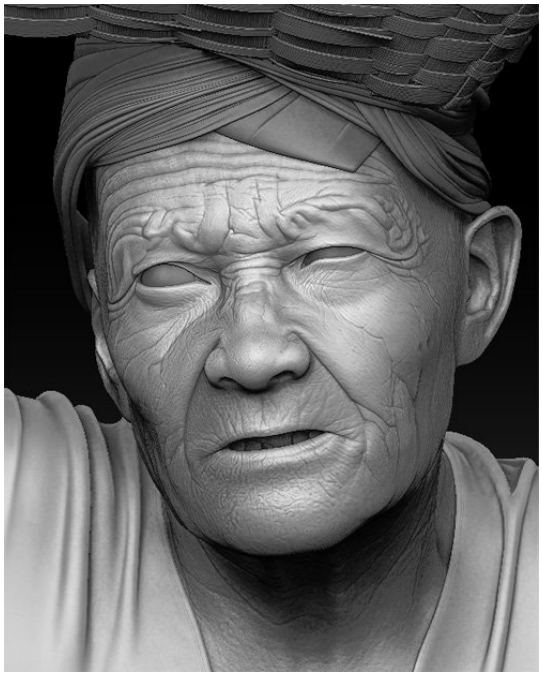


Рис. 5.20. 3D – модель без текстур

В якості карт можуть бути використані зображення, збережені у файлах різних форматів (BMP, TIP, JPG, EPS), чи процедурні текстури, які є наборами правил швидкої побудови потрібного візерунку.

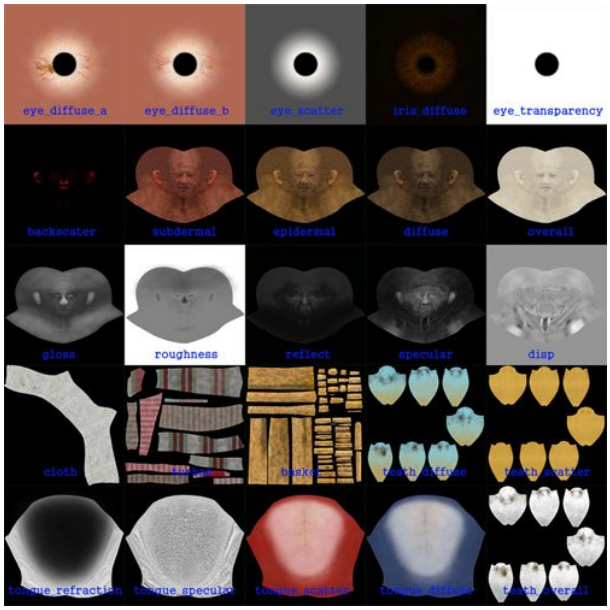


Рис. 5.21. Набір текстур

Додаткові властивості матеріалів. У залежності від конкретної програми трьохмірного моделювання пропонується різна сукупність додаткових властивостей матеріалів. Наприклад, пакет 3D Studio MAX дозволяє моделювати динаміку твердих тіл, що приводить у цьому випадку до необхідності задати коефіцієнти пружності матеріалу, сили тертя спокою і тертя ковзання (рис. 5.23).

Анімація. Базовим принципом комп'ютерної анімації (як, власне кажучи, і будь-якої іншої) є швидка зміна послідовності кадрів, фіксуючих проміжні фази руху, перед очима спостерігача.

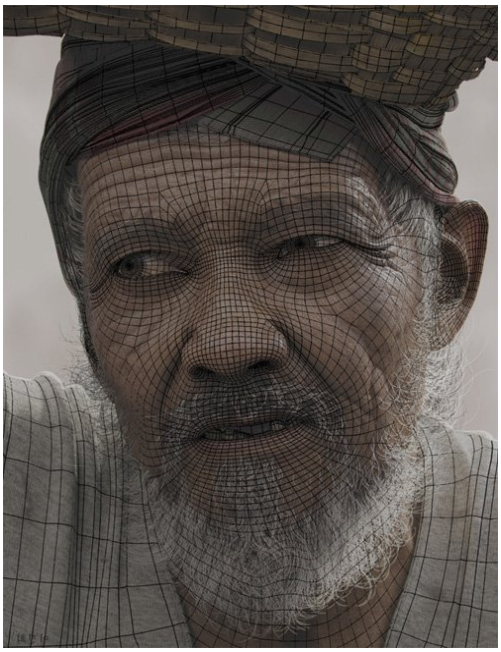


Рис. 5.22. 3D – модель з текстурами

Під рухом мається на увазі як безпосередньо переміщення чи поворот об'єкта в просторі сцени, так і будь-яку зміну його форми, кольору і т.п. Кадри повинні змінювати один одного під час перегляду з досить високою швидкістю, інакше в спостерігача не створиться ілюзії неперервності змін, що відбуваються.

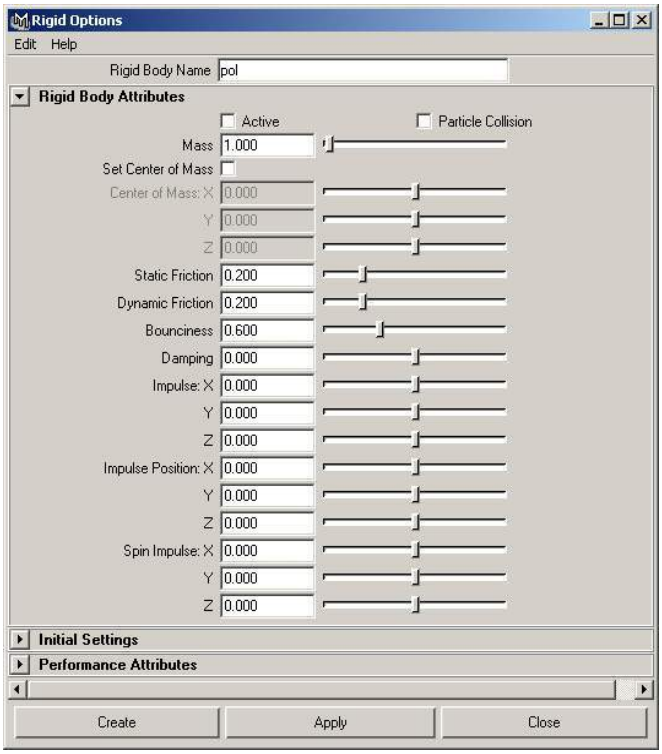


Рис. 5.23. Додаткові властивості матеріалу в 3D Studio MAX

Рендерінг. Кінцева реалізація (rendering) – заключний етап роботи зі сценою. До цього етапу сцена буде містити інформацію про геометрію об'єктів, їх матеріали і освітлення. Завдання модуля візуалізації в тому, щоб обчислити колір кожного пікселя кінцевого зображення, базуючись на інформації про моделі і вибране положення віртуального спостерігача (камери).

Колір кожної точки на поверхні обмальовуючого об'єкта розраховується, виходячи з фізичних властивостей матеріалу і освітлюючого його світла.

Рендерінг *– процес інтерпретації всього об'єкта і даних про його освітленість для створення закінченої картини у тому вигляді, в якому вона повинна виглядати на перспективі з вибраної точки зору.*

5.4. Редактори тривимірної графіки

Програма VRML, стала першою мовою тривимірного моделювання для Web. VRML 97 набула найбільшого поширення і має розширення VRL. Вона застосовує формат ASCII і являє собою звичайний текстовий файл із списком об'єктів, які названі вузлами (nodes). VRML використовує декартову тривимірну систему координат. VRML 2.0 має наступні елементи й механізми: елементи для представлення інформації про 2D – і 3D-примітиви; елементи для визначення характеристик цих примітивів; елементи для перегляду й моделювання 2D- і 3D-інформації; механізм для збереження й передачі мережами 2D- і 3D-даних; механізм для включення даних із інших метафайлів; механізм для визначення нових типів і форм інформації.

Пакет 3D Studio Max – це 32-розрядний, багатопоточний пакет, орієнтований на ринок інструментів старшого класу, де дуже важлива точність. Функція Track View керує анімаційними ефектами. Він володіє унікальною можливістю під назвою «історія даних»; ця функція дозволяє розглядати будь-який етап роботи незалежно від того, як давно цей етап був виконаний. 3ds Max 2010 – це повнофункціональне рішення для 3D моделювання, анімації і рендерингу, що застосовується у розробці найпопулярніших ігор, зйомці фільмів і відеопрограм.

Пакет TrueSpace 3.0 компанії Caligari є сумісним з VRML 2.0 і містить вбудований браузер VRML, а також анімацію, інверсну кінематику і функцію відображення колізій (коли два об'єкти, які перетинаються під час руху реалістично взаємодіють між собою). У програмі реалізоване затінення Фонга і використовується засіб рендерингу з трасуванням променів. Програма дозволяє маніпулювати об'єктами у повністю в тривимірній перспективі.

Перевагами пакета Extreme 3D компанії Macromedia є інтеграція з іншими продуктами компанії Macromedia й можливість виконувати рендеринг зображень у

змішаних багатоплатформенних мережах. Він дозволяє пареміщувати двовимірні профілі в Extreme 3D з пакету FreeHand і застосовувати до них тривимірні ефекти з іншими продуктами компанії Macromedia й можливість виконувати рендеринг забражень у змішаних багатоплатформенних мережах, а також переміщувати двовимірні профілі в Extreme 3D з пакету FreeHand і застосовувати до них тривимірні ефекти.

Пакет Sculpt D популярний серед виробників цифрових ілюстрацій, які розробляють складні скульптурні об’єкти й виконують рендеринг результатів із високою роздільною здатністю під час виводу на друк. Програма надає більше 1,5 кубічних метрів робочого простору із точністю у межах 0.01 мм.

У пакет Ray Dream Studio компанії Fractal Design входять програми Ray Dream Designer 4.1 і Ray Dream Animator, які не продаються окремо.

LightWave 3D 5.0 — це одна з небагатьох дійсно незалежних від платформи програм. Пакет підтримує трасування променів, тому дає можливість керувати переломленням світла і глибиною поля. Версія для Macintosh підтримує технологію QuickDraw 3D, яка дозволяє реалізувати рух графіки з тінями у режимі реального часу.

Пакет Strata StudioPro — це професійний повнофункціональний інструмент, який володіє широким набором можливостей з моделювання, рендерингу і анімації тривимірних композицій в інтерактивному режимі. Ця програма, існуюча тільки у версії для Macintosh, дозволяє виводити файли VRML і підключати URL або адресу AppleScript до моделей VRML. Вона дозволяє створювати тривимірні моделі і застосовувати до них фотореалістичні текстури, а також ефекти освітлення; для рендерингу використовується засіб трасування променів. Крім того, програма пропонує засоби створення анімації з можливістю обробки подій.

Maya є програмою для створення тривимірної графіки і анімації, заснованих на моделях, створених користувачем у віртуальному просторі, освітлених віртуальними джерелами світла і показаних через об'єктиви віртуальних камер. Програма дозволяє створювати фотореалістичні растрові зображення, подібні тим, які отримуються за допомогою цифрової камери. При цьому робота над будь-якою сценою починається з порожнього простору. Будь-який параметр можна змусити змінюватися з часом. У результаті після візуалізації набору кадрів виходить анімована сцена.

Maya працює як на комп'ютерах PC з операційною системою Windows Nt/2000, так і в операційних системах Linux, IRIX, Macintosh.

Прикладами застосування Maya є створення мультфільмів «Життя комах», «Шрек».

Існують такі версії цієї програми, як: Maya Complete, Maya Unlimited, Maya Builder, Autodesk Maya. Autodesk Maya представляє собою інтегроване програмне рішення для 3D моделювання, анімації і рендерингу, засноване на відкритій архітектурі. Ці можливості у вигляді єдиного додатка представляють виняткову

цінність для фахівців із комп'ютерної графіки, виробників комп'ютерних ігор, яким потрібен високий рівень контролю за процесами створення продукції. Продукт для 3D анімації Autodesk Maya 2011 забезпечує покращену взаємодію з користувачами всіх підтримуючих платформ (оперативних систем Mac OS X, 32 - і 64-розрядної Windows та 64-розрядної Linux). Версія відрізняється оновленим для користувача інтерфейсом з його дизайном, можливістю прикріплення елементів, більш гнучкими засобами редагування, інструментами вибору кольору, оглядачем файлів, покращеними процедурами анімації персонажів і взаємодії з видовим екраном, можливостями 3D монтажу, вбудованими функціями керування кольором. Завдяки новому засобу задання послідовності кадрів (Camera Sequencer) використовуються в одній анімації кадри з різних камер. Ця програма дозволяє швидко виконувати покращений скіннінг персонажів з більш реалістичними деформаціями. Покращення прийнятих процесів роботи з об'єктами, здійснюється за допомогою створення об'єктів з трансформаціями (DAG-об'єкти), які можна розміщувати в просторі, встановлювати батьківсько-дочірні відносини між ними та здійснювати їх входження та пов'язування з джерелами світла.

Зараз основними конкурентами Maya є програми Lightwave, Softimage XSI і 3ds max, вартість яких складає від 2000 до 7000 доларів. Серед програм, що коштують менше 1000 доларів, можна вказати TrueSpace, Inspire 3d, Cinema 4d, Bryce і Animation Master. Існує навіть безкоштовна програма для роботи з тривимірною графікою, що називається Blender. Більшість зазначених редакторів добре працюють на базі персональних комп'ютерів і мають версії для різних операційних систем, таких як Macintosh.

Blender — є пакетом, що включає засоби моделювання, анімації, рендерингу, післяобробки відео, а також створення інтерактивних ігор. Програма є вільним програмним забезпеченням і розповсюджується під ліцензією GNU GPL. Blender був розроблений як робочий інструмент голландською анімаційною студією NeoGeo і розповсюджувалася за принципом shareware.

Особливостями пакету є малий розмір (біля 10 МБ), висока швидкість рендерингу, наявність версій для безлічі операційних систем таких, як: FreeBSD, GNU/Linux, Mac OS X, SGI Irix 6.5, Sun Solaris 2.8 (sparc), Microsoft Windows, SkyOS, MorphOS та Pocket PC. До особливостей редактора відносяться функції динаміки твердих і м'яких тіл, рідин, наявність системи гарячих клавіш, велика кількість легкодоступних розширень, написаних на мові Python. У базове постачання не входять розгорнена документація та велика кількість демонстраційних сцен, існує хороший механізм експорту в різноманітні формати, такі як obj, dxf, stl, 3ds та інші.

Blender мав репутацію програми складної для вивчення. Практично кожна функція мала відповідне їй поєднання клавіш і кожна клавіша включена в більш ніж одне поєднання (shortcut). З моменту відкриття джерельних кодів Blender зазнав переробки основного коду програми. Це зробило процес додавання нових

можливостей набагато легшим. І до програми були додані повні контекстні меню до усіх функцій, поліпшено користувацький інтерфейс із введенням колірних схем, прозорих плаваючих елементів, нову систему проглядання дерева об'єктів та інші зміни.

Хоча Blender повнофункціональна програма, проте вона має недолік процесу моделювання на основі N-Gon і дещо незавершений набір інструментів моделювання, методи чисельного вимірювання та маніпулювання, неможливість побудувати поєднання клавіш на свій розсуд, обмежену сумісність з іншими 3d форматами файлів, систему моделювання складного руху одягу (зараз знаходиться в розробці) і дещо незручне представлення бібліотек матеріалів.

Версія Blender 2.40 має велику кількість нових можливостей: оновлену систему анімації; менший стек модифікаторів; поліпшення в Інтерфейсі Користувача; нову систему частинок (включаючи волосся) and guides, а також динаміку рідин і покращені інструменти булевого моделювання (Boolean Modelling). У той же час новіша версія програми Blender 2.41 володіє GLSL піксельними та вершинними шейдерами для Ігрового Двигунця (Game Engine), Subsurf UV Unwrapping and a sculpting tool. Вийшла у світ новіша версія цього редактора, а саме Blender 2.42. Значним професійним проектом, у ході якого був використаний Blender є Людина-Павук 2.

Отже, враховуючи особливості пакетів вони знайшли широке застосування в тривимірній графіці. Провести їх порівняльний аналіз досить складно, хоча в основному, чим складніша програма, тим більш складну анімацію вона дозволяє створювати і тим простіший в ній процес моделювання складних об'єктів або процесів.

Питання для самоперевірки

1. Дайте визначення двовимірної і тривимірної графіки.
2. Здійсніть порівняльний аналіз двовимірної і тривимірної графіки.
3. Вкажіть основні функції 3-D графіки.
4. Охарактеризуйте варіанти просторів, які найчастіше надають програми тривимірного моделювання.
5. Здійсніть порівняльний аналіз створюючих тривимірних об’єктів.
6. Вкажіть основні типи геометричних об’єктів.
7. Назвіть основні види негеометричних об’єктів.
8. Які редактори застосовуються у 3-D графіці?

Список рекомендованої літератури

1. Блінова Т.О., Порєв В.М. Комп’ютерна графіка / За ред. В.М. Порєва. – К.: Видавництво “Юніор”, 2004. – 456 с.

2. Веселовська Г.В., Ходаков В.Є., Веселовський В.М. Комп’ютерна графіка: Навч.посібник для студентів вищих навчальних закладів / Під ред. В.Є. Ходакова. – Херсон: ОЛДІ-плюс, 2004. – С. 199–218

3. Глушаков С.В., Крабе Г.А. Компьютерная графика. – Харьков: Фолио, 2002. – 500 с.

4. Ренди Дж. Рост. OpenGl. Трехмерная графика и язык программирования шейдеров. Для профессионалов. – Питер, 2005. – 432 с.

РОЗДІЛ 6. КОЛІР У КОМП'ЮТЕРНІЙ ГРАФІЦІ

6.1. Основні відомості про колір і колірні моделі

Колір, особливо відмінності в кольорі – це є здатність організму реагувати на різні довжини електромагнітних хвиль.

Колірне охоплення екрану монітора, офсетного друку чи фотознімка набагато менше, ніж людського ока. Їх колірне охоплення – це діапазон кольорів, що можуть бути відтворені, зафіксовані чи описані будь-яким чином.

У зв'язку з різницею в колірних охопленнях різних пристроїв із різними принципами утворення кольору, для передачі й одержання зображень були створені різні моделі кольорів. Оскільки жодна з моделей не є ідеальною і повною за колірним охопленням.

Призначенням колірної моделі надання засобам опису кольору в межах деякого колірного охоплення і забезпечення інтерполяції кольорів для взаємозв'язку між пристроями з різними колірними охопленнями і способами отримання кольору [2].

Колірні моделі описують колірні відтінки за допомогою змішування декількох основних кольорів. Основні кольори розбиваються на відтінки за яскравістю (від темного до світлого), і кожній градації яскравості привласнюється цифрове значення (наприклад, найтемнішій – 0, найсвітлішій – 255). Таким чином, будь-який колір можна розкласти на відтінки основних кольорів і позначити його набором цифр – колірних координат.

Під час вибору колірної моделі можна визначати тривимірний колірний координатний простір, усередині якого кожен колір представляється точкою. Такий простір ***називається простором колірної моделі.***

Професійні графічні програми зазвичай дозволяють оперувати з декількома колірними моделями, більшість з яких створена для спеціальних цілей чи особливих типів фарб: RGB, CMY, CMYK, Lab, HSV/HSB і HLS.

6.2. Колірна модель RGB

Геніальний «локус», був затверджений Міжнародною Освітлювальною Комісією, – МОКНУВ багато десятків років тому назад. Принципом роботи локуса було складання будь-якого кольору не з семи барв веселки, а всього з трьох R – red (червоний), G – green (зелений), B – blue (блакитний), тобто RGB.

Стисло історія системи RGB така. Томас Юнг (1773-1829) узяв три ліхтарі і пристосував до них червоний, зелений і синій світлофільтри. Так були отримані

джерела світла відповідних кольорів. Жовтий колір давало змішування червоного й зеленого, блакитний – зеленого й синього, пурпуровий – синього і червоного, а білий колір – усіх трьох основних кольорів. Джеймсом Максвеллом (1831-1879) був виготовлений перший колориметр, за допомогою якого порівнювалися монохроматичний колір і змішування в заданій пропорції компонентів RGB. Регулюючи яскравість кожного з компонентів, можна домогтися вирівнювання кольорів суміші й монохроматичного випромінювання.

До теперішнього часу система RGB є офіційним стандартом. Рішенням Міжнародної Комісії з Освітлення – МКО у 1931 році були стандартизовані основні кольори, які було рекомендовано використовувати в якості R, G і B, рис. 6.1.

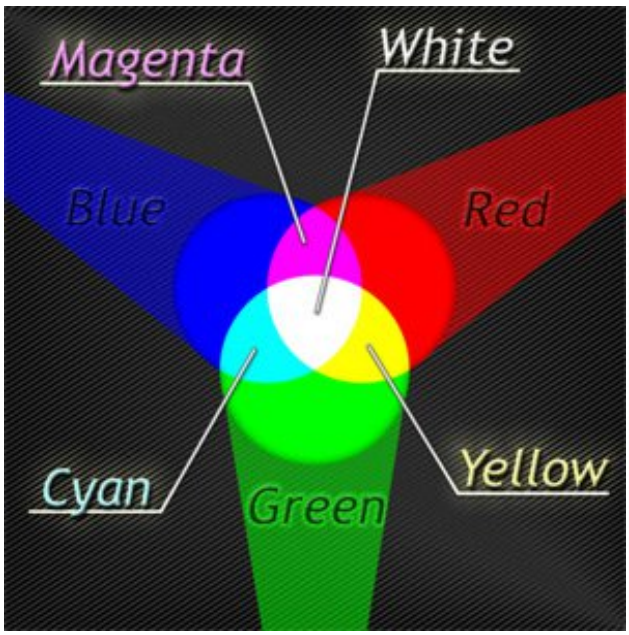


Рис. 6.1. Комбінація базових кольорів колірної моделі RGB

RGB знайшла широке застосування в пристроях, що світяться (телевізійні кінескопи, комп'ютерні монітори), а також зображень, що отримуються під час сканування. По замовчуванню екран монітора працює саме в цьому діапазоні. Серйозною проблемою є великий колірний обхват і апаратна залежність (не зовсім аналогічний показ кольорів на різних у основному ЕЛТ-моніторах).

Оскільки монітори різних моделей і виробників відрізняються, було запропоновано декілька альтернативних колірних просторів, відповідних «усередненому» монітору. До таких відносяться, наприклад sRGB, Adobe RGB, Wide-Gamut RGB.

sRGB має найменший кольоровий обхват і тому підходить для тих, хто працює з веб-графікою, друкує на струменевих принтерах. Однак для професійної якості друку вона не придатна. Adobe RGB 1998 – отриманий з телестандартів, вважається найбільш оптимальним видом під час роботи з графічними пакетами. Останній Wide-

Gamut RGB має найбільший обхват і може бути застосований до 48-розрядних робіт. Монітор комп'ютера має інший принцип показу кольорів, і тому модель RGB (з її трьома видами), для друку майже не придатна.

Кольори цієї моделі називають адитивними. Оскільки кольори в RGB генеруються підсумовуванням світлових потоків.

Колірний простір моделі RGB іноді представляють у вигляді колірного куба (рис. 6.2, 6.3). По осях відкладаються значення колірних каналів, кожен з яких може приймати значення від 0 до 255. Всередині куба знаходяться всі кольори цієї моделі. У точці початку відліку координатних осей усі значення каналів дорівнюють нулю (чорний колір), а в протилежній по діагоналі куба точці максимальні значення каналів під час змішування утворюють білий колір. Якщо ці дві точки з'єднати відрізком, то на ньому буде розташована шкала відтінків від чорного до білого – сіра шкала. Три вершини куба дають три чистих вихідних кольори. У той же час, кожна з трьох інших вершин між ними дає чистий, змішаний з двох основних, колір. Графічні програми дозволяють комбінувати необхідний RGB-колір з 256 відтінків червоного, зеленого і синього, що в сумі становить 16,7 мільйонів кольорів.

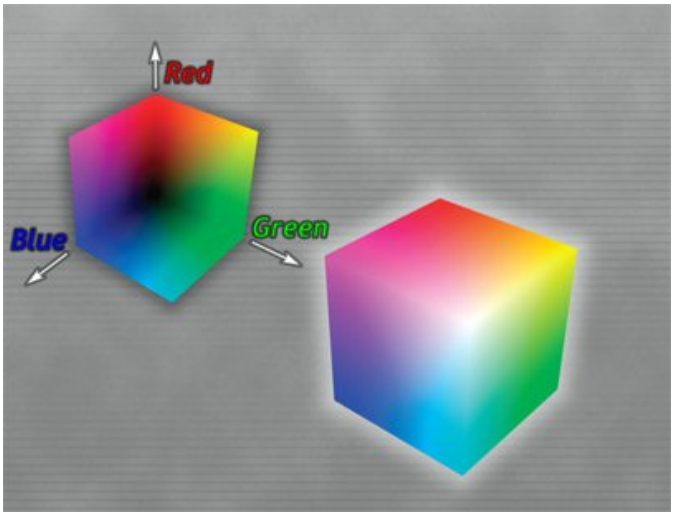


Рис. 6.2. Колірна модель RGB представлена у вигляді куба

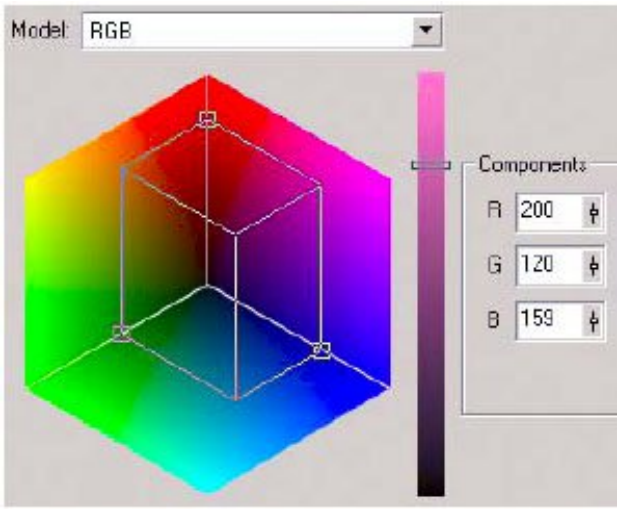


Рис. 6.3. Колірна модель RGB у графічному вигляді

Модель RGB залежна від пристрою. Під час заміни пристрою змінюються і кольори. Вона не дуже підходить для відтворення кольору, коли в одному комплексі повинні працювати сканер, принтер і монітор. Оскільки модель використовує три первинні кольори, вона не підходить для розфарбовування чи для фарбників і пігментів, які використовуються під час друку, оскільки ті застосовують інший набір первинних кольорів (блакитний, пурпуровй, жовтий).

Фахівці ізраїльської компанії Genoa Color Technologies розробили алгоритм перетворення RGB-кольору в новий стандарт, названий MPC (multi-primary colour,

тобто – множинні первинні кольори), і розробила чіпи, що проробляють таке перетворення на апаратному рівні.

Відтворення шести сигналів яскравості має здійснюватися на новій техніці, що має шість первинних кольорів для кожного пікселя.

Найлегше таке перетворення можна зробити з проекційними телевізорами, заснованими на мініатюрних рідкокристалічних панелях або так званих мікродзеркальних процесорах (або проекторах DLP). Все, що потрібно – це замінити кольорові фільтри на ЖК-панелі.

У принципі аналогічно можна модифікувати комп'ютерні дисплеї на рідких кристалах, що випускаються, і плазмові панелі. Але краще – створювати під новий стандарт нові моделі.

Мікросхема працює з роздільною здатністю до 1280 на 1024 точок при частоті оновлення екрану 60 герц (у співпраці з комп'ютерною технікою – дисплеями, проекторами), а також з телевізійною прогресивною (у сенсі – відрядковою) розгорткою стандарту 720 p (один із стандартів телебачення високої чіткості). Схема підтримує до 8 біт представлення яскравості для кожного з 6 базових кольорів. Genoa вже виготовила проекційний телевізор, що працює в новій системі і виміряла його параметри. Відмічалася надзвичайна точність передачі природних кольорів і правильний білий колір. І крім цього новий стандарт дав можливість підвищити яскравість зображення на 40% без втрати колірного обхвату. Технологія «звертається» до засобів відображення, не зачіпаючи джерела сигналу, і повністю сумісна з алгоритмами компресії MPEG-2 і стандартом DVD.

Ізраїльські науковці розробили алгоритм перетворення RGB-кольору в новий стандарт «множинні первинні кольори» і розробили чіпи, що здійснюють таке перетворення на апаратному рівні. При цьому для кожного пікселя повинно бути шість базових кольорів, а комп’ютерні дисплеї і плазмові панелі потрібно модифікувати, що в свою чергу призведе до створення нових колірних моделей.

6.3. Колірна модель CMY і CMYK

На відміну від колірної моделі RGB моделі CMY застосовуються в поліграфії для друкуючих пристроїв, фотонабірних автоматів.

Кольори цієї моделі засновані на відніманні частини спектра падаючого світла (білого) і називаються субтрактивними. Канали CMY являють собою залишок віднімання основних RGB компонентів з білого кольору (як відомо, білий колір складається з повного спектра кольорів). Під час змішування двох основних кольорів результат виявиться темнішим за кожен із вихідних, оскільки кожен колір поглинає свою частину спектра. При цьому утворюються такі кольори: Cyan – блакитний (білий колір мінус червоний), Magenta – пурпуровий (білий мінус зелений), Yellow –

жовтий (білий мінус синій), рис. 6.4. Кольори, які виходять змішуванням базових фарб у субтрактивній системі, ***називаються складеними чи тріадними*** (process color). Складеними тому, що виходять змішуванням основних складових, а прикметник «тріадний» знаходить своє пояснення в технології друку. Такі кольори утворюються в результаті трьох прогонів друкарського верстата, і в кожному з них друкарська машина наносить необхідну кількість жовтої, пурпурової і блакитної фарби. Тільки її розуміють растрові процесори.

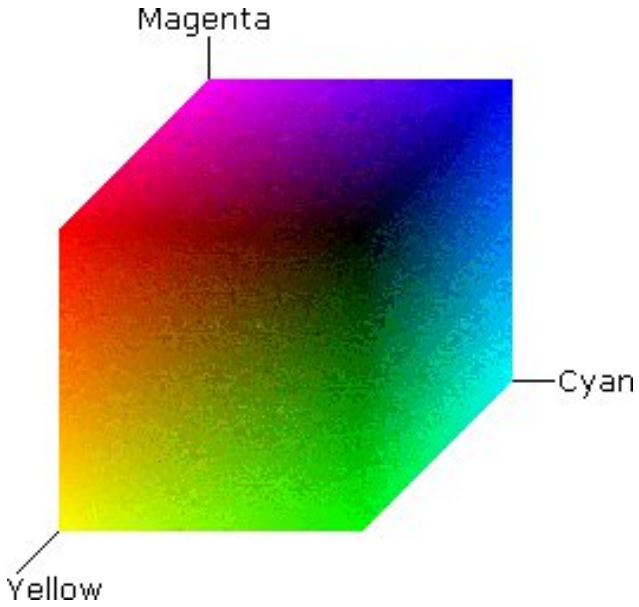


Рис. 6.4. Колірна модель CMY представлена у вигляді куба

Оскільки для відтворення чорного кольору потрібне нанесення трьох фарбників, а матеріали дорогі, використання CMY-моделі є неефективним. Крім того, під час застосування цієї моделі існують небажані візуальні ефекти, які виникають за рахунок того, що під час виведення точки три базові кольори можуть лягати з невеликими відхиленнями. Тому до базових кольорів CMY-моделі додали чорний (black). Завданням якого стало підсилити поглинання світла в темних частинах, зробити їх максимально чорними, тобто збільшити тоновий діапазон друку. Система CMY з додатковою чорною складовою називається CMYK. Чорний колір (Black) представлений в назві останньою буквою для того, щоб не плутати його в скороченнях і абревіатурах із синім (Blue), рис. 6.5. Ця система слугує теоретичною основою цифрового друку. Колірні координати розглядаються як фарбники, які наносяться на поверхню паперу, тому інтенсивність кожної координати вимірюється у відсотках від 0 (відсутність фарби) до 100 (максимальна щільність фарби).

Змішування всіх трьох компонентів за максимальних значень дає чорний колір. З іншого боку, при повній відсутності фарби і, відповідно, нульових значеннях основних компонентів, отримаємо білий колір. Для моделі CMYK білий колір варто сприймати як колір чистого білого паперу. Під час змішування основних компонентів

із рівними значеннями одержуємо відтінки сірого кольору і утворюється сіра шкала (діагональ куба), рис. 6.6.

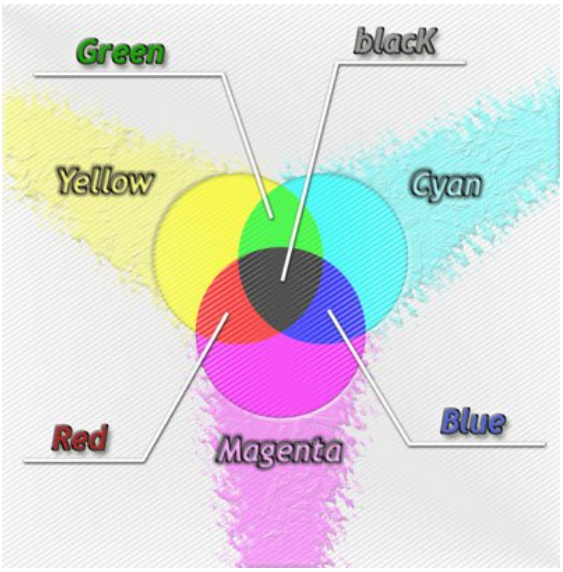


Рис. 6.5. Комбінація базових кольорів колірної моделі CMYK

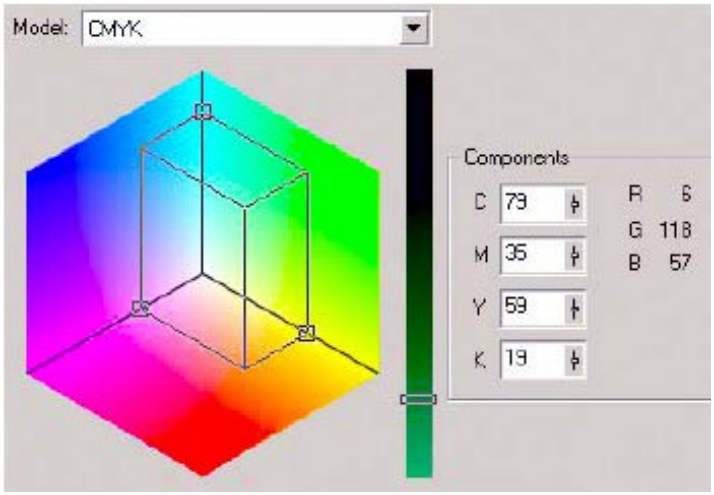


Рис. 6.6. Колірна модель CMYK у графічному вигляді

Проте модель CMYK має вузький колірний обхват, тому що вона описує відбиті кольори, інтенсивність яких завжди менша, ніж у випромінюючих. Коли виникає необхідність переведення зображення в CMYK і назад у RGB, то найімовірніше можна сказати, що це призведе до погіршення якості зображення, оскільки багато фільтрів растрових програм у цій моделі не працюють, на 30% потрібний більший об'єм пам'яті в порівнянні з моделлю RGB. Як і модель RGB, модель CMYK є апаратнозалежною.

Отже, колірні моделі CMY і CMYK призначені підготувати зображення і вивести його на друк. Застосування CMY доцільне лише теоретично для чорно-білих принтерів, де картридж можна замінити на кольоровий. Додавання чорної фарби дозволило зробити модель CMYK повністю функціональною (але не досконалою) в кольоровому друці. Також покращилась якість виводу діапазону сірих відтінків. Як і RGB, CMYK залишається апаратно залежною, з недостатньо великим кольоровим діапазоном моделлю. Однак при всіх своїх недоліках ця модель досить достатньо відображає необхідний для друку спектр, але разом із тим може нести в собі неадекватну кольоропередачу на виводі, тому деякі зображення краще з самого початку редагувати в ній. Якість, яка отримується під час друку, залежить від якості паперу. У професійній поліграфії CMYK майже не застосовують, а користуються різними її модифікаціями (системи Pantone, Trumatch і ін.) інтегрованими в серйозні графічні програми.

Зручним геометричним способом представлення взаємодії основних колірних моделей RGB і CMYK є колірний круг, рис. 6.7. У ньому на рівній відстані один від одного розміщені адитивні і субтрактивні кольори.

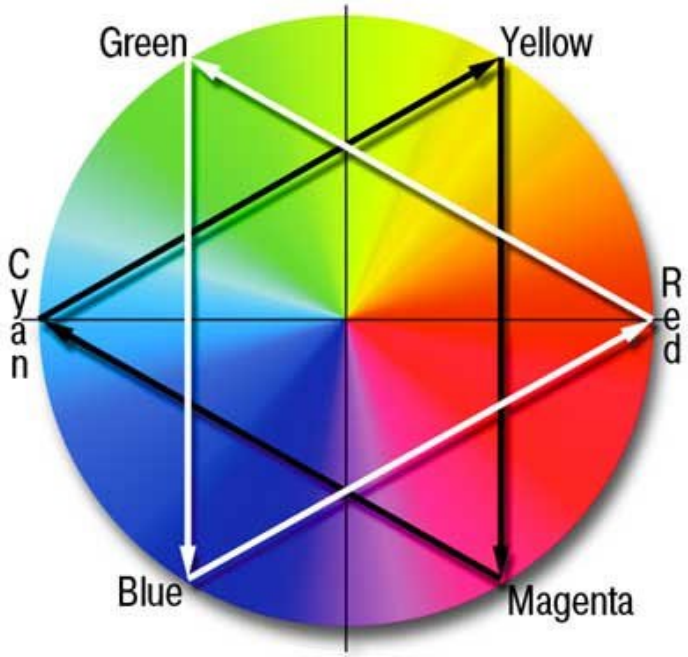


Рис. 6.7. Колірний круг

Пари кольорів, розташовані під кутом 180◦ ***називаються компліментарними чи додатковими.***

У різних джерелах наводяться різні зображення колірного круга. Ці відмінності не мають принципового значення і не впливають на прогностичні властивості моделі.

Основними положеннями колірного синтезу по круговій моделі є: додавання будь-якої фарби колірного круга компенсує додаткову фарбу, ніби розбавляє її в результуючому кольорі. Наприклад, щоб змінити колірне співвідношення у бік зелених тонів, слід знизити кількість пурпурового кольору, який є додатковим до зеленого. Підвищення кількості компонентів RGB спричиняє за собою зниження концентрації параметрів CMY і навпаки; кожен субтрактивний (адитивний) колір знаходиться між двома адитивними (субтрактивними). Складання будь-яких двох кольорів RGB дає колір CMY, що лежить між ними. Справедливе і зворотне твердження. Наприклад, Red+Green=Yellow, Blue+Green = Cyan, Red +Blue = Magenta, Cyan +Magenta = Blue, Cyan +Yellow = Green, Magenta +Yellow = Red; освітлення чи затемнення кольору граничної насиченості спричиняє зниження його насиченості.

Отже, колірна модель RGB широко застосовується для відображення на моніторі і після сканування, CMY і CMYK - для друкуючих пристроїв, фотонабірних автоматів.

6.4. Колірна модель HSV/HSB

Колірна модель HSV/HSB є нелінійним перетворенням колірної моделі RGB. Вона широко застосовується дизайнерами і художниками завдяки практичній універсальності для кольорокорекції. У всіх графічних додатках ця модель присутня в тому чи іншому вигляді.

HSV/HSB не залежить від устаткування і зручна для сприйняття людиною. Оскільки людина інтуїтивно сприймає колір, розділяючи його на відтінок, насиченість і яскравість. Саме тому з нею часто працюють різні програми, надалі перетворюючи кольори в модель RGB для показу на екрані монітора чи в модель CMYK для друку на принтері. Крім того, модель HSV/HSB зручно використовувати при редагуванні малюнків.

Отже, компонентами моделі HSV/HSB є Hue (тон або відтінок), Saturation (насиченість) і Value (значення кольору, колірний тон, кількість світла чи світлота), Brightness (яскравість).

Модель HSV/HSB зручно представляти у вигляді кольорового круга, уздовж якого розташовуються відтінки кольорів. Значення кольору вибирається як точка на крузі (чи вектор, що виходить з центру окружності і вказує на дану точку), рис. 6.8. Різні відтінки розміщуються по окружності, яка становить 360◦. Кольори йдуть у спектральному порядку і замикаються пурпуровим.

Модель HSV/HSB має досить широкий колірний обхват. Він не такий великий, як у Lab, але більший ніж у CMYK.

Модель HSV/HSB зручно представляти у вигляді колірного круга. Значення кольору вибирається як точка на колі (чи вектор, що виходить з центра окружності і вказує на дану точку). Різні відтінки розміщуються по окружності, яка становить 360◦. Зокрема, 0 – червоний, 60 – жовтий, 120 – зелений, 180 – блакитний, 240 – синій, 300 – пурпуровий, тобто додаткові кольори розташовані один проти одного відрізняються на 180º, рис. 6.8. Застосовується спосіб представлення цієї колірної моделі у вигляді світлової шестигранної піраміди. При цьому по вертикальній осі відкладається значення V, а відстань від осі до бічної грані в горизонтальному перетині відповідає параметру S (за діапазон зміни цих величин приймається інтервал від нуля до одиниці), рис. 6.10. Шестикутник, що лежить в основі піраміди, являє собою проекцію колірного куба в напрямку його головної діагоналі, перевернутий шестигранний конус. Тон кольору H задається кутом, відкладеним навколо вертикальної осі відрахованим від червоного. Точки на самій окружності відповідають чистим (максимально насиченим) кольорам. Точка в центрі відповідає нейтральному кольору мінімальної насиченості (білий, сірий, чорний – це залежить від яскравості). Тобто можна сказати, що кут нахилу вектора визначає відтінок, довжина вектора – насиченість кольору. Величина S змінюється від нуля на осі конуса, до одиниці на його гранях. Значенню V=0 відповідає вершина піраміди (чорний колір), значенню V=1 – основа піраміди; кольори при цьому найбільш

інтенсивні. Точка з координатами V=1, S=0 – центр основи піраміди відповідає білому кольору. Проміжні значення координати V при S=0, тобто на осі піраміди, відповідають сірим кольорами, якщо S=0, то значення відтінку H вважається невизначеним, а S=1, якщо точка лежить на бічній грані піраміди.

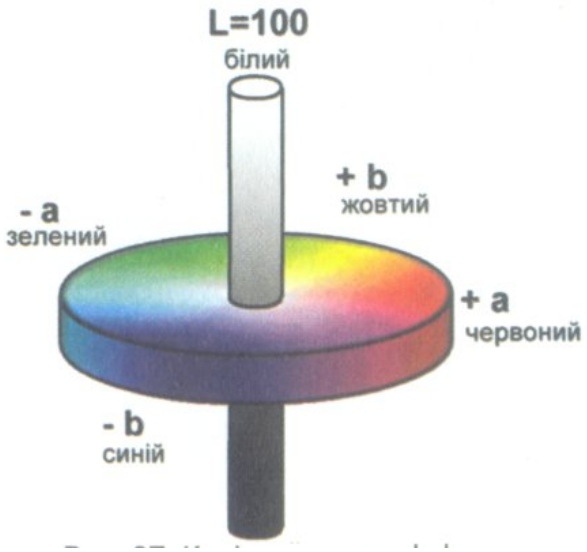


Рис. 6.8. Графічне представлення колірної моделі HSV/HSB

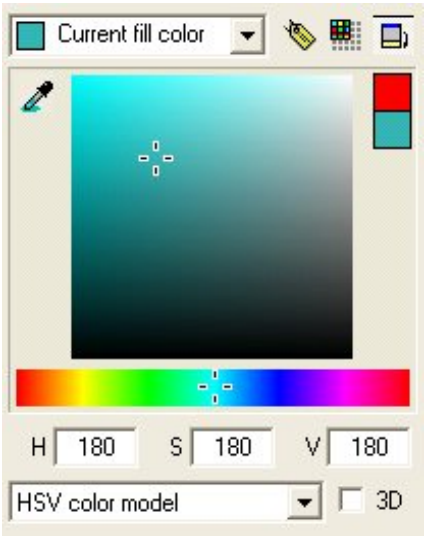
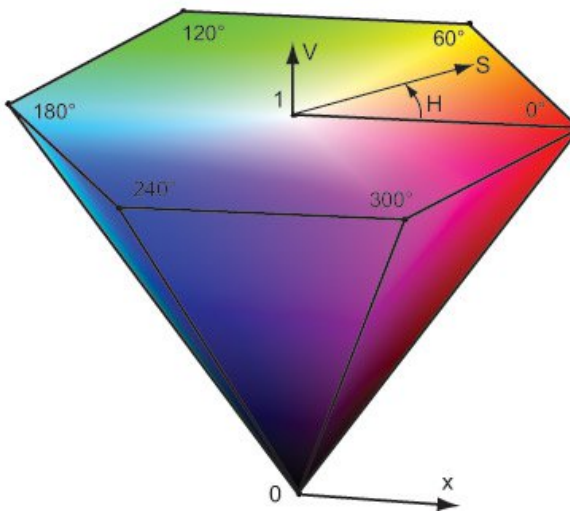
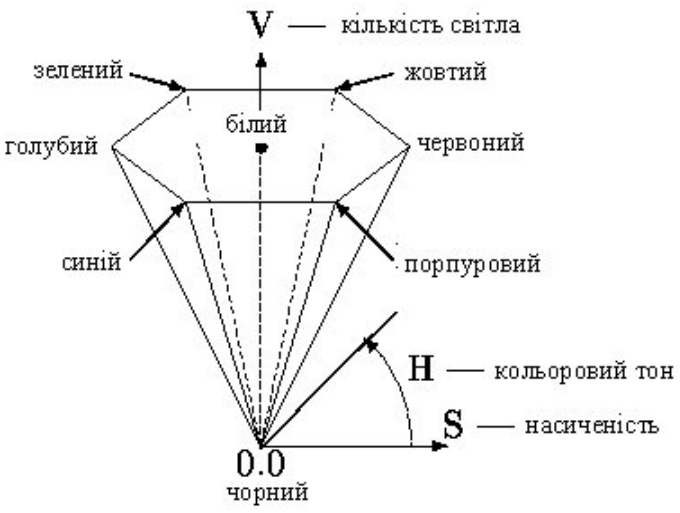


Рис. 6.10. Утворення кольору в HSV/HSB моделі



а)

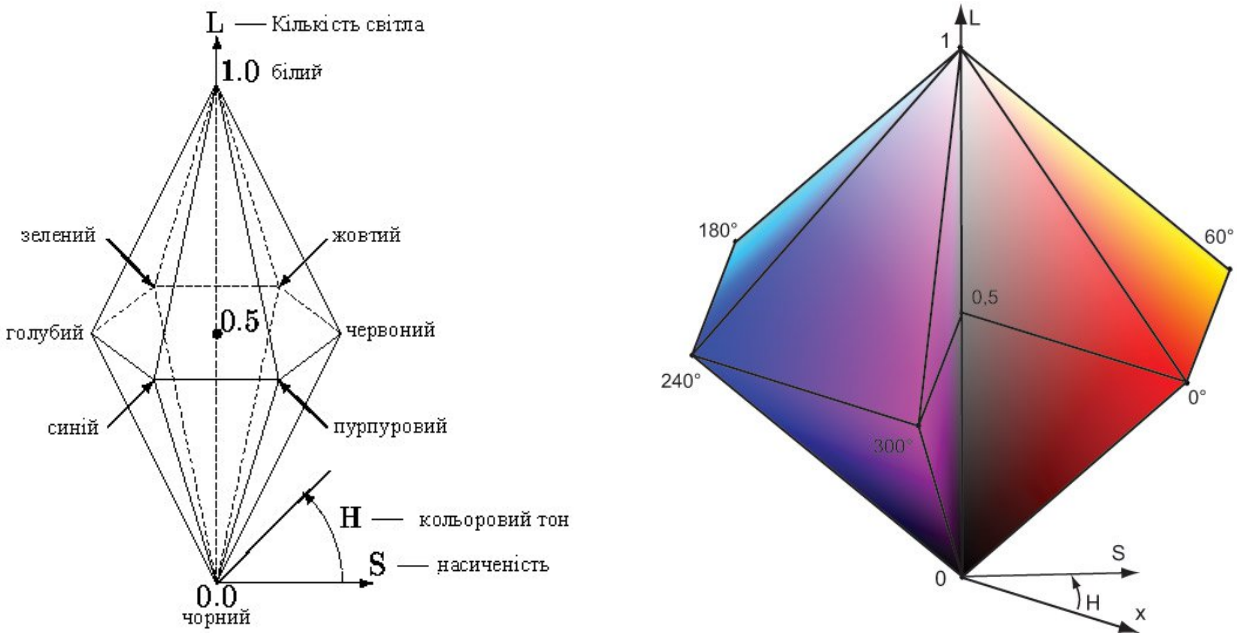


б)

Рис. 6.9. Колірна модель HSV представлена у вигляді світлової шестигранної піраміди

6.5. Колірна модель HLS

Розширенням колірної моделі HSV/HSB є HLS. Параметрами якої є Hue, Lightness, Saturation, відповідно: кольоровий тон, світимість, насиченість, рис. 6.11.



а) б)

Рис. 6.11. Колірна модель HLS у вигляді світлової подвійної шестигранної піраміди

Різниця між моделями HSV/HSB і HLS полягає в заміні нелінійного компоненту яскравість на лінійний компонент освітлений. А також в основній моделі HSV/HSB є власна яскравість об'єкта (ніби приймаємо його за джерело світла), а в HLS враховується світимість об'єкта (яскравість відбитого від нього світла). Іншими словами, в HSB «джерело» – Сонце, а в HLS – Місяць... В основі колірної моделі HLS лежить система Освальда. Ця модель утворює простір у формі подвійного конуса. Колірний тон задається кутом повороту навколо вертикальної осі конусів. За початок відліку прийнятий синій колір. Кольори йдуть у спектральному порядку і замикаються пурпуровим. Отже, по вертикальній осі відкладається L (світимість), а інші два параметри задаються як і в HSV/HSB. Ця модель утворює підпростір, що представляє собою подвійний конус, у якому чорний колір задається вершиною нижнього конуса і відповідає значенню L=0, білий колір максимальної інтенсивності задається вершиною верхнього конуса і відповідає значенню L=1. Максимально інтенсивні кольорові тони відповідають основі конусів з L=0,5, що не зовсім зручно. Тон кольору H, аналогічно системі HSV/HSB, задається кутом повороту. Насиченість

S змінюється в межах від 0 до 1 і задається відстанню від вертикальної осі L до бічної поверхні конуса. Тобто максимально насичені кольори розташовуються при L=0,5 і S=1.

6.6. Колірна модель Lab

У комп'ютерній графіці широко застосовується кольорова модель Lab. Вона була розроблена в 1931 році. У 1976 році модель Lab була вдосконалена і названа CIE Lab. Оскільки модель апаратно-незалежна, то вона відтворює одні й ті ж кольори незалежно від особливостей пристрою (монітора, принтера чи комп'ютера).

Модель Lab базується на людському сприйнятті кольору. При однаковій інтенсивності око людини сприймає зелений колір променів найбільш яскравим, дещо менш яскравим – червоний колір, і ще більш темним – синій. Яскравість при цьому є характеристикою сприйняття, а не характеристикою самого кольору. Під час розробки моделі Lab переслідувалася мета математичного коректування нелінійності сприйняття кольору людиною.

Цій моделі віддають перевагу в основному професіонали, оскільки вона суміщає переваги як CMY, так і RGB, а саме забезпечує доступ до всіх кольорів, працюючи з досить великою швидкістю. А також вона відрізняється трохи незвичайною побудовою ніж у інших колірних моделях. Побудова кольорів тут, так як і в RGB, базується на злитті трьох каналів.

Будь-який колір у колірній моделі Lab обумовлюється параметрами L – яскравість (Lightness) і хроматичними складовими – двома декартовими координатами: **a, (**змінюється від зеленого до червоного, через сірий) і **b (**змінюється від синього до яскраво жовтого через сірий), рис. 6.12–6.14. Lightness здійснює контроль за яскравістю кольорів, утворених a і b. Білий колір співставляється з максимальною інтенсивністю. **L** лежать в межах 0-100, а **a i b** – 200–200. Якщо a і b рівні 0, змінюючи L, отримуємо зображення, що містить градації сірого (grayscale). Під час змішування двох кольорів результуючий буде більш яскравим, що є ще однією подібністю з колірною моделлю RGB.

Професіонали застосовують цей колірний простір навіть для створення складних масок і кардинальних змін кольорів документа. Оскільки модель має величезний колірний обхват, перевід у неї не пов'язаний з втратами. Можна в будь-який момент перевести зображення з RGB в Lab і назад, і при цьому його кольори не зміняться.

На відміну від кольорових просторів RGB або CMYK, які є, по суті, набором апаратних даних для відтворення кольору на папері чи на екрані монітора (колір може залежати від типу друкарської машини, марки фарб, вологості повітря в цеху чи виробника монітора і його настройок), Lab визначає колір. Оскільки, яскравість у

моделі Lab цілком відділена від кольору, то це робить модель зручною для регулювання тонової характеристики (підвищення контрасту, виправлення похибки тонових діапазонів) і видалення кольорового шуму (у т.ч. розмивка растру і видалення регулярної структури зображень у форматі JPEG), різкості та інших тонових характеристик зображення.

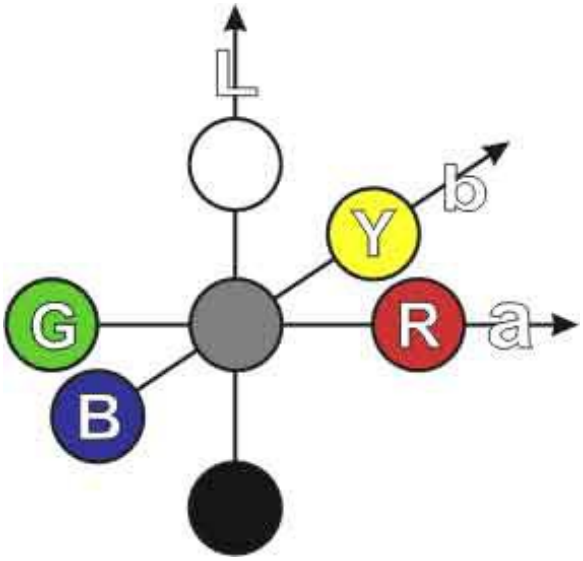


Рис. 6.12. Графічне представлення колірної моделі CIE Lab

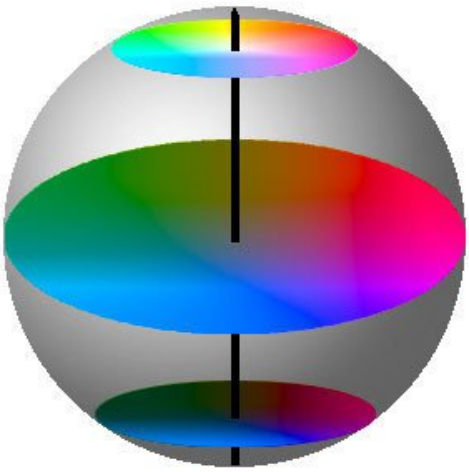


Рис. 6.13. Графічне представлення колірної моделі Lab

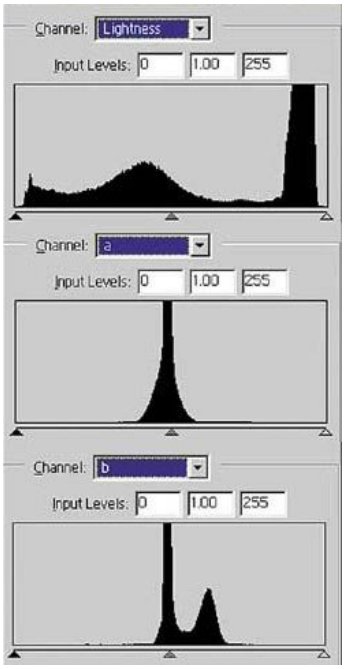
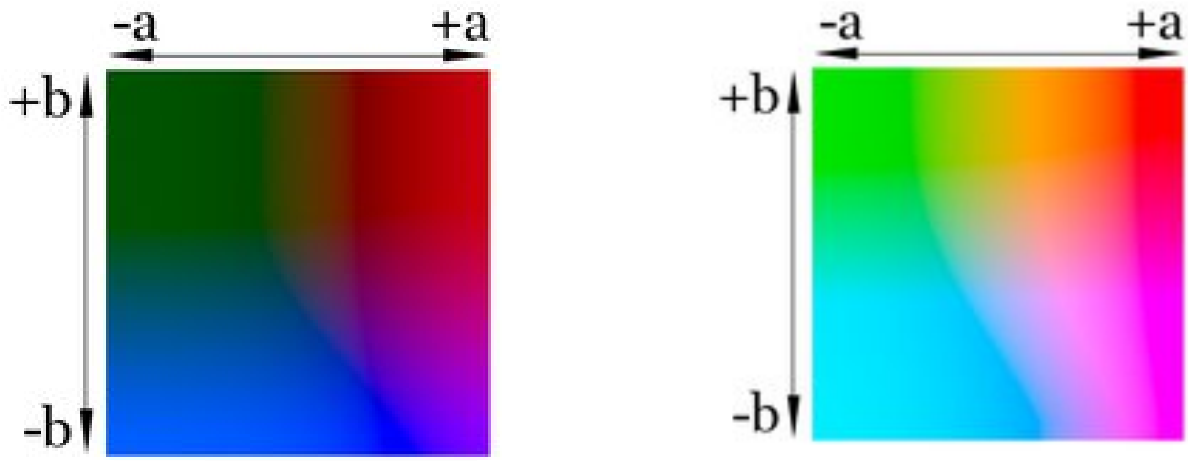


Рис. 6.14. Канали моделі Lab



Яскравість 25% Яскравість 75%

Рис. 6.15. Приклад зміни яскравості в кольорій моделі Lab

Враховуючи позитивні характеристики, а саме величезний колірний обхват, модель Lab знайшла широке застосування в програмному забезпеченні для обробки зображень як проміжна колірного простору, через яку відбувається конвертація даних між іншими кольоровими просторами під час їх підготовки (наприклад, з RGB сканера в CMYK друкованого пристрою). При цьому особливі властивості Lab зробили редагування в цьому просторі потужним інструментом корекції кольору.

Зображення, що готуються для публікації в Інтернеті, прийнято створювати в так званій безпечній палітрі («web-safe» palette) кольорів. Вона є варіантом індексної палітри. Але файли зображень у Web-графіці повинні мати мінімальний розмір, тому необхідно відмовитися від включення в їх склад індексної палітри. Для цього була прийнята єдина фіксована палітра кольорів, названа безпечною, тобто така, що забезпечує правильне відображення кольорів на будь-яких пристроях (і в програмах), підтримуючих єдину палітру. Безпечна палітра містить всього 216 кольорів, що пов'язане з обмеженнями, вимогами сумісності, накладаючими, з комп'ютерами, що не відносяться до класу IBM PC.

Для створення кольорів, відсутніх у індексній палітрі, застосовують спеціальний метод Error Diffusion Dithering**.** Це метод імітації (Dithering), в якому передача кольорового півтону в точковому зображенні, відсутнього в стандартній палітрі, досягається за рахунок змінних пікселів двох або доступніших кольорів. Оптичне зміщення кольорів у оці глядача, що не має можливості розглянути окремо суміжні пікселі, створює ілюзію присутності на малюнку кольору, насправді відсутнього в палітрі. Для кольорів, що сильно відрізняються від кольорів палітри, ця техніка приводить до створення зернистих зображень.

Отже, колірні моделі HSV/HSB, HLS широко застосовуються для роботи дизайнерів і художників, Lab – для обробки зображень як проміжна колірного простору через яку відбувається конвертація даних між іншими кольоровими

просторами. Для подання графічних об’єктів у Інтернеті набула поширення спеціальна палітра «web-safe» palette.

6.7. Чорно-біле зображення

6.7.1. Створення зображень

Існують різні методи створення зображень – вручну, на дисплеї, з використанням програм графічного редагування та верстки. Для делікатних графічних робіт існує безліч спеціальних пристроїв – мишки у формі товстого олівця, різноманітні креслярські планшети, екрани, чутливі на дотик. Спеціальні програми сприймають зазначені координати та супровідні команди, і зберігають утворені малюнки у растрових (BMP, GIF, JPEG, PCX, PNG, PSD, TIFF та ін.) або векторних (SVG, SVGZ, EPS, WMF, EMF, CDR, CMX, AI та ін.) форматах.

Існують такі типи зображень як: монохромні, кольорові, растрові, векторні та ін.

Монохромне зображення *– це зображення, що містить колір одного спектра (довжини хвилі), що сприймається, як один відтінок (на відміну від кольорового зображення, що містить різні кольори). Монохромними зображеннями, наприклад, є малюнки тушшю, олівцем або вугіллям, чорно-білі фотографії, зображення на екрані чорно-білими телевізорами або комп'ютерними моніторами (незалежно від істинного кольору їх свічення).*

Кольорове зображення – це зображення, що містить кольори різних спектрів.

Растрове зображення – це зображення, що являє собою масив пікселів.

Векторне зображення – це зображення, що складається з об'єктів (ліній, кіл, кривих, багатокутників), які можна описати математичними рівняннями.

Чорно-біле зображення є растровим. Існує два типи растрових зображень, які можна відносити до монохромних: бінарне і півтонове.

Бінарне (двійкове) зображення іноді називають «монохромним» і «чорно-білим», що в загальному випадку не правильно і веде іноді до плутанини. Бінарне растрове зображення може бути монохромним у випадках, коли: один вид пікселів (не важливо «0» або «1») відображається абсолютно чорним кольором, а інший будь-яким іншим (наприклад, «чорно-білий» растр, «чорно-зелений» і т. д.); обидва види (і «0», і «1») відображаються одним відтінком, але з різною яскравістю (наприклад, «темно- та світло-сірий», «яскраво-зелений і темно-зелений», «синій і сірий» і т. д.). Бінарне зображення не буде монохромним, якщо різні види пікселів відображаються різними відтінками кольору (наприклад, «жовто-зелене», «червоно-синє» растрове зображення та ін.)

Півтонове растрове зображення завжди є монохромним за визначенням, незалежно від того півтони (яскравості) якого відтінку кольору воно містить. Цей

термін є досить неоднозначним, тому що в різних сферах людської діяльності розуміється по різному: наприклад, в науці, в обробці космічних знімків, в програмній індустрії.

У деяких програмних продуктах цим терміном називають бінарні зображення, що складаються з білих і чорних пікселів. Однак те ж зображення, візуалізоване іншим програмним продуктом, може виглядати, наприклад, як червоно-синє. При цьому півтонове зображення, що є суворо монохромним, так називають значно рідше.

У зв'язку з цим рекомендується утриматися від терміна монохромне зображення при першому використанні, коли його значення не цілком зрозуміле з контексту, на користь більш однозначних: бінарне зображення і півтонове зображення.

6.7.2. Основні відомості про чорно-біле зображення

Найбільш примітивний тип зображення – це монохромне зображення, кожна точка якого може бути пофарбована тільки чорним або білим кольором. Монохромні зображення потребують дуже мало пам’яті для збереження і виводу інформації. Конвертувати в чорно-біле зображення можна будь-який напівтоновий малюнок. Однак у такий спосіб можна зберігати не всі типи зображень.

Портрет або малюнок олівцем, збережені у монохромному режимі, дають погані результати. Оскільки людське око дуже чутливе до деталей обличчя, то конвертування портрета в монохромне зображення, що видаляє велику частину дрібних деталей, робить фрагменти залишених зображень грубішими, і може змінити обличчя на портреті до невпізнанності.

З іншого боку, малюнок, виконаний тушшю і збережений у вигляді монохромного зображення, при високій роздільній здатності виглядатиме дуже реалістично, оскільки туш має дуже однорідний чорний колір.

Чорно-біле зображення найбільше застосовується у фотографії.

6.7.3. Методи перетворення кольорового зображення в чорно-біле у програмі Photoshop

Практично всі RAW-конвертери мають можливість переведення зображення в чорно-біле. Працюючи з фотографіями в Photoshop, застовуючи конвертер тільки для конвертації RAW-файлів, доцільно переводити фотографії в 16-бітовий формат з невеликим контрастом. Оскільки низький контраст дасть більше можливостей для маніпуляцій градаціями яскравостей, а 16-бітовий формат запобігає випаданню тонів із гістограми.

Зйомка і підготовка до обробки. Багато камер мають можливість перетворення зображень у чорно-білі, здійснення різних налаштувань і тонувань. Зйомка в кольорі з подальшим перетворенням на комп’ютері у чорно-білий вид надає багато можливостей для роботи з тонами і контрастом, рис. 6.16.

Рис. 6.16. Композиція чорно-білої і кольорової фотографії

У деяких випадках корисніше зробити не одну фотографію, а кілька знімків з різною експозицією, створити з них HDRI (паномарна фотографія, яка охоплює всі кути зору з однієї точки і містить велику кількість даних, як правило 32 біта в пік селі на канал, який може застосовуватися для освітлення сцени) з подальшою тональною компресією і тоді переводити зображення в чорно-біле. Деякі ефекти, які вносять під час тональної компресії такі програми, як Photomatix, можуть виглядати непривабливо на кольорових фотографіях, але застосування їх цілком доречне на чорно-білих.

Для зміни балансу світлих і темних областей можна застосувати інструмент Shadow/Highlight. Під час роботи з кольоровими фотографіями параметри в цьому діалоговому вікні потрібно змінювати обережно, оскільки можна легко порушити не тільки природний контраст мотиву, а й спотворити колір. У той же час у чорно-білій фотографії цим інструментом можна користуватися набагато ширше, особливо під час зйомки архітектури, де зміна балансу світла і тіней може зробити знімок цікавішим, рис. 6.17.

Розглянемо методи перетворення зображення в чорно-біле, які надає Photoshop. У більшості випадків для роботи з фотографіями достатньо одного чи двох описаних нижче методів.

Для більш наочного показу методів можна взяти зображення подане на рис. 6.18. У середньому ряду знаходяться кольори в чистому вигляді, над ними – квадрати зі зменшеною насиченістю в HSB-моделі. Нижче середнього ряду зменшується яскравість кольорів у HSB-моделі.

Рис. 6.17. Результат застосування інструменту Shadow/Highlight

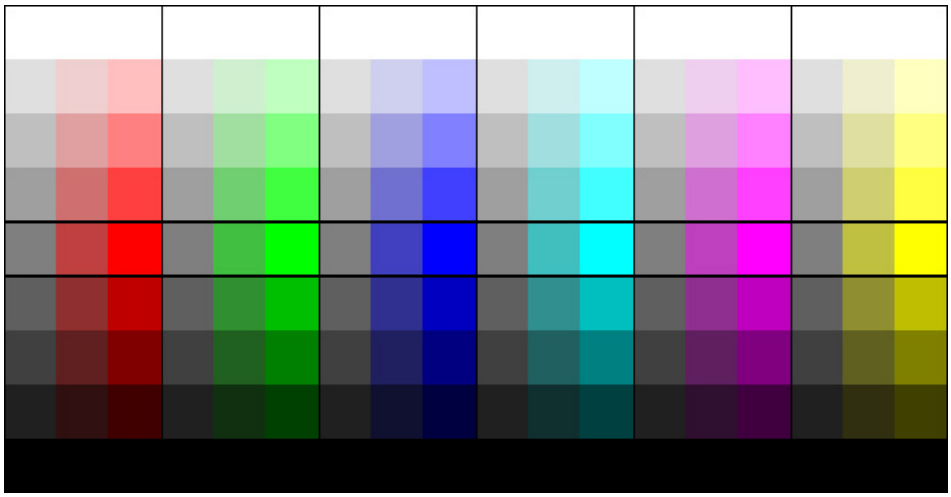


Рис. 6.18. Гамма кольорів у Photoshop

Метод Black/White

Метод Black/White є найбільш зручним і зрозумілим методом переводу зображення в чорно-біле. Використання його в більшості випадків призводить до оптимальних результатів. Він з'явився тільки у версії Photoshop CS3. Користувачі попередніх версій використовують схожий метод – Channel Mixer.

За допомогою діалогового вікна Black/White визначають яскравість кольорових тонів. При значенні 50 для певних колірних тонів всі пікселі цього кольору приймають такі ж значення HSL-яскравості, як і під час використання Hue/Saturation. А при значенні 0 відповідні насичені кольори стають чорними, при 100 – білими, а малонасичені кольори, відповідно, затемнюються чи стають світлішими. А при негативних значеннях затемнюються також малонасичені і світлі пікселі цього колірного тону, рис. 6.19. Незважаючи на те, що Black/White нагадує Channel Mixer,

принцип його дії зовсім інший. А саме будь-які зміни у вікні Channel Mixer впливають на всі зображення, у той час як Black/White – тільки на певні колірні відтінки.

Рис. 6.19. Результат застосування інструменту Black/White

У ранніх версіях Photoshop фільтр частково замінюють двома коригуючими шарами спочатку Selective Color і над ним – Hue/Saturation з насиченістю 100. Тепер, змінюючи параметр Black для певних кольорів у шарі Selective Color, можна змінити їх яскравість. Вплив цього методу буде помітний в першу чергу для насичених тонів. Обійти це обмеження можна створенням ще одного шару Hue/Saturation безпосередньо над шаром із фотографією і збільшити в ньому насиченість всього зображення чи тільки певних кольорів.

Метод Channel Mixer

До появи версії Photoshop CS3 метод Channel Mixer був найкращим методом переводу зображення в чорно-біле. Проте він ще широко застосовується, оскільки в каналах вже міститься чорно-біле зображення і результатом комбінування декількох каналів є хороше зображення, рис. 6.20.

Застосування методу Channel Mixer вимагає деякого часу і досвіду під час переводу фотографій у чорно-біле зображення. Для цього треба проаналізувати всі три канали окремо і з’ясувати, наскільки вигляд того чи іншого каналу відповідає бажаному результату. Потім, відзначивши опцію Monochrome в діалоговому вікні Channel Mixer, змінити процентний вміст інформації з певних каналів. Якщо ефект фільтра недостатньо виражений, можна додати новий коригувальний шар Hue/Saturation безпосередньо під шаром Channel Mixer і збільшити насиченість всього зображення або тільки певних кольорів. Щоб результати підвищення насиченості

були більш якісними, можна скористатися виборчою насиченістю. Найчастіше для оптимальних результатів сума впливу всіх каналів повинна дорівнювати 100%. Найбільш природно виглядає результат 30/60/10.

Рис. 6.20. Вихідне зображення і результати застосування Channel Mixer з параметрами 80/20/0 і -20/-30/150

Метод Calculations

Метод Calculations можна вважати логічним продовженням роботи з каналами через Channel Mixer. Діалогове вікно Calculations дозволяє розраховувати результат накладання каналів один на одний з різними режимами перекриття. Викликавши діалог через Image-Calculation, можна задати вихідні канали і режим перекриття.

Цей метод дозволяє працювати тільки з двома каналами. Для роботи з усіма трьома каналами треба перейти до списку каналів, в опціях вибрати Split Channels і отримати три зображення в режимі Grayscale. Скопіювавши їх в один файл у вигляді шарів, можна експериментувати з режимами перекриття і прозорістю, а за необхідності додавати до верств-каналів маски і закривати на ній частини.

Цей метод працює, тільки коли зображення представлене одним-єдиним базовим шаром Background. Якщо ні, то шари зображення потрібно спочатку скласти (Ctrl + Shift + E) і створити базовий шар через Layer-New-Background from Layer.

Метод Gradient Map

Застосування методу Gradient Map зобов'язує перед початком роботи встановити чорний і білий кольори, натиснувши клавішу D, створити новий шар Gradient Map і вибрати перший у списку градієнт Foreground to Background.

Цей метод базується на формулі 30% інформації червоного каналу, 60% – зеленого і 10% синього з невеликими варіаціями. Отримані результати сильніше відрізняються від візуального сприйняття яскравостей кольорового зображення, ніж результати наступних більш простих методів. Однак контраст отриманої фотографії вищий від 18 для чистого синього кольору до 237 для жовтого, рис. 6.21.

Рис. 6.21. Результат застосування інструменту Gradient Map

Як варіант цього методу можна використовувати «зонний градієнт», своєрідну імітацію методу роботи з зонами яскравості, який пропонує програма Lightzone. Для

цього створюється чорно-білий градієнт і ставиться на ньому між двома крайніми каретками ще 9 кареток з кроком 10. Тобто друга каретка буде знаходитися на позиції 10, третя – 20 і т. д. Кожній каретці привласнюється відтінок сірого, відповідний її позиції, – перша буде зі значенням яскравості 0, друга – 10 і т. д. Тоді, змінюючи позиції кареток, маніпулюють контрастом і розподілом яскравостей на зображенні. Зрушуючи каретки в центр градієнта, отримуємо більш контрастне зображення, зрушуючи каретки до країв, – більше відтінків сірого, рис. 6.22.

Для оптимальних результатів краще застосовувати «зонний градієнт» до 16-бітових зображень.

Метод зрушення колірного відтінку практичного застосування немає, за винятком для знімків з неяскравими кольорами.

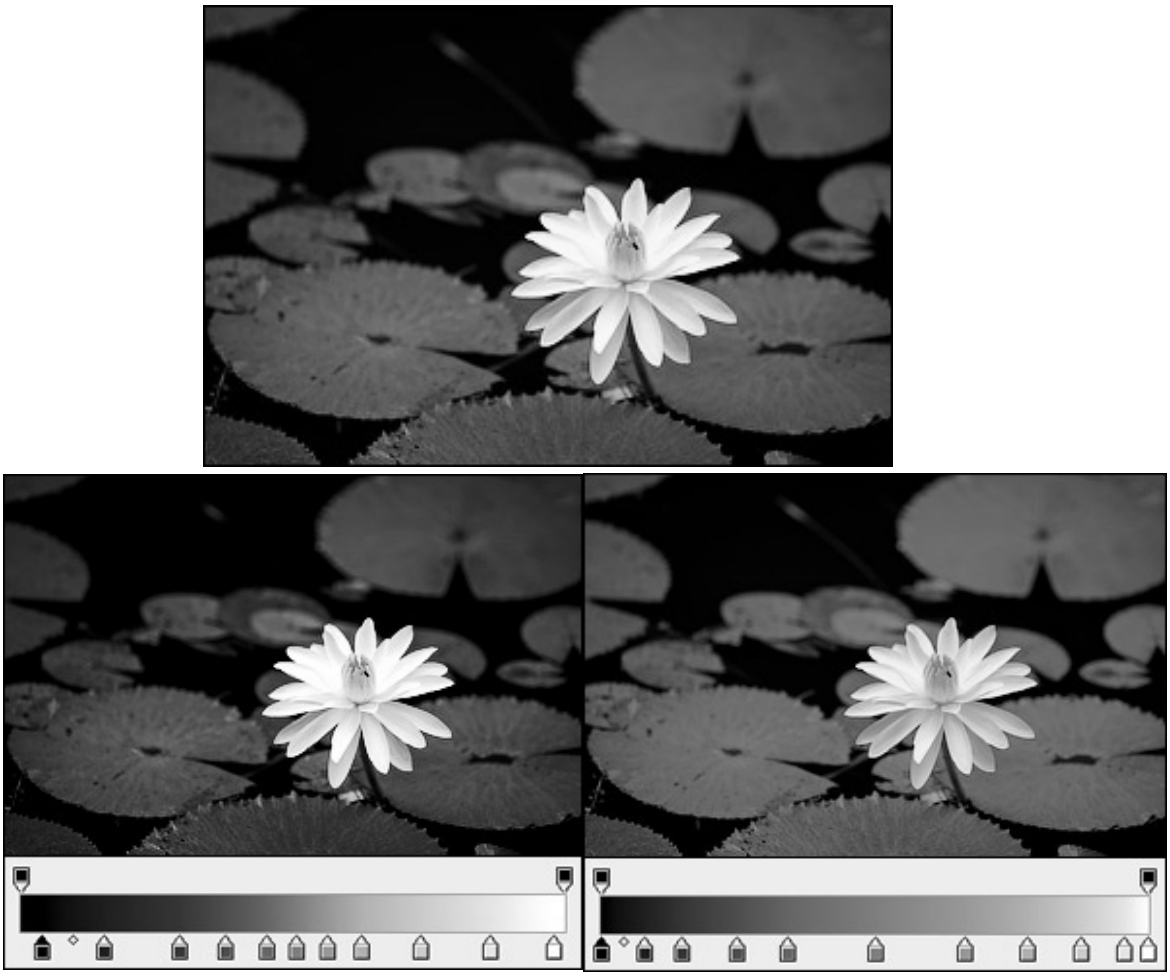


Рис. 6.22. Вихідна чорно-біла фотографія і результати застосування зонного градієнта з різними параметрами

Методи Hue/Saturation і Desaturate практичне застосування знайшли у випадках, якщо на вихідній фотографії немає кольору, який потрібно враховувати, рис. 6.23.

Рис. 6.23. Результат застосування методів Hue/Saturation і Desaturate

Метод Grayscale

Під час застосування методу Grayscale для заповнення інформацією єдиного результуючого каналу використовується приблизно класична формула 30/60/10. Крім застосування цієї формули Photoshop проводить додаткові маніпуляції над інформацією зображення, для рівномірнішого розподілу ступенів яскравості між різними колірними відтінками.

Контраст фотографій досить невисокий, проте це дає фотографу більший простір для маніпуляцій над зображенням з допомогою кривих, рівнів або інших методів роботи з контрастом. Цей метод застосовується, якщо на фотографії є тільки слабкі кольори. У такому випадку цей метод все ж підходить краще, ніж попередній, оскільки враховує колірні відтінки, рис. 6.24.

Рис. 6.24. Перетворення зображення в чорно-біле методом Grayscale

Якщо за переведенням зображення слідують маніпуляції з тонуванням, можна перевести зображення в Grayscale і відразу ж в RGB.

Метод Lab

Результати методу Lab схожі на попередній, тільки діапазон яскравостей, в якому розподіляються різні кольори, менший. Коли під час переведення в режим Grayscale яскравості чистих кольорів розподілялися від 70 для синього до 248 для жовтого, то під час цього методу – від 250 до 110. Зображення переводиться в колірний простір

Lab, після чого вибирається канал L (в списку каналів або через Ctrl +3) і переводиться в Grayscale, рис. 6.25.

Рис. 6.25. Результат застосування методу Lab

6.8. Напівтонове зображення

6.8.1. Основні відомості про напівтонове зображення

Напівтоновим зображенням вважається зображення, *що має безліч значень тону, і їх безперервну, плавну зміну.*

Прикладами напівтонових зображень можуть бути малюнки, картини, виконані фарбами, фотографії.

Як і всі растрові зображення півтонове зображення кодується в цифровому вигляді за допомогою бітової карти (матриці), що зберігає значення елементів зображення (пікселів). Кожен піксель напівтонового зображення може кодуватися різною кількістю біт, що визначає кількість можливих півтонів. Наприклад: 2 біт – 4 півтони; 3 – 8; 4 – 16; 8 – 256 і т. д.

При цьому, однобітове бінарне зображення (1 біт на 1 піксель) можна вважати виродженим напівтоновим, здатним передати лише 2 півтони (наприклад чорний і білий), або ж окремим видом напівтонового.

Безліч можливих півтонів називають рівнями сірого (англ. gray scale), незалежно від того, півтони якого кольору або його відтінку передаються. Рівні сірого не відрізняються за спектральним складом (відтінком кольору), але відрізняються за яскравістю. Кількість можливих півтонів у даному випадку є глибина кольору, яку часто передають не в кількості самих півтонів, а в кількості біт на піксель (англ. bit per pixel, bpp).

Наприклад, в аерофотозйомці і космічній зйомці на виході можна отримати напівтонові зображення з глибиною кольору (кількістю біт на піксел, bpp від англ. Bit per pixel) 16 або 32. Яке значеннння у допустимому діапазоні буде вважатися найяскравішим, а яке найтемнішим не має значення, тому що число, яке є значенням кожного пікселя – лише умовний код яскравості. Досить вказати напрямок відліку.

Наприклад, можуть існувати напівтонові растри, де на кожен піксель відведено по 8 біт, а зображення має 256 півтонів. При цьому пікселі зі значенням 0 або 255 будуть чорними, і навпаки, пікселі зі значенням 255 або 0 – білими, решта півтонів сірого будуть рівномірно розподілені між даними значеннями колірного індексу.

В образотворчому мистецтві та побуті найчастіше застосовують напівтонові растри з глибиною кольору 8 біт (що дорівнює 1 байт), тобто кожен піксель зображення може приймати 256 різних умовних значень яскравості від 0 до 255. Цього цілком достатньо, щоб правильно відобразити чорно-білу фотографію.

У науці і техніці часто такого діапазону і дискретності подання яскравості не достатньо.

Деякі формати зберігання растрових зображень (наприклад, TIFF) дозволяють задавати за допомогою палітри через умовне значення індексу точні фотометричні характеристики зображення. Така палітра являє собою таблицю, де кожному умовному рівню сірого (задається цілим числом – кодом) ставиться у відповідність будь-яка фотометрична величина. Це також часто використовується на практиці в тих випадках, коли умовної відмінності яскравості однієї ділянки зображення від іншої не достатньо. Наприклад, під час дешифрування аерокосмічних знімків з метою прогнозування врожаю чи оцінки ураженості шкідниками, необхідно знати точну кількість зареєстрованого випромінювання.

6.8.2. Структурна модель напівтонових зображень та її використання у задачі сегментації зображень

Форма об'єкту повністю визначає його опис і представлена його граничним контуром і функцією оптичної щільності, яка визначена в межах цього контуру.

Обробка візуальної інформації, зокрема напівтонових зображень, відноситься до найбільш складних завдань штучного інтелекту і, в той же час, все більше є актуальним методом для практичного використання в найрізноманітніших галузях науки і технологій. У даний час у засобах штучного інтелекту напівтонові зображення представлені у растровому вигляді. Таке уявлення виключає можливість обробки – перетворення, ідентифікації об'єктів, що відрізняються від афінних перетворень – масштабом, положенням у полі зображення, поворотом. У сучасних засобах обробки візуальної інформації практично не використовуються такі поняття як контури об'єктів напівтонових зображень (за винятком контурів об'єктів попередньо бініаризованих напівтонових зображень).

У той же час однією з найважливіших і найбільш природних особливостей зорового сприйняття людини є її здатність до сегментації поля зору на об'єкти, які відрізняються від фону оптичною щільністю, кольором, текстурою та ін. Основною характеристикою будь-якого об'єкта є його форма, яка визначена контуром – кордоном між об'єктом і фоном. Контур, у свою чергу, сприймається людиною як послідовність відрізків прямих і дуг кривих ліній. Форма напівтонових і кольорових

об'єктів визначається, крім того, функцією оптичної щільності з урахуванням кольору, текстури всередині контуру кожного з об'єктів. Ці особливості зорового сприйняття людини відображені у пропонованій структурній моделі напівтонового зображення.

Структурна модель дає можливість представлення довільних зображень за одноманітною формою. Завдання приведення до структурної моделі довільних зображень, заданих у растровому вигляді, спотворених перешкодами у загальному випадку ще не вирішена. Однак в окремих випадках приведення зображень до структурної моделі дозволяє істотно підвищити швидкість і якість обробки візуальної інформації, що, у свою чергу, забезпечить якісне функціонування інформаційних технологій які використовують ці засоби. Об'єкти напівтонового зображення, представлені в структурному вигляді, інваріантному щодо афінних перетворень, найкращим чином підходять у якості вихідних даних для обробки засобами зростаючих пірамідальних мереж і теорії розпізнавання і пам'яті.

Такий структурний аналіз форми візуальних об'єктів, спотворених перешкодами, добре узгоджується з відомим стандартом MPEG-7, і може бути до нього адаптований.

Розглянемо структурну модель напівтонового зображення і приклад її використання підчас розробки інформаційних технологій для медичних діагностичних систем, зокрема, для структурного аналізу напівтонових зображень з метою автоматичного виділення об'єктів на прикладі зображень медичних препаратів, отриманих за методом Кирліан.

6.8.3. Структурний аналіз напівтонового зображення

Основою структурного аналізу напівтонового зображення є модель, яка визначає його структурні елементи. У відповідності з відомими уявленнями про механізми зорового сприйняття такими структурними елементами зображення, зокрема, є об'єкти, розташовані на тлі, який визначається двовимірною функцією оптичної щільності. Об'єкти, в свою чергу, визначаються контурами, які їх обмежують, і двовимірною функцією оптичної щільності в межах об'єкта. Контури є замкнутими послідовностями, які утворені відрізками прямих і дугами кривих ліній.

Під зображенням розуміють частину площини, обмежену деякою геометричною фігурою, зазвичай прямокутником, кожна точка якої характеризується певним значенням оптичної щільності. Іншими словами, на частині площини, обмеженої прямокутником з розмірами X, Y визначена $p = f(x, y)$, $(0 < x < X; 0 < y < Y)$. Цій функції можна поставити у відповідність деяку поверхню $z = f(x, y)$ в просторі $OXYZ$.

Таким чином, кожному напівтоновому зображенню можна поставити у відповідність регулярну незамкнену поверхню в просторі $OXYP$ яка складається з простих елементів поверхні. Для поверхні, яка відповідає напівтоновому зображенню, справедливе наступне обмеження. Кожному значенню пари координат

(*x, y*) відповідає одне і тільки одне значення функції r (*x, y*), тобто перпендикуляр до площини зображення в будь-якій точці *x*, *y* перетинає уявну поверхню один і тільки один раз.

Контур кожного елемента регулярної поверхні є замкнутою послідовністю регулярних дуг кривих і відрізків прямих. Точки контуру не є регулярними точками елементів простих поверхонь. Точки контуру – це граничні точки елементів простих поверхонь. Точки контуру утворюють особливі лінії поверхні, які є граничними, що розділяють різні елементи простих поверхонь. На відміну від бінарних зображень, точки яких можуть мати тільки два значення оптичної щільності – чорний або білий, області напівтонових зображень, обмежені контуром, можуть мати різні закони зміни оптичної щільності. Відповідно, кількість відмінних один від одного сусідніх елементів простих поверхонь, як правило, більше двох. Значить, контури напівтонового зображення можуть і не бути, в загальному випадку, однозв'язними послідовностями регулярних дуг кривих і відрізків прямих, а складаються з гілок, що з'єднують вузли. Гілки є особливими лініями зображення. Точки контуру, за винятком вузлів, є регулярними точками гілок. Вузли є особливими точками контуру і зображення. Гілки і вузли разом із законом зміни оптичної щільності кожного елемента простої поверхні повністю визначають регулярну поверхню і відповідну їй область зображення. У багатьох практичних випадках контури напівтонового зображення є однозв'язними послідовностями регулярних дуг кривих і відрізків прямих, що істотно спрощує завдання структурного аналізу.

У напівтоновому зображенні завжди можна виділити області, для яких значення оптичної щільності постійне, або змінюється за певним законом. Закон зміни оптичної щільності визначається grad(r) – градієнтом оптичної щільності. Зазвичай в межах однієї області r = const, або $\partial r / \partial x + \partial r / \partial y = \text{const}$, або $\partial^2 r / \partial x^2 + \partial^2 r / \partial y^2 = \text{const}$. У той же час можливі і інші закони зміни оптичної щільності. Згідно з наведеними визначеннями півтонове зображення можна розглядати як деяку область регулярної поверхні, що складається з регулярних елементів простих поверхонь, причому кожен об'єкт зображення відповідає одному чи декільком елементам простих поверхонь.

6.8.4. Цифрова строкова модель довільного напівтонового зображення

З поверхнею в просторі OXY^r, якій відповідає півтонове зображення, поєднана решітка N′ M′ P, і для кожного пікселя зображення визначено середнє в межах його площі значення оптичної щільності p (n, m), яка набирає цілі значення p (n, m) = (0, P); n = (0, *N*); m = (0, *M*). Сторона решітки з *N* клітинками розташована вздовж осі *Ox*, сторона решітки з *M* клітинками розташована уздовж осі *OY*, сторона решітки з P клітинками розташована вздовж осі O^r. Нехай (yn^r); n = (0, *N*) – безліч паралельних площин, перпендикулярних осі *Ox* в тривимірному просторі OXY^r. Точно так само

($xm\,\mathbf{r}$); m = (0, *M*) – безліч паралельних площин, перпендикулярних осі *OY*. Перетин поверхні зображення з цими площинами утворює на кожній з площин $Yn\,\mathbf{r}$ лінію контуру $\mathbf{r}_n(x)$, а на кожній з площин $Xm\,\mathbf{r}$ лінію контуру $\mathbf{r}_m(y)$, або $\mathbf{r}_n(m)$ і $\mathbf{r}_m(\mathrm{n})$ для випадку дискретизованого зображення.

Виділення регулярних і особливих точок регулярних поверхонь може бути виконано в процесі структурного аналізу функцій $\mathbf{r}_n(m)$ і $\mathbf{r}_m(n)$ дискретизованого напівтонового зображення, що дає можливість представити їх як послідовності відрізків цифрових прямих і дуг цифрових кривих у площинах $\mathbf{r}\,On$ для будь-яких значень m = (0, *M*) і $\mathbf{r}\,Om$ для будь-яких значень n = (0, *M*) відповідно. Граничні точки відрізків і дуг є особливими точками ліній перетину, в той час як інші точки є регулярними точками ліній перетину. Кожна точка *t* дискретизованого зображення належить одночасно двом взаємно перпендикулярним площинам $yn_t\,\mathbf{r}$ і $xm_t\,\mathbf{r}$ та двом пересічним лініям контуру $\mathbf{r}_{nt}(m)$ і $\mathbf{r}_{mt}(n)$ відповідно.

З визначення регулярної поверхні випливає, що точка поверхні є регулярною, якщо вона є регулярною точкою горизонтальної і вертикальної ліній перетину.

Якщо ж точка поверхні є особливою точкою хоча б однієї з ліній – горизонтальної і / або вертикальної ліній перетину, то така точка є особливою – граничною точкою регулярної поверхні – області напівтонового зображення.

Граничні точки областей напівтонового зображення (його регулярної поверхні) утворюють лінії контурів у площині *XOY*. Лінії контурів, у свою чергу, містять регулярні та особливі точки.

Структурний аналіз напівтонового зображення повинен містити такі операції:

1) виділення особливих точок регулярних поверхонь (областей зображення);

2) побудова особливих ліній зображення (контурів), які обмежують об'єкти в особливих точках регулярних поверхонь;

3) виділення структурних елементів контурів – відрізків прямих і дуг кривих ліній.

На рис. 6.26 представлений приклад структурного аналізу з використанням строкової моделі, а саме виділення контурів на напівтоновому зображенні. Програми виконують над зображенням (рис. 6.26, a) такі операції. Для кожного вертикального і горизонтального рядків зображення будуються графіки функцій оптичної щільності, приклади яких зображені на рис. 6.26, c, d. Для кожного графіка визначається послідовність елементів, з яких він складається, – відрізків цифрових прямих і дуг цифрових кривих. Граничні точки між елементами графіка є особливими точками графіка цього рядка і всього напівтонового зображення. Особливі точки зображення виділені на рис. 6.26, b білим кольором. Особливі точки належать лініям контуру напівтонового зображення. За особливими точкам побудовані контури напівтонового зображення. На рис. 6.26,e представлені контури в растровому вигляді, утворені

окремими особливими точками. Їм відповідають контури у векторному вигляді – рис. 6.26, f.

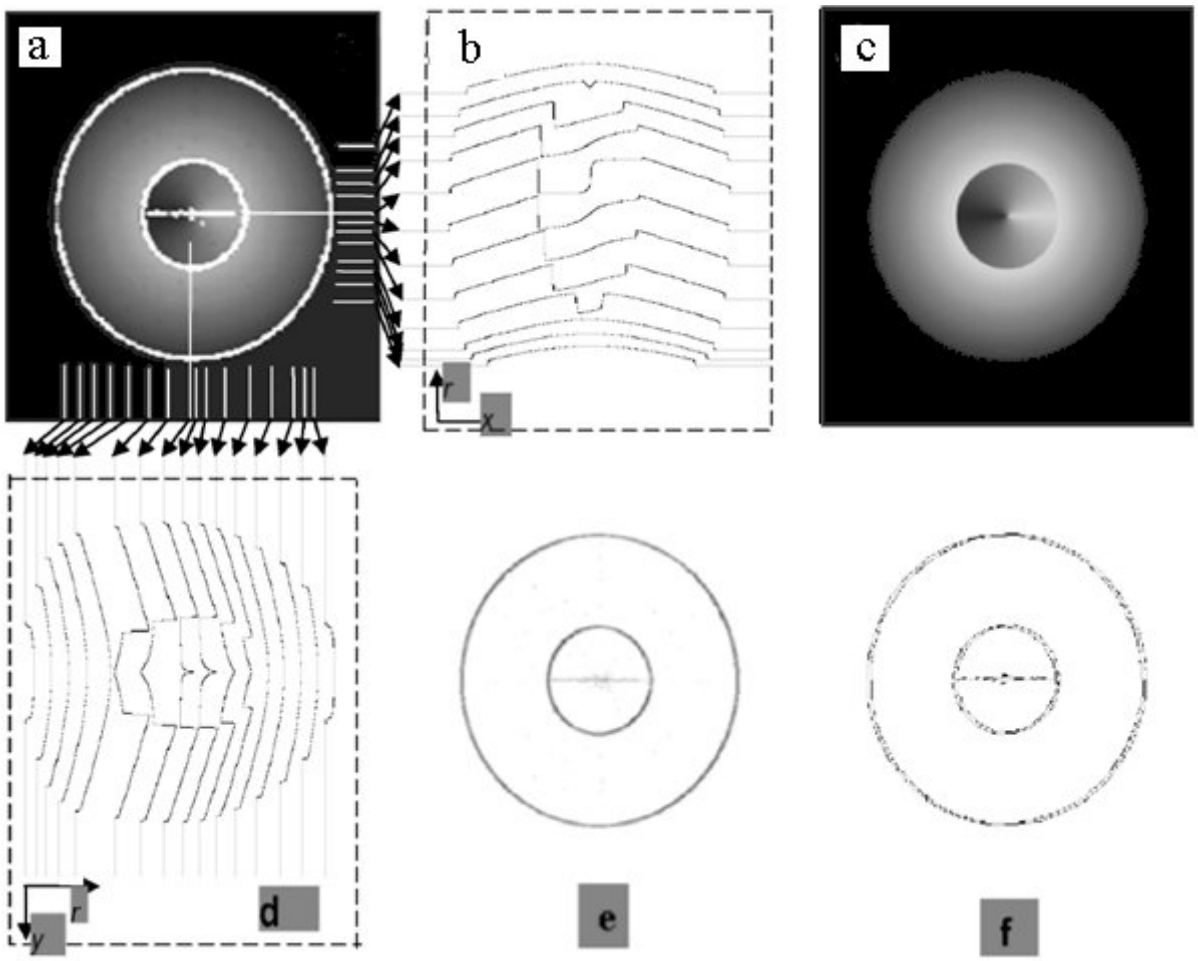


Рис. 6.26. Виділення контурів на напівтоновому зображенні, здійснено програмами, що реалізують строкову модель зображення та обробку контурів: a – модельне півтонове зображення з виділеними особливими точками; b – криві оптичної щільності – *r* – горизонтальних рядків; c – модельне півтонове зображення; d – криві оптичної щільності – *r* – вертикальних рядків; e – зображення контурів, утворені окремими особливими точками; f – зв'язкові зображення контурів

6.8.5. Експерименти з обробки зображень, отриманих за методом Кирліан

Розглянемо використання структурної строкової моделі цифрового зображення на прикладі обробки зображень медичних препаратів, отриманих за методом Кирліан.

Зображення медичних препаратів взагалі містять об'єкти, форма яких дуже мінлива, але, в той же час, саме у формі міститься діагностична інформація візуальної оцінки. Як правило, такі зображення спотворені перешкодами. Зображення медичних препаратів використовуються в процесі прийняття рішень в медичних діагностичних системах. Зображення, які отримані в процесі функціонування таких систем не

завжди високої якості, що значно знижує можливість їх швидкого і повного сприйняття експертами з мінімальними витратами часу. У той же час саме в процесі аналізу великих кількостей таких зображень можуть бути отримані нові знання про стан здоров'я груп населення. Час обробки і прийняття рішення, так само як і кількість експертів у медичних діагностичних системах, як правило, обмежена. Без автоматизації обробка таких обсягів візуальної інформації перестає бути ефективною. Зокрема знижується якість обробки і зростає кількість помилок.

Зображення, отримані за методом Кирліан (далі зображення Кирліан), є знімками, виконаними на спеціальній фотоплівці, розміром А4, на яких зафіксовані світіння від кожного з десяти пальців (рис. 6.27).

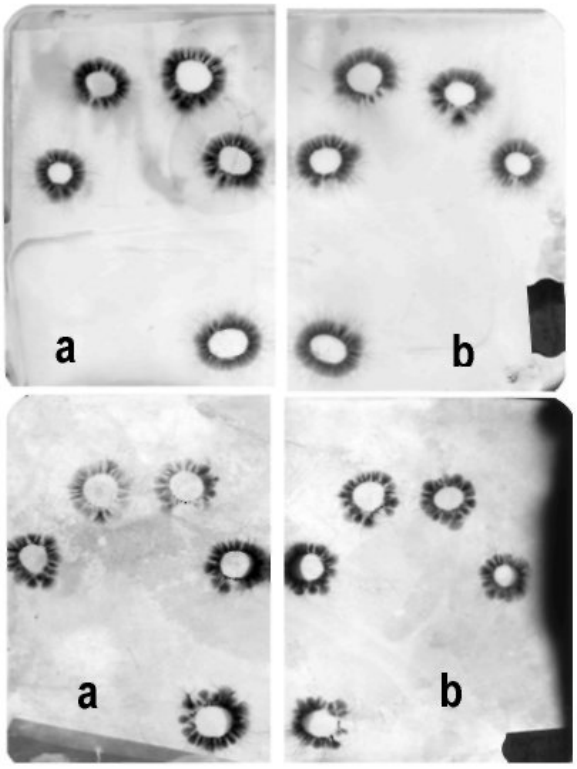


Рис. 6.27. Приклади зображень Кирліан світіння пальців: a) лівих, b) правих рук

M_o, M_f – середні значення яскравості об'єктів і фону; o_{min}, o_{max}, $f_{\min}$, $f_{\max}$ – мінімальні і максимальні значення яскравості об'єктів і фону.

Для зображень характерні нестабільність і нерівномірність фону, значна кількість перешкод, які за рівнем яскравості і величиною, в порівнянні з об'єктами, мають нестійкість форми та рівень яскравості.

Хоча за змістом обробки в даний час ці зображення могли б вважатися бінарними, проте, навіть завдання бінаризації таких зображень не може вважатися тривіальним, як і завдання подальшої обробки. Зокрема, завдання виділення та ідентифікації об'єктів з метою діагностики. Існує стандартне програмне забезпечення для обробки зображень Кирліан, отриманих на спеціальних приладах окремо для кожного пальця. Однак використання таких приладів ускладнює діагностику,

оскільки стан досліджуючого організму за період послідовної фіксації світінь кожного з десяти пальців може істотно змінитися.

Щоб використовувати стандартне програмне забезпечення для зображень Кирліан, на яких одночасно зафіксовані всі пальці рук (рис. 6.27), необхідно попередньо сегментувати ці зображення і повернути зображення кожного пальця таким чином, щоб воно відповідало його вертикальному положенню.

Пропоноване програмне забезпечення призначене для автоматичної сегментації зображень Кирліан від п'яти пальців на зображення до кожного пальця окремо. Робота програми полягає у виконанні наступних операцій:

1) по осі абсцис визначають значення яскравості, а по осі ординат – умовні значення, пропорційні кількості пікселів, що відповідають даному значенню яскравості. Обчислення гістограми оптичної щільності досліджуючого зображення Кирліан зображено на рис. 6.28. Після її згладжування, визначають мінімальне значення яскравості об'єктів o_{min}, як мінімальне значення яскравості пікселів зображення та максимальне значення яскравості фону f_{max}, як максимальне значення яскравості пікселів зображення. Встановлюють середнє значення яскравості об'єктів M_o, як перший максимум під час зростання значень яскравості, починаючи з нуля, і фону M_f, як перший максимум під час зменшення значень яскравості, починаючи з максимального (255);

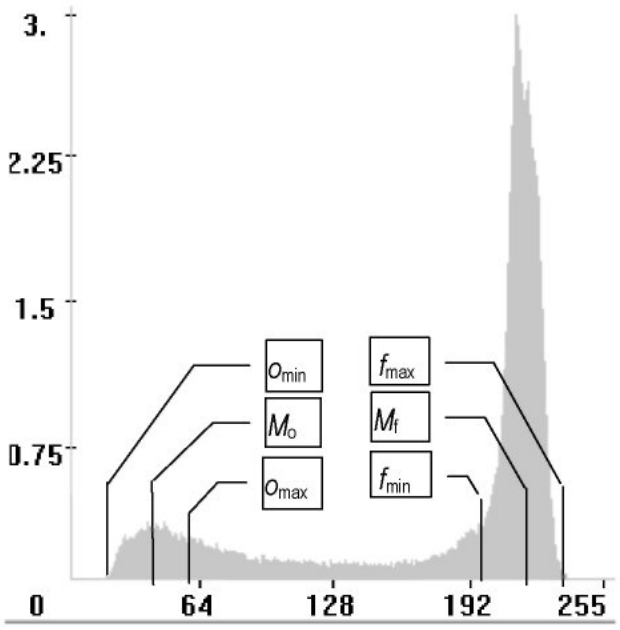


Рис. 6.28. Гістограма яскравості (оптичної щільності) зображення

2) на підставі припущення про симетричність розподілу випадкових величин яскравості пікселів фону та об'єктів, обчислюють мінімальне значення яскравості пікселів фону, як $f_{min} = M_f - (f_{max}-M_f)$ і максимальне значення яскравості пікселів об'єктів, як $o_{\max} = M_o + (M_f - o_{min})$;

3) обчислюють функції яскравості (оптичної щільності) горизонтальної рядка $r(m, n)$ $m = 1,M$ для всіх горизонтальних рядків зображення $n = 1,N$ (рис. 6.29).

Ця функція не є результатом бінаризації, оскільки не розглянуті значення функції яскравості $o_{max} < r(m, n) < f_{min}$. Пікселі з такими значеннями функції яскравості є проміжними між пікселями фону та об'єкта і не мають суттєвого значення для виділення об'єктів на зображенні, принаймні, для вирішення завдань з обробки зображень Кирліан.

Дане перетворення зображення, з урахуванням динамічного діапазону є нелінійним із зміною кількості рівнів квантування від 256 до 3 і дозволяє в значній мірі виключити вплив перешкод. Приклади функції $u_b(m, n)$ зображені на рис. 6.29.

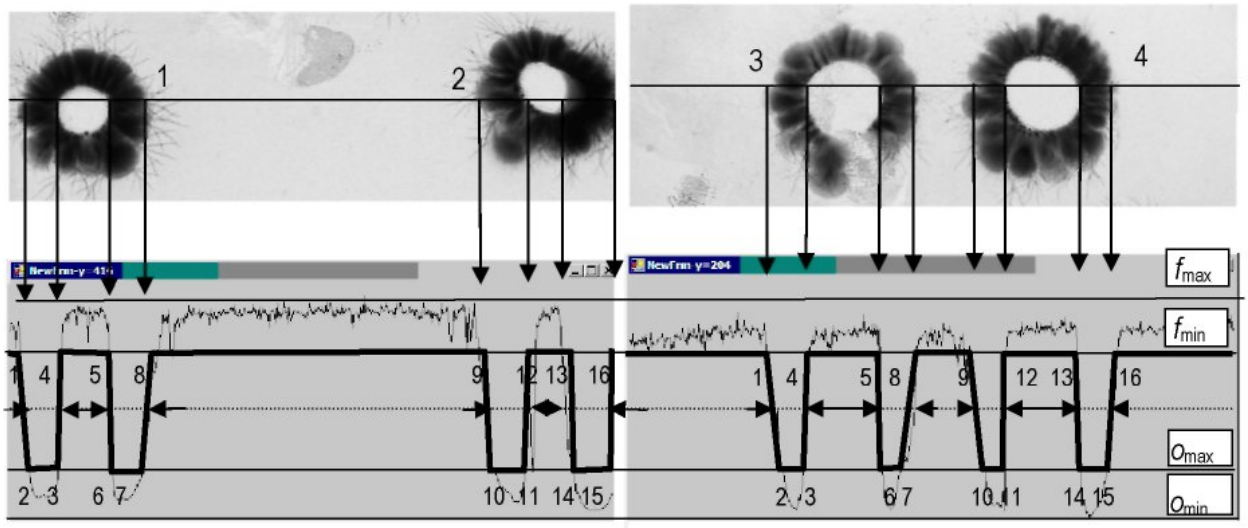


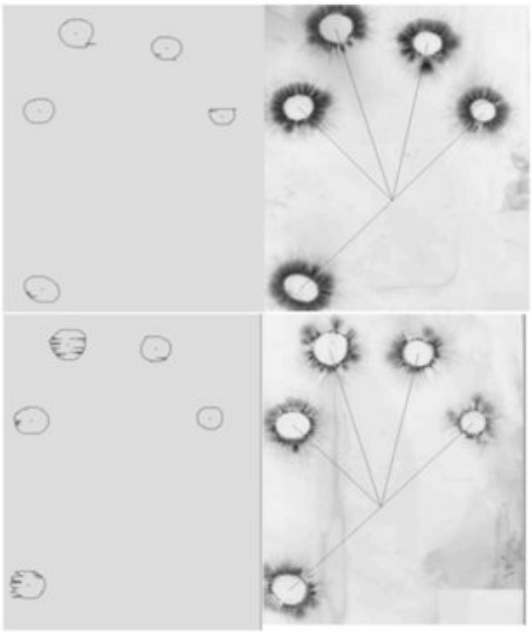
Рис. 6.29. Виділення особливих точок у місцях перетину об'єктів горизонтальними рядками

4) визначають функцію на зображенні:

$$u_b = (m,n) = \begin{cases} f_{\min,} \; при \; r(m,n) \geq f_{\min} \\ o_{\max,} \; при \; r(m,n) \leq o_{\max,} \end{cases}$$

ламаною лінією. Числами 1, 4, 5, 8, 9, 12, 13, 16, визначені особливі точки зображення, які належать фону, а числами 2, 3, 6, 7, 10, 11, 14, 15 – особливі точки зображення, які належать об'єктам.

Будують внутрішній і, за необхідності, зовнішній контури об'єктів, за виділеними особливими точками, рис.6.30. Якщо контури визначено, то об'єкти виділені успішно.



a b

Рис. 6.30. Контури об’єктів: a – внутрішні контури; b – апроксимація контурів еліпсами

Визначають положення об'єктів світіння пальців щодо центру долоні. Апроксимують внутрішні контури еліпсами, визначають кут повороту кожного пальця, обертають зображення світіння пальця до відповідності вертикального положення пальця (рис. 6.30) і формують результуючі файли зображень.

Структурна модель дозволяє будувати описи об'єктів напівтонового зображення інваріантні до афінних перетворень.

Отже, існує можливість і ефективність обробки напівтонових зображень з використанням структурної моделі напівтонових зображень.

6.8.6. Кодування напівтонових зображень, представлених набором бітових перерізів

Напівтонове зображення являє собою матрицю пікселів, в якій кожен піксель містить інформацію про яскравість складової сигналу. Під час розгляду зображення на двовимірній площині слід виділити наявність у ньому областей рівної чи приблизно рівної яскравості. Найпростіші алгоритми кодування, такі як RLE, LZ, не виділяють ці геометричні (просторові) області, розглядаючи зображення у вигляді довгого розгорнутого рядка, і тому не можуть повною мірою використовувати всі особливості зображення в процесі стиснення, забезпечуючи невисокий коефіцієнт компресії. Найближче до вирішення даної проблеми підійшли розробники фрактальних алгоритмів кодування напівтонових зображень.

Фрактальні алгоритми забезпечують досить великі коефіцієнти компресії за хорошого відновлення зображення приймачем. Недоліком даних алгоритмів є

тривалий час кодування, яке визначається пошуком структури (атрактора) у зображенні. Істотним поштовхом до розвитку фрактального кодування напівтонових зображень стало обґрунтування можливості використання системи кусково-визначених функцій (PIFS) для кодування напівтонових зображень, отриманих не штучним (комп'ютерне моделювання) шляхом.

Робота алгоритму здійснюється наступним чином. Вихідне зображення розбивається на доменні (опорні) і рангові (відновлювані) растрові області. На підставі певного набору функціональних перетворень для кожного рангового блоку проводиться пошук відповідного доменного блоку. Швидкість і якість кодування можна регулювати введенням порогових значень відповідності під час порівняння рангових і доменних областей. З урахуванням того, що в процесі пошуку кращого доменного блоку порівнюються растрові області, відповідний блок повинен мати відповідну яскравість і контрастність, що знижує швидкість кодування і збільшує обсяг закодованого файла.

Для уникнення цього явища запроваджують альтернативний комплексний алгоритм, заснований на представленні цифрового напівтонового зображення у вигляді набору бітових перетинів. Для цього вихідне зображення рівномірно квантують зазвичай на вісім бітових перерізів. Можливе також застосування нерівномірного квантування, що дозволить знизити необхідну кількість бітових перерізів для представлення зображення. Сусідні бітові перетини мають схожі області чи просто є інверсними копіями один одного. Особливо чітко даний момент простежується в старших бітових перетинах, що володіють високою просторовою кореляцією, як по рядках, так і по стовпцях зображення. У міру зниження розрядності бітового перетину середня протяжність областей однакової яскравості також знижується. Сильний кореляційний зв'язок між сусідніми бітовими перерізами дозволяє значно спростити процедуру пошуку, що збільшує компресію даних і підвищує швидкість кодування. У даному випадку в якості доменних блоків будуть області старшого (восьмого) бітового перерізу, оскільки просторова кореляція в ньому зазвичай має найбільше значення. Цей перетин буде опорним, що безпосередньо передається на приймаючий пристрій без кодування. Всі інші перетини розбиваються на рангові блоки, для яких проводиться відповідно до алгоритму пошук найкращого доменного блоку. Зі зменшенням розрядності бітового перетину ступінь подібності перерізів значно падає. Тому процес кодування обмежують тільки старшими чотирма бітами. У процесі порівняння рангова область зазнає різних функціональних перетворень, що дозволяють досягти схожості на початкових етапах кодування. Набір і кількість цих функціональних перетворень дозволяє регулювати швидкість кодування з одного боку і ступінь компресії зображення з іншого. Для простоти обчислень бітові перерізи вихідного зображення розбиваються на рангові та доменні області квадратної форми, кратні двом. У якості функціональних перетворень використовується інверсія і поворот області.

Використання бінарних зображень значно спрощує процес порівняння на відміну від роботи алгоритму з напівтоновими областями, оскільки сигнал може приймати лише два значення, і даний алгоритм може бути легко апаратно реалізований. Якість відновленого зображення визначається ступенем схожості рангового і доменного блоків. Регулювання цього процесу здійснюється шляхом заздалегідь обраного порогового значення відповідності. Якщо в процесі перебору всіх доменних блоків потрібний так і не був знайдений, ранговий блок розбивається на частини, кожна з яких знову порівнюється з доменними областями. Даний процес триває до досягнення необхідного порогового рівня відповідності. Змінюючи це значення можна керувати як часом кодування, так і якістю відновленого зображення. Велика частина найбільш поширених алгоритмів стиснення застосовує не одну, а кілька методів стиснення, що дозволяє підлаштуватися до характеристик кодованого зображення. З цією ж метою файл, отриманий після безпосередньо фрактального алгоритму кодування, додатково стискається архіватором WINZIP. При цьому досягається значна ступінь компресії, тому що файл після фрактального етапу кодування містить повторюючу структуру.

Питання для самоперевірки

1. Чим умовлена необхідність створення різних моделей кольорів?
2. Вкажіть призначення колірної моделі.
3. Що таке простір колірної моделі?
4. Охарактеризуйте основні колірні моделі, які застосовуються в комп’ютерній графіці.
5. Які кольори називаються компліментарними чи додатковими?
6. Вкажіть методи створення зображень.
7. Який тип зображення вважається найбільш примітивним?
8. Назвіть основні типи зображень.
9. Охарактеризуйте методи перетворення кольорового зображення в чорно-біле у програмі Photoshop.
10. Вкажіть, яке зображення вважається напівтоновим. Охарактеризуйте його.

Список рекомендованої літератури

1. Блінова Т.О., Порєв В.М. Комп’терна графіка / За ред. В.М. Порєва. – К.: Видавництво «Юніор», 2004. – 456 с.
2. Горобець С.М. Основи ком’ютерної графіки: Навч. пос. / За ред. М.В. Левківського. – К.: Центр навчальної літератури, 2006. – 232 с.

РОЗДІЛ 7. ПРАКТИЧНА ЧАСТИНА

7.1. Цифровий диференціальний аналізатор

Один з методів розкладання відрізка в растр полягає у вирішенні диференціального рівняння, що описує цей процес. Для прямої лінії маємо

$$\frac{dx}{dt} = const \quad або \quad \frac{Dy}{Dx} = \frac{y_2 - y_1}{x_2 - x_1}$$

Рішення представляється у вигляді:

$$y_{i+1} = y_i + \Delta y,$$

$$y_{i+1} = y_i + \frac{y_2 - y_1}{x_2 - x_1}\Delta x \qquad (7.1)$$

де x_1, y_1 і x_2, y_2 – кінці відрізка, який розкладається і y_i – початкове значення для наступного кроку уздовж відрізка. Фактично рівняння (7.1) являє собою рекурентне співвідношення для послідовних значень уздовж потрібного відрізка. Цей метод, що використовується для розкладання в растр відрізків, називається цифровим диференціальних аналізатором (ЦДА). У простому ЦДА або Δ_x, або Δ_y (більше із збільшень) вибирається в якості одиниці растру. Нижче наводиться простий алгоритм, який працює у всіх квадрантах.

Процес розкладання в растр відрізка за методом цифрового диференціального аналізатора (ЦДА)

Передбачається, що кінці відрізка (x_1, y_1) *і* (x_2, y_2) *не збігаються.*

Integer - *функція перетворення дійсного числа в ціле.*

Примітка: у багатьох реалізаціях функція **Integer** *означає взяття цілої частини, тобто*

Integer *(-8.5) = -9, а не -8. У даному алгоритмі використовується саме така функція,*

Sign - *функція, що повертає -1, 0, 1 для негативного, нульового і позитивного аргументу відповідно апроксимуємо довжину відрізка*

if $abs(x_2 - x_1) \geq abs(y_2 - y_1)$ **then**

Довжина = $abs(x_2 - x_1)$

else

Довжина = $abs(y_2 - y_1)$

end if

вважаємо більше із збільшень Δx *або* Δy *рівним одиниці растра*

$\Delta x = (x_2 - x_1)$/Довжина

$\Delta y = (y_2 - y_1)$/Довжина

округлюємо величини, а не відкидаємо дробову частину

використання знакової функції робить алгоритм придатним для всіх квадрантів

$x = x_1 + 0.5 *$**Sign** (Δx)

$y = y_1 + 0.5 *$**Sign** (Δy)

початок основного циклу

i = 1

while (i £ довжина)

Plot (**Integer**(x), **Integer**(y))

$x = x + \Delta x$

$y = y + \Delta y$

$i = i + 1$

end while

finish

Наведемо приклад, що ілюструє роботу алгоритму.

Приклад 7.1. Простий ЦДА у першому квадранті.

Розглянемо відрізок з точки (0, 0) в точку (5, 5). Використовуємо ЦДА для розкладання цього відрізка в растр. Результати роботи алгоритму такі:

початкові дані

$x_1 = 0$

$y_1 = 0$

$x_2 = 5$

$y_2 = 5$

Довжина = 5

$\Delta x = 1$

$\Delta y = 1$

$x = 0.5$

$y = 0.5$

результати покрокового виконання основного циклу

i	Plot	x	y
1	(0, 0)	0.5	0.5
		1.5	1.5
2	(1, 1)		
		2.5	2.5
3	(2, 2)		
		3.5	3.5
4	(3, 3)		
		4.5	4.5
5	(4, 4)	5.5	5.5

Отримане растрове подання відрізка наведено на рис. 7.1. Зауважимо, що кінцеві точки визначені точно і вибрані пікселі рівномірно розподілені вздовж відрізка. Зовнішній вигляд прямої цілком задовільний. Однак, якщо початковим значенням змінної i зробити нуль замість одиниці, то виявиться активованим піксель з координатами (5, 5), що не бажано. Якщо адреса пікселя задана цілими координатами лівого нижнього кута, то активація цього пікселя дасть явно невірну кінцеву точку відрізка (рис. 7.1). До того ж при виведенні серії послідовних відрізків піксель (5, 5) буде активований двічі, в кінці цього відрізка і на початку наступного. Такий піксель може виглядати як більш яскравий або мати інший (мабуть, неприродний) колір. Наступний приклад ілюструє роботу алгоритму в третьому квадранті.

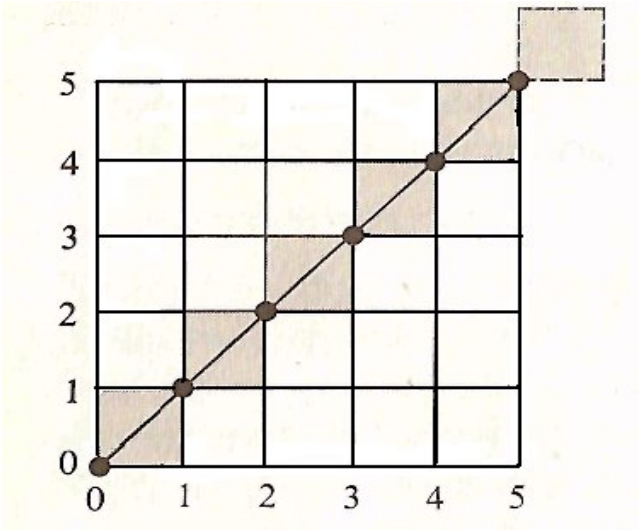


Рис. 7.1. Результати роботи простого ЦДА в першому квадранті

Приклад 7.2. Простий ЦДА в третьому квадранті.

Розглянемо відрізок з точки (0, 0) в точку (-8, -4) у третьому квадранті. Результати роботи алгоритму такі:

початкові дані

$x_1 = 0$

$y_1 = 0$

$x_2 = -8$

$y_2 = -4$

Довжина = 8

$\Delta x = -1$

$\Delta y = -0.5$

$x = -0.5$

$y = -0.5$

результати покрокового виконання основного циклу в припущенні, що використовується функція округлення, такі:

i	Plot	x	y
		-0.5	-0.5
1	(-1, -1)		
		-1.5	-1.0
2	(-2, -1)		
		-2.5	-1.5
3	(-3, -2)		
		-3.5	-2.0
4	(-4, -2)		
		-4.5	-2.5
5	(-5, -3)		
		-5.5	-3.0
6	(-6, -3)		
		-6.5	-3.5
7	(-7, -4)		
		-7.5	-4.0
8	(-8, -4)		
		-8.5	-4.5

Попри те, що результати, наведені на рис. 7.2, виглядають цілком прийнятними, аналіз відрізків, проведених з точки (0, 0) в точку (-8, 4) і (8, -4), показує, що розкладений у растр відрізок лежить по одну сторону від

реального і що на одному з кінців відрізка з'являється зайва точка. Тобто результат роботи алгоритму залежить від орієнтації. Отже, точність у кінцевих точках погіршується. Далі, якщо замість взяття цілої частини використовувати округлення до найближчого цілого, то результати знову вийдуть різними.

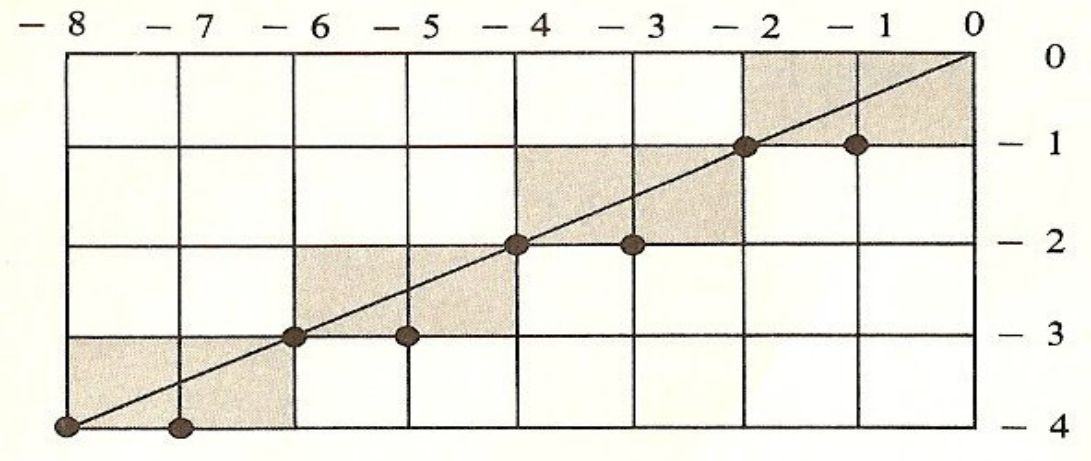


Рис. 7.2. Результати роботи простого ЦДА в третьому квадранті

Таким чином, або потрібно використовувати більш складний і більш повільний алгоритм, або потрібно відступитися від вимоги максимально точної апроксимації. До того ж запропонований алгоритм має той недолік, що описаний більш прийнятний алгоритм.

7.2. Алгоритм Брезенхема

Хоча алгоритм Брезенхема був спочатку розроблений для цифрових графопобудовувачів, однак, також підходить і для використання растровими пристроями з ЕПТ. Алгоритм вибирає оптимальні растрові координати для подання відрізка. У процесі роботи одна з координат − або x, або y (в залежності від кутового коефіцієнта) − змінюється на одиницю. Зміна іншої координати (або на нуль, або на одиницю) залежить від відстані між координатами сітки. Таку відстань ми назвемо помилкою.

Алгоритм побудований так, що потрібно перевіряти лише знак цієї помилки. На рис. 7.3 це ілюструється для відрізка в першому октанті, тобто, для відрізка з кутовим коефіцієнтом, що лежить в діапазоні від нуля до одиниці. З рисунка зрозуміло, що якщо кутовий коефіцієнт відрізка з точки (0, 0) більший 1/2, то його перетинання з прямою = 0. Отже, точка растру (1, 1) краще апроксимує проходження відрізка, ніж точка (1, 0). Якщо кутовий коефіцієнт менше 1/2, то вірно зворотнє. Для кутового коефіцієнта, рівного

1/2, немає будь-якого бажаного вибору. У даному випадку алгоритм обирає точку (1, 1).

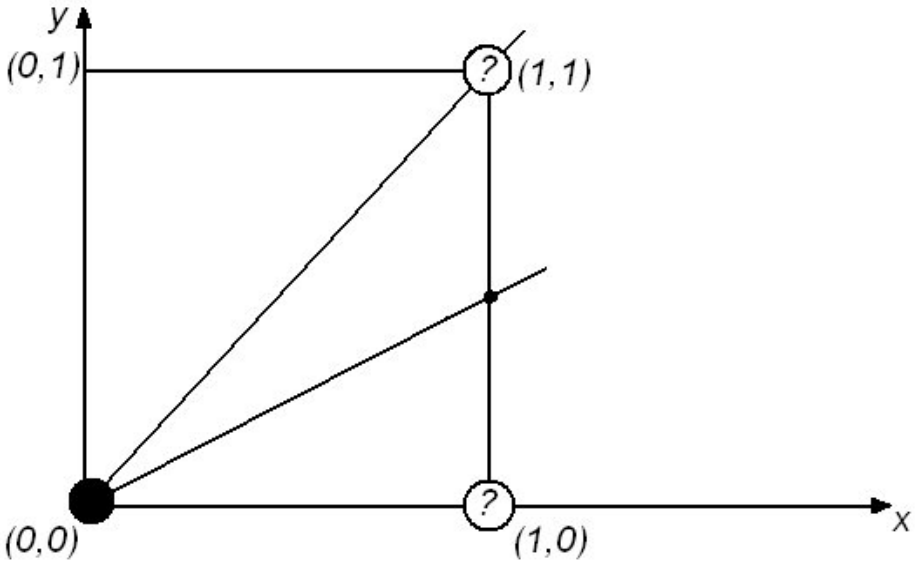


Рис. 7.3. Основна ідея алгоритму Брезенхема

Не всі відрізки проходять через точки растру. Подібна ситуація ілюструється на рис. 7.4, де відрізок з тангенсом кута нахилу 3/8 спочатку проходить через точку растра (0, 0) і послідовно перетинає три пікселя. Також ілюструється обчислення помилки при поданні відрізка дискретними пікселями. Так як бажано перевіряти лише знак помилки, то вона спочатку встановлюється рівною − 1/2. Таким чином, якщо кутовий коефіцієнт відрізка більший чи дорівнює 1/2, то величина помилки в такій точці растру з координатами (1, 0) може бути обчислена як

$$e = e + m,$$

де m - кутовий коефіцієнт. У нашому випадку при початковому значенні помилки − 1/2

$$e = -1/8 + 3/8 = 1/4,$$

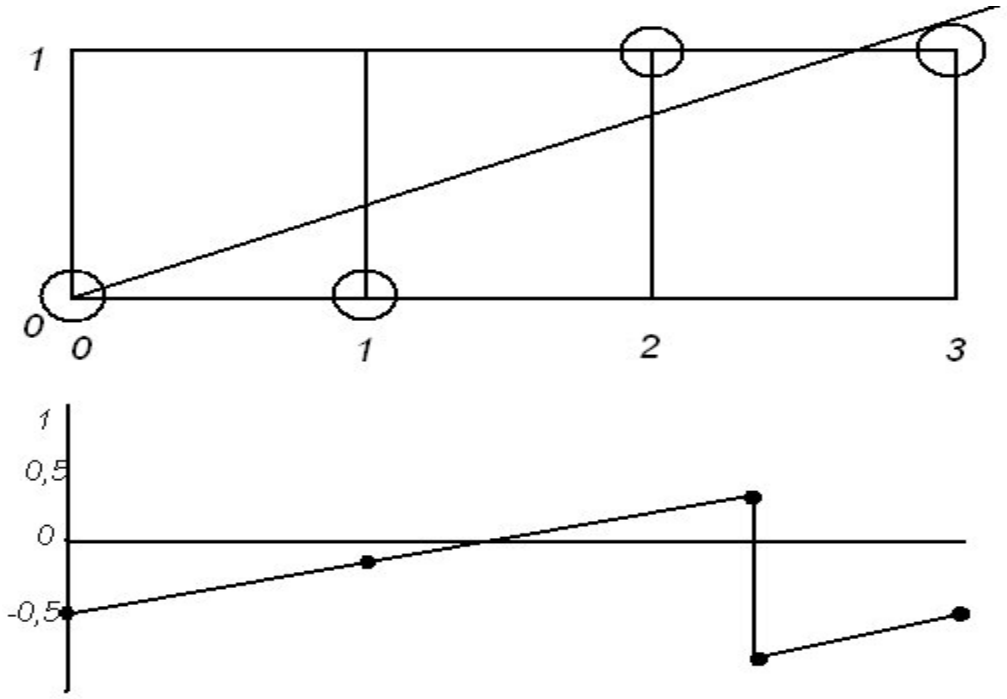


Рис. 7.4. Графік помилки в алгоритмі Брезенхема

Оскільки e негативне, то відрізок пройде нижче середини пікселя. Отже, піксель на тому ж самому горизонтальному рівні краще апроксимує положення відрізка, тому y не збільшується. Аналогічно обчислюємо помилку

$$e = -\frac{1}{8} + \frac{3}{8} = 1/4,$$

в наступній точці растру (2, 0). Тепер e позитивне, а значить, відрізок пройде вище середньої точки. Растровий елемент (2, 1) з наступною за величиною координатою y краще апроксимує положення відрізка. Отже, y збільшується на одиницю. Перш ніж розглядати наступний піксель, необхідно відкоригувати помилку вирахуванням з неї одиниці. Маємо:

$$e = \frac{1}{4} - 1 = -3/4.$$

Зауважимо, що перетин вертикальної прямої x = 2 із заданим відрізком лежить на 1/4 нижче прямої y = 1. Якщо ж перенести відрізок 1/2 вниз, то ми отримаємо саме величину − 3/4. Продовження обчислень для наступного пікселя дасть

$$e = -\frac{3}{4} + \frac{3}{8} = -3/8$$
,

Оскільки e негативне, то Y не збільшується. З усього сказаного випливає, що помилка − це інтервал, що відсікається по осі y розглянутим відрізком у кожному растровому елементі (відносно − 1/2).

Наведемо алгоритм Брезенхема для першого октанта, тобто для випадку 0 £ D*y* £ D*x* .

Алгоритм Брезенхема розкладання в растр відрізка для першого октанту

передбачається, що кінці відрізка (x_1, y_1) *і* (x_2, y_2) *не збігаються.*

Integer − *функція перетворення в ціле*

$x, y, \Delta x, \Delta y$ − *цілі*

y − *дійсне*

ініціалізація змінних

$x = x_1$

$y = y_1$

$\Delta x = x_2 - y_1$

$\Delta y = y_2 - y_1$

ініціалізація e *з поправкою на половину пікселя*

$$e = \frac{\Delta y}{\Delta x} + 1/2$$

початок основного циклу

for i = 1 **to** Δx

Plot (x, y)

While (e ³ 0)

$y = y + 1$

$e = e - 1$

end while

$x = x + 1$

$e = e + \Delta x/\Delta y$

next i

finish

Блок-схема алгоритму наведено на рис. 7.5. Приклад подано нижче.

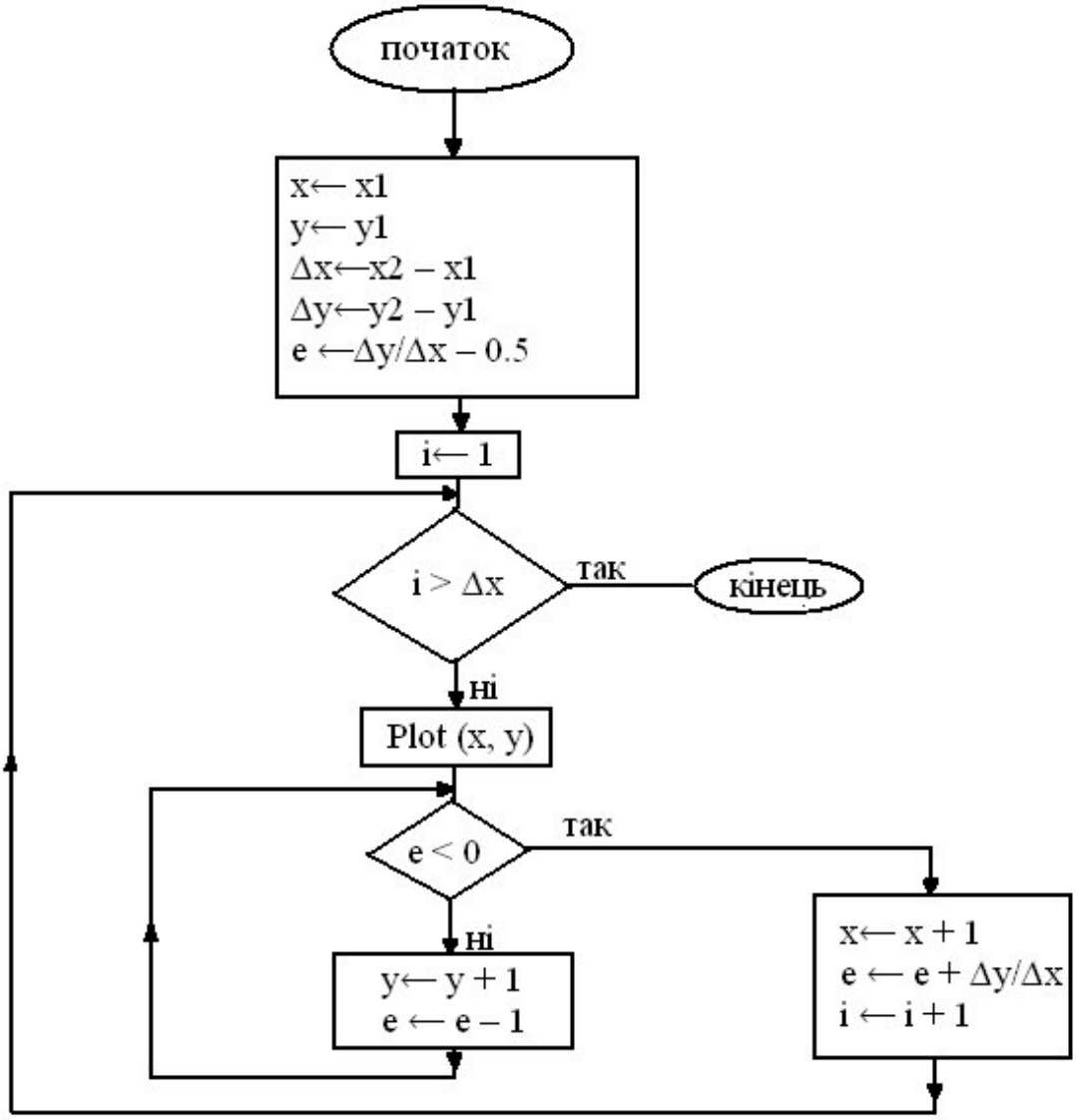


Рис. 7.5. Блок-схема алгоритму Брезенхема

Приклад 7.3. Алгоритм Брезенхема.

Розглянемо відрізок проведений з точки (0, 0) в точку (5, 5). Розкладання відрізка в растр за алгоритмом Брезенхема призводить до такого результату:

початкові дані

$x = 0$

$y = 0$

$\Delta x = 5$

$\Delta y = 5$

$e = 1 - 1/2 = 1/2$

Результати покрокового виконання основного циклу

i	Plot	e	X	Y
		1/2	0	0
1	(0, 0)			
		−1/2	0	1
		1/2	1	1
2	(1, 1)			
		−1/2	1	2
		1/2	2	2
3	(2, 2)			
		−1/2	2	3
		1/2	3	3
4	(3, 3)			
		−1/2	3	4
		1/2	4	4
5	(4, 4)			
		−1/2	4	5
		1/2	5	5

Результат наведений на рис. 7.6 збігається з очікуваним. Зауважимо, що точка растру з координатами (5, 5) не активована. Цю точку можна активувати шляхом зміни умови циклу **for-next** на 0 **to** Δx. Активацію точки (0, 0) можна усунути, якщо поставити оператор **Plot** безпосередньо перед рядком **next** i.

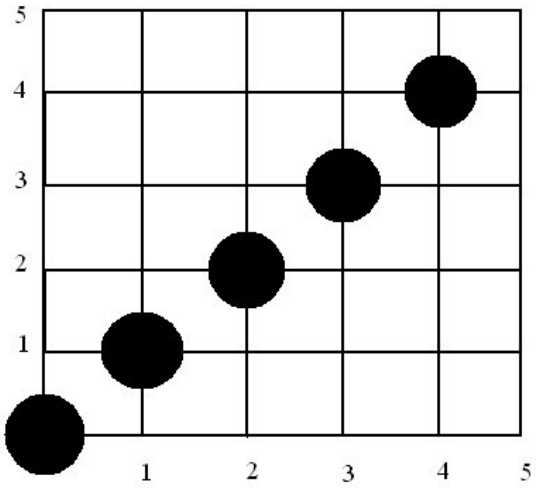


Рис. 7.6. Результат роботи алгоритму Брезенхема в першому октанті

7.3. Цілочисельний алгоритм Брезенхема

Алгоритм Брезенхема в такому вигляді, як він представлений вище, вимагає використання арифметики з плаваючою точкою і ділення (для обчислення кутового коефіцієнта та оцінки помилки). Швидкодію алгоритму можна збільшити, якщо використовувати лише цілочисельну арифметику і виключити ділення. Оскільки важливий лише знак помилки, то просте перетворення

$$\bar{e} = 2e\,\Delta x,$$

перетворить попередній алгоритм у цілочисельний і дозволить ефективно реалізувати його на апаратному чи мікропрограмного рівні. Модифікований цілочисельний алгоритм для першого октанта, тобто для 0 £ D*y* £ D*x* , такий:

Цілочисельний алгоритм Брезенхема для першого октанта

передбачається, що кінці відрізка (x_1, y_1) *і* (x_2, y_2) *не збігаються та всі змінні − цілі.*

$x = x_1$

$y = y_1$

$\Delta x = y_2 - x_1$

$\Delta y = y_2 - x_1$

ініціалізуємо $\bar{e}$ *з поправкою на половину пікселя.*

$\bar{e} = 2 * \Delta y - \Delta x$

Початок основного циклу

for i = 1 **to** Δx

Plot (x, y)

while ($\bar{e}$ ³ 0)

$y = y + 1$

$\bar{e} = \bar{e} - 2 * \Delta y$

end while

$x = x + 1$

$\bar{e} = \bar{e} + 2 * \Delta y$

next i

finish

Блок-схему, наведену на рис. 7.5, можна застосувати і в даному випадку з відповідними змінами в обчисленні помилки.

7.4. Загальний алгоритм Брезенхема

Для того, щоб реалізація алгоритму Брезенхема була повною, необхідно обробляти відрізки у всіх октантах. Модифікацію легко зробити, враховуючи в алгоритмі номер квадранта, в якому лежить відрізок і його кутовий коефіцієнт. Коли абсолютна величина кутового коефіцієнта більше 1, y використовується для прийняття рішення про зміну величини x. Вибір постійно змінюється (на +1 або −1) координати залежить від квадранта (рис. 7.7).

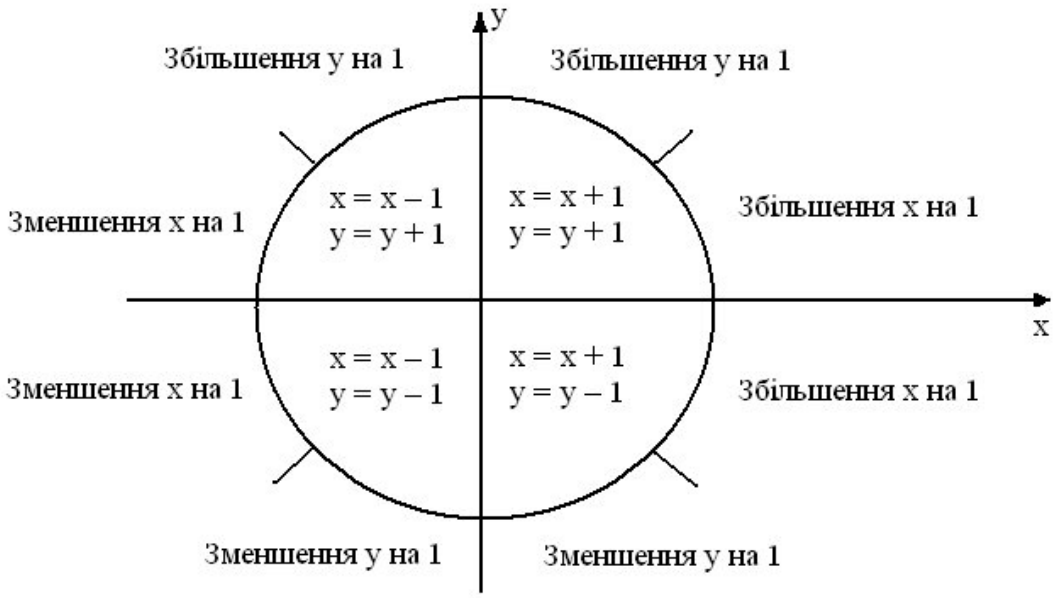


Рис. 7.7. Розбір випадків для узагальненого алгоритму Брезенхема

Загальний алгоритм може мати такий вигляд:

Загальнений цілочисельний алгоритм Брезенхема квадрантів

передбачається, що кінці відрізка (x_1, y_1) *і* (x_2, y_2) *не збігаються*

всі змінні вважаються цілими

функція ***Sign*** *повертає -1, 0, 1 для негативного, нульового і позитивного аргументу відповідно*

ініціалізація змінних

$x = x_1$

$y = y_1$

$\Delta x = abs(x_2 - x_1)$

$\Delta y = abs(y_2 - y_1)$

$s_1 = \mathbf{Sign}\ (x_2 - x_1)$

$s_2 = \mathbf{Sign}\ (y_2 - y_1)$

обмін значень Δx *та* Δy *у залежності від кутового коефіцієнта нахилу відрізка*

if $\Delta y > \Delta x$ **then**

тим = Δx

$\Delta x = \Delta y$

Δy = тим

Обмін =1

else

Обмін = 0

end if

ініціалізація $\bar{e}$ *з поправкою на половину пікселя*

$\bar{e} = 2 * \Delta y - \Delta x$

основний цикл

for *i = 1* **to** Δx

Plot (x, y)

while ($\bar{e}$ ³ 0)

if Обмін = 1 **then**

$y = y + s_1$

else

$y = y + s_2$

end if

$\bar{e} = \bar{e} - 2 * \Delta x$

end while

if Обмін = 1 **then**

$y = y + s_2$

else

$x = x + s_1$

end if

$\bar{e} = \bar{e} + 2 * \Delta y$

next i

finish

Приклад 7.4. Узагальнений алгоритм Брезенхема

Для ілюстрації загального алгоритму Брезенхема розглянемо відрізок із точки (0, 0) в точку (-8, -4). У прикладі 7.2. цей відрізок був оброблений за допомогою простого ЦДА

початкові дані

$x = 0$

$y = 0$

$\Delta x = 8$

$\Delta y = 4$

$s_1 = -1$

$s_2 = -1$

Обмін = 0

$e = 0$

послідовне виконання основного циклу

i	Plot	e	x	y
		0	0	0
1	(0, 0)			
		−16	0	−1
		−8	−1	−1
2	(−1,−1)			
		0	−2	−1
3	(−2, 1)			
		−16	−2	−2
		−8	−3	−2
4	(−3,−2)			
		0	−4	−2
5	(−4, 2)			
		−16	−4	−3
		−8	−5	−3
6	(−5,−3)			
		0	−6	−3
7	(−6, 3)			
		−16	−6	−4
		−8	−7	−4
8	(−7, 4)			
		0	−8	−4

На рис. 7.8 наведений результат. Результати роботи двох алгоритмів відрізняються, рис. 7.2 і 7.8.

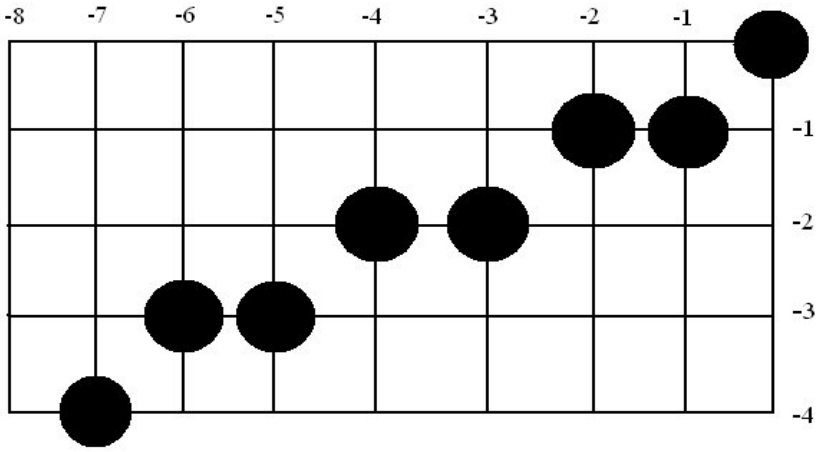


Рис. 7.8. Результат роботи узагальненого алгоритму Брезенхема в третьому квадранті

7.5. Алгоритм Брезенхема для генерування кола

У растр потрібно розкладати не лише лінійні, але й інші, більш складні функції. Розкладання конічних перерізів, тобто кіл, еліпсів, парабол, гіпербол, було присвячено значну кількість робіт. Один з найбільш ефективних і простих для розуміння алгоритмів генерації кола належить Брезенхему. Для початку зауважимо, що необхідно згенерувати лише одну восьму частину кола. Решта її частин можуть бути отримані послідовними відбитками, як це наведено на рис. 7.9. Якщо згенерований перший октант (від 0 до 45° проти годинникової стрілки), то другий октант можна отримати дзеркальним відображенням відносно прямої $y = x$, що дає в сукупності перший квадрант. Перший квадрант відбивається відносно прямої $x = 0$ для отримання відповідних перетворень.

Для виведення алгоритму розглянемо першу чверть кола з центром у початку координат. Зауважимо, що якщо робота алгоритму починається в точці $x = 0$, $y = R$, то при генерації кола за годинниковою стрілкою в першому квадранті y є монотонно спадною функцією аргументу x (рис. 7.10). Аналогічно, якщо вихідною точкою є $y = 0, x = R$, то при генерації кола проти годинникової стрілки x буде монотонно спадаючою функцією аргументу y.

У нашому випадку вибирається генерація за годинниковою стрілкою з початком в точці $x = 0, y = R$. Передбачається, що центр кола та початкова точка знаходяться точно в точках растру.

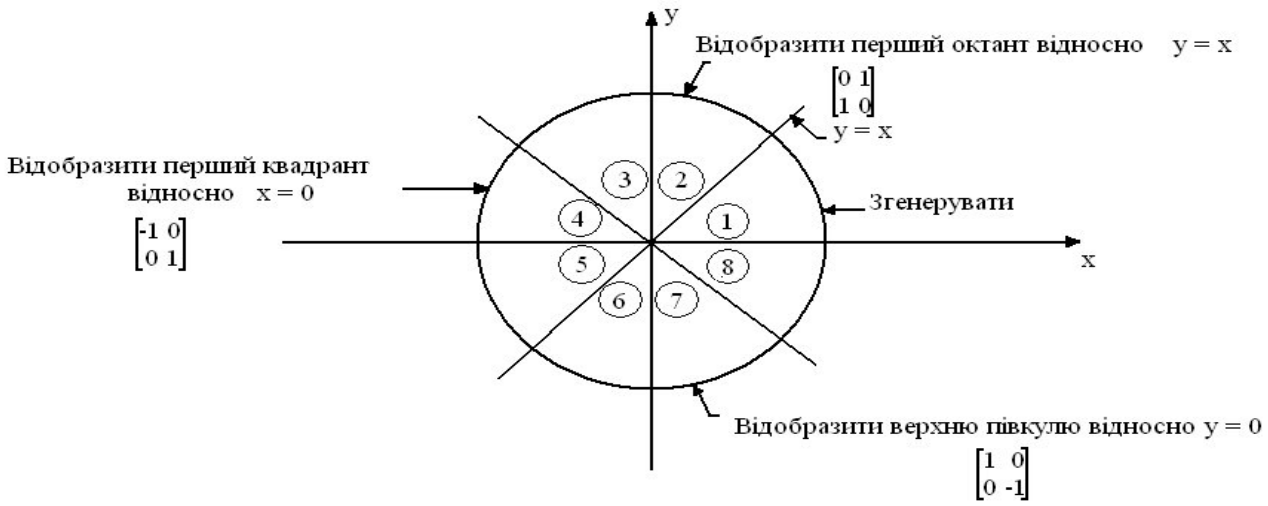


Рис. 7.9. Генерація повної окружності з дуги в першому октанті

Для будь-якої заданої точки на колі при генерації за годинниковою стрілкою існує лише три можливості вибрати наступний піксель, що найкращим чином наближає коло: горизонтально вправо, по діагоналі вниз і вправо, вертикально вниз.

На рис. 7.11 ці напрямки позначені відповідно m_n, m_D, m_V. Алгоритм вибирає піксель, для якого мінімальний квадрат відстані між одним із цих пікселів і колом, тобто мінімум

$$m_H = |(x_1 + 1)^2 + (y_i)^2 - R^2|$$
$$m_D = |(x_i + 1)^2 + (y_i - 1)^2 - R^2|$$
$$m_V = |(x_i)^2 + (y_i - 1)^2 - R^2|$$

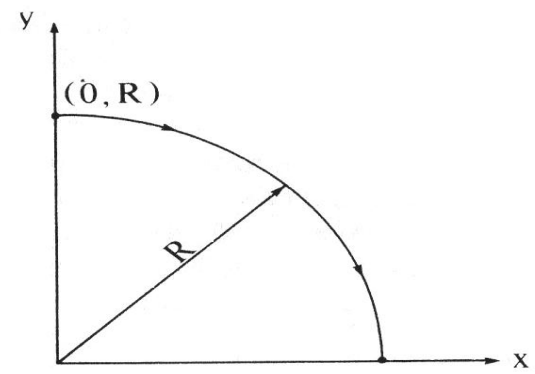


Рис. 7.10. Коло в першу квадранті

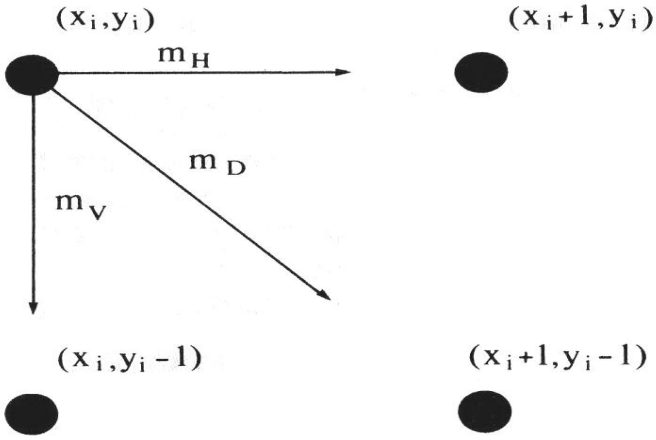


Рис. 7.11. Вибір пікселів в першому квадранті

Обчислення можна спростити, якщо зауважити, що в околі точки (x_i, y_i) можливі лише п'ять типів перетинів кола та сітки растру, що наведені на рис. 7.12.

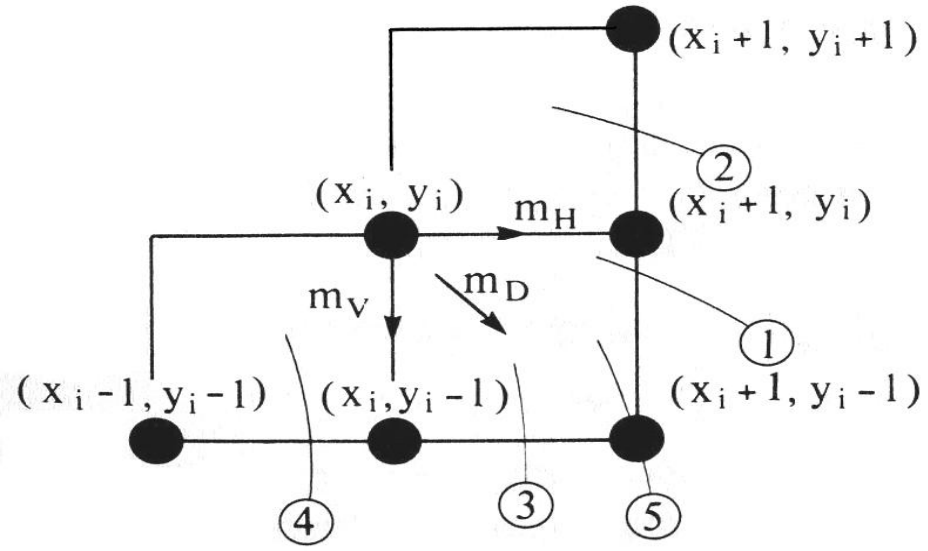


Рис. 7.12. Перетин кола та сітки растру

Різниця між квадратами відстаней від центра кола до діагонального пікселя $(x_i + 1, y_i - 1)$ і від центра до точки на колі R_2 дорівнює:

$$\Delta_i = (x_i + 1)^2 + (y_i - 1)^2 - R^2.$$

Як і в алгоритмі Брезенхема для відрізка, для вибору відповідного пікселя бажано використовувати лише знак помилки, а не її величину.

При $\Delta_i < 0$ діагональна точка $(x_i + 1, y_i - 1)$ знаходиться всередині реального кола, тобто, це випадки 1 або 2 наведені на рис. 7.12. Ясно, що в даній ситуації слід вибрати чи піксель $(x_i + 1, y_i)$, тобто m_H, чи піксель $(x_i + 1, y_i - 1)$, тобто, m_D. Для цього спочатку розглянемо випадок 1 і перевіримо різницю квадратів відстаней від кола до пікселів у горизонтальному і діагональному напрямках:

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

При $\delta < 0$ відстань від кола до діагонального пікселя (m_D) більша, ніж до горизонтального (m_H). Навпаки, якщо $\delta > 0$, відстань до горизонтального

пікселя (m_H) більше. Таким чином,

$$\text{при } \delta \ll 0 \text{ вибираємо } m_H \text{ в } (x_i + 1, y_i),$$
$$\text{при } \delta > 0 \text{ вибираємо } m_D \text{ в } (x_i + 1, y_i - 1).$$

При $\delta = 0$, коли відстань від кола до обох пікселів однакова, вибираємо горизонтальний крок.

Кількість обчислень, необхідних для оцінки величини δ, можна скоротити, якщо зауважити, що у випадку 1:

$$(x_i + 1)^2 + (y_i)^2 - R^2 \gg 0,$$
$$(x_i + 1)^2 + (y_i - 1)^2 - R^2 < 0,$$

оскільки діагональний піксель $(x_i + 1, y_i - 1)$ завжди лежить в колі, а горизонтальний $(x_i + 1, y_i)$ − поза ним. Таким чином, δ можна обчислити за формулою:

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

Доповнення до повного квадрата члена $(y_i)^2$, з допомогою додавання і віднімання $-2y, +1$ дає

$$\delta = 2[(x_i + 1)^2 + (y_i - 1)^2 - R^2] + 2y_i - 1.$$

У квадратних дужках стоїть за визначенням Δ_i і його підстановка істотно спрощує вираз.

Розглянемо випадок 2 наведений на рис. 7.12 і зауважимо, що тут повинен бути вибраний горизонтальний піксель $(x_i + 1, y_i)$, оскільки y є монотонно спадною функцією. Перевірка, компонента δ показує, що

$$(x_i + 1) + (y_i)^2 - R^2 < 0,$$
$$(x_i + 1) + (y_i - 1)^2 - R^2 < 0.$$

оскільки в випадку 2 горизонтальний $(x_i + 1, y_i)$ та діагональний $(x_i + 1, y_i - 1)$ пікселі лежать у колі. Отже, $\delta < 0$, і при використанні того ж самого критерію, що й у випадку 1, вибирається піксель $(x_i + 1, y_i)$.

Якщо $\Delta_1 > 0$, то діагональна точка $(x_i + 1, y_i - 1)$ знаходиться за колом, тобто це випадок 3 і 4 наведені на рис. 7.12. У даній ситуації зрозуміло, що потрібно обрати або піксель $(x_i + 1, y_i - 1)$, тобто, m_D, або $(x_i, y_i - 1)$, тобто, m_V. Аналогічно розгляду попереднього випадку критерій вибору можна отримати, розглядаючи спочатку випадок 3 і перевіряючи різницю між квадратами відстаней від кола до діагонального m_D і вертикального m_V, пікселів, тобто,

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

При $\delta' < 0$ відстань від кола до вертикального пікселя $(x_i, y_i - 1)$ більша і слід вибрати діагональний крок m_D до пікселя $(x_i + 1, y_i - 1)$. І навпаки, у випадку $\delta' > 0$ відстань від кола до діагонального пікселя більша і слід вибрати вертикальний рух до пікселя $(x_i, y_i - 1)$. Таким чином,

при d¢£ 0 вибираємо m_D в $(x_i + 1, y_i - 1)$,
при $\delta' > 0$ вибираємо m_V в $(x_i, y_i - 1)$.

Тут у випадку $\delta' = 0$, тобто коли відстані рівні, обраний діагональний крок.

Перевірка компонента δ' показує, що:

$$(x_i + 1)^2 + (y_i - 1)^2 - R^2 £ 0,$$
$$(x_i)^2 + (y_i - 1)^2 - R^2 < 0.$$

Оскільки для випадку 3 діагональний піксель $(x_i + 1, y_1 - 1)$ знаходиться за колом, тоді як вертикальний піксель $(x_i, y_i - 1)$ лежить всередині його. Це дозволяє записати δ' у вигляді:

$$\delta' = (x_i + 1)^2 + (y_i - 1)^2 - R^2 + (x_i)^2 + (y_i - 1)^2 - R^2.$$

Доповнення до повного квадрата члена $(x_i)^2$ за допомогою додавання і віднімання $2x_i + 1$ дає:

$$\delta' = 2[(x_i + 1)^2 + (y_i - 1)^2 - R^2] - 2x_i - 1$$

Використання визначення Δ_i приводить вираз до виду:

$$\delta' = 2(\Delta_i - x_i) - 1.$$

Тепер розглядаємо випадок 4, та знову зауважимо, що слід вибрати вертикальний піксель $(x_i, y_i - 1)$, оскільки y є монотонно спадаючою функцією при зростанні x.

Перевірка компонент δ' для випадку 4 показує, що:

$$(x_i + 1)^2 + (y_i - 1)^2 - R^2 > 0,$$
$$(x_i)^2 + (y_i - 1)^2 - R^2 > 0.$$

Оскільки обидва пікселя знаходяться за колом то, і при використанні критерію, розробленого для випадку 3, відбувається вірний вибір m_V.

Залишилось перевірити лише випадок 5 наведений на рис. 7.12, який зустрічається, коли діагональний піксель $(x_i + 1, y_i - 1)$ лежить на колі, тобто $\Delta_i = 0$. Перевірка компонента δ показує, що:

$$(x_i + 1)^2 + (y_i)^2 - R^2 > 0,$$
$$(x_i + 1)^2 + (y_i - 1)^2 - R^2 = 0.$$

Отже, $\delta > 0$ і обирається діагональний піксель $(x_i + 1, y_i - 1)$. Аналогічним чином оцінюємо компоненти δ':

$$(x_i + 1)^2 + (y_i - 1)^2 - R^2 = 0,$$
$$(x_i)^2 + (y_i - 1)^2 - R^2 < 0.$$

І $\delta' < 0$, що є умовою вибору правильного діагонального кроку до $(x_i + 1, y_i - 1)$. Таким чином, випадок $\Delta_i = 0$ підпорядковується тому ж критерію, що й випадок $\Delta_i < 0$ або $\Delta_i > 0$.

Підіб'ємо підсумок отриманих результатів:

$\Delta_i < 0$

d £ 0 Вибираємо піксель $(x_i + 1, y_i) \rightarrow m_H$

$\delta > 0$ Вибираємо піксель $(x_i + 1, y_i - 1) \rightarrow m_D$

$\Delta_i > 0$

d¢£ 0 Вибираємо піксель $(x_i + 1, y_i - 1) \rightarrow m_D$

$\delta' > 0$ Вибираємо піксель $(x_i, y_i - 1) \rightarrow m_V$

$\Delta_i = 0$ Вибираємо піксель $(x_i + 1, y_i - 1) \rightarrow m_D$.

Легко розробити прості рекурентні співвідношення для реалізації покрокового алгоритму. Спочатку розглянемо горизонтальний крок m_H до пікселя $(x_i + 1, y_i)$. Позначимо це нове положення пікселя як $(i + 1)$. Тоді координати нового пікселя і значення Δ_i рівні

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

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

$$\Delta_{i+1} = (x_{i+1} + 1)^2 + (y_{i+1} - 1)^2 - R^2$$

$$= (x_{i+1})^2 + 2x_{i+1} + 1 + (y_i - 1)^2 - R^2$$

$$= (x_i + 1)^2 + (y_i - 1)^2 - R^2 + 2x_{i+1} + 1$$

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

Аналогічно координати нового пікселя і значення Δ_i для кроку m_D до пікселя $(x_i + 1, y_i - 1)$ такі:

$$x_{i+1} = x_i + 1$$
$$y_{i+1} = y_i - 1$$
$$\Delta_{i+1} = \Delta_i + 2x_{i+1} - 2y_{i+1} + 2$$

Те ж саме для кроку m_V до $(x_i, y_i - 1)$

$$x_{i+1} = x_i$$
$$y_{i+1} = y_i - 1$$
$$\Delta_{i+1} = \Delta_i - 2y_{i+1} + 1$$

Реалізація алгоритму Брезенхема на псевдокоді для кола подана нижче.

Покроковий алгоритм Брезенхема для створення кола в першому квадранті

всі змінні – цілі

ініціалізація змінних

$x_i = 0$

$y_i = R$

$\Delta_i = 2(1 - R)$

межа = 0

Plot (x_i, y_i)

if $y_i \ll$ Межа **then** 4

Виділення випадку 1 або 2, 4 або 5, або 3

if $\Delta_i < 0$ **then** 2

if $\Delta_i > 0$ **then** 3

if $\Delta_i = 0$ **then** 20

визначення випадку 1 або 2

$\delta = 2\Delta_i + 2y_i - 1$

If d £ 0 **then** 10

If $\delta > 0$ **then** 20

визначення випадку 4 або 5

$\delta = 2\Delta_i + 2y_i - 1$

if d £ 0 **then** 20

if $\delta > 0$ **then** 30

виконання кроків

крок m_H

$x_i = x_i + 1$

$\Delta_i = \Delta_i + 2x_i + 1$

go to 1

крок m_D

$x_i = x_i + 1$

$y_i = y_i - 1$

$\Delta_i = \Delta_i + 2x_i - 2y_i + 2$

go to 1

крок m_V

$y_i = y_i - 1$

$\Delta_i = \Delta_i - 2y_i + 1$

go to 1

finish

Зміна межі встановлюється в нуль для закінчення роботи алгоритму на горизонтальній осі в результаті генерується коло в першому квадранті. Якщо необхідний лише один з октантів, то другий октант можна отримати – за допомогою установки Межа = **Integer** $(R/\sqrt{2})$, а перший - за допомогою відображення другого октанта відносно прямої $y = x$. Блок-схема алгоритму наведена на рис. 7.13.

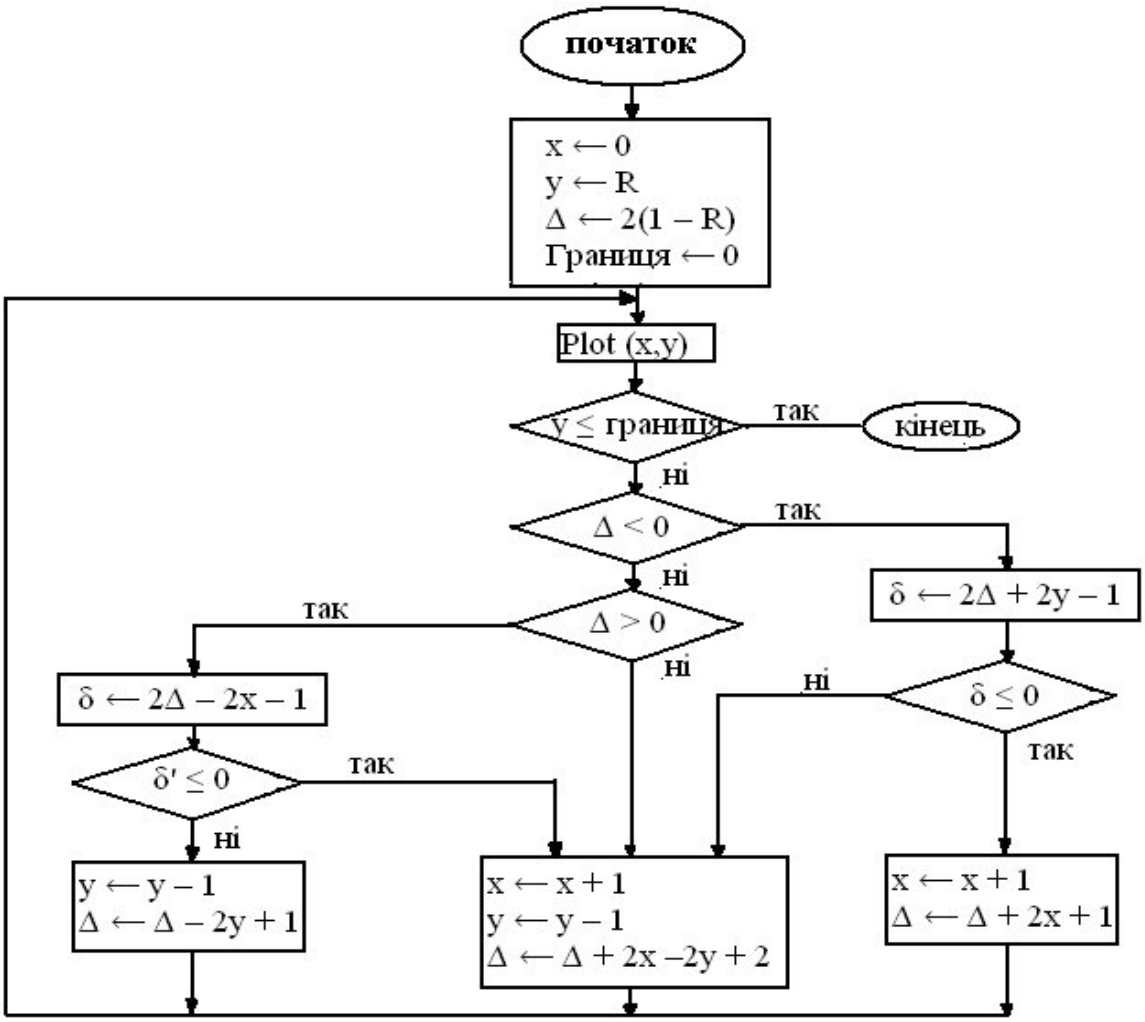


Рис. 7.13. Блок-схема покрокового алгоритму Брезенхема створення кола в першому квадранті

Приклад 7.5. Алгоритм Брезенхема для кола.

Для ілюстрації роботи алгоритму генерації кола розглянемо коло радіусом 8, з центром у початку координат. Генерується лише перший квадрант

початкові дані

$x = 0$

$y = 8$

$\Delta_i = 2(1 - 8) = -14$

Межа = 0

Покрокове виконання основного циклу

Plot (0,8)

$y_i >$ Межа продовжувати

$\Delta_i < 0$ **go to** 2

$\delta = 2(-14) + 2(8) - 1 = -13 < 0$ **go to** 10

$x = 0 + 1 = 1$

$\Delta_i = -14 + 2 + 1 = -11$

g**o to** 1

Plot (1, 8)

$y_i >$ Межа продовжувати

$\Delta_i < 0$ **go to** 2

$\delta' = 2(-11) + 2(8) - 1 = -7 < 0$ **go to** 10

$x = 1 + 1 = 2$

$\Delta_i = -11 + 2(2) + 1 = -6$

go to 1

Plot (2, 8)

–

–

–

продовжувати

Результати всіх послідовних проходжень алгоритму зведені в таблицю. Список пікселів, обраних алгоритмом, складається з (0, 8), (1, 8), (2, 8), (3, 7), (4, 7), (5,6), (6, 5), (7, 3), (8, 2), (8, 1), (8, 0)

Plot	Δ_i	δ	δ'	x	Y
	−14			0	8
(0, 8)					
	−11	−13		1	8
(1, 8)					
	−6	−7		2	8
(2, 8)					
	−12	3		3	7
(3, 7)					
	−3	−11		4	7
(4, 7)					
	−3	−7		5	6
(5, 6)					
	1	5		6	5
(6, 5)					
	9		−11	7	4
(7, 4)					
	4		3	7	3
(7, 3)					
	18		−7	8	2
(8, 2)					
	17		19	8	1
(8, 1)					
	18		17	8	0
(8, 0)					
Алгоритм закінчено					

Результати наведені на рис. 7.14 разом із реальною окружністю. Алгоритм легко узагальнюється для інших квадрантів або дуг кіл.

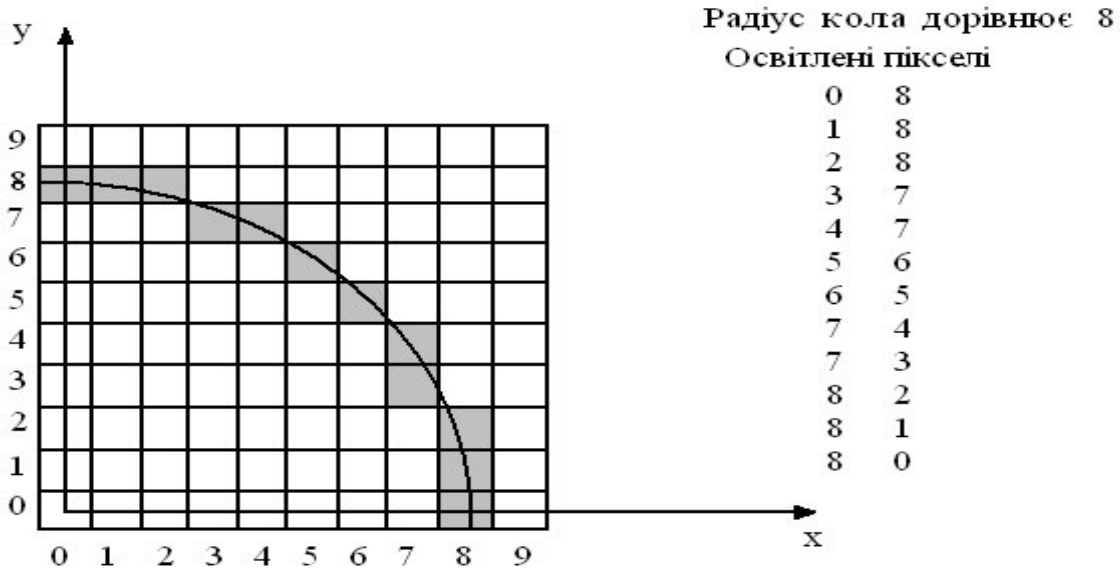


Рис. 7.14. Результати роботи покрокового алгоритму Брезенхема генерації кола

Приклади реалізації

Приклад 7.6. Використовуючи ЦДА та алгоритм Брезенхема, написати програму для викреслювання відрізків з однієї довільної точки в іншу на псевдорастрі 32×32. Використовуйте псевдобуфер кадру у вигляді одномірного масиву спочатку для зберігання зображення, а тоді для виводу його з псевдобуфера на екран. Демонстраційний текст повинен складатися, принаймні, з 16 відрізків з початком у центрі кола і кінцями, що рівномірно розташовані по колу. Необхідно забезпечити можливість встановлювати центр кола в довільну точку растру. Візуально порівняєте результати. Введіть список активованих пікселів для відрізка з (0,0) в (-8, -3) для обох алгоритмів. Як впливає на результат ініціалізація? Порівняйте обчислювальну ефективності двох алгоритмів шляхом хронометражу розкладених в растр 100 випадково вибраних відрізків.

```
uses graph;
var
  grDriver: Integer;
  grMode: Integer;

function sign(v:real): integer;
begin
```

```
    if v>0 then sign:=1;
    if v=0 then sign:=0;
    if v<0 then sign:=-1;
end;

procedure p_line(x1,y1,x2,y2:integer);
var
  x,y,dx,dy,len:real;
  i:integer;
begin
    if abs(x2-x1)>=abs(y2-y1) then begin
    len:=abs(x2-x1);
 end else
      len:=abs(y2-y1);
 dx:=(x2-x1)/len;
 dy:=(y2-y1)/len;
 x:=xl+0.5*sign(dx);
 y:=yl+0.5*sign(dy);
 i:=l;
 while (i<=len) do begin
      putpixel(trunc(x),trunc(y),10);
    x;=x+dx;
    y:=y+dy;
      inc(i);
 end;

end;
procedure b_line(xl,y1,x2,y2:integer);
var
  tmp,e,x,y,dx,dy:real;
  xchng,i,s1,s2:integer;
begin
    x:=x1;
    y:=y1;
    dx:=abs(x2-x1);
    dy:=abs(y2-y1);
    s1:=sign(x2-x1);
    s2:=sign(y2-y1);
```

```
    if dy>dx then begin
      tmp:=dx;
      dx:=dy;
      dy:=tmp;
      xchng:=1;
  end else
      xchng:=0;
  e:=2*dy-dx;
  for i:=l to trunc(dx) do begin
    putpixel(trunc(x),tnmc(y),12);
   while e>=0 do begin
       if xchng=1 then x:=x+s1
       else
          y:=y+s2;
          e:=e~2*dx;
   end;
       if xchng=1 then y:=y+s2
       else
          x:=x+sl;
          e:=e+2*dy;
   end;
end;

function readkey:word;
var

  ww:word;
begin
    asm
    mov ah,10h
    int$16
    mov ww,ax
    end;
    readkey:=ww;
end;
```

```
begin
   {init graphiX}
     grDriver:=EGA;
     grMode:=2;
     InitGraph(grDriver,grMode,'');
   p_line(50,100,2,160);
   b_line(200,l 10,2,150);
     readkey;
end.
```

Приклад 7.7. Коло, розкладене в растр, можна отримати за допомогою алгоритму Брезенхема для генерації кола, описаного в розділі 7.6. його можна також згенерувати шляхом розкладення в растр ребер вписаного багатокутника за допомогою алгоритму Брезенхема для відрізка. Напишіть програму, яка б реалізувала обидва методи, та розкладіть в растр коло радіусом R=15 на растрі 32x32. Порівняйте результати для вписаних багатокутників з 4, 8, 16, 32, 64, 128 сторонами для алгоритму генерації кола. Використайте псевдобуфер кадра у вигляді одновимірного масиву спочатку для збереження інформації, а тоді для виводу його з псевдобуфера на дисплей. Передбачте вивід списку растрових точок, використовуючи формат «стрічка-колонка» для кожного алгоритму у припущенні, що початок координат (0,0) знаходиться в нижньому лівому куті. Результати порівняйте візуально і чисельно.

```
uses graph;
var
 grDriver: Integer;
 grMode: Integer;
 ErrCode: Integer;

 procedure putP(x,y:integer);
 begin
 putpixel(x+100,y+50,15);
 end;

 procedure BrezenchemPlot(x1,y1,x2,y2:integer);{-- процедура побудови відрізка
алгоритмом Брезенхема}
```

```
var i,x,y,dx,dy,tmp,ch,s1,s2,e:integer;
begin
x:=x1;
y:=y1;
dx:=abs(x2-x1);
dy:=abs(y2-y1);
if dx=0 then s1:=0 else s1:=(x2-x1) div dx;
if dy=0 then s2:=0 else s2:=(y2-y1) div dy;
if dy>dx then
 begin
 tmp:=dx;
 dx:=dy;
 dy:=tmp;
 ch:=1;
 end
 else ch:=0;
 e:=2*dy-dx;
 for i:=1 to dx do
   begin
   putpixel(x+150,y+50,10);
   while e>=0 do
       begin
       if ch=1 then x:=x+s1
       else
          y:=y+s2;
          e:=e-2*dx;
       end;
   if ch=1 then y:=y+s2
   else
      x:=x+s1;
      e:=e+2*dy;
   end;
 end;

procedure BrezenchemCircle;{--процедура побудови кола за алгоритмом
Брезенхема}
var x,y,d,g,r,l:integer;
```

```
begin
r:=15;
x:=0;
y:=r;
d:=2*(1-r);
l:=trunc(r/sqrt(2));
repeat
putp(x,y);
putp(-x,y);
putp(x,-y);
putp(-x,-y);
putp(y,x);
putp(-y,x);
putp(y,-x);
putp(-y,-x);
if d=0 then
 begin
 x:=x+1;
 y:=y-1;
 d:=d+2*x-2*y+2;
 end
 else if d<0 then
 begin
 g;=2*d+2*y-1;
 if g<=0 then
  begin
  x:=x+1;
  d:=d+2*x+1;
  end;
 end
 else
 begin
 g:=2*d+2*x-1;
 if g<=0 then
  begin
  x:=x+1;
  y:=y-1;
```

```
   d:=d+2*x-2*y+2;
   end
   else
   begin
   y:=y-1;
   d:=d-2*y+1;
   end;
 end;
until y<=1;
end;

procedure VidrCircle(sg:integer);{-- процедура побудови кола по відрізкам вписаного багатокутника}
var r,x1,y1,x2,y2,i:integer;
phi:real;
begin
r:=15;
phi:=PI*2/sg;
for i:=1 to sg do
begin
x1:=round(r*sin(phi*(i-1)));
y1:=round(r*cos(phi*(i-1)));
x2:=round(r*sin(phi*i));
y2:=round(r*cos(phi*i));
BrezenchemPlot(x1,y1,x2,y2);
end;
end;

begin
 grDriver:=Detect;    {--ініціалізація графічного режиму}
 InitGraph(grDriver,grMode,'');  { ..}
 ErrCode:=GraphResult;             { ..}
 if ErrCode <>grOk then Writeln('Graphics error:',GraphErrorMsg(ErrCode));
  VidrCircle(128);      {--побудова кола за відрізками (4,8,16,32,64,128)}
```

```
 BrezenchemCircle;    {-- побудова кола за алгоритмом Брезенхема}
 Readln;
 CloseGraph;
end.
```

7.6. Буфер кадрів

Растровий дисплей реалізується у вигляді буфера кадрів, що складається з напівпровідникової пам'яті з довільним доступом. Хоча це метод реалізації, що найбільш часто зустрічається, однак для буфера кадрів може бути використана і вторинна пам'ять − диск або барабан.

Буфери кадрів можна також реалізувати за допомогою зсувних регістрів. Схематично зсувний регістр можна вважати стеком типу FIFO (першим прийшов − першим обслужений). Якщо стек заповнений, то при додаванні в вершину стека нових бітів даних з дна виштовхуються перші біти даних. Дані, що виштовхуються зі стека, можна інтерпретувати як інтенсивність пікселя скануючого рядка. Буфери кадрів на зсувних регістрах можна реалізувати, використовуючи по одному регістру на піксель в скануючому рядку при довжині кожного регістру, що рівний кількості рядків. Інший варіант − використання єдиного регістра з довжиною, що рівна кількості пікселів в скануючому рядку помноженому на число рядків.

На рис. 7.15 наведений найпростіший шестистроковий дисплей по вісім пікселів на рядок. На рисунку наведено також буфер кадрів на зсувних регістрах, до якого входять вісім регістрів по 6 бітів кожний. У буфері знаходиться набір бітів зображення. Біти для рядка 3 показані в момент виштовхування з дна зсувних регістрів. Для узгодженості зі швидкістю відеогенерації потрібно ретельно керувати послідовністю виведення зсувних регістрів.

Для буферів кадрів на вторинній пам'яті і на зсувних регістрах рівень інтерактивності невисокий. Для вторинної пам'яті причина полягає у великому часі доступу, а для зсувних регістрів зниження ефективності інтерактивної роботи пов'язане з тим, що зміни можуть бути зроблені лише при додаванні бітів в регістр.

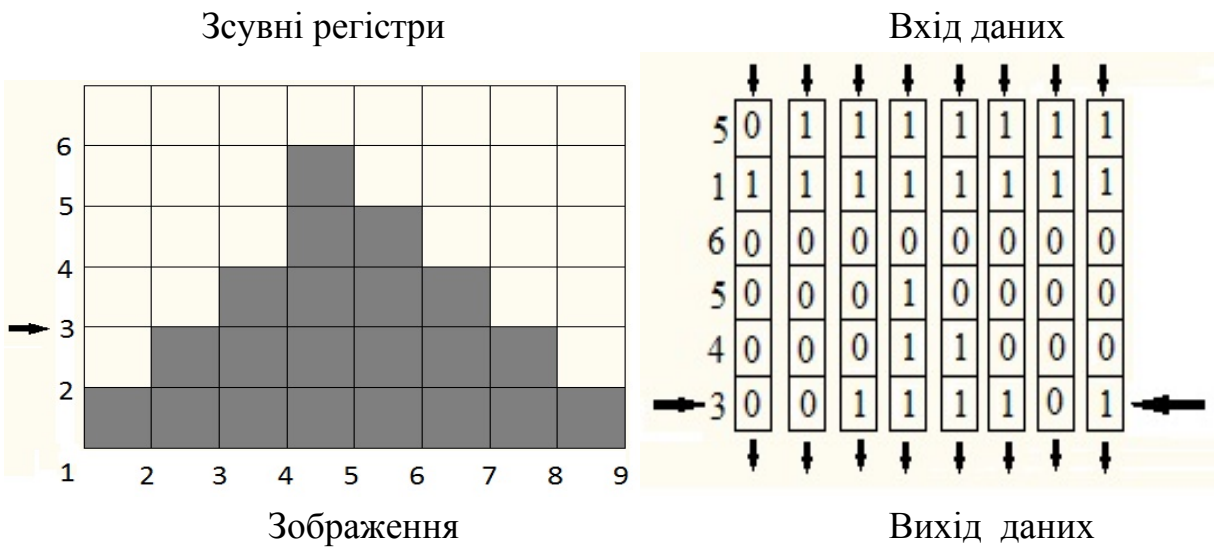


Рис. 7.15. Буфер кадрів на зсувних регістрах

Як наведено на рис. 7.16, схема графічної системи з буфером кадрів схожа на схему для векторного дисплея з регенерацією. При необхідності прикладна програма на головному комп'ютері модифікує буфер кадрів. Дисплейний контролер періодично обробляє його в порядку сканування рядків і передає відеомонітору інформацію, що необхідна для регенерації зображення. Буфер кадрів можна реалізувати або як частину пам'яті головного комп'ютера, або як окрему пам’ять. На рис. 7.17 показані дві ці схеми, що реалізовані зі структурою загальної шини.

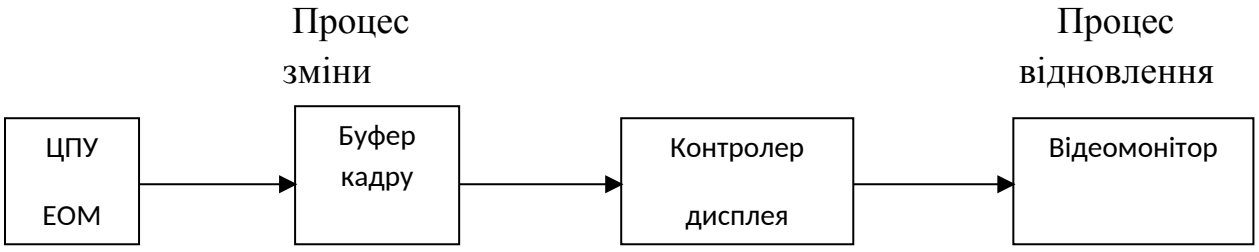


Рис. 7.16. Графічна система з буфером кадру

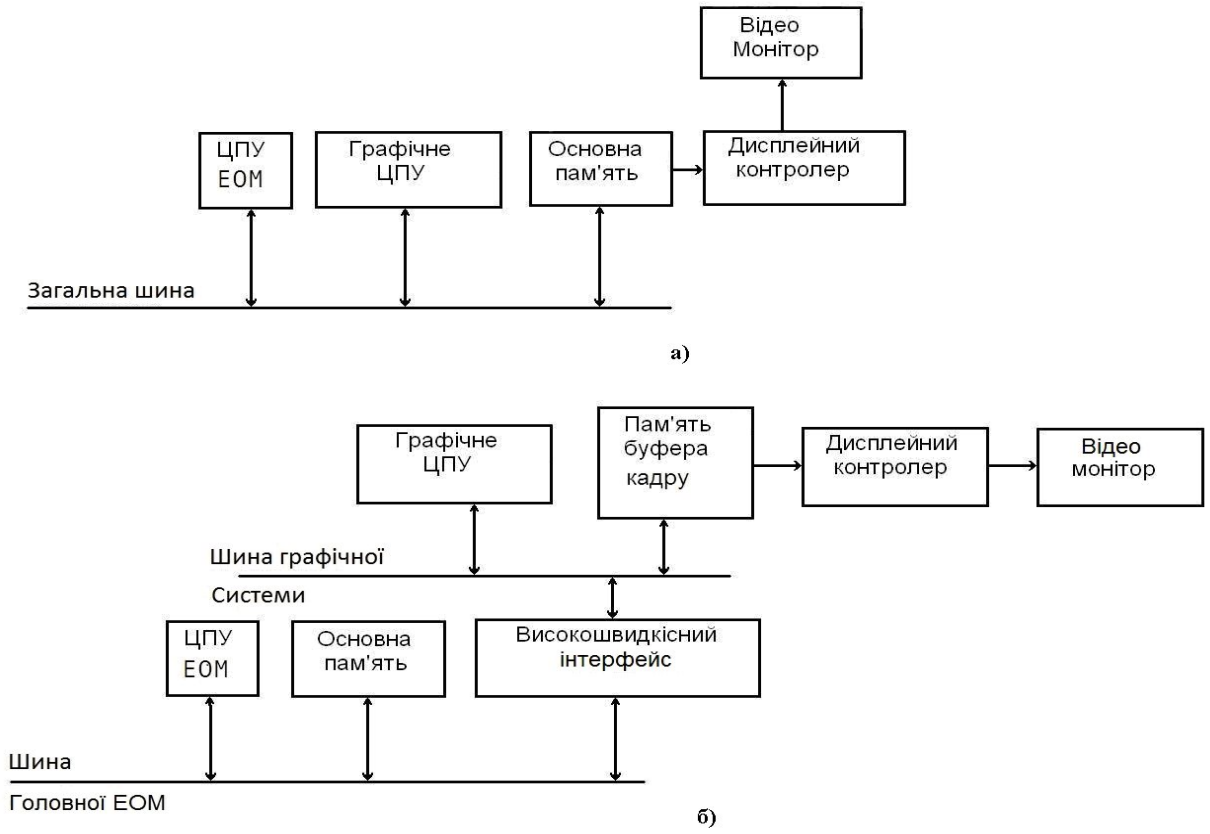


Рис. 7.17. Архітектура графічних систем с буфером кадру

7.7. Зображення відрізків

Подібна адресація буфера кадру дозволяє поводитися з ним як з графічним дисплеєм на запам'ятовуючій трубці. Спочатку буфер кадру очищується або встановлюється в фонову інтенсивність або колір. Замість того щоб записувати вектори прямо на екран дисплея, для розкладання в растр відрізка застосовується або алгоритм Брезенхема, або ЦДА, і відповідні пікселі записують в буфер кадру. Коли зображення або кадр побудовані, контролер дисплея читає буфер кадру в порядку сканування рядків і виводить результат на відеомонітор.

Вибіркове стирання відрізків можна реалізувати за допомогою повторного використання алгоритму розкладу в растр і запису відповідних пікселів з фоновою інтенсивністю або кольором. Проблема, яка виникає при використанні цього методу, наведена на рис. 7.18. Якщо відрізок, який видаляється, перетинає інший відрізок, то в останньому з'явиться розрив. На рис. 7.18, а показані два відрізка, що перетинаються. Якщо горизонтальний відрізок y = 5 стирається за допомогою запису пікселів з фоновою інтенсивністю або кольором в буфер кадру, то в результаті в іншого відрізка

з'явиться розрив у пікселі (5, 5). Виявити і заповнити розриви не складає труднощів, потрібно лише визначити перетин видаляючого відрізка з усіма іншими відрізками у зображенні. Дана операція для складного зображення може зайняти багато часу.

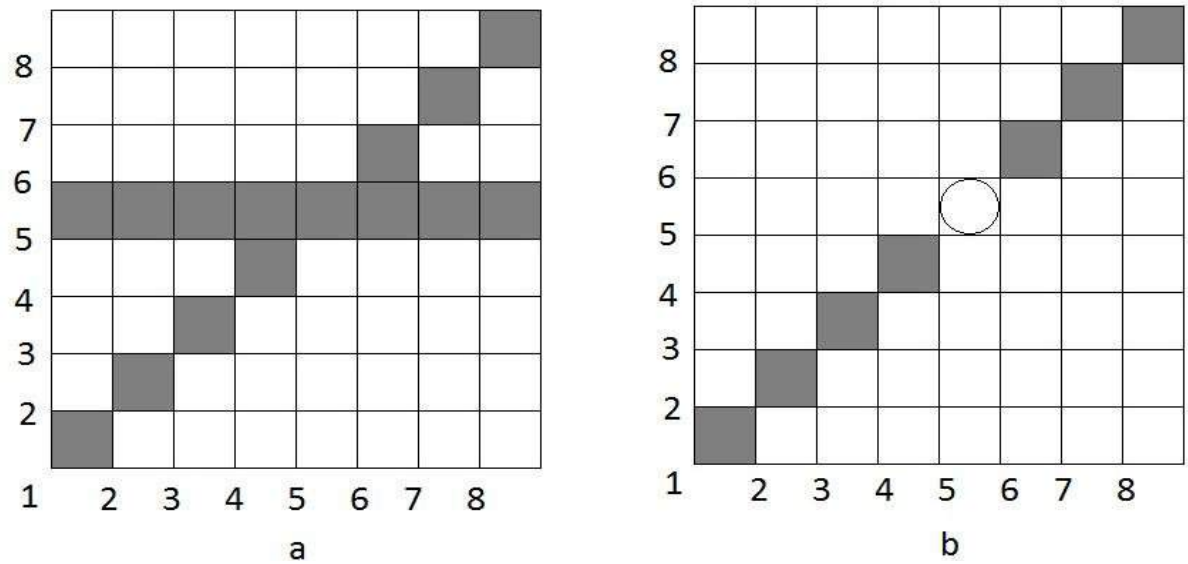


Рис. 7.18. Вибіркове стирання відрізків в буфері кадру

Для зменшення витрат можна використовувати оболонковий або мінімаксний текст. Цей метод наведено на рис. 7.19. Відрізок ab можуть перетинати лише ті відрізки, які проходять через намальовану пунктиром прямокутну оболонку, що сформована з мінімальних і максимальних значень координат *x*, *y* відрізка *ab*.

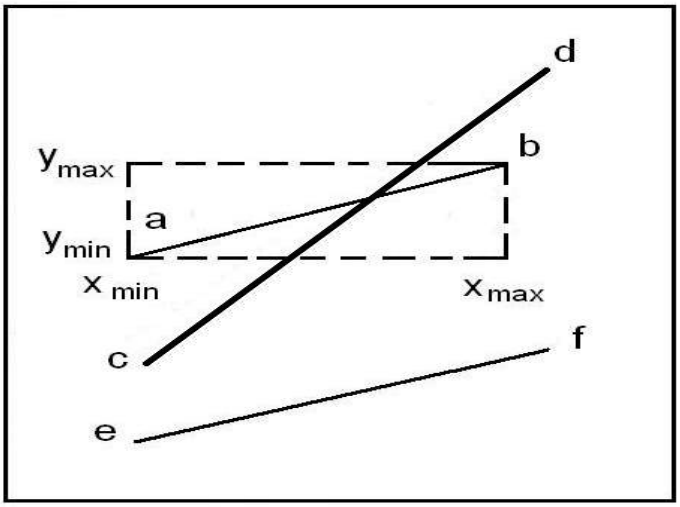


Рис. 7.19. Оболонковий або мінімаксний текст

Тести для наступного відрізка виглядають наступним чином:

Мінімаксний або оболонковий текст
If (Хотрmax < Хоболmin) **or**
(Хотрmax > Хоболmin) **or**
(Yотрmax < Yоболmin) **or**
(Yотрmax < Yоболmin)
Then
перетину немає
Else
обчислити перетин
Finish.

7.8. Зображення літер

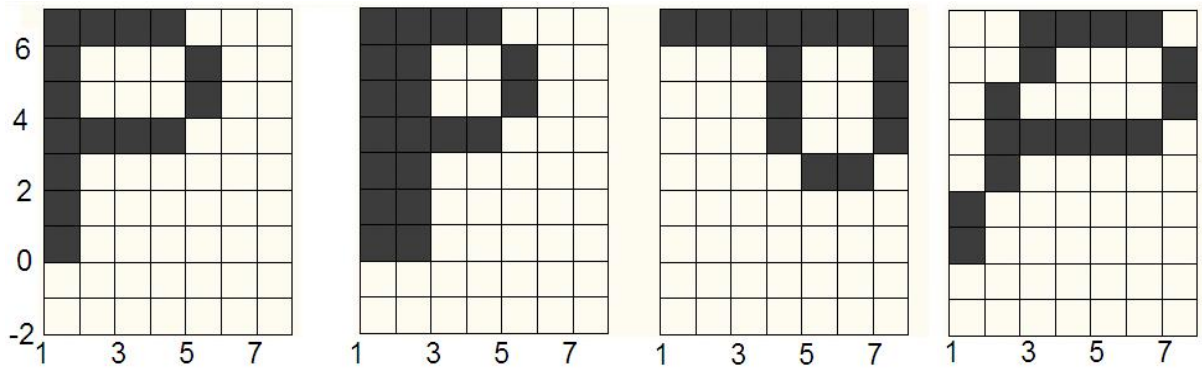
Алфавітно-цифрові символи (літери) записуються в буфер кадру за допомогою маски. Літерна маска − це маленький растр, який містить відносні адреси пікселів, що використовуються для представлення літери. За допомогою літерної маски також можна представити і спеціальні символи, що специфічні в конкретній прикладній галузі, наприклад, резистори, конденсатори або математичні символи. Сама маска просто містить двійкові величини, що позначають, використовується чи ні конкретний піксель в масці для подання форми літери або символу. Для простих чорно-білих зображень 1 зазвичай означає, що піксель використовується в представленні, а 0 − не використовується. Для кольорових зображень застосовуються додаткові біти в якості індексів у таблиці кольорів.

Літеру можна вставити в буфер кадру, вказавши адресу (x_0, y_0) початку маски в буфері. Кожен піксель в масці зміщується на величину x_0, y_0. Простий алгоритм для бінарної маски наводиться нижче.

Вставка маски *в буфер кадру*
Xmin, Xmax, Ymin, Ymax – *границі маски*
x_0, y_0 – *адреса в буфері кадру*
for j = Ymin **to** Ymax − 1
for I = Xmin **to** Xmax – 1
If маска(i,j) < > 0 **then**
записати Маска(i,j) в буфер кадру в (x_0 + i, y_0 + j)
else
end if
next i

next j
finish.

Для створення літер різних шрифтів або орієнтації перед записом в буфер, маску можна модифікувати. Деякі з таких простих модифікацій наведені на рис. 7.20. На рис. 7.20, а зображена вихідна літерна маска. Записуючи її в дві послідовні комірки x_0 та x_0+1, отримаємо жирну літеру (рис. 7.20, б). Літеру можна повернути (рис. 7.20, с) або нахилити, в результаті останньої операції отримаємо курсив (рис. 7.20, д).



а) б) с) д)

Рис. 7.20. Перетворені маски букв

7.9. Растрова розгортка суцільних областей

Генерація суцільних областей з простих описів ребер або вершин називається растровою розгорткою суцільних областей, заповненням багатокутників або ж заповненням контурів.

Для цього можна використовувати кілька методів, які діляться на дві категорії: растрова розгортка та растрове заповнення.

В методах растрової розгортки намагаються визначити в порядку сканування рядків, чи лежить точка всередині багатокутника чи контуру. Ці алгоритми зазвичай йдуть з «верху» багатокутника або контуру до «низу». Методи розгортки також можна застосувати і до векторних дисплеїв, у яких вони використовуються для штрихування або зафарбовування контурів, як наведено на рис. 7.21.

У методах затравного заповнення передбачається, що відома деяка точка (затравка) всередині замкненого контуру. У алгоритмах шукають точки, які сусідні з затравною і розташовані всередині контуру. Якщо сусідня точка розташована не в середині, то означає, що виявлена межа контуру. Якщо ж точка виявилася всередині контуру, то вона стає новою затравною точкою і

пошук продовжується рекурсивно. Подібні алгоритми застосовні лише до растрових пристроїв.

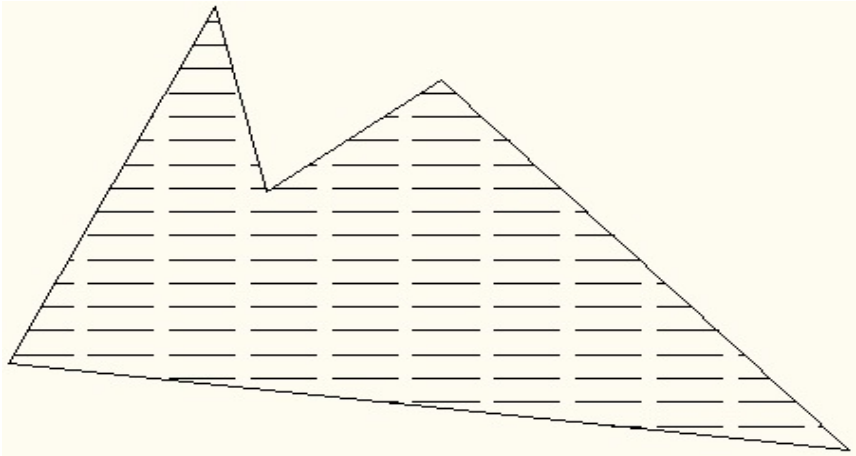


Рис. 7.21. Штрихування або зафарбовування контуру

7.10. Заповнення багатокутників

Багато замкнених контурів є простими багатокутниками. Якщо контур складається з кривих ліній, то його можна апроксимувати багатокутником або багатокутниками. Найпростіший метод заповнення багатокутника полягає в перевірці на приналежність вмісту багатокутника кожного пікселя в растрі. Оскільки зазвичай більшість пікселів лежить поза багатокутником, то даний метод дуже трудомісткий. Затрати можна зменшити шляхом обчислення для багатокутника прямокутної оболонки − найменшого прямокутника, що містить всередині себе багатокутник. Як наведено на рис. 7.22, перевіряються лише внутрішні точки цієї оболонки. Використання прямокутної оболонки для багатокутника, що наведений на рис. 7.22,a, набагато скорочує кількість пікселів, що перевіряються. В той же час для багатокутника, наведеного на рис. 7.22,b, скорочення істотно менші.

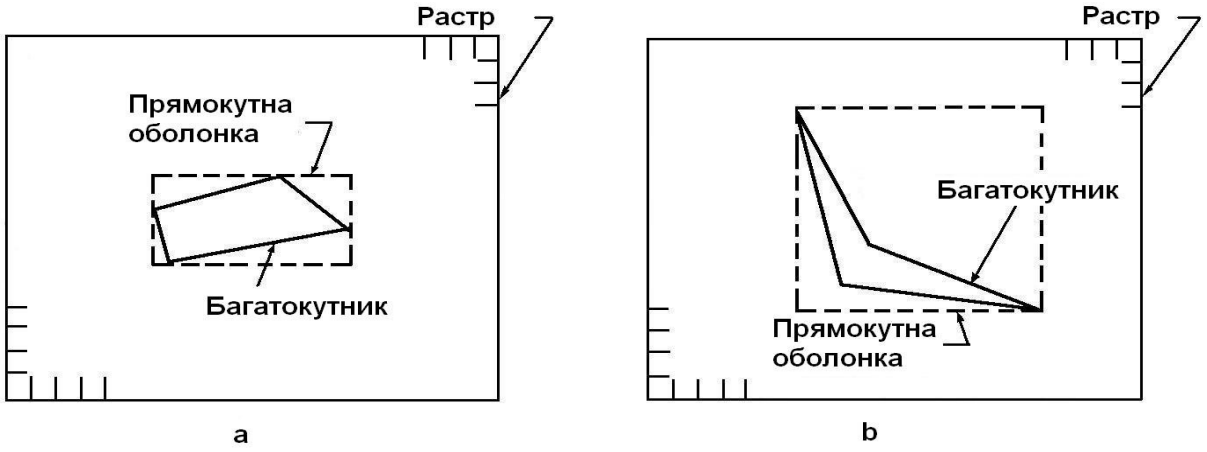


Рис. 7.22. Прямокутна оболонка

7.11. Растрова розгортка багатокутників

Можна розробити більш ефективний метод, ніж тест на належність внутрішньої частини, якщо скористатися тим фактом, що сусідні пікселі, ймовірно, мають однакові характеристики (крім пікселів граничних ребер). Ця властивість називається просторовою когерентністю. Для растрових графічних пристроїв сусідні пікселі на скануючому рядку, ймовірно, мають однакові характеристики. Це когерентність растрових рядків.

Характеристики пікселів на цьому рядку змінюються лише там, де ребро багатокутника перетинає рядок. Ці перетини ділять скануючий рядок на області.

Для простого багатокутника на рис. 7.23 рядок 2 перетинає багатокутник при x = 1 і x = 8. Отримуємо три області:

$x < 1$ поза багатокутником

$1 \le x \le 8$ всередині багатокутника

$x > 8$ поза багатокутником

Рядок 4 ділиться на 5 областей:

$x < 1$ поза багатокутника

$1 \le x \le 4$ всередині багатокутника

$4 < x < 6$ поза багатокутника

$6 \le x \le 8$ всередині багатокутника

$x > 8$ поза багатокутника

Зовсім необов'язково, щоб точки перетину для рядка 4 відразу визначалися у фіксованому порядку (зліва направо). Наприклад, якщо багатокутник

задається списком вершин $P_1P_2P_3P_4P_5$, а список ребер - послідовними парами вершин $P_1P_2, P_2P_3, P_3P_4, P_4P_5, P_5P_1,$ то для рядка 4 будуть знайдені наступні точки перетину з ребрами багатокутника: 8, 6, 4, 1. Ці точки потрібно відсортувати у зростаючому порядку по x, тобто отримати 1, 4, 6, 8.

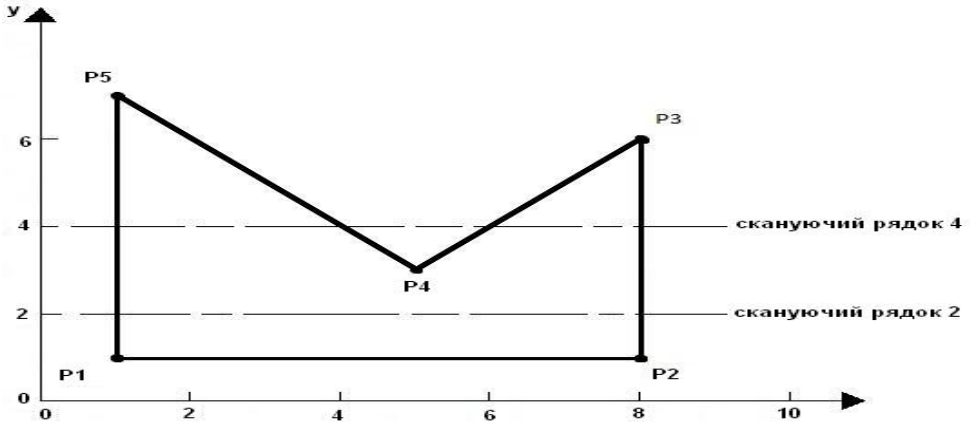


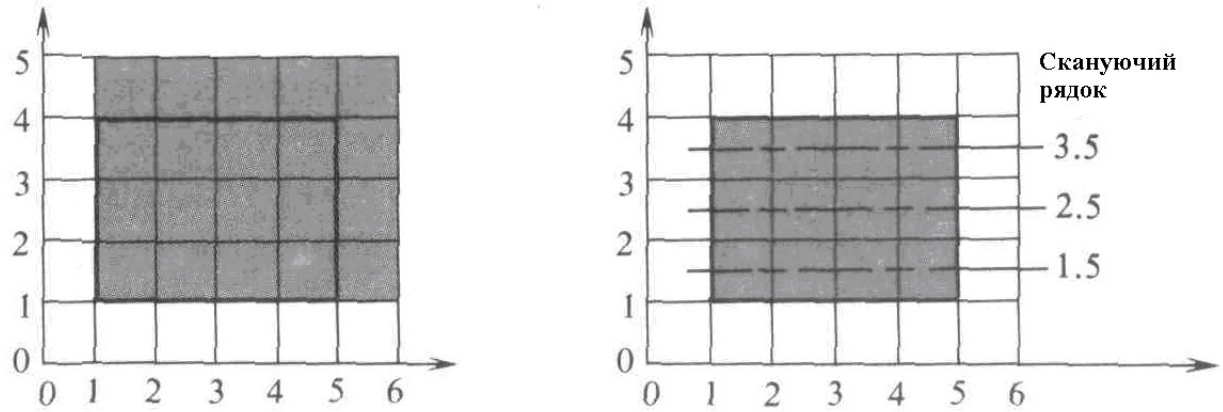
Рис. 7.23. Растрова розгортка суцільної області

При визначенні інтенсивності, кольору і відтінку пікселів на скануючому рядку розглядаються пари відсортованих точкових перетинів. Для кожного інтервалу, що задається парою пересічень, використовується інтенсивність або колір багатогранника, що заповнюється. Для інтервалів між парами перетинів і крайніх (від початку рядка до першої точки перетину і від останньої точки перетину до кінця рядка) використовується фонова інтенсивність або колір. На рис. 7.23 для рядка 4 у фоновий колір встановлені пікселі: від 0 до 1, від 4 до 6, від 8 до 10, тоді як пікселі від 1 до 4 та від 5 до 8 зафарбовані в колір багатокутника.

Точне визначення пікселів, які повинні активуватися, вимагає деякої обережності. Розглянемо простий прямокутник, наведений на рис. 7.24. Прямокутник має координати (1,1) (5,1), (5,4), (1,4). Скануючі рядки з 1 по 4 мають перетин з ребрами багатокутника при x=1 і x=5. Оскільки піксель адресується координатами свого лівого нижнього кута, то для кожного з цих скануючих рядків будуть активовані пікселі з координатами 1, 2, 3, 4 і 5. На рис. 7.24,а наведений результат. Зауважимо, що площа, яка покривається активованими пікселями, дорівнює 20, в той час як справжня площа прямокутника дорівнює 12.

Модифікація системи координат скануючого рядка і тесту активації усуває цю проблему, як це наведено на рис. 7.24,б. Вважається, що скануючі рядки проходять через центр рядків пікселів, тобто через середину інтервалу, як це наведено на рис. 7.24,б. Тест активації модифікується наступним чином: перевіряється, чи лежить всередині інтервалу центр пікселя, що розташований праворуч від перетину. Однак пікселі все ще адресуються координатами лівого

нижнього кута. Як наведено на рис. 7.24,б, результат даного методу є коректним.



а) б)

Рис. 7.24. Системи координат рядків сканування

Горизонтальні ребра не можуть перетинати скануючий рядок і, таким чином, ігноруються. Це зовсім не означає, що їх немає на рисунку. Ці ребра формуються верхніми і нижніми рядками пікселів, як наведено на рис. 7.24. Він ілюструє коректність верхнього та нижнього ребер багатокутника, що отримані в результаті модифікації системи координат скануючих рядків.

Додаткова складність виникає при перетині скануючого рядка і багатокутника точно по вершині, як це наведено на рис. 7.25. При використанні погодження про середину інтервала між скануючими рядками отримуємо, що рядок y=3.5 перетне багатокутник в 2, 2 і 8, тобто вийде непарна кількість перетинів. Отже, розбивання пікселів на пари дасть невірний результат, тобто пікселі (0, 3), (1, 3) і від (3, 3) до (7, 3) будуть фоновими, а пікселі (2, 3), (8, 3), (9, 3) зафарбуються в колір багатокутника. Тому виникає ідея враховувати лише одну точку перетину з вершиною. Тоді для рядка y=3.5 отримаємо правильний результат. Проте результат застосування методу до рядка y=1.5, що має два перетини в (5, 1), показує, що метод невірний. Для цього рядка саме розбиття на пари дасть вірний результат, тобто зафарбований буде лише піксель (5, 1). Якщо ж враховувати у вершині лише одне перетинання, то пікселі від (0, 1) до (4, 1) будуть фоновими, а пікселі від (5, 1) до (9, 1) будуть зафарбовані в колір багатокутника.

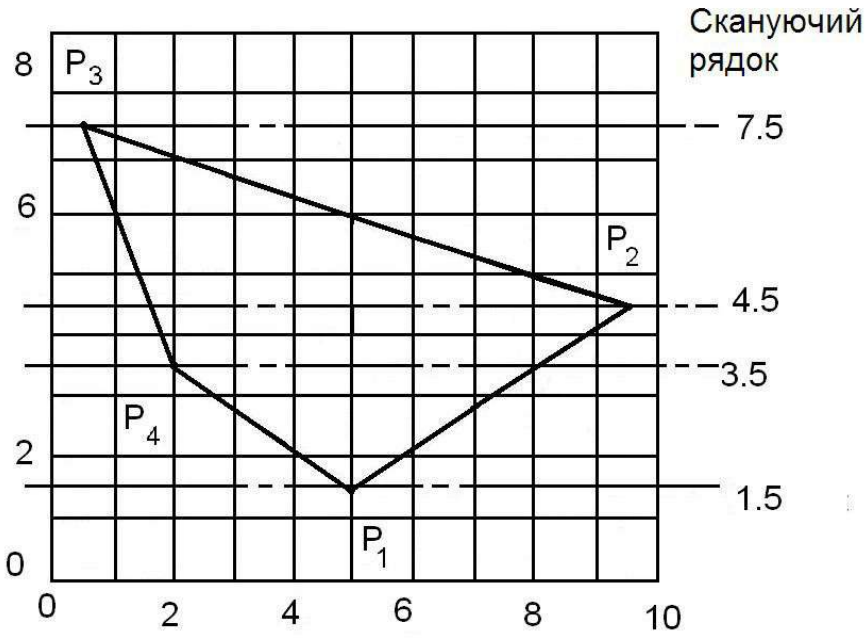


Рис. 7.25. Особливості перетину з рядками сканування

Правильний результат можна отримати, враховуючи точку перетину у вершині два рази, якщо вона є точкою локального мінімуму або максимуму і враховувати її лише один раз в протилежному випадку. Визначити локальний максимум або мінімум багатокутника у розглянутій вершині можна за допомогою перевірки кінцевих точок двох ребер, що з'єднані у вершині. Якщо у обох кінців координати *y* більше, ніж у вершини, то вершина є точкою локального мінімуму. Якщо менше, то вершина − точка локального максимуму. Якщо одна більша, а інша менша, то, вершина не є ні точкою локального мінімуму, ні точкою локального максимуму. На рис. 7.25 точка P_1 − локальний мінімум, P_3 − локальний максимум, а P_2 і P_4 − ні те й ні інше. Отже, в точках P_1 і P_3 враховуються два перетини з скануючими рядками, а в P_2 і P_4 − один.

Хоча перша схема дозволяє процесору ЕОМ самому маніпулювати буфером кадру (рис. 7.17, а), зазвичай, більш ефективно додати до системи спеціалізований графічний процесор. При отриманні команд від головного процесора графічний процесор керує детальною обробкою буфера кадру. При двох процесорах на загальній шині та однієї пам'яті на шині може відбутися конфліктна ситуація, що скорочує середню продуктивність системи. Таким чином, для високопродуктивних систем більш кращою є архітектура, наведена на рис. 7.17, б. У цьому випадку пам'ять буфера кадру відокремлена від основної, що виключає конфлікт на шині. Більш того, графічну підсистему можна оптимізувати для покращення характеристик модифікації буфера кадрів і, отже, для збільшення продуктивності системи.

7.12. Простий алгоритм з впорядкованим списком ребер

Використовуючи описані вище методи, можна розробити ефективні алгоритми растрової розгортки суцільних областей, які називаються алгоритмами з впорядкованим списком ребер. Вони залежать від сортування у порядку сканування точок перетину ребер багатокутника зі скануючими рядками. Ефективність цих алгоритмів залежить від ефективності сортування. Наведемо дуже простий алгоритм.

Простий алгоритм з впорядкованим списком ребер

Підготувати дані.

Визначити для кожного ребра багатокутника точки петинів зі скануючими рядками, що проведені через середини інтервалів, для чого можна використовувати алгоритм Брезенхема або ЦДА. Горизонтальні ребра ігноруються. Занести кожен перетин $(x, y + \frac{1}{2})$ у список.

Відсортувати список по рядках і по зростанню x у рядку; тобто (x_1, y_1) передує (x_2, y_2), якщо $y_1 > y_2$ або $y_1 = y_2$ та $x_1 \le x_2$.

Перетворити ці дані в растрову форму:

Виділити з відсортованого списку пари елементів (x_1, y_1) і (x_2, y_2). Структура списку гарантує, що $y = y_1 = y_2$ і $x_1 \le x_2$. Активувати на скануючому рядку y пікселі для цілих значень x таких, що

$$x_1 \le x + \frac{1}{2} \le x_2.$$

Приклад 7.8. Простий упорядкований список ребер

Розглянемо багатокутник, наведений на рис. 7.23. Його вершини $P_1(1,1)$, $P_2(8,1)$, $P_3(8,6)$, $P_4(5,3)$ та $P_5(1,7)$. Перетини з серединами скануючих рядків наступні:

скан. рядок 1.5: (8, 1.5), (1, 1.5)

скан. рядок 2.5: (8, 2.5), (1, 2.5)

скан. рядок 3.5: (8, 3.5), (5.5, 3.5), (4.5, 3.5), (1, 3.5)

скан. рядок 4.5: (8, 4.5), (6.5, 4.5), (3.5, 4.5), (1, 4.5)

скан. рядок 5.5: (8, 5.5), (7.5, 5.5), (2.5, 5.5), (1, 5.5)

скан. рядок 6.5: (1.5, 6.5), (1, 6.5)

скан. рядок 7.5: немає

Весь список сортується в поряду сканування спочатку зверху вниз і тоді – зліва направо

(1, 6.5), (1.5, 6.5), (1, 5.5), (2.5, 5.5), (7.5, 5.5), (8, 5.5), (1, 4.5), (3.5, 4.5), (6.5 , 4.5), (8, 4.5), (1, 3.5), (4.5, 3.5), (5.5, 3.5), (8, 3.5), (1,2.5), (8, 2.5), (1,1.5), (8, 1.5)

Виділяючи зі списку пари перетинів і застосовуючи алгоритм, що описаний вище, отримаємо список пікселів, які повинні бути активовані:

(1.6)

(1,5), (2, 5), (7, 5)

(1,4), (2, 4), (3,4), (6, 4), (7, 4)

(1, 3), (2, 3), (3, 3), (4, 3), (5, 3), (6, 3), (7, 3)

(1, 2), (2, 2), (3, 2) (4, 2), (5, 2), (6, 2), (7, 2) (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1)

Результат наведений на рис. 7.26. Зауважимо, що обидва вертикальних і нижнє ребра зображені вірно.

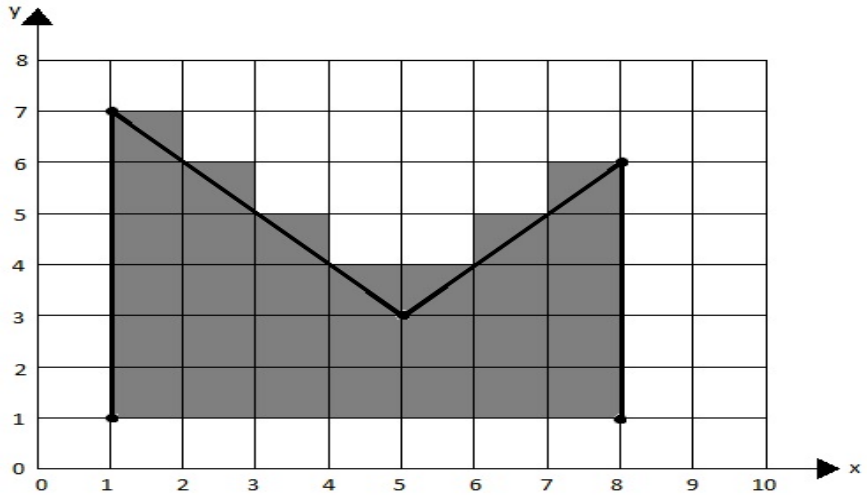


Рис. 7.26. Результат растрової розгортки суцільної області, що наведена на рис. 7.33

7.13. Алгоритм зі списком ребер і прапорцем

Алгоритм, який використовує список ребер і прапорець, є двокроковим. Перший крок полягає в обрисовуванні контура, в результаті чого на кожному скануючому рядку утворюються пари обмежуючих пікселів. Другий крок полягає у заповненні пікселів, що розташовані між обмежуючими. Більш точно алгоритм можна сформулювати в наступному вигляді:

Обрисовування контуру.

Використовуючи погодження про середину інтервалу між скануючими рядками для кожного ребра, що перетинає скануючий рядок, відмітити самий

лівий піксель, центр якого лежить праворуч від перетину;

тобто

$$x \le x+1/2 \le x_{\text{перетину}}$$

Заповнення.

Для кожного скануючого рядка, що перетинає багатокутник

Всередині = FLASH

For x = 0 (ліва межа) **to** $x=x_{max}$ (права межа)

if піксель в точці x має граничне значення

then інвертувати значення змінної Всередині

if Всередині = TRUE **then**

присвоїти пікселю в *x* значення кольору багатокутника

else

присвоїти пікселю в x значення кольору фону

end if

next x.

Приклад 7.9. Алгоритм, що використовує список ребер і прапорець

Розглянемо застосування даного алгоритму до тестового багатокутника наведеного на рис. 7.23. Спочатку вимальовується контур, результат цього кроку наведений на рис. 7.27,а. Активовані пікселі в точках (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (2, 6), (3 , 5), (4, 4), (5, 3), (6, 3), (7, 4), (8, 4), (8, 3), (8, 2), (8, 1).

Тоді багатокутник заповнюється. Для ілюстрації цього процесу виділений і показаний рядок 3 на рис. 7.27,б. Для обрисовування контуру на цьому рядку активовані пікселі при *x* = 1, 5, 6 і 8. Застосування алгоритму заповнення дає наступні результати:

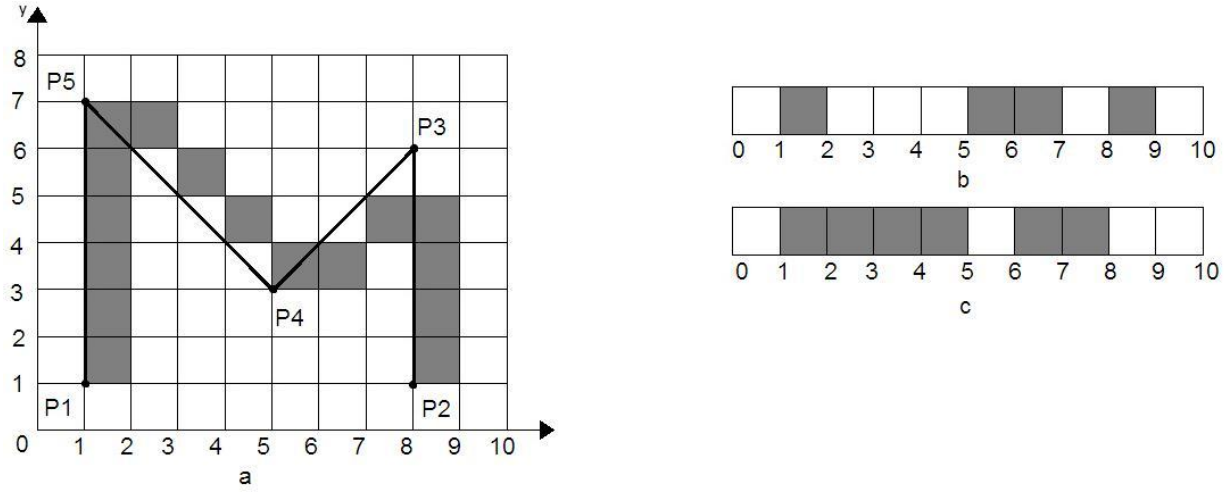


Рис. 7.27. Алгоритм заповнення по ребрах і прапорцеві

Спочатку

Всередині = FALSE

Для $x = 0$ Піксель – не граничний і Всередині = FALSE, отже, вживати ніяких дій не потрібно.

Для $x = 1$ Піксель - межовий, тому змінна Всередині інвертується і отримує значення TRUE. Змінна Всередині = TRUE, тому пікселю присвоюється колір або інтенсивність багатокутника.

Для $x = 2, 3, 4$ Піксель – не граничний і Всередині = TRUE, тому пікселю присвоюється колір багатокутника.

Для $x = 5$ Піксель - межовий, тому змінна Всередині інвертується. Всередині = FALSE, тому пікселю присвоюється фоновий колір.

Для $x = 6$ Піксель – межовий, тому змінна Всередині інвертується і отримує значення TRUE. Змінна Всередині = TRUE, тому пікселю присвоюється колір багатокутника.

Для $x = 7$ Піксель – не граничний і Всередині = TRUE, тому пікселю присвоюється колір багатокутника.

Для $x = 8$ Піксель – межовий, тому змінна Всередині інвертується і отримує значення FALSE. Мінлива всередині = FALSE, тому пікселю присвоюється колір фону.

Результат наведений на рис. 7.27,с, а кінцевий результат для всього багатокутника співпадає з результатом роботи алгоритму зі списком ребер.

У даному алгоритмі кожен піксель обробляється лише один раз, так що затрати на ввід/вивід значно менше, ніж в алгоритмі зі списком ребер або алгоритмі з перегородкою. Жоден з цих алгоритмів, якщо вони працюють з

буфером кадру, не вимагає побудови, підтримки та сортування будь-яких списків. При програмній реалізації алгоритм з впорядкованим списком ребер і алгоритм зі списком ребер і прапорцем працюють приблизно з однаковою швидкістю. Однак алгоритм зі списком ребер і прапорцем придатний для апаратної або мікропрограмної реалізації, в результаті чого він виконується на один-два порядки швидше, ніж алгоритм з впорядкованим списком ребер. Для простих зображень навіть можлива анімація в реальному режимі часу.

7.14. Алгоритм заповнення із затравкою

У наведених вище алгоритмах заповнення відбувається в порядку сканування. Інший підхід використовується в алгоритмах заповнення із затравкою. В них передбачається, що відомий хоча б один піксель з внутрішньої області багатокутника. Алгоритм намагається знайти і зафарбувати всі інші пікселі, що належать внутрішній області. Області можуть бути або внутрішньо-, або гранично-визначеними. Якщо область відноситься до внутрішньо-визначеної, то всі пікселі, що належать внутрішній частині, мають один і той же колір або інтенсивність, а всі пікселі, які зовнішні по відношенню до області, мають інший колір. Це наведено на рис. 7.28. Якщо область відноситься до гранично-визначеної, то всі пікселі на межі області мають виділене значення або колір, як це наведено на рис. 7.29. Жоден з пікселів з внутрішньої частини такої області не може мати це виділене значення. Тим не менше пікселі, які зовнішні у відношенні до межі, також можуть мати граничне значення. Алгоритми, що заповнюють внутрішньо-визначені області, називаються внутрішньо-заповнюючими, а алгоритми для гранично-визначених областей – гранично-заповнюючими. Далі будуть обговорюватися гранично-заповнюючі алгоритми, однак відповідні внутрішньо-заповнюючі алгоритми можна отримати аналогічним чином.

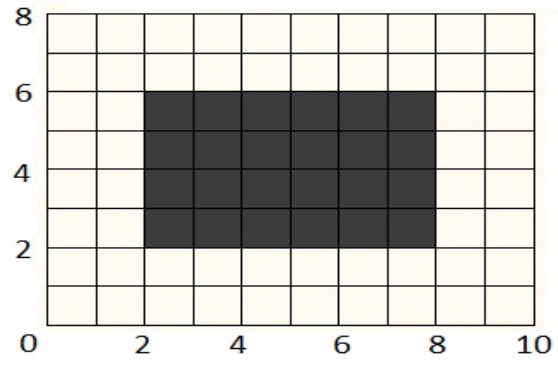


Рис. 7.28. Внутрішньо-визначена область

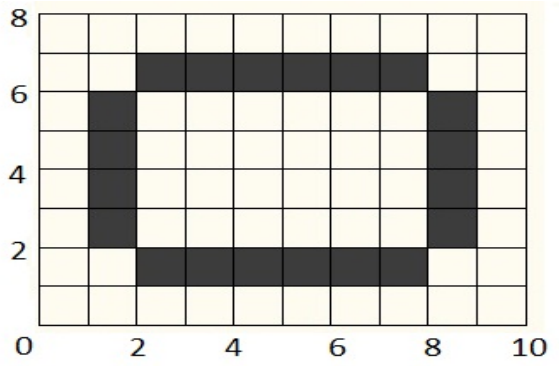


Рис. 7.29. Гранично-визначена область

Внутрішньо- або гранично-заповненні області можуть бути 4- або 8-зв'язними. Якщо область 4-зв'яна, то будь-якого пікселя в області можна досягти за допомогою комбінації рухів лише в 4 напрямках: наліво, направо, вгору, вниз. Для 8-зв'язної області пікселя можна досягти за допомогою комбінації рухів у двох горизонтальних, двох вертикальних і 4 діагональних напрямках.

Алгоритм заповнення 8-зв'язної області заповнить і 4-зв'язну область, проте зворотне невірно. На рис. 7.30 наведені прості приклади 4- і 8-зв'яних внутрішньо-визначених областей. Хоча кожна з підобластей 8-зв'язної області на рис. 7.30,b є 4-зв'язною, для переходу з однієї підобласті в іншу потрібен 8-зв'язний алгоритм. Однак у ситуації, коли потрібно заповнити різними кольорами дві окремі 4-зв'язні підобласті, використання 8-зв'язного алгоритму викличе неправильне заповнення обох областей одним і тим же кольором.

На рис. 7.31 наведена 8-зв'язна область з рис. 7.30, що перевизначена у вигляді гранично-визначеній області. На рис. 7.31 ілюструється той факт, що для 8-зв'язної області, у якої дві підобласті дотикаються кутами, межа 4-зв'язна, а межа 4-зв'язної області 8-зв'язна. Далі мова, в основному, піде про алгоритми для 4-зв'язних областей, проте їх можна легко переробити для 8-зв'язкних областей, якщо заповнення проводити не в 4, а в 8 напрямах.

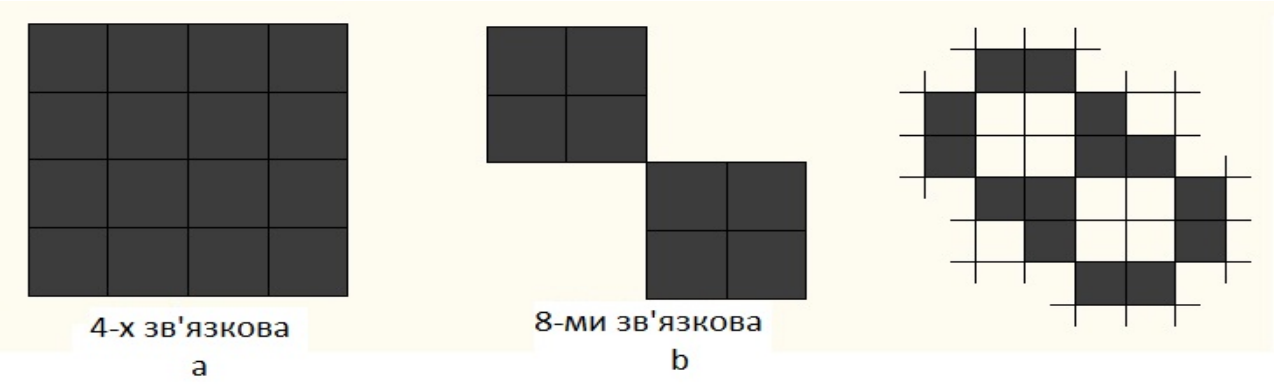


Рис. 7.30. 4 - і 8-зв'язні внутрішньо-визначені області

Рис. 7.31. - і 8-зв'язні гранично-визначені області

7.15. Простий алгоритм заповнення із затравкою

Використовуючи стек, можна розробити простий алгоритм заповнення гранично-визначеної області. Стек – це просто масив або інша структура даних, в яку можна послідовно поміщати значення і з якої їх можна послідовно витягувати. Коли нові значення додаються або поміщаються в стек, всі інші значення опускаються вниз на один рівень. Коли значення видаляються або витягуються із стека, інші значення спливають або піднімаються вгору на один рівень. Такий стек називається стеком прямої дії або стеком з дисципліною обслуговування «першим прийшов, останнім обслужений» (FIFO). Простий алгоритм заповнення із затравкою можна представити у наступному вигляді:

Простий алгоритм заповнення із затравкою і стеком

Помістити затравочний піксель в стек

Доки стек непорожній.

Витягти піксель зі стека

Присвоїти пикселю потрібне значення

Для кожного із сусідніх до поточного 4-зв'язого пікселя перевірити: чи є він граничним пікселем або чи не присвоєно вже пікселю потрібне значення. Проігнорувати піксель в будь-якому з цих двох випадків. В іншому випадку помістити піксель в стек.

Алгоритм можна модифікувати для 8-зв'язних областей, якщо переглядати 8-зв'язні пікселі, а не лише 4-зв'язні. Наведемо більш формальний виклад алгоритму, в якому припускається існування затравочного пікселя і

гранично-визначеної області

Простий алгоритм заповнення

Затравка (x, y) видає затравочний піксель

***Push** – процедура, яка поміщає піксель в стек*

***Pop** – процедура, яка витягує піксель зі стека*

Піксель (x, y) = затравка (x, y)

ініціалізуємо стек

Push Піксель (x, y)

while (стек не порожній)

витягаємо піксель зі стека

Pop Піксель (x, y)

if Піксель (x, y) <> Нов___ значення **then**

Піксель (x, y) = Нов___значення

end if

перевіримо, чи потрібно поміщати сусідні пікселі в стек

if (Піксель (x + 1, y) <> Нов___значення **and**

Піксель (x + 1, y) <> Гран___значення) **then**

Push Піксель (x + 1, y)

if (Піксель (x, y + 1) <> Нов___значення **and**

Піксель (x, y + 1) <> Гран___значення) **then**

Push Піксель (x, y + 1)

if (Піксель (x - 1, y) <> Нов___значення **and**

Піксель (x - 1, y) <> Гран___значення) **then**

Push Піксель (x - 1, y)

if Піксель (x, y – 1) <> Нов___значення **and**

Піксель (x , y- 1) <> Гран___значення) **then**

Push Піксель (x, y - 1)

end if

end while

В алгоритмі перевіряються і поміщаються в стек 4-зв'язні пікселі, починаючи з правого від поточного пікселя. Напрямок обходу пікселів – проти годинникової стрілки.

Приклад 7.10. Простий алгоритм заповнення із затравкою.

Як приклад застосування алгоритму розглянемо гранично-визначену полігональну область, задану вершинами (1, 0), (7, 0), (8, 1), (8, 4), (6, 6), (1, 6), (0, 5) і (0, 1). Область наведена на рис. 7.32. Затравний піксель – (4, 3). Область заповнюється піксель за піксельом в порядку, зазначеному на рис. 7.32 лінією зі стрілками. Числа, зображені на пікселі, позначають позицію пікселя в стеці, що займається при роботі алгоритму. Для деяких пікселів таких чисел декілька. Це означає, що піксель поміщали в стек більше одного разу. Коли алгоритм доходить до пікселя (5, 5), стек налічує 25 рівнів глибини і містить пікселі (7, 4), (7, 3), (7, 2), (7, 1), (6, 2), (6, 3), (5, 5), (6, 4), (5, 5), (4, 4), (3, 3), (3, 4), (3, 5), (2, 4), (2, 3), (2, 2), (2, 2), (3, 2), (5, 1), (3, 2), (5, 2), (3, 3) , (4, 4), (5, 3).

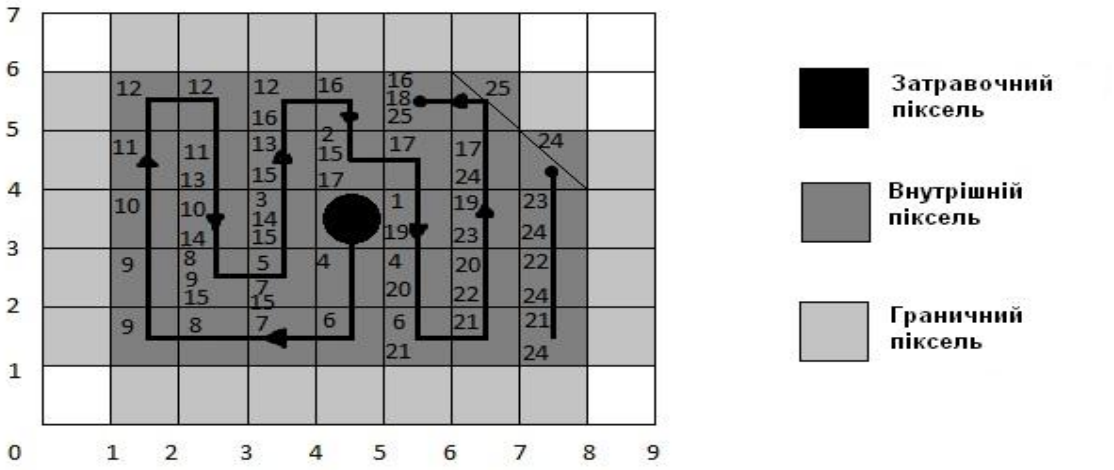


Рис. 7.32. Затравне заповнення за допомогою простого стекового алгоритму

Оскільки всі пікселі, що оточують (5, 5), містять або граничні, або нові значення кольору, то жоден з них в стек не поміщається. Отже, зі стека витягується піксель (7, 4) і алгоритм продовжує заповнювати колонку (7, 4), (7, 3), (7, 2), (7, 1). При досягненні пікселя (7, 1) перевірка знову показує, що суміжні пікселі або вже заповнені, або є граничними пікселями. Так як в даній момент багатокутник повністю заповнений, то витягання пікселів зі стека до повного його ощищення не викликає появи додаткових пікселів, які слід заповнити. Коли стек стає порожнім, алгоритм завершує роботу.

Багатокутник з прикладу 7.12 є однозв'язною областю, але алгоритм буде правильно заповнювати ті області, що містять порожнини. Це показано в наступному прикладі.

Приклад 7.11. Алгоритм заповнення із затравкою для багатокутника з порожниною.

Як приклад застосування алгоритму розглянемо гранично-визначену область, що містить порожнину. Вона наведена на рис. 7.33. Як і в попередньому прикладі, вершини багатокутника задані пікселями (1, 0), (7, 0), (8, 1), (8, 4), (6, 6), (1, 6), (0 , 5) і (0, 1). Внутрішня порожнина визначається пікселями (3, 2), (5, 2), (5, 3), (3, 3). Затравний піксель – (4, 4). Через порожнини багатокутник заповнюється не в тому порядку, як у прикладі 7.10. Новий порядок заповнення наведений на рис. 7.33 лініями зі стрілками. Як і в попередньому прикладі, числа у квадраті пікселя показують позицію, що займається піксельом в стеці. Коли обробка доходить до пікселя (3, 1), всі суміжні його 4-зв'язні пікселі або вже заповнені, або являються граничними. Тому ні один з пікселів не поміщається в стек. Глибина стека в цей момент дорівнює 15. У стеці знаходяться пікселі (7, 1), (7, 2), (7, 3), (6, 5), (7, 4), (6, 5), (3, 1), (1, 2), (1, 3), (1, 4), (2, 5), (3, 5), (4, 5), (5, 4).

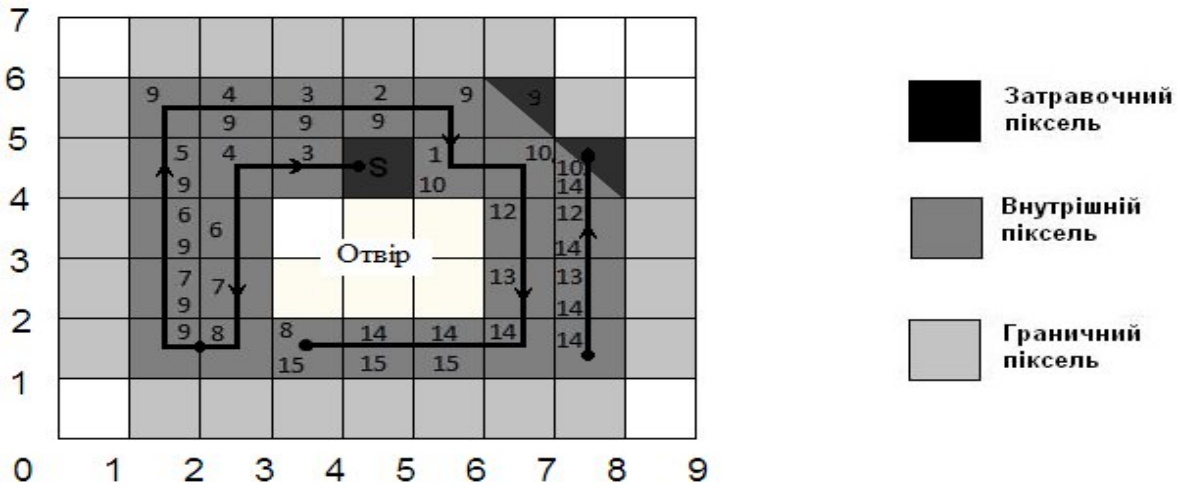


Рис. 7.33. Затравне заповнення області з порожниною за допомогою простого стекового алгоритму

Після видалення зі стека пікселя (7, 1) заповнюється колонка (7, 1), (7, 2), (7, 3), (7, 4), при цьому ні один новий піксель в стек не додається. Для пікселя (7, 4) знову всі 4-зв'язні суміжні пікселі або вже заповнені, або є граничними. Звертаючись до стека, алгоритм витягує піксель (6, 5), його заповнення завершує заповнення цього багатокутника. Подальша обробка відбувається без будь-якого затравлення, і коли стек стає порожнім, алгоритм завершує роботу.

7.16. Построковий алгоритм заповнення із затравкою

Як видно з обох прикладів, стек може стати досить великим. Ще один недолік попереднього алгоритму – стек найчастіше містить дублюючу або непотрібну інформацію. В порядковому алгоритмі заповнення із затравкою розмір стека мінімізується за рахунок зберігання лише одного затравочного пікселя для будь-якого безперервного інтервалу на скануючому рядку. Неперервний інтервал – це група приєднаних один до одного пікселів (обмежена вже заповненими або граничними пікселями). Для розробки алгоритму використовуємо евристичний підхід, проте також можливий і теоретичний підхід, що заснований на теорії графів.

Даний алгоритм застосуємо до гранично-визначених областей. Гранично-визначена 4-зв'язна область може бути як опуклою, так і не опуклою, а також може містити порожнини.

В області, зовнішньої і прилеглої до нашої гранично-визначеної області, не повинно бути пікселів з кольором, яким область або багатокутник заповнюється. Схематично роботу алгоритму можна розбити на чотири етапи.

Порядковий алгоритм заповнення із затравкою – затравний піксель на інтервалі витягується зі стека, що містять затравні пікселі.

Інтервал із затравним піксельом заповнюється вліво і вправо від затравки вздовж скануючого рядка до тих пір, поки не буде знайдена межа.

У змінних $X_{лів}$ і $X_{прав}$ запам'ятовуються крайній лівий і крайній правий піксель інтервалу.

У діапазоні $X_{лів} \le x \le X_{прав}$. перевіряються рядки, що розміщуються безпосередньо над і під поточним рядком. Визначається, чи є на них ще не заповнені пікселі. Якщо пікселі є (тобто, не всі пікселі граничні або вже занаповнені), то в зазначеному діапазоні крайній правий піксель в кожному інтервалі відзначається як затравний і поміщається в стек.

При ініціалізації алгоритму в стек поміщається єдиний затравний піксель, робота завершується при очищенні стека. Як наведено на рис. 7.34 і в прикладі нижче, алгоритм справляється з порожнинами та зубцями на межі. Нижче наводиться більш докладний опис алгоритму на псевдокоді.

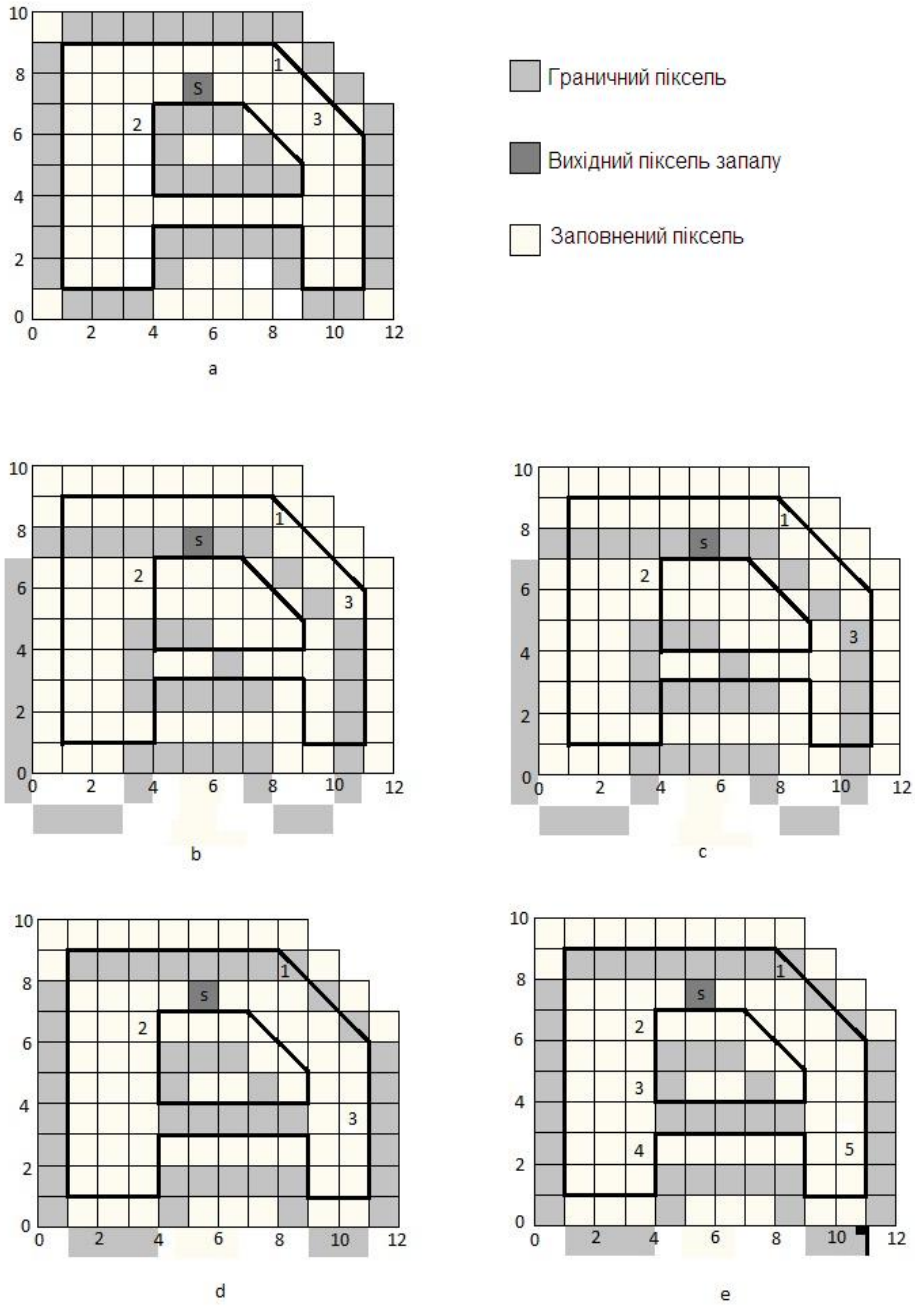


Рис. 7.34. Порядковий алгоритм заповнення із затравкою для багатокутника

Построковий алгоритм заповнення із затравкою

Затравка (x, y) видає затравний піксель

Pop – процедура, яка витягує піксель зі стека.

Push – процедура, яка поміщає піксель в стек

Ініціалізуєм стек

Push Затравка (x, y)

while (стек не порожній)

витягуємо піксель зі стека і присвоюємо йому нове значення

Pop Піксель (x, y)

Піксель (x, y) = Нов значення

зберігаємо x-координату затравного пікселя

Час___x = x

заповнюємо інтервал праворуч від затравки

X = X + 1

while Піксель (x, y) <> Гран___ значення

Піксель (x, y) = Нов___значення

x = x + 1

end while

зберігаємо крайній праворуч піксель

X-прав = x - 1

відновлюємо x-координату затравки

x = Час x

заповнюємо інтервал ліворуч від затравки

x = x - 1

While Піксель (x, y) <> Граничні значення

Піксель (x, y) = Нові значення

X = X - 1

end while

зберігаємо крайній ліворуч піксель

Хлів = x + 1

відновлюємо x-координату затравки

x = Час x

перевіримо, чи рядок вище не є ні межею багатокутника, ні вже повністю заповненим; якщо це не так, то знайти затравку, починаючи з лівого краю підінтервалу скануючого рядка

x = Хлів

y = y + 1

while x <Хправ

шукаємо затравку на рядку вище

Флажок = 0

while (Піксель (x, y) <> Гран___значення **and**

Піксель (x, y) <> Нов___значення **and** x <Хправ

if Флажок = 0 **then** Флажок = 1

X = X + 1

end while

поміщаємо в стек крайній праворуч піксель

if Флажок = 1 **then**

if (x = Хправ **and** Піксель (x, y) < > Гран_значення **and** Піксель (x, y) < > Нов_значення) **then**

Push Піксель (x, y)

else

Push Піксель (x - 1, y)

end if

Флажок = 0

end if

продовжимо перевірку, якщо інтервал був перерваний

Хвхід = x

while ((Піксель (x, y) = Гран_значення **or** Піксель (x, y) =

= Нов значення) **and** x <Хправ)

X = X + 1

end while

переконаємось, що координата пікселя збільшена

If x = Хвхід **then** x = x + 1

end while

перевіримо, чи рядок нижче не є ні межею багатокутника, ні вже повністю заповненою

ця частина алгоритму абсолютно аналогічна перевірці для рядка вище, за винятком того, що замість y = y + 1 *потрібно підставити* y = y - 1

end while

finish

Тут функція **Pop** витягує координати x,y пікселя зі стека, а функція **Push** поміщає їх в стек.

Приклад 7.12. Построковий алгоритм заповнення із затравкою.

Розглянемо роботу алгоритму для гранично-визначеної області наведеної на рис. 7.34. При ініціалізації в стек поміщається затравний піксель, позначений як затравка (5,7) на рис. 7.34,а. Спочатку в якості затравки інтервалу зі стека витягується цей піксель. Інтервал заповнюється праворуч і ліворуч від затравки. Знайдені при цьому кінці інтервал Хправ = 9 та Хлів = 1.

Тоді перевіряється строка, що розміщена вище поточної і виявляється, що вона не гранична і не заповнена. Крайнім правим пікселем в діапазоні $1 \le x \le 9$ виявляється піксель (8,8), позначений цифрою 1 на рис. 7.34,а. Цей піксель потрапляє у стек. Тоді аналогічним чином обробляється рядок нижче поточного. У діапазоні Хправ є два підінтервали на цьому рядку. Для лівого інтервалу в якості затравки виступає піксель (3,6), що позначений цифрою 2 на рис. 7.34,а, він теж поміщається в стек. Затравка для правого підінтервала – піксель (9,6), він потрапляє у стек. Зауважимо, що піксель (9,6) – це не самий крайній правий піксель на інтервалі, однак це самий крайній правий піксель в діапазоні $Хлів \le x \le Хправ$, тобто, в діапазоні $1 \le x \le 9$. На цьому завершується перший прохід алгоритму.

Далі зі стека витягується верхній піксель. Тут заповнюються інтервали, розміщені в правій частині багатокутника на послідовних скануючих рядках. Для рядка 3 затравкою служить піксель (10, 3) (рис. 7.34, d). У результаті заповнення інтервалу праворуч і ліворуч від затравки отримуємо нові межі діапазону: Хлів = 1 та Хправ = 10. Обробляючи рядок вище, отримуємо для лівого підінтервалу затравний піксель (3,4), який потрапляє у стек. Правий підінтервал до цього часу вже заповнений. Обробка рядка нижче дає затравку (3,2) для лівого і (10,2) для правого підінтервалів. Ці пікселі також помішаються в стек. Саме зараз досягається максимальна глибина стека для оброблюваного багатокутника.

Тепер залишається лише один цікавий момент. Після заповнення 4-зв'язних полігональних підобластей з затравними пікселями 5, 4 і 3 на рис. 7.34,е зі стека витягується піксель, що позначений цифрою 2. Тут ми виявляємо, що всі пікселі на цьому рядку і на сусідніх рядках (вище і нижче) вже заповнені. Таким чином, ні один піксель в стек не поміщається. Зі стека витягується піксель 1 і рядок обробляється, при цьому знову додаткових пікселів не з'являється. Тепер стек порожній, багатокутник заповнений і робота алгоритму завершена.

Максимальна глибина стека в цьому випадку дорівнює п'яти.

Приклади реалізації

Приклад 7.13. У площині х-у знайти точки перетину площини з колом. Вершини: P0(4:4), P1(4:26), P2(20:26), P3(28:18), P4(21:4), P5(21:8), P6(10:8),

P7(10:4), V0(10:12), V1(10:20), V2(17:20), V3(21:16), V4(21:12)

```
Uses crt,graph;
Type TPoint = record
X,y:integer;
End;

Const n=7; m=4; s=10; max_elem=32;

Var
grDriver:integer;
grMode:integer;
ErrCode:integer;
P:array[0..n] of tpoint;
V:array[0..m] of tpoint;
R:array[0..max_elem-1,0..20] of integer;
I,j,y,k,bf,im:integer;
Buf:tpoint;

Function det(a11,a12,a21,a22:integer):integer;
Begin
Det2:=a11*a22-a12*a21;
End;

Procedure myline(p1,p2:tpoint; vara,b,c:integer);
Begin
A:=p2.y-p1.y;
B:=-p2.x+p1.x;
C:=p1.x*(p2.y-p1.y)-p1.y*(p2.x-p1.x);
End;

Function mix(a,b:integer):integer;
Begin if a<=b then min:=a else min:=b; end;
Function max(a,b:integer):integer;
Begin if a>=b then max:=a else max:=b; end;

Function intersect (p11, p12, p21, p22:tpoint; var p:t point): boolean;
```

```
Var d, a1, b1, c1, a2, b2, c2: integer;
Begin
 Line(p11, p12, a1, b1, c1);
 Line(p21, p22, a2, b2, c2);
D:=det2(a1, b1, a2, b2);
If d=0 then peretun:=false
Else begin
p.x:=round(det2(c1,b1,c2,b2)/d);
p.y:=round(det2(a1,c1,a2,c2)/d);
if p.y in [min(p1.y,p2.y)..max(p1.y,p2.y)] then
peretunsc:true;
end;
end;
function intersectline (p1,p2:tpoint; y:integer; var p:tpoint):boolean;
var a,b,c,d:integer;
begin
peretunsc:=false;
priama(p1,p2,a,b,c);
if a<>0 then
begin
p.x:=round((c-b*y)/a);
p.y:=round((a*y)/a);
if p.y in [min(p1.y,p2.y)..max(p1.y,p2.y)] then
peretunsc:=true; end;
end;
begin
grDriver:=detect;
InitGraph(grDriver, grMode,'');
ErrCode:=GraphResult;
If ErrCode=grOk then
Begin
P[0].x:=4
P[0].y:=4
P[1].x:=4
P[1].y:=26
P[2].x:=20
P[2].y:=26
```

```
P[3].x:=28
P[3].y:=18
P[4].x:=21
P[4].y:=4
P[5].x:=21
P[5].y:=18
P[6].x:=10
P[6].y:=8
P[7].x:=10
P[7].y:=4
V[0].x:=10
V[0].y:=12
V[1].x:=10
V[1].y:=20
V[2].x:=17
V[2].y:=20
V[3].x:=21
V[3].y:=16
V[4].x:=21
V[4].y:=12
For i:=0 to max_elem-1 do
For j:=0 to 20 do r [I,j]:=0;
Moveto(p[n].x*s,p[n].y*s);
For i:=0 to n do
Lineto (p[i].x*s,p[i].y*s);
Moveto(v[m].x*s,v[m].y*s);
For i:=0 to m do
Lineto (v[i].x*s,v[i].y*s);
For y:=0 to max_elem-1 do
For i:=1 to n do
If peretunsc (p[i-1],p[i],y,buf) then begin
Circle(buf.x*s, buf.y*s,s div 2);
K:=r[buf.y,0];inc(k);
R[buf.y,k]:=buf.x;
R[buf.y,0]:=k;
End;
For y:=0 to max_elem-1 do
```

```
Begin
K:=r[y,0];
For i:=1 to k do begin
im:=i;
For j:=i+1 to k-1 do
If r[y,j]<r[y,im] then im:=j;
Bf:=r[y,im];
R[y,im]:=r[y,i];
R[y,i]:=bf;
End;
End;
For i:=0 to k do writeln(r[i].x:4,r[i].y:4);
For i:=1 to k do
If (r[i].y<r[i-1].y) or ((r[i].y=r[i-1].y) and (r[i].x<r[i=1].x)) then
Begin
Buf:=r[i-1];
R[i-1]:=r[i];
R[i]:=buf;
End;
Readln;
clrscr;
For i:=0 to k do writeln (r[i].4:4,r[i].y:4);
Readln;
Circle(r.x*s,r.y*s,s div 2);
closeGraph;
writeln;
for i:=0 to max_elem-1 do begin
for j:=1 to r[I,0] do
write(r[i,j],'');
writeln;
end;
end
end.
```

Приклад 7.14. Даний багатокутник, зовнішня частина якого задана точками (4, 5), (4, 26), (20, 26), (28, 18), (21, 4), (21, 8), (10, 8), (10, 4), а внутрішня точками (10, 12), (10, 20), (17, 20), (21, 16), (21, 12) на растрі 32x32.

Напишіть програму, в якій реалізовано стрічковий алгоритм з затравкою для заповнення внутрішньої частини багатокутника. Затравка – піксель (14, 20)

```
Uses graph;
Label Label1, Label2, Label3, Label4, Label5, Label6, Label7, Label8;
Type tpoint=record
X,y:integer;
End;

Var
grDriver:integer;
grMode:integer;
ErrCode:integer;
Right, left, top, down, x1, x2, y1, y2:integer;
T1, t2: array[1..4]  of integer;
Flag:integer;
P, p1, p2, ppn, pn, pp1, pp2:tpoint;
N, I, h, s1, s2, m, x, y:integer;

Begin
If x1<left then t1[4]:=1 else t1[4]:=0;
If x1>right then t1[3]:=1 else t1[3]:=0;
If y1<down then t1[2]:=1 else t1[2]:=0;
If x1<top then t1[1]:=1 else t1[1]:=0;
If x2<left then t2[4]:=1 else t2[4]:=0;
If x2<right then t2[3]:=1 else t2[3]:=0;
If y2<down then t2[2]:=1 else t2[2]:=0;
If y2<top then t2[1]:=1 else t2[1]:=0;
Flag:=0;
s1:=0; s2:=0;
Pp1:=p1;
Pp2:=p2;
M:=32000;
For i:=1 to 4 do begin
s1:=s1+t1[i];
s2:=s2+t2[i]; end;
If (s1=0) and (s2=0) then goto Label7;
```

```
H:=0;
For i:=1 to 4 do begin
H:=h+trunk((t1[i]+t2[i])/2);
If h<>0 then begin
Flag:=-1;
Goto Label7; end; end;
If s1=0 then begin
N:=1;
Pp1:=p1;
P:=p2;
Goto Label2;
end;
If s2=0 then begin
N:=2;
Pp1:=p2;
P:=p1;
Goto Label2;
end;
N:=0;
Label1: if n<>0 then ppn:=p;
N:=n+1;
If n>2 then goto Label7;
P:=pn;
Label2: if x2-x1=0 then goto Label4;
M:= round((y2-y1)/(x2-x1));
If left < p.x then goto label3;
y:=m*(left-p.x)+p.y;
If y>top then goto Label3;
If y<down then goto Label3;
p.y:=y;
p.x:=Left;
goto Label1;
Label3: if right>p.x then goto Label4;
y:= m(right-p.x)+p.y;
If y>top then goto label4;
If y<down then goto label4;
p.y:=y;
```

```
p.x:=right;
goto Label1;
If m=0 then goto Label1;
If top>p.y then goto Label5;
X:=trunk(1/m)*(top-p.y)+p.x;
If x<left then goto Label5;
If x>right then goto Label5;
p.x:=x;
p.y:=top;
goto label1;
If down < p.y then goto Label8;
x:=trunc(1/m*(down-p.y)+p.x);
If x<left then goto Label6;
If x>right then goto Label6;
p.x:=x;
p.y:=down;
goto label1;
Label6: flag:=-1;
Label7: if flag=-1 then goto Label 8;
grDriver:=detect;
initGraph(grDriver, grMode,'');
ErrCode:=Graphresult;
If ErrCode:=grOk then begin
Line(pp1.x,pp1.y,pp2.x,pp2.y);
End;
Readln;
Label8;
      End.
```

Список рекомендованої літератури

1. Роджерс Д. Алгоритмические основы машинной графики: Пер. с англ. – М.: Мир, 1989. – 512 с.
2. Роджерс Д., Адамс Дж. Математические основы машинной графики. – М.: Машиностроение, 1980. – 231 с.
3. Павлидис Т. Алгоритмы машинной графики и обработки изображений. – М: Радио и связь, 1986. – 400 с.

РОЗДІЛ 8. КОНТРОЛЬНІ ЗАВДАННЯ

Лабораторна робота № 1

Тема. Особливості синтезу статичного і динамічного зображень в графічному режимі

1. Синтез статичного зображення

Широке застосування в інженерну діяльність персональних комп'ютерів (ПК) дозволило різко розширити галузь застосування графічних зображень, як найбільш інформативних засобів для сприйняття і передачі інформації. Існуючі засоби формування зображень визначаються одним із двох принципово різних режимів роботи адаптера (контролера) дисплея − текстовим або графічним. У текстовому режимі екран монітора розбивається на матрицю знакомісць розміром 40×25, 80×25, 80×43 або 80×50 залежно від типу адаптера і номера формату. Стандартним вважається формат, розбиваючий весь простір екрану на 25 стрічок довжиною у 80 знакомісць. У той же час адаптер BGA допускає формат 43×80, а VGA і вище − 50×80. Кожне знакомісце характеризується такими параметрами:

1. координатами;
2. складом;
3. кольором символу;
4. кольором фону.

Координати знакомісця − номери стрічки і стовпця, на перетині яких розміщене шукане знакомісце. Нумерація стрічок здійснюється зверху донизу (1−25 чи 0−24), а стовпців зліва направо (1−80 чи 0−79).

Склад знакомісця − один із символів розширеного набору ASCII, який називається зазвичай кодовою сторінкою і завантажуючого в пам'ять ПК у момент включення комп'ютера чи запуску програмного пакета. Набір містить 256 символів, з яких перші 128 зазвичай постійні для всіх кодових сторінок, а решта 128 змінюються залежно від алфавіту країни чи призначення програмного пакету. Усі символи мають свої числові коди в діапазоні від 0 до 255, тобто для збереження складових одного знакомісця потрібен один байт. У той же час даний код дозволяє відобразити будь-який символ кодової таблиці шляхом набору коду на правій цифровій клавіатурі при натисненій кнопці Alt.

У сучасних адаптерах для кодування кольору символу зазвичай застосовується чотири біта, що дає 16 можливих кольорів, які нумеруються: 0 − чорний, 1 − синій, 2 − зелений, 3 − голубий, 4 − червоний, 5 − рожевий, 6 − коричневий, 7 − сірий, 8 − темно-сірий, 9 − яскраво-сииій, 10 − яскраво-зелений, 11 − яскраво-голубий, 12 − яскраво-червоний, 13 − яскраво-рожевий, 14 − жовтий, 15 − білий.

Для зафарбовування фону зазвичай застосовуються 8 кольорів палітри, що потребує для кодування 3 біта. Ще один біт призначений для фіксації режиму мигання символу чи його відсутності. У відеопам'яті ПК зберігається інформація тільки про три останні параметри знакомісця, яка подається одним байтом.

Таким чином, під час роботи в символьному режимі одне знакомісце потребує два байти відеопам'яті (один на опис складу, інший − на опис параметрів), а загальна потреба в такій пам'яті становить всього 4 кБ (формат 80×25) чи 8 кБ (80×50). Низька вимогливість до відеопам'яті робить символьний режим привабливим для синтезу найпростіших графічних зображень (таблиць, стовпчикових діаграм, комп'ютерних ігор). Для цього в кодову таблицю ПК вводяться символи псевдографіки, вивід яких здійснюється за правилом, описаним раніше.

Опис середовища програмування

Статичне графічне зображення в символьному режимі може бути створене засобами будь-якого текстового редактора (Лексикона, Фотона, WD та ін.), але подібний підхід не дозволяє застосовувати різні кольори в силу відсутності відповідних засобів в складі вказаних пакетів. Такі засоби мають різні мови програмування, тому лабораторна робота базується на синтезі програми на мови Паскаль, яка створює на екрані потрібне зображення.

Зазвичай для виводу інформації на екран у мові Паскаль застосовуються такі процедури, які перебувають у модулі Crt:

TextBackground *(колір)* − задає колір фону знакомісця. Колір задається номерами 0÷7.

TextColor *(колір)* − задає колір символу за допомогою номерів 0÷15.

Window *(x1, y1, x2, y2)* − задає розміри вікна на екрані і встановлює курсор у лівий верхній кут вікна з координатами (1, 1), оскільки нумерація стрічок і стовпців у Паскалі починається з 1. Під час входу в середовище ТурбоПаскаль по замовчуванню встановлюється вікно з параметрами (1, 1, 80, 25). Під час встановлення вікна меншого розміру увесь ввід/вивід прикладної програми здійснюється через нього. Під час повернення до повного екрану необхідно виконати процедуру Window (1, 1, 80, 25).

ClrScr − очищає поточне вікно, заповнюючи його кольором фону і встановлює курсор у його верхній лівий кут.

GotoXY *(x, y)* − встановлює курсор до елемента екрана з заданими координатами *x, y.*

Сам же вивід на екран конкретного символу чи символьної стрічки здійснюється процедурами Write чи WriteLn. Фіксація синтезованого зображення на певний час (до натиснення будь-якої кнопки) можлива за допомогою циклічної структури repeat .until і процедури KeyPressed, повертаючої значення True під час натиснення будь-якої кнопки. На певний час зображення на екрані фіксується за допомогою процедури Delay *(кількість мілісекунд)*.

Приклад програми. Створити програму, яка реалізує графічними засобами мови програмування Turbo Pascal таке статичне зображення: фасад будинку з двома вікнами

Лістинг програми

```
program s;
uses crt, graph;
var
driver, mode:integer;
begin
clrscr;
driver:=0;
mode:=0;
InitGraph (driver,mode,'');
rectangle (200,200,300,300);
rectangle (210,230,230,270);
rectangle (270,230,290,270);
line (200,200,250,150);
line (250,150,300,200);
readln;
end.
```

2. Синтез динамічного зображення

Динамічне зображення, яке генерується на екрані дисплея ПК, широко застосовується для ілюстрації протікання в часі різних процесів, пов’язаних як із обслуговуванням самого комп'ютера (копіювання файлів, тестування жорсткого диска), так і виконання на ПК різних прикладних програм (моделювання руху судна чи робота, комп'ютерні ігри, переміщення по екрану редагуючого тексту і т.п.). Таке зображення має більшу інформативність, ніж статичне, але потребує і значних затрат праці для своєї реалізації. Найбільш повноцінне, реалістичне динамічне зображення може бути отримане тільки під час роботи дисплея в графічному режимі.

Але часто потрібний результат може бути доступний і засобам символьного режиму, що знижує, в кінцевому результаті, вимоги до ресурсів комп'ютера і підвищує його швидкодію. Динамічне зображення може бути синтезоване тільки програмним шляхом.

Можливі різні підходи до формування динамічних зображень. Існує варіант формування зображення, яке не містить статичних елементів, що потребує на кожному кроці його стирання з наступним перемальовуванням на новому місці. Стирання може бути виконане очищенням (команда ClrScr) чи заповненням екрану пробілами потрібного кольору.

Якщо зображення формується тільки на частину екрану, то рекомендується виділити командою Window вікно потрібного розміру і працювати в його межах.

Якщо потрібне зображення містить нерухомий фрагмент, то на кожному кроці рекомендується стирати і синтезувати на новому місці тільки рухому частину, що знижує загальну трудоємкість генерації рисунка. При цьому можливі такі варіанти:

1. із зафарбовуванням кольором фону рухомої частини;
2. із накладанням на рухому частину вікна і його очищенням кольором фону.

Опис середовища програмування

Синтез рухомого зображення може бути реалізований засобами будь-якої алгоритмічної мови високого рівня, яка містить процедури і функції роботи з екраном. У даній роботі, як і в попередній, пропонується вирішити завдання на основі мови ТурбоПаскаль.

Приклад програми. Створити програму, реалізуючи засобами псевдографіки мови програмування Turbo Pascal таке динамічне зображення: на екрані в лівому верхньому і правому нижньому кутах написано різним кольором «ЛНТУ». Лівий напис плавно опускається, а правий піднімається до краю екрану протягом 13 сек.

Лістинг програми

```
program semafor;
uses
Crt;
var
i:integer;
begin
TextBackground(6);
Clrscr;
for i:=1 to 25 do
begin
TextColor(0);
GotoXY(1,i);
Write('LNTY');
TextColor(6);
GotoXY(1,i-1);
Write('    ');
TextColor(5);
GotoXY(77,26-i);
Write('LNTY');
TextColor(6);
GotoXY(77,27-i);
Write('    ');
```

```
GotoXY(1,1);
write('  ');
GotoXY(1,i);
delay(50000);
end;
end.
```

Варіанти завдань

Створити програму, реалізуючу засобами псевдографіки мови програмування Turbo Pascal такі динамічні зображення:

1. на екрані відображається календар, на якому написані дні місяця. Червона рамка рухається по всіх числах із інтервалом у 1,1 сек;
2. на екрані по горизонталі намальовані 4 вікна. У кожному з них по черзі загорається світло зі збільшуючимся інтервалом від 1 до 4 сек;
3. на екрані по горизонталі намальований спідометр до 200 км/год. Максимальна швидкість досягається за 25 сек, а потім падає до 80 км/год. за 15 сек;
4. на екрані зображаються осі координат, а потім у динаміці 10°/сек символами * будується графік одного періоду синусоїди з амплітудою 10 стрічок;
5. на екрані вертикально намальований термометр зі шкалою від -20° до 60°. Температура на першому етапі від -10° до 30° за 12 сек, а потім за 10 сек падає до нуля;
6. намальоване перехрестя, яким рухаються назустріч одна одній дві машини, швидкість однієї в два рази вище, ніж іншої;
7. по екрану зліва направо протягом 60 сек рухається стрілка =>, швидкість початку в два рази вище швидкості хвоста;
8. по екрану справа наліво протягом 20 сек рухається стовпчик коричневого кольору, на кожному кроці збільшуючись у висоту на одну стрічку;
9. по периметру екрану з кроком у 0,4 сек рухаються один за одним три трикутники: червоного, зеленого і жовтого кольору;
10. на екрані намальована вертикальна драбина жовтого кольору. Щабель червоного кольору рухається знизу вверх з кроком у 2 сек;
11. прямокутник червоного кольору, рухаючись по екрану справа наліво з кроком 0.6 сек, тягне за собою подвійну лінію зеленого кольору;
12. дві башти: жовтого кольору висотою у 15 стрічок і коричневого − у 1 стрічку. Із дискретом у 1 сек висота жовтої башти зменшується до одиничної, а коричневої збільшується до 15 стрічок;
13. на екрані зображена буква «Н» висотою у 14 стрічок. Щабель букви повільно декілька разів піднімається і опускається. Під час зміни напряму змінюється колір букви;
14. на екрані зліва направо переміщується світлофор із кроком по горизонталі одна позиція, а за часом – 0,5 сек. На кожному кроці змінюється активний колір світлофора;

15. фігура ⊓, обертаючись за годинниковою стрілкою зі швидкістю 90°/сек, здійснює чотири оберти;

16. у прямокутнику 12×14 коричневого кольору намальований по центру прямокутник 6×4 білого кольору. Білий прямокутник повільно переміщується до лівої границі внутрішнього прямокутника, а потім − до правої;

17. на екрані намальована стилізована зелена ялинка. На ній спалахують і гаснуть три червоні зірочки з інтервалом 1 сек, а ялинка переміщується зверху екрану вниз;

18. дві рамки, початково знаходяться в правому нижньому і лівому верхньому кутах екрану, переміщуються по діагоналі назустріч одна одній до повного співпадання з дискретом 0.7 сек/крок;

19. вздовж екрану намальований забор одного кольору, а за ним переміщуючий прямокутник іншого кольору. Дискрет переміщення - символ/сек;

20. прямокутник висотою в 12 стрічок, у якому написане слово «Інформація», рухається повільно по екрану справа наліво, а слово рухається зверху вниз прямокутника.

Зміст звіту

1) Зміст завдання;
2) лістинг програми;
3) надруковане зображення.

Контрольні питання

1. Яким чином реалізується на мові Turbo Pascal затримка в часі?
2. Для чого застосовується динамічне зображення, яке створюється на екрані комп'ютера?
3. Чому дорівнює дискрет переміщення зображення по горизонталі і вертикалі в символьному режимі роботи комп'ютера?
4. Яким чином можна стерти зображення на екрані комп'ютера?
5. Як можна засобами псевдографіки намалювати на екрані комп'ютера трикутник?

Лабораторна робота № 2

Тема. Синтез статичного зображення в графічному режимі

Існує два режими роботи дисплея персонального комп'ютера (ПК): символьний і графічний. Другий режим є універсальним і дозволяє створити зображення високої складності, оскільки реалізує керування кольором кожної точки (пікселя) екрану. Можливості графічного режиму визначаються такими параметрами комп'ютерної системи:

1. роздільною здатністю (допустимою кількістю точок) екрану;
2. розміром відеопам'яті;
3. встановленим відеоадаптером (відеоконтролером);
4. встановленим графічним драйвером.

У наш час дисплеї широкого застосування допускають роздільну здатність 320×200, 640×200, 640×400, 640×350, 720×348, 640×480, 720×350 чи 1024×768. Більш висока роздільна здатність, як правило, застосовується в дисплеях спеціального призначення.

Відеопам'ять (ВП) зберігає код кольору кожного пікселя. У кольоровій палітрі в 4 кольори на кожну точку потрібно 2 біта ВП, у 16 кольорів − 4 біта (півбайта). Сучасні дисплеї допускають роботу з палітрою в 224 кольори, тобто на кожен піксель застосовується 24 біта (3 байта) ВП. За обмеженого розміру ВП збільшення кольоровості може привести до зниження роздільної здатності дисплея і навпаки.

Адаптер (відеоадаптер, відеоконтролер, відеокарта) являє собою електронну плату в складі ПК, яка керує роботою дисплея. У наш час у ПК широко застосовуються такі види адаптерів: CGA, EGA, VGA, SVGA і ін. Кожен адаптер, у той же час, допускає декілька графічних підрежимів, які відрізняються кольоровою палітрою, роздільною здатністю та кількістю графічних сторінок.

Графічний режим роботи дисплея потребує також підтримки графічного драйвера − спеціальної програми з розширенням .bgi.

Опис середовища програмування

Вивчення графічного режиму здійснюватимемо в середовищі мови програмування TurboPascal. Під час запуску програми по замовчуванню дисплей перебуває в символьному режимі роботи. Для переведення його в графічний режим застосовується спеціальна процедура мови InitGraph (нк, нр, пд), де: **нк** − номер відеокарти. Існують такі номери для найбільш розповсюджених відеокарт: 0 − автоматичне визначення комп'ютером існуючої відеокарти, 1 − CGA; 3 − EGA; 9 − VGA; **нр** − номер графічного підрежиму для даної відеокарти. Для EGA можливі підрежими: 0 − 640×200, 16 кольорів, 4 графічні сторінки; 1 − 640×350, 16 кольорів, 2 сторінки, а для VGA: 0 − 640×200, 16 кольорів, 4 сторінки; 1 − 640×350, 16 кольорів, 2 сторінки; 2 − 640×480, 16 кольорів, 1 сторінка; **пд** − шлях до драйвера, тобто стрічка, оформлена у відповідності з синтаксисом, прийнятим у мові TurboPascal, і яка містить маршрут до каталогу з графічним драйвером. Якщо драйвер розміщується в поточному каталозі, то стрічка може бути пустою.

Повернення з графічного режиму дисплея в символьний здійснюється процедурою CloseGraph без параметрів.

Всі процедури графічного режиму розміщуються в модулі Graph, який повинен бути підключений до програми командою Uses.

Установка кольорів фону і зображення здійснюється відповідно процедурами SetBkColor (колір) і SetColor (колір).

У сучасних адаптерах для кодування кольору символу звичайно застосовується 4 біта, що дає 16 можливих кольорів, які нумеруються 0−15.

Для забарвлення фону звичайно застосовують перші 8 кольорів палітри, що потребує для кодування 3 біта. Ще один біт призначений для фіксації режиму мигання символу чи його відсутності. У відеопам'яті ПК зберігається інформація тільки про три останні параметри знакомісця, які представляються одним байтом. Таким чином, під час роботи в символьному режимі одне знакомісце потребує 2 байта відеопам'яті (один на опис вмісту, інший — на опис параметрів), а загальна потреба в такій пам'яті складає всього 4 кБ (формат 80×25) чи 8 кБ (80×50). Низька вимогливість до відеопам'яті робить символьний режим привабливим для синтезу найпростіших зображень (таблиць, стовпчикових діаграм, комп'ютерних ігор.

Під час встановлення графічного режиму по замовчуванню, доступний для роботи весь екран дисплея. Його очищення із зафарбовування кольором фону здійснюється процедурою ClearDevice. При цьому показник поточної позиції встановлюється в точку з координатами (0, 0).

Під час роботи в межах менших ніж екран області можна оформити її у вигляді вікна процедурою SetViewPort (xl, yl, x2, y2, Clip), де: x1, y1— координати лівого верхнього кута вікна; x2, y2 − координати правого нижнього кута вікна; Clip − обмежувач фігур. Якщо CIip=True, то всі побудови здійснюються тільки в межах вікна, в іншому випадку частини фігур можуть виходити за межі вікна.

Процедура ClearViewPort аналогічна за своїми діями процедурі ClearDevice, але стосовно вікна.

Графічне зображення може будуватися поточково, за допомогою процедури PutPixel (x, y, колір), *де x, y* − координати пікселя. Номери кольорів відповідають вказаним раніше.

Під час синтезу зображення на основі ліній попередньо потрібно вказати їх параметри за допомогою процедури SetLineStyle (стиль, шаблон, товщина). Параметр стиль може приймати такі значення: 0 – суцільна лінія; 1 – пунктирна; 2 – штрих пунктирна; 3 – штрихова; 4 – задається користувачем. Параметр шаблон визначає вид лінії, який задається користувачем, тобто стилем 4. В іншому випадку доцільно задати шаблон 0. Параметр товщина може приймати такі значення: 1 – нормальна; 3 – товста.

За допомогою ліній можна будувати такі фігури:

1. дугу на основі процедури Arc (x, y, a1, a2, r), де: x, y – координати центра дуги, a1 – кут до початкової точки дуги, який відраховується проти годинникової стрілки від горизонтальної вісі, направленої зліва направо, a2 – кут до кінцевої точки дуги, який відраховується проти годинникової стрілки від горизонтальної вісі, направленої зліва направо, r – радіус дуги;

2. коло на основі процедури Circle (x, y, r), де: x, y – координати центра кола, r – радіус кола;

3. пряму лінію на основі процедур: Line (xl, y1, x2, y2), де x1, y1, x2, y2 – координати точок, між якими проводиться лінія;

LineRel (Dx, Dy), де: Dx, Dy – зміщення координат точки кінця лінії по відношенню до координат початкової точки;

LineTo (x, y), де x, y – координати точки, в яку проводиться лінія з поточної точки.

Дві останні процедури мають на увазі, що показник координат вже встановлений в початкову точку лінії. Дана операція може бути виконана за допомогою процедур MoveTo (x, y) чи MoveRel (Dx, Dy). Значення параметрів x, y, Dx, Dy визначені раніше;

4. прямокутник за допомогою процедури Rectangle (xl, yl, x2, y2), де: x1, y1 – координати лівого верхнього кута прямокутника, а x2, y2 – правого нижнього; TurboPascal містить також процедури побудови зафарбованих фігур;

5. зафарбований прямокутник на основі процедури Bar (xl, y1, x2, y2). Значення параметрів пояснене вище;

6. сектор кола на основі процедури PieSlice (x, y, a1, a2, r). Значення параметрів аналогічні процедурі Arc.

Зафарбовані фігури передбачають попереднє встановлення стилю заповнення за допомогою процедури SetFillStyle (шаблон, колір), де шаблон допускає такі значення:

0 – заповнення кольором фону,

1 – однорідне заповнення вказаним у даній процедурі кольором,

2 – заповнення - - - -,

3 – заповнення ///,

4 – заповнення товстими ///,

5 – заповнення товстими \\\,

6 – заповнення \\\,

7 – заповнення кліткою,

8 – заповнення косою кліткою,

9 – заповнення густою сіткою,

10 - заповнення рідкими точками,

11 - заповнення густими крапками.

Споживач може сам побудувати замкнений контур і зробити його заповнення за допомогою процедури FloodFill (x, y, контур), де: x, y – координати будь-якої точки

всередині зафарбовуючої області, контур – колір лінії контуру, всередині якого здійснюється зафарбовування.

Приклад складання програми. Створити програму, яка реалізує графічними засобами мови програмування Turbo Pascal таке статичне зображення: фасад будинку з двома вікнами.

Лістинг програми

```
program s;
uses crt, graph;
var
driver, mode:integer;
begin
clrscr;
driver:=0;
mode:=0;
InitGraph (driver,mode,'');
rectangle (200,200,300,300);
rectangle (210,230,230,270);
rectangle (270,230,290,270);
line (200,200,250,150);
line (250,150,300,200);
readln;
end.
```

Варіанти завдань

Створити програму, яка реалізує графічними засобами мови програмування TurboPascal таке статичне зображення:

1. голубе небо, червоне сонце, синє море;
2. годинник із годинною, хвилинною і секундною стрілками різного кольору;
3. стакан води з трьома бульбашками повітря;
4. червоне сонце з десятьма вихідними промінцями;
5. в'язку різнокольорових куль;
6. стилізоване зображення крану;
7. колесо з шістьма спицями різного кольору;
8. дорожній знак „цеглина»;
9. світлофор;
10. стіл більярду з трьома кулями;
11. вертикальний термометр;
12. жовтий прямокутник із червоним контуром; всередині прямокутника симетрично розмістити чотири білих кола з зеленим контуром;
13. драбину з восьми перекладин;

14. стилізоване зображення парашуту;
15. диск телефону;
16. тріумфальну арку;
17. анфас письмового столу з двома тумбами;
18. шахову дошку;
19. м’яч, верхня половина якого зафарбована в жовтий, нижня – в зелений колір, а посередині червона смуга;
20. букву „Н» висотою в половину екрану, елементи букви різного кольору;
21. зелене дерево;
22. м’яч всередині деякої труби;
23. м’яч на поверхні;
24. кулька і сходи вверх;
25. циркового клоуна;
26. цар-рибу;
27. людину з мультфільму «Папка, − палка, − огуречек…»;
28. хмару, сонце;
29. телевізор, всередині якого риба;
30. ялинку, падаючий сніг.

Зміст звіту

1) Зміст завдання;
2) лістинг програми;
3) надруковане зображення.

Контрольні питання

1. Які параметри лінії потрібно вказати під час застосування даного графічного примітива?
2. Які параметри потрібно задати під час переходу до графічного режиму?
3. Які процедури переміщують вказівник по екрану в графічному режимі?
4. Що таке піксель?
5. Чи можна засобами графіки наносити на екран комп’ютера зафарбований трикутник?

Лабораторна робота № 3

Тема. Синтез динамічного зображення в графічному режимі

Формування динамічного зображення в графічному режимі опирається на ті ж алгоритми, що були розглянуті в попередній лабораторній роботі. Проте даний режим містить звичайно ряд спеціальних засобів, дозволяючи створити більш плавний, наближений до природного, рух. До таких засобів відносяться:

1. наявність двох або більше графічних сторінок, кожна з яких може бути видимою (тобто відображатися на екрані дисплею) і/чи активною (саме на ній модифікується зображення засобами алгоритмічної мови). По замовчуванню під час переходу в графічний режим одна й та сама сторінка є і видимою і активною, що зручно під час синтезу статичного зображення;

2. інструментарій роботи з виділеною прямокутною ділянкою графічного зображення, який зберігається в оперативній пам'яті і в подальшому може застосовуватися для:

- очищення на екрані частини чи всього зображення шляхом вибірки фрагменту з ОЗП (оперативного запам'ятовуючого пристрою) і накладання на стираючу ділянку;
- переміщення частини чи всього зображення шляхом вибірки фрагмента з пам'яті і розміщення на новому місці екрану.

Даний інструментарій може застосовуватися в рамках всіх розглянутих раніше алгоритмів формування динамічного зображення.

Опис середовища програмування

Для роботи з розглянутим раніше інструментарієм алгоритмічна мова TyrboPascal містить такі процедури:

SetActivePage (номер сторінки) – для встановлення номеру активної сторінки. Необхідно врахувати, що нумерація сторінок у графічному режимі мови TyrboPascal починається з нуля,

SetVisualPage (номер сторінки) – для встановлення номера видимої (відображаючої на екрані) сторінки,

Getlmage (xl, yl, x2, y2, пам'ять) – для збереження зображення заданої ділянки екрану в ОЗП. Параметри xl, y1, x2, y2 задають координати лівого верхнього і правого нижнього кутів зберігаючої ділянки екрану, а пам'ять являє собою змінну – ім'я буфера в ОЗП, зберігаючи дане зображення,

Putlmage (x, y, пам'ять, спосіб) – для видачі на екран збереженої в ОЗП ділянки. Змінна спосіб вказує вид об'єднання передаючої на екран інформації з вже відомою і може приймати такі значення: 0 – MOV, 1 – XOR, 2 – OR, 3 – AND, 4 – NOT. Значення 0 (MOV) дозволяє під час накладання з деяким зсувом стерти попереднє зображення і регенерувати його на новому місці, що створює ефект переміщення. Інші значення змінної дозволяють створювати різні відеоефекти.

Робота з виділеним фрагментом досить складна, оскільки передбачає застосування змінних типу Pointer (показник) і потребує попереднього визначення

розміру виділяючого прямокутника за допомогою функції ImageSize (x1, yl, x2, y2) і резервування в ОЗП для нього пам'яті за допомогою процедури GetMem (пам'ять, розмір). Більш детально даний інструментарій розглянутий в прикладі.

Графічний режим дозволяє як у статиці, так і в динаміці виводити на екран фрагменти тексту. Для цього застосовуються такі процедури як:

1. SetTextJustify (горизонталь, вертикаль) – для вирівнювання тексту по горизонталі і вертикалі, де: горизонталь – змінна, задаюча режим горизонтального вирівнювання і приймаюча одне зі значень: 0 – нульове вирівнювання (вивідний символ розміщується справа від задаючої координати виводу), 1 – вирівнювання по центру, 2 – праве вирівнювання; вертикаль – змінна, задаюча режим вертикального вирівнювання і приймаюча одне з таких значень: 0 – нижнє вирівнювання (вивідний символ розміщується зверху по відношенню до задаючої координати виводу), 1 – верхнє вирівнювання;
2. SetTextStyle (шрифт, напрям, розмір) – для встановлення поточних параметрів вивідного тексту, де: шрифт – змінна, задаюча тип шрифта і приймаюча одне зі значень в діапазоні 0…4, напрям – змінна, задаюча напрям виводу тексту (0 – по горизонталі, 1 – по вертикалі), розмір – змінна, задаюча розмір символів і визначається дослідним шляхом виходячи з умов до зображення;
3. OutTextXY (x y, стрічка) – для виводу на екран послідовності символів стрічка, починаючи з точки з координатами x, y;
4. OutText (стрічка) – для виводу на екран послідовності символів стрічка, починаючи з поточної позиції.

Приклад програми. Створити програму, яка реалізує графічними засобами мови програмування Turbo Pascal таке динамічне зображення: вікна будинку різного кольору рухаються на зустріч одне одному і в кінцевому випадку міняються місцями.

Лістинг програми

```
program s;
uses crt, graph;
var
driver, mode,i:integer;
begin
driver:=0;
mode:=0;
InitGraph (driver,mode,'');
for i:=0 to 60  do begin
clrscr;
rectangle (200,200,300,300);
setcolor(3) ;
rectangle (210+i,220,230+i,270);
```

```
setcolor(5) ;
rectangle (270-i,220,290-i,270);
setcolor(white) ;
line (200,200,250,150);
line (250,150,300,200);
delay (5000)  ;
end;
readln;
end.
```

Варіанти завдань

Створити програму, яка реалізовує засобами графіки мови програмування TyrboPascal таке динамічне ображення, яке базується на завданні попередньої лабораторної роботи:

1. червоне сонце піднімається з-понад моря і переміщується вверх і вправо;
2. секундна стрілка здійснює повний оберт;
3. дві бульбашки спливають за 15 сек;
4. сонце обертається проти годинникової стрілки з дискретою 10 °/сек.; довжина циклу 20 сек;
5. одна куля зі в'язки піднімається у верх, а друга − з подвоєною швидкістю опускається вниз;
6. гак крану рухається по стрілці вправо й опускається вниз;
7. колесо обертається 15 сек з дискретою 0,5 сек, обертаючись на кожному кроці на 10 °;
8. знак «цеглина» поступово (за 20 сек з дискретою 0,4 сек на кожному кроці) ховається за горизонт;
9. колір світлофора змінюється з дискретою 0,7 сек; дана процедура повторюється три рази;
10. одна куля на більярдному столі рухається вгору, а друга з подвоєною швидкістю вправо;
11. покази термометра змінюються від -10° до 40°, а потім у два рази повільніше до 0°;
12. серпоподібний місяці повільно заходить за горизонт;
13. драбина переміщується справа наліво, а потовщена нижня перекладина - за цей же час вверх;
14. до парашуту підвішені слова «ЛНТУ група АВ-21»; все зображення опускається вниз і відноситься вліво з подвійною швидкістю;
15. диск телефону повертається на п'ять цифр з дискретою 2 сек;
16. на верху арки ліхтар; арка переміщується по екрану справа наліво, при цьому змінюється колір ліхтаря;

17. у кожній тумбі по три ящики; з дискретою у 1 сек всі ящики висуваються з тумб;

18. стилізоване зображення шашки проходить по головній діагоналі дошки з дискретою 0,7 сек,

19. м'яч переміщується знизу вверх і до кінця руху зменшується в два рази;

20. буква «Н» переміщується зліва направо і зменшується до кінця руху в два рази;

21. дерево зеленіє і росте, потім жовтіє і зменшується в розмірі;

22. зигзагоподібний рух м'яча всередині деякої труби зліва направо;

23. рух м'яча, кинутого на поверхню: спочатку частіше вдаряється об землю і вкінці зупиняється;

24. шарик піднімається по сходах вверх, рухається по площадці, падає вниз, розділяється на два нових шарики, які починають рухатися в різні сторони;

25. циркового клоуна, в якого зі сторони в сторону рухаються очі;

26. цар-рибу, повз якої пропливають пухирці повітря різного розміру;

27. йдучу людину з мультфільму «Палка, − палка, − огуречек...);

28. хмару, закриваючу сонце, коли хмара закриє сонце, починається дощ;

29. телевізор, всередині якого рухається риба спочатку зліва направо потім справа-наліво, то пливе вздовж верхнього краю екрану, то посередині, то вздовж нижнього краю екрану;

30. ялинку, сніговика, падаючий сніг.

Зміст звіту

1) Зміст завдання;
2) лістинг програми;
3) надруковане зображення.

Контрольні питання

1. Яким чином здійснюється зміна графічних сторінок у алгоритмічній мові Turbo Pascal?
2. Чи можна змінювати висоту символів, які відображаються в графічному режимі?
3. Чи можна розміщувати текст вертикально по екрану в графічному режимі?
4. Чи можна застосовувати вікна під час створення динамічного зображення в графічному режимі?
5. Які попередні операції потрібні для збереження в пам'яті ПК виділеного фрагменту?

Лабораторна робота № 4

Тема. Перетворення двовимірних координат

Перетворення координат широко використовується під час синтезу як статичних, так і динамічних зображень з метою реалізації таких операцій над рисунком, як перенесення, поворот, подібність, симетрія.

Перенесення є достатньо простою операцією, оскільки реалізується шляхом зміни значень першої і/чи другої координат. Інші операції, а тим більше їх комбінація, характеризуються більшою складністю і вимагають відповідних математичних перетворень.

У лабораторній роботі розглядаються тільки двовимірні перетворення. Звичайно в даному процесі початково задаються основні (первинні) координати x1, y1 а через них у вигляді рівнянь прораховуються вторинні координати x22, y22.

$$x22 = f(xl, yl),$$
$$y22 = f(xl, yl).$$

Найбільш поширені так звані афінні перетворення, які припускають збереження після розрахунку вторинних координат паралель і форми у прямих. Загальний вигляд рівнянь таких перетворень має вигляд:

$$x22 = axxxl+axyyl+ax,$$
$$y22 = ayxx1 +ayyy1+ay.$$

При цьому зсув без повороту характеризується матрицею коефіцієнтів:

$$\begin{vmatrix} 1 & 0 & a_x \\ 0 & 1 & a_y \end{vmatrix},$$

де ax, ay - задають зсув відповідно по осях x і y, тобто x2 = xl+ax, y2=y1+ay.

Операція масштабування (перетворення подібності) реалізується матрицею:

$$\begin{vmatrix} M & 0 & 0 \\ 0 & M & 0 \end{vmatrix}.$$

Дзеркальне відображення щодо діагоналі першого квадранта (зміна координатних осей) припускає матрицю вигляду:

$$\begin{vmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{vmatrix}.$$

Повороту на кут α проти годинникової стрілки відповідає матриця:

$$\begin{vmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha\alpha & \cos\alpha & 0 \end{vmatrix}.$$

Комбінації двох або більше операцій перетворення відповідає комбінація коефіцієнтів відповідних матриць. Так, наприклад, зсув із одночасним поворотом реалізується матрицею:

$$\begin{vmatrix} \cos\alpha & -\sin\alpha & a_x \\ \sin\alpha\alpha & \cos\alpha & a_y \end{vmatrix}.$$

Опис середовища програмування

Під час перетворення координат у середовищі мови програмування TurboPascal необхідно врахувати, що графічний екран ПК є четвертим квадрантом прямокутної системи координат, тобто початок координат знаходиться в лівому верхньому кутку. Повноцінна система координат створюється самим користувачем шляхом перенесення її початку вниз і/чи вправо на величини Δx і/чи Δy, тобто реалізується локальна система, зсунута на дані величини щодо глобальної. Надалі глобальні координати всіх точок ($x_г$, $y_г$) перераховуються через локальні ($x_л$, $y_л$) за такими рівняннями:

$$x_г = x_л + \Delta x,$$
$$y_г = \Delta y - y_л.$$

Друга проблема, що виникає звичайно при складних перетвореннях, − різна роздільна здатність (кількість пікселів на одиницю довжини) графічного екрану по горизонтали і вертикалі. Спроба зобразити квадрат шляхом задання однакової кількості пікселів по його сторонах приводить до відображення прямокутника, витягнутого по висоті. Даний ефект відсутній тільки за роздільної здатності 640×480 і особливо посилюється під час встановлення підрежимів з низькою кількістю пікселів по вертикалі, наприклад, 640×200. Для усунення вказаного ефекту засобами мови Паскаль проводиться попереднє масштабування зображення за допомогою процедури GetAspectRatio (mx, my), де mx, my − змінні, в які записується число точок по горизонталі і вертикалі. Надалі, під час побудови потрібного зображення всі координати по осі X повинні множитися на відношення my/mx, або, навпаки, всі координати по осі Y множаться на відношення mx/my. При цьому виникає ще одна проблема пов'язана з особливостями мови Паскаль, які отримуються під час множення. Масштабовані координати можуть приймати суттєві значення, що заборонено для параметрів графічних примітивів даної мови. Тому всі результати таких множень повинні округлятися функцією Round.

Приклад програми. У першому квадранті локальної системи координат зобразити прямокутник, ліва нижня точка якого має координати (10,10), а елементи

зображення мають розмір не менше 50 пікселів і різне забарвлення, а потім перемістити його вгору на 120 і в право на 200 пікселів з поворотом за годинниковою стрілкою на 70°.

Лістинг програми

```
program dd;
uses crt,graph;
var dr,md,i:integer;
p:pointer;    size:word;
begin
initgraph(dr,md,'');
setlinestyle(0,0,3);
setcolor(lightgreen);
rectangle(2,25,150,100);
 size:=imagesize(0,0,153,103);
 getmem(p,size);
 getimage(0,0,153,103,p^);
 cleardevice;
for i:= 1 to 100 do
 begin
 putimage(50,200-i,p^,0);
 delay(5000);
 end;
for i:= 1 to 150 do
begin
 putimage(50+i,100,p^,0);
 delay(5000);
 end;
 cleardevice;
 line(215,150,255,100);
 line(215,150,325,230);
 line(325,230,360,175);
 line(360,175,255,100);
 readln;
 end.
```

Варіанти завдань

Створити програму, що реалізує графічними засобами мови програмування TurboPascal зображення, включаючи: локальну систему координат; на початку першого квадранта координат початкове зображення, ліва нижня точка якого має координати (10, 10), а елементи зображення мають розмір не менше 50 пікселів і різне забарвлення; перетворене відповідно до завдання зображення.

1. —|— – подібна фігура дзеркально відображається щодо осі X і збільшується в 1,5 рази;
2. —|— – подібна фігура переміщається вгору на 90 і вліво на 150 пікселів із поворотом на 40° за годинниковою стрілкою;
3. ┌— − подібна фігура дзеркально відображається щодо діагоналі першого квадранта і збільшується в 2 рази;
4. прямокутний трикутник переміщується вниз на 40 і вправо на 200 пікселів із поворотом проти годинникової стрілки на 50°;
5. дві вертикальні паралельні лінії і зсунути вгору і вправо на 100 пікселів і повернути на 30º проти годинникої стрілки щодо початку локальних координат;
6. T − фігура дзеркально відображається на осі Y і повертається на 15° за годинниковою стрілкою;
7. |—| − подібна фігура зміщується на 120 і вліво на 140 пікселів з поворотом проти годинникової стрілки на 73°;
8. правильна трапеція відображається відносно діагоналі першого квадранта і зменшується в 2 рази;
9. ||- − подібна фігура повертається на 180° і збільшується в 1,3 рази;
10. зафарбований трикутник зміщується вліво на 230 і вниз на 50 пікселів з поворотом за годинниковою стрілкою на 49°;
11. -|—|- – подібна фігура дзеркально відображається відносно осі X з поворотом за годинниковою стрілкою на 36°;
12. зафарбований рівнобедренний трикутник зміщується вліво на 130 і вниз на 150 пікселів з поворотом за годинниковою стрілкою на 79°;
13. ± – подібна фігура зміщується вправо на 60 і вниз на 200 пікселів зі збільшенням в 2,3 рази;
14. ├ − подібна фігура дзеркально відображається відносно осі Y і зменшується в 1,4 рази;
15. X − подібна фігура повертається на 70° проти годинникової стрілки зі зміщенням вліво на 60 і вниз на 140 пікселів;
16. Δ – подібна фігура повертається на 180°, зміщуючись вліво і вниз на 100 пікселів;
17. H – подібна фігура дзеркально відображається відносно діагоналі першого квадранта зі збільшенням в 1.5 рази;

18. И – подібна фігура дзеркально відображається відносно осі X з поворотом за годинниковою стрілкою на 34°;

19. N – подібна фігура зміщується вверх на 90 і вліво на 120 пікселів із поворотом за годинниковою стрілкою на 56°;

20. ┴┴ – подібна фігура зміщується вправо на 200 і вниз на 70 пікселів із збільшенням в 1,8 рази;

21. ┌┐ – подібна фігура зміщується вниз на 120 і вліво на 140 пікселів із поворотом проти годинникової стрілки на 73°;

22. ┃├ – подібна фігура повертається на 180° і збільшується в 1,3 рази;

23) ┼┼ – подібна фігура зміщується вправо на 200 і вниз на 70 пікселів із збільшенням у 1,8 рази;

24. ┌┘ – подібна фігура дзеркально відображається відносно осі X з поворотом проти годинникової стрілки на 54°;

25. E – подібна фігура зміщується на 100 і вліво на 160 пікселів з поворотом проти годинникової стрілки на 23°;

26. П – подібна фігура зміщується вліво на 130 і вниз на 80 пікселів з поворотом за годинниковою стрілкою на 69°;

27. O – подібна фігура зміщується вверх на 90 і вліво на 120 пікселів із поворотом за годинниковою стрілкою на 26°;

28. L – подібна фігура зміщується вверх на 90 і вліво на 120 пікселів із поворотом за годинниковою стрілкою на 56°;

29. # – подібна фігура дзеркально відображається відносно осі У і зменшується в 1,4 рази;

30. ↑ – подібна фігура повертається на 180° і збільшується в 1,3 рази.

Зміст звіту

1) Зміст завдання;
2) лістинг програми;
3) надруковане зображення.

Контрольні питання

1. Які геометричні перетворення називаються афінними?
2. Які існують види геометричного перетворення?
3. Яким квадрантом прямокутної системи координат поданий екран дисплея по замовчуванню?
4. Для чого застосовується під час геометричної побудови процедура мови Паскаль GetAspectRatio?
5. Чи можна сумістити при геометричному перетворенні зсув і масштабування?

Лабораторна робота № 5

Тема. Динамічне перетворення двовимірних координат

Даний вид зображень знаходить широке застосування в комп'ютерній графіці для синтезу мнемосхем, моделювання технологічних процесів і переміщень роботів, розширення образотворчих можливостей людиномашинного інтерфейсу. Сам процес побудови базується на алгоритмах, розглянутих раніше, в попередніх лабораторних роботах, а методи перетворення координат детально представлені в попередній роботі.

Опис середовища програмування

Дані перетворення, як і в попередніх лабораторних роботах, реалізуються в середовищі мови програмування TurboPascal. Всі необхідні засоби мови розглянуті раніше. Реалізувати завдання на основі алгоритму із зміною графічних сторінок, що створює більш якісне зображення. Використання для регенерації рухомої частини зображення вікон або виділених прямокутних фрагментів не рекомендується, оскільки поворот зображення під час руху чи його масштабуванні ускладнюють визначення координат кутів ділянки екрану, що використовується.

Приклад програми. Створити програму, яка реалізує засобами графіки мови програмування Turbo Pascal динамічне зображення попередньої роботи, початкове зображення будується на початку першого квадранта, ліва нижня точка зображення має координати (20,10) і здійснюється її зсув до точки (-100, -110) зі збільшенням до кінця руху фігури в 2 рази.

Лістинг програми

```
program linii;
uses
Graph,crt;
var
GraphDriver:integer;
GraphMode:integer;
mx,my:word;
x11,x12,x21,x22,x31,x32,y11,y12,y21,y22,y31,y32:integer;
ax,ay,dx,dy:integer;
m, y:real;
ns,i:byte;
sx,sy, x0,y0:integer;
procedure XY(xn,yn:integer);
begin
SetLineStyle(0,0,1);
Line(xn,320,xn,10);
Line(xn,10,xn-5,20);
```

```
Line(xn,10,xn+5,20);
Line(40,yn,600,yn);
Line(600,yn,593,yn-3);
Line(600,yn,593,yn+3);
SetColor(4);
SetTextJustify(0,0);
SetTextStyle(0,0,1);
OutTextXY(597,yn+15,'X');
OutTextXY(xn-15,15,'Y');
end;
begin
GraphDriver:=3;
GraphMode:=1;
InitGraph(GraphDriver,GraphMode,'C:\BPZ\BGI');
SetBkColor(9);
GetAspectRatio(mx,my);
dx:=320;
dy:=125;  ax:=0;ay:=0;
y:=0; sy:=0;  ns:=0;
x11:=50;y11:=50;
x12:=80;y12:=50;
x21:=80;y21:=50;
x22:=80;y22:=150;
x31:=80;y31:=150;
x32:=110;y32:=150;
for sx:=0 to 20 do
begin
if ns=0 then
begin
SetActivePage(0);
SetVisualPage(1);
end
else     begin
SetActivePage(1);
SetVisualPage(0);
end;
ClearDevice;
XY(dx,dy);
SetColor(7);
SetLineStyle(0,0,3);
```

```
Line(dx+Round(my/mx*x11),dy-Round(mx/my*y11),dx+Round(my/mx*x12),dy-
Round(mx/my*y12));
Line(dx+Round(my/mx*x21),dy-Round(mx/my*y21),dx+Round(my/mx*x22),dy-
Round(mx/my*y22));
Line(dx+Round(my/mx*x31),dy-Round(mx/my*y31),dx+Round(my/mx*x32),dy-
Round(mx/my*y32));
SetColor(13);
m:=y;
Line(dx+Round(my/mx*(x11*m))-20*ax,dy-
Round(mx/my*y11*m)+15*ay,dx+Round(my/mx*(x12*m))-20*ax,
dy-Round(mx/my*(y12*m))+15*ay);
Line(dx+Round(my/mx*(x21*m))-20*ax,dy-
Round(mx/my*y21*m)+15*ay,dx+Round(my/mx*(x22*m))-20*ax,
dy-Round(mx/my*(y22*m))+15*ay);
Line(dx+Round(my/mx*(x31*m))-20*ax,dy-
Round(mx/my*y31*m)+15*ay,dx+Round(my/mx*(x32*m))-20*ax,
dy-Round(mx/my*(y32*m))+15*ay);
ax:=ax+1;ay:=ay+1;
y:=y+0.1;
sy:=sy+1;
if ns=0 then
ns:=1
else
ns:=0;
Delay(50000);
end;
readln;
CloseGraph;
end.
```

Варіанти завдань

Створити програму, що реалізовує засобами графіки мови програмування TurboPascal наступне динамічне зображення:

а) фігура для побудови визначається відповідно до завдань, наведених у попередній лабораторній роботі;

б) початкове зображення будується на початку першого квадранта. Ліва нижня точка зображення має координати (20,10);

в) зображення переміщується відповідно до наступних варіантів завдання:

1. зсув до точки (120, -30) з поворотом на кожному кроці на 1° за годинниковою стрілкою;
2. зсув до точки (-60, 60) із збільшенням до кінця руху фігури в два рази;

3. зсув до точки (100, 90) з поворотом проти годинникової стрілки на 2° на кожному кроці;

4. зсув до точки (-60, -30) з поворотом за годинниковою стрілкою на 3° на кожному кроці;

5. зсув до точки (100, -80) із зменшенням до кінця руху фігури в два рази;

6. зсув до точки (-80,60) з поворотом проти годинникової стрілки на кожному кроці на 1;

7. зсув до точки (140, 130) з поворотом за годинниковою стрілкою на кожному кроці на 2°;

8. зсув до точки (70, -80) зі збільшенням до кінця руху фігури в 1,5 рази;

9. зсув до точки (70, 80) з поворотом проти годинникової стрілки на кожному кроці на 3;

10. зсув до точки (-130, -120) зі зменшенням фігури до кінця руху в 2 рази;

11. зсув до точки (-60, 170) з поворотом за годинниковою стрілкою на кожному кроці на 1°;

12. зсув до точки (80, -110) з поворотом проти годинникової стрілки на кожному кроці на 2°;

13. зсув до точки (140, 130) з поворотом проти годинникової стрілки на кожному кроці на 3°;

14. зсув до точки (60, 90) з поворотом на кожному кроці на 1º проти годинникової стрілки;

15. зсув до точки (-200,120) зі зменшенням до кінця руху фігури в 2 рази;

16. зсув до точки (150, -120) з поворотом за годинниковою стрілкою на кожному кроці на 1°;

17. зсув до точки (-280, 160) з поворотом проти годинникової стрілки на кожному кроці на 2°;

18. зсув до точки (300, 150) з поворотом за годинниковою стрілкою на кожному кроці на 3°;

19. зсув до точки (-280, -140) з поворотом проти годинникової стрілки на кожному кроці на 1;

Зміст звіту

1) Зміст завдання;
2) лістинг програми;
3) надруковане зображення.

Контрольні питання

1. Чи можна отримати динамічне зображення на основі однієї графічної сторінки? Чи можна сумістити під час руху три види перетворень: зсув, поворот і масштабування? Чому рівна мінімальна дискрета переміщення зображення по осях X і Y? Чим визначається швидкість переміщення зображення по екрану? Які варіанти завдання допускають однократну побудову координатних осей?

Лабораторна робота № 6

Тема. Тривимірні перетворення координат

Більшість геометричних перетворень, з якими стикається інженер у практичній діяльності, реалізується в тривимірному просторі (рух робота із захопленням об'єкта, зображення об'ємної деталі і т.д.). Існує ряд нелінійних тривимірних систем відліку: циліндричні, сферичні та ін. Але найбільш часто застосовуються афінні перетворення, оскільки вони не змінюють форму тіл. Загальна структура рівнянь, що використовуються в процесі таких перетворень, така ж, як і за двох координат, але рівнянь три:

$$x_2=a_{xx}x_1+a_{xy}y_1+a_{xz}z_1+a_x,$$
$$y_2=a_{yx}x_1+a_{yy}y_1+a_{yz}z_1+a_y,$$
$$z_2=a_{zx}x_1+a_{zy}y_1+a_{zz}z_1+a_{z..}$$

При цьому 9-членна квадратна матриця використовується для реалізації повороту точки навколо осей, а 3-членний вектор здійснює її перенесення в тримірному просторі. На відміну від повороту на площині в просторі поворот задається послідовним обертанням навколо трьох координатних осей. При цьому для кожної осі існує своя матриця повороту. Поворот на кут 0 щодо осі 2 здійснюється матрицею:

$$R_z = \begin{vmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{vmatrix}.$$

Аналогічно виконується поворот на кут 0 щодо осі Y:

$$R_y = \begin{vmatrix} \cos\Theta & 0 & \sin\Theta \\ 0 & 1 & 0 \\ -\sin\Theta & 0 & \cos\Theta \end{vmatrix}.$$

Поворот на кут Θ щодо осі X проводиться за допомогою аналогічної матриці перетворення:

$$R_x = \begin{vmatrix} 1 & 0 & \sin\Theta \\ 0 & cos\Theta & -\sin\Theta \\ 0 & \sin\Theta & \cos\Theta \end{vmatrix}.$$

Масштабування має також свою матрицю, але в даній роботі обмежуємося тільки двома розглянутими вище видами перетворень як найважливішими.

Окремою і важливою проблемою, що виникає під час виконання даної лабораторної роботи, є відображення тривимірного простору на площину. З курсу інженерної графіки відомо, що найбільш поширені аксонометричні проекції які, у

свою чергу, діляться на триметрію (всі показники спотворення по осях різні), диметрію (два показники спотворення рівні, третій відрізняється від них) і ізометрію (всі три показники спотворення рівні). Конкретні види аксонометричних проекцій наведені на рис. 8.1. Для прикладу розглянемо відображення прямокутної ізометричної проекції на двовимірну систему координат, відповідну екрану дисплея (рис. 8.2). Центри систем відліку суміщені. Вісь у двовимірній системі співпадає з віссю 2 тримірного простору. З рис. 8.2 видно, що:

$$M_{x2} = M_{y3}*\cos30° - M_{x3}*\cos30° = \cos30°*(M_{y3} - M_{x3}) = 0.87*(M_{y3} - M_{x3}),$$
$$M_{y2} = M_{z3} – (M_{y3}*\sin30° + M_{x3}*\sin30°) = M_{z3} – 0.5*(M_{y3} + M_{x3}).$$

Відповідно до вимог ізометричної проекції всі тривимірні координати повинні домножуватися на коефіцієнт 0,82. Тому остаточні вирази для двовимірних координат мають вигляд:

$$M_{x2} = 0.82*0.87*(M_{y3} - M_{x3}) = 0.71*(M_{y3} - M_{x3}),$$
$$M_{y2} = 0.82*(M_{z3} – 0.5*(M_{y3} - M_{x3})).$$

Опис середовища програмування

Для спрощення процесу реалізації програми пропонується використовувати відеорежим з роздільною здатністю 640×480, що усуває необхідність в масштабуванні кінцевого зображення на екрані, дозволяючи зосередити основну увагу на процедурі перетворення тривимірних координат на площину.

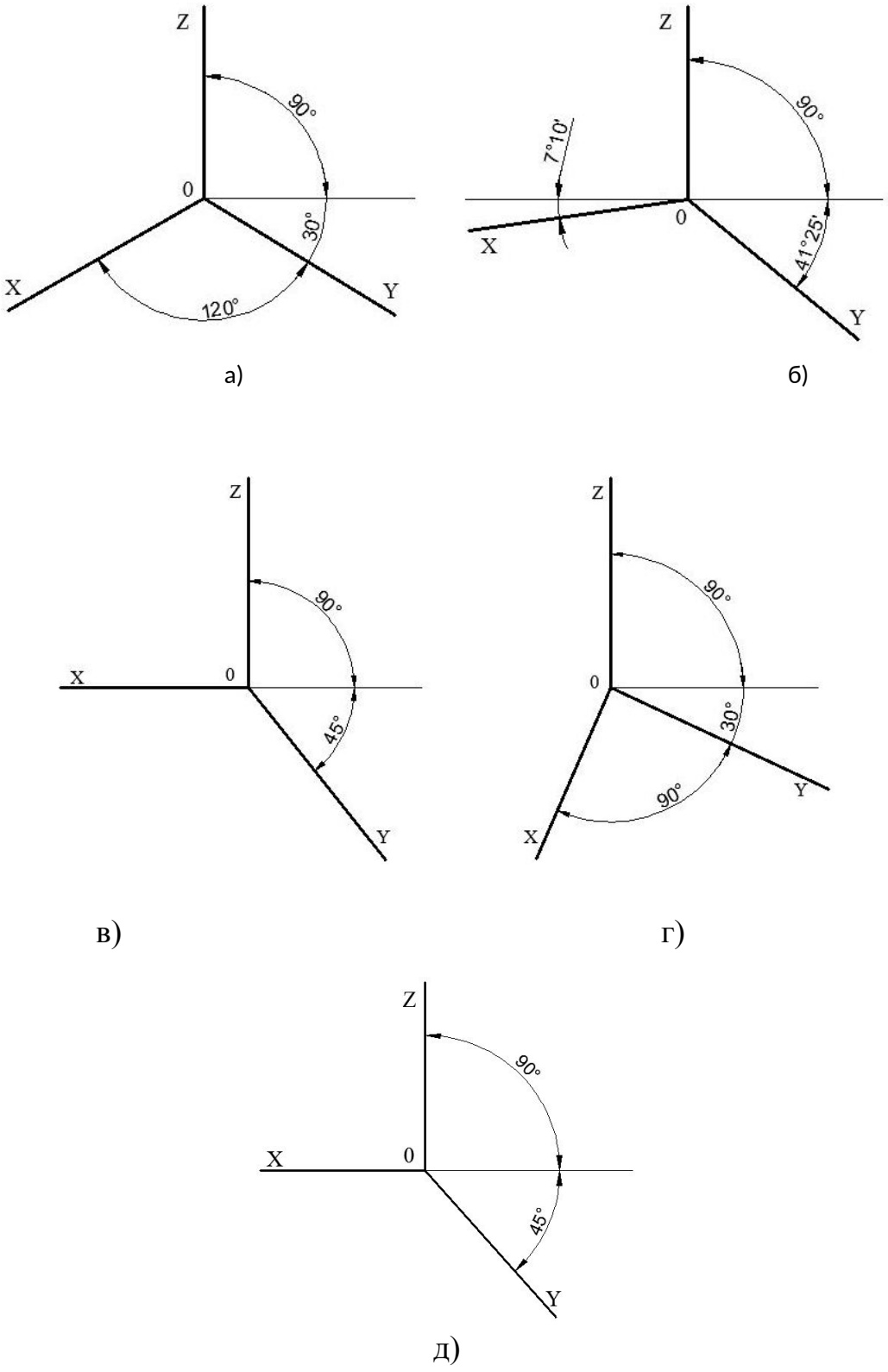


Рис. 8.1. Види аксонометричних проекцій:

а) прямокутна ізометрична проекція; U=W=V=0.82; б) прямокутна диметрична проекція; U=W=0.94, V=0.47; в) косокутна фронтальна ізометрична проекція U=W=V=1; г) косокутна горизонтальна ізометрична проекція U=W=V=1; д) косокутна фронтальна ізометрична проекція U=W=1, V=0.5; д) косокутна фронтальна симетрична проекція: U=W=1, V=0.5

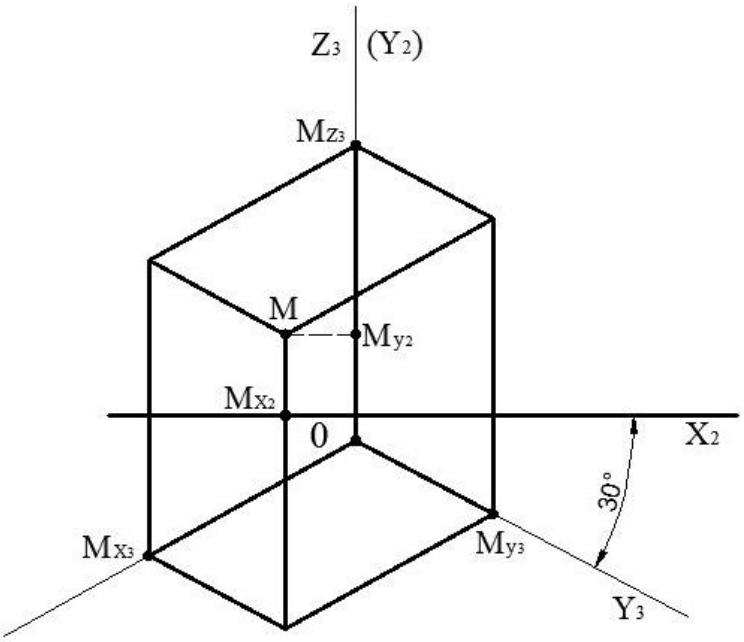


Рис. 8.2. Відображення прямокутної ізометричної проекції на двовимірну систему координат

Приклад програми. Створити програму, яка реалізовує засобами графіки мови програмування Turbo Pascal таке зображення: косокутна горизонтально ізометрія Т –подібна фігура початково розміщена в площині XY, а потім зсувається по осі X на 150 пікселів, а по осі Y – на 40 пікселів, повертаючись навколо осі Y на 20° проти годинникової стрілки, навколо осі X на 40° за годинниковою стрілкою.

Лістинг програми

```
program linii3;
uses
Graph,Crt;
var
GraphDriver:integer;
GraphMode:integer;
x11,x12,x21,x22,y11,y12,y21,y22,z11,z12,z21,z22:real;
x11y,x12y,x21y,x22y,y11y,y12y,y21y,y22y,z11y,z12y,z21y,z22y:real;
x11x,x12x,x21x,x22x,y11x,y12x,y21x,y22x,z11x,z12x,z21x,z22x:real;
x11m,x12m,x21m,x22m,y11m,y12m,y21m,y22m,z11m,z12m,z21m,z22m:real;
dx,dy:integer;
sy,sx:real;
ax,ay:real;
procedure XYZ(xn,yn:integer);
begin
SetLineStyle(0,0,1);
Line(xn,yn,xn,5);
```

```
Line(xn,5,xn-5,15);
Line(xn,5,xn+5,15);
Line(xn,yn,200,450);
Line(200,450,200,440);
Line(200,450,210,445);
Line(xn,yn,580,yn+Round((580-dx)*0.5774));
Line(580,yn+Round((580-dx)*0.5774),576,yn+Round((580-dx)*0.5774)-10);
Line(580,yn+Round((580-dx)*0.5774),569,yn+Round((580-dx)*0.5774));
SetColor(14);
SetTextJustify(0,0);
SetTextStyle(0,0,1);
OutTextXY(195,435,'X');
OutTextXY(590,yn+Round((580-dx)*0.5774),'Y');
OutTextXY(xn-15,15,'Z');
end;
procedure
Linii(x11p,x12p,y11p,y12p,z11p,z12p,x21p,x22p,y21p,y22p,z21p,z22p:real);
begin
SetLineStyle(0,0,3);
SetColor(8);
Line(dx+Round(0.5*(y11p-x11p)),dy+Round(-z11p+0.5*(x11p+y11p)),
dx+Round(0.5*(y12p-x12p)),
dy+Round(-z12p+0.5*(x12p+y12p)));
SetColor(10);
Line(dx+Round(0.5*(y21p-x21p)),
dy+Round(-z21p+0.5*(x21p+y21p)),
dx+Round(0.5*(y22p-x22p)),
dy+Round(-z22p+0.5*(x22p+y22p)));
end;
begin
GraphDriver:=Detect;
InitGraph(GraphDriver,GraphMode,'C:\BPZ\BGI');
SetBkColor(9);
ClearDevice;
dx:=320;
dy:=250;
XYZ(dx,dy);
x11:=40;
y11:=40;
z11:=0;
```

```
x12:=40;
y12:=140;
z12:=0;
x21:=40;
y21:=90;
z21:=0;
x22:=80;
y22:=90;
z22:=0;
Linii(x11,x12,y11,y12,z11,z12,x21,x22,y21,y22,z21,z22);
Readln;
ay:=20*3.14/180;
ax:=40*3.14/180;
sx:=150;
sy:=80;
X11y:=x11+z11*sin(ax);
Y11y:=y11*cos(ax)-sin(ax)*z11;
Z11y:=y11*sin(ax)+z11*cos(ax);
X12y:=x12+z12*sin(ax);
Y12y:=y12*cos(ax)-sin(ax)*z12;
Z12y:=y12*sin(ax)+z12*cos(ax);
X21y:=x21+z21*sin(ax);
Y21y:=y21*cos(ax)-sin(ax)*z21;
Z21y:=y21*sin(ax)+z21*cos(ax);
X22y:=x22+z22*sin(ax);
Y22y:=y22*cos(ax)-sin(ax)*z22;
Z22y:=y22*sin(ax)+z22*cos(ax);
Linii(x11y,x12y,y11y,y12y,z11y,z12y,x21y,x22y,y21y,y22y,z21y,z22y);
Readln;
x11x:=x11y*cos(ay)+z11y*sin(ay);
y11x:=y11y;
z11x:=-x11y*sin(ay)+z11y*cos(ay);
x12x:=x12y*cos(ay)+z12y*sin(ay);
y12x:=y12y;
z12x:=-x12y*sin(ay)+z12y*cos(ay);
x21x:=x21y*cos(ay)+z21y*sin(ay);
y21x:=y21y;
z21x:=-x21y*sin(ay)+z21y*cos(ay);
x22x:=x22y*cos(ay)+z22y*sin(ay);
y22x:=y22y;
```

```
z22x:=-x22y*sin(ay)+z22y*cos(ay);
Linii(x11x,x12x,y11x,y12x,z11x,z12x,x21x,x22x,y21x,y22x,z21x,z22x);
Readln;
x11m:=x11x+sx;
x12m:=x12x+sx;
x21m:=x21x+sx;
x22m:=x22x+sx;
y11m:=y11x+sy;
y12m:=y12x+sy;
y21m:=y21x+sy;
y22m:=y22x+sy;
z11m:=z11x;
z12m:=z12x;
z21m:=z21x;
z22m:=z22x;
Linii(x11m,x12m,y11m,y12m,z11m,z12m,x21m,x22m,y21m,y22m,z21m,z22m);
Readln;
CloseGraph;
end.
```

Варіанти завдань

Створити програму, яка реалізує засобами графіки мови програмування TurboPascal зображення, включаючи:

а) згідно з наведеними нижче по варіантах завданням проекцію тривимірних координат;

б) на початку координат початкове зображення;

в) перетворене у відповідності з завданням зображення:

1. прямокутна ізометрична проекція. ± − подібна фігура початково розміщена в площині XY, а потім зсувається по осі X на 70 пікселів, осі Y − на 40 пікселів і повертається навколо осі Z на 60°, осі X − на 120° за годинниковою стрілкою;

2. прямокутна симетрична проекція. L − подібна фігура початково розміщена в площині XZ, а потім зсувається по осі Y на 140 пікселів, по осі Z − на 70 пікселів і повертається навколо осі Y на 70° за годинниковою стрілкою, навколо осі X на 50° проти годинникової стрілки;

3. косокутна фронтальна ізометрична проекція ┌─ − подібна фігура початково розміщена в площині YZ, а потім зсувається по осі Y на 80 пікселів, а по осі Z − на 100 пікселів, повертаючись навколо осі Z на 90° за годинниковою, а навколо осі Y на 120° проти годинникової стрілки;

4. косокутна горизонтальна ізометрична проекція, < − подібна фігура початково розміщена в площині XY, а потім зсувається по осі X на 50 пікселів, а по

осі Y на 130 пікселів, повертаючись навколо осі Y на 80° проти годинникової стрілки, навколо осі X на 90° за годинниковою стрілкою;

5. прямокутна ізометрична проекція || – подібна фігура початково розміщена в площині XY, а поті повертається на 90º проти годинникової стрілки навколо осей X і Y, зсуваючись на 150 пікселів по осі X і на 100 пікселів по осі Y;

6. прямокутна ізометрична проекція. —| – подібна фігура початково розміщена в площині XY, а потім зсувається по осі X на 90 пікселів, осі Y − на 140 пікселів і повертається навколо осі Z на 80°, осі X − на 100° за годинниковою стрілкою;

7. прямокутна симетрична проекція. X − подібна фігура початково розміщена в площині XZ, а потім зсувається по осі Y − на 90 пікселів по осі Z − на 170 пікселів і повертається навколо осі Y на 45° за годинниковою стрілкою, навколо осі X на 70 проти годинникової стрілки;

8. косокутна фронтальна ізометрична проекція. |—| − подібна фігура початково розміщена в площині YZ, а потім зсувається по осі Y на 180 пікселів, по осі Z − на 70 пікселів, повертаючись навколо осі Z на 90° за годинниковою, а навколо осі Y на 110° проти годинникової стрілки;

9. косокутна горизонтальна ізометрична проекція. ||- – подібна фігура початково розміщена в площині XY, а потім зсувається по осі X на 150 пікселів, а по осі Y на − 130 пікселів, повертаючись навколо осі Y на 50° проти годинникової, навколо осі X на 40° за годинниковою стрілкою.

10. косокутна фронтальна симетрична проекція. II − подібна фігура початково розміщена в площині XZ, а потім зсувається по осі X на 90 пікселів, по осі Z − на 70 пікселів, повертаючись навколо осі Y на 30°, навколо осі Z на 110° проти годинникової стрілки;

11. прямокутна ізометрична проекція , ├ – подібна фігура початково розміщена в площині XY, а потім зсувається по осі X на 70 пікселів, по осі Y - на 800 пікселів і повертається навколо осі Z на 50°, осі X - на 100° за годинниковою стрілкою;

12. прямокутна симетрична проекція, ⌂-подібна фігура початково розмішена в площині XZ, а потім зсувається по осі Y на 60 пікселів, по осі Z − на 120 пікселів і повертається навколо осі Y на 90° за годинниковою стрілкою, навколо осі X на 80 пікселів проти годинникової стрілки;

13. косокутна фронтальна ізометрична проекція, Δ – подібна фігура початково розміщена в площині YZ, а потім зсувається по осі Y на 80 пікселів, по осі Z − на 120 пікселів, повертаючись навколо осі Z на 90° за годинниковою, а навколо осі Y на 120 пікселів проти годинникової стрілки;

14. косокутна горизонтальна ізометрична проекція, р – подібна фігура початково розміщена в площині XY, а потім зсувається по осі X на 150 пікселів, а по осі Y − на 40 пікселів, повертаючись навколо осі Y на 20 проти годинникової стрілки, навколо осі X на 40° за годинниковою стрілкою;

15. косокутна фронтальна симетрична проекція. Н – подібна фігура початково розміщена в площині XZ, а потім зсувається по осі X на 40 пікселів, по осі Z - на 140 пікселів, повертаючись навколо осі Y на 40°, навколо осі Z на 150° проти годинникової стрілки;

16. прямокутна ізометрична проекція. ╪ – подібна фігура початково розміщена в площині XY, а потім зсувається по осі X на 40 пікселів, осі Y − на 90 пікселів і повертається навколо осі Z на 50°, осі X − 100° за годинниковою стрілкою;

17. прямокутна симетрична проекція. И – подібна фігура початково розміщена в площині XZ, а потім зсувається по осі Y на 130 пікселів, по осі Z − на 70 пікселів і повертається навколо осі Y на 45° за годинниковою стрілкою, навколо осі X на 90° проти годинникової стрілки;

18. косокутна фронтальна ізометрична проекція. Х − подібна фігура початково розміщена в площині YZ, а потім зсувається по осі Y на 80 пікселів, по осі Z − на 120 пікселів, повертаючись навколо осі Z на 90° за годинниковою, а навколо осі Y на 120 пікселів проти годинникової стрілки;

19. косокутна горизонтальна ізометрична проекція. ◊ − подібна фігура початково розміщена в площині XY, а потім зсувається по осі X на 80 пікселів, а по осі Y на 110 пікселів, повертаючись навколо осі Y на 70° проти годинникової стрілки, навколо осі X на 70° за годинниковою стрілкою;

20. косокутна фронтальна ізометрична проекція. ≠ − подібна фігура початково розміщена в площині XZ, а потім зсувається по осі X на 80, по осі Z − на 50 пікселів, повертаючись навколо осі Y на 60°, навколо осі Z на 140° проти годинникової стрілки;

21. ┌┘ − подібна фігура початково розміщена в площині YZ, а потім зсувається по осі Y на 80 пікселів, по осі Z − на 120 пікселів, повертаючись навколо осі Z на 90° за годинниковою, а навколо осі У на 120 пікселів проти годинникової стрілки;

22. ┴┴ – подібна фігура початково розміщена в площині YZ, а потім зсувається по осі У на 80 пікселів, по осі Z − на 120 пікселів, повертаючись навколо осі Z на 90° за годинниковою, а навколо осі У на 120 пікселів проти годинникової стрілки;

23. ┌┐− подібна фігура початково розміщена в площині XY, а потім зсувається по осі X на 150 пікселів, а по осі Y на − 130 пікселів, повертаючись навколо осі Y на 50° проти годинникової, навколо осі X на 40° за годинниковою стрілкою;

24. # – подібна фігура початково розміщена в площині XY, а потім зсувається по осі X на 90 пікселів, осі Y − на 140 пікселів і повертається навколо осі Z на 80°, осі X - на 100° за годинниковою стрілкою;

25. |-| − подібна фігура початково розміщена в площині XY, а потім зсувається по осі X на 70 пікселів, осі Y − на 40 пікселів і повертається навколо осі Z на 60°, осі X - на 120° за годинниковою стрілкою;

26. |—|− – подібна фігура початково розміщена в площині XY, а потім зсувається по осі X на 70 пікселів, по осі Y - на 800 пікселів і повертається навколо осі Z на 50°, осі X - на 100° за годинниковою стрілкою;

27. ┃├ − подібна фігура початково розміщена в площині XZ, а потім зсувається по осі Y на 140 пікселів, по осі Z − на 70 пікселів і повертається навколо осі Y на 70° за годинниковою стрілкою, навколо осі X на 50° проти годинникової стрілки;

28. ↑ − подібна фігура початково розміщена в площині XZ, а потім зсувається по осі Y на 130 пікселів, по осі Z − на 70 пікселів і повертається навколо осі У на 45° за годинниковою стрілкою, навколо осі X на 90° проти годинникової стрілки;

29. ┃├ − подібна фігура початково розміщена в площині XZ, а потім зсувається по осі X на 40, по осі Z − на 70 пікселів, повертаючись навколо осі У на 60°, навколо осі Z на 100° проти годинникової стрілки;

30. _┃_┃__− подібна фігура початково розміщена в площині XZ, а потім зсувається по осі X на 80, по осі Z - на 50 пікселів, повертаючись навколо осі У на 60°, навколо осі Z на 140° проти годинникової стрілки.

Зміст звіту

1) Зміст завдання;
2) лістинг програми;
3) надруковане зображення.

Контрольні питання

1. Чи зміниться результат тривимірних перетворень, якщо змінити місцями зсув і поворот?
2. Які існують види тривимірних проекцій?
3. Чому проміжні координати перетворюючого зображення в програмі прикладу визначені як змінні типу real, а кінцеві − типу integer?
4. Яка роздільна здатність екрану не потребує введення масштабуючих коефіцієнтів?
5. Чому під час відображення тривимірної прямокутної системи координат на площині розрахунок координати X двомірної системи не враховує значення координати Z тривимірної системи?

Лабораторна робота № 7

Тема. Динамічне перетворення тривимірних координат

Задачі подібного вигляду виникають в процесі відображення на екрані комп'ютера результатів тривимірного моделювання динамічних переміщень технологічного устаткування (роботів, автоматичних ліній і т.п.). Математичний апарат таких перетворень був розглянутий в попередній лабораторній роботі, а методи відображення динамічного тривимірного зображення на екрані комп'ютера базуються на алгоритмах, запропонованих у попередніх лабораторних роботах. Тривимірне моделювання складної системи, що складається з декількох елементів, припускає два варіанти прорахунку тримірних координат: елементи рухаються незалежно один від одного − їх координати також розраховуються незалежно один від одного в тривимірній системі; елементи в процесі переміщення всієї системи здійснюють також відносні переміщення по відношенню один до одного (окремі ланки маніпулятора робота і т.д.) − даний випадок вимагає введення для деяких елементів модельованої системи локальних тривимірних координат із постійним прорахунком координат у даній системі і подальшим переведенням у координати глобальної тривимірної системи.

Опис середовища програмування

Як і в попередніх лабораторних роботах, що реалізовують динамічне зображення, пропонується використовувати алгоритм із зміною графічних сторінок. Оскільки динамічне тривимірне зображення, в загальному випадку, допускає зміну шести параметрів (три координати і три кути повороту), то при програмній реалізації вимагається чітко визначити параметр, що використовується як змінна циклу при перетворенні. Звичайно дане завдання вирішується на основі аналізу конкретного завдання на моделювання.

Приклад програми. Створити програму, яка реалізує засобами графіки мови програмування Turbo Pascal наступне динамічне зображення: проекція тривимірних координат відповідає варіанту, заданому в попередній лабораторній роботі; вид зображення і тип переміщення відповідає таким параметрам: лінія 1 (0,0,0;110,0,0); лінія 2 (110,0,0;110,90,0).

Лістинг програми

```
program s;
uses crt, graph;
var
driver, mode:integer;
h,dx,dy,j,y,x11,y11:integer;
mx,my:word;
a,i:real;
begin
```

```
driver:=0;
mode:=0;
x11:=100; y11:=1;
y:=0; j:=0;
dx:=170;dy:=265;
setcolor(white);
setlinestyle(0,0,3);
line (300,0,300,200);
line (300,0,197,15);
line (300,0,303,15);
line (300,100,100,300);
line (100,300,115,197);
line (100,300,111,189);
line (500,300,483,197);
line (500,300,487,189);
outtextxy(288,0,'z');
outtextxy(98,181,'y');
outtextxy(500,190,'x');
outtextxy(188,191,'0');
i:=0  ;
while i<6.1 do
begin
setcolor(red);
putpixel(170,265,red);
putpixel(dx+round(cos(i)*(x11+y)*sin(i)*(y11+y)-j),
dy-(round(mx/my*(sin(i)*(x11+y)+cos(i)*(y11+y))-j)), white);
setcolor(red);
line (170,265,dx+round(cos(i)*(x11+y)*sin(i)*(y11+y)-j),
dy-(round(mx/my*(sin(i)*(x11+y)+cos(i)*(y11+y))-j)));
setcolor(0);
delay (5000)  ;
line (170,265,dx+round(cos(i)*(x11+y)*sin(i)*(y11+y)-j),
dy-(round(mx/my*(sin(i)*(x11+y)+cos(i)*(y11+y))-j)));
setcolor(red);
line (170,265,300,200);
i:=i+0.1;
end  ;
readln;
end.
```

Варіанти завдань

Створити програму, що реалізовує засобами графіки мови програмування TurboPascal наступне динамічне зображення:

а) проекція тривимірних координат відповідає варіанту, заданому в попередній лабораторній роботі;

б) вид зображення і тип переміщення відповідає таким параметрам:

варіанти 1-5: лінія 1 (0, 0, 0; 0, 90, 0); лінія 2 (40, 0, 0; 40, 90, 0)

1. лінія 2 здійснює повний оберт навколо лінії 1 за годинниковою стрілкою. Все зображення при цьому переміщується уздовж осі Y на 180 пікселів;

2. лінія 1 − здійснює повний оберт навколо осі X проти годинникової стрілки, а лінія 2 в два рази повільніше обертається навколо осі X за годинниковою стрілкою. Все зображення переміщується вздовж осі X на 100 пікселів;

варіанти 16-20: лінія 1 (0, 0, 0; 110, 0, 0); лінія 2 (110, 0, 0; 110, 90, 0)

16. лінія 1 здійснює повний оберт навколо лінії 2 за годинниковою стрілкою. При цьому вся система зсувається вздовж осі X на 120 пікселів;

17. лінія 2 здійснює повний оберт навколо осі X проти годинникової стрілки. Вся система зсувається уздовж осі 2 на 180 пікселів;

18. вся система здійснює повний оберт навколо точки з'єднання за годинниковою стрілкою і переміщується уздовж осі Y на 90 пікселів;

19. лінії здійснюють повний оберт, обертаючись в протилежні сторони. При цьому вся система переміщується уздовж осі 2 на 120 пікселів;

20. лінія 1 здійснює повний оберт навколо осі Y за годинниковою стрілкою, а лінія 2 − повний оберт навколо осі X проти годинникової стрілки.

Зміст звіту

1) Зміст завдання;
2) лістинг програми;
3) надруковане зображення.

Контрольні питання

1. За яких умов виявляється некоректність графічного зображення моделюючої системи на екрані в даній лабораторній роботі?
2. Як виглядатиме алгоритм програми під час використання в даній лабораторній роботі тільки однієї графічної сторінки?
3. Чи можна використовувати як змінну циклу зсув по координатних осях?
4. Чи можна змоделювати переміщення по різних осях тримірної системи з різною швидкістю?

Чи можна в даній лабораторній роботі змоделювати зміну розмірів системи, що переміщується?

Лабораторна робота № 8

Тема. Контроль видимості елементів зображення

Аналіз результатів виконання попередньої лабораторної роботи показує, що в зображенні можуть виникнути некоректності, пов'язані з відображенням елементів, які повинні бути в глибині чи невидимі, поверх елементів, які знаходяться ближче до спостерігача. Причина некоректності − відсутність аналізу відносно розміщення елементів по об'єму простору і методів усунення невидимих ліній. Лінії відображались на екрані у відповідності з чітко заданою послідовністю в програмі, а не з їх природною послідовністю розміщення по глибині моделюючого простору.

Завдання контролю видимості (усунення невидимих ліній) виявилась для машинного вирішення досить складним і потребуючим підвищеної витрати машинного часу й оперативної пам'яті. Основна причина в тому, що в загальному випадку кожен із елементів змодельованого в тривимірному просторі об'єкта може бути закритий, під час спостереження, будь-яким іншим елементом. Тому під час збільшення кількості елементів кількість їх взаємних співставлень збільшується в квадраті. До теперішнього часу розроблений ряд методів і алгоритмів вирішення подібного завдання: алгоритм Галімберті і Монтанарі, алгоритм Варнока, алгоритм Уоткінса, алгоритм Ньюелла, Ньюелла і Санча, алгоритм плаваючого горизонту, алгоритм Роберта, алгоритм Вейлера-Азертона і ін. Для прикладу коротко розглянемо сутність алгорптма Уоткінса, а потім алгоритму Ньюелла, Ньюелла і Санча.

Алгоритм Уоткінса застосовується в терміналах із телевізійною розгорткою зображення. Відображаючий об'єкт подумки розсікається площинами, перпендикулярними поверхні екрану, паралельними одна одній і проходячими по стрічках розгортки. Кожна така площина визначає сукупність відрізків прямих, являючи собою результат перетину граней відображаючого об'єкта з даною площиною. Для формування зображення в кожній площині аналізується ця сукупність відрізків і визначається їх положення відносно користувача. Оскільки відрізки, розміщені спереду, закривають повністю чи частково відрізки, розміщені ззаду, то досить накреслити на лінії розгортки тільки частини відрізків, які видно спостерігачу.

Ідея алгоритму Ньелла, Ньюелла і Санча досить проста. Спочатку визначається порядок розміщення всіх граней зображаючого об'єкта по віддаленості від спостерігача. Далі всі грані почергово зображаються на екрані в порядку їх наближення до спостерігача, що приводить до природного відображення перекриттів граней.

Як зазначалося вище, контроль видимості є досить складним завданням і потребує великої обчислювальної потужності від комп'ютера. Вказані алгоритми реалізуються зазвичай спеціально розробленими графічними програмними пакетами чи спеціально створеними графічними мікропроцесорами, що обладнуються

спеціалізованими комп'ютерами, які називаються графічними станціями (ще одна їх відмінність від інших комп'ютерів − великий екран).

Опис середовища програмування

У міру складності вирішення завдання контролю видимості елементів обмежимося найпростішим варіантом, який передбачає, що користувач сам визначає умови коректного відображення моделюючого об'єкта на екрані і явно це фіксує в ревізуючій програмі.

Приклад програми. Аналіз прикладу, реалізованого в попередній лабораторній роботі, показує, що некоректність має місце під час співпадання на екрані зображень лінії 1 і 2. При цьому лінія 2 знаходиться далі по осі X, але зображається поверх лінії 1 згідно з фіксованим порядком алгоритму програми. Для ускладнення завдання задамо зсув всієї системи по осі X на 120 пікселів, що приводить і до перекриття віссю Y лінії 2 в процесі обертання. Усунути некоректність можна зміною порядку викреслювання ліній 1, 2 і вісі Y. Поки значення координати X лінії 2 позитивне зберігається порядок, встановлений в попередній лабораторній роботі, а при від'ємних значеннях лінія 2 відображається після лінії 1 і осі Y. Для зменшення лістингу програми вводиться процедура відображення кожної лінії і осі Y.

Лістинг програми

```
Program Lr9;
uses {підключення потрібних модулів}
Graph, Crt;
var
GraphDriver:integer; {номер графічного драйвера}
GraphMode:integer; {номер графічного підрежиму}
mx, my:word; {параметри масштабування зображення}
k:real; {коефіцієнт масштабування зображення}
x11, x12, y11, y12, z11, z12:integer; {координати лінії 1 в локальній системі координат}
x21, x22, y21, y22, z21, z22:integer; {початкові координати лінії 2 в локальній системі координат}
x21y, x22y, y21y, y22y, z21y, z22y:real; {координати лінії 2 в локальній системі координат після повороту навколо осі Y}
x11m, x12m:integer; {координати кінців лінії 1 по осі X після зсуву }
x21m, x22m:integer; {координати кінців лінії 2 по осі X після зсуву}
dx, dy:integer; {зміщення глобальної тривимірної системи координат відносно координатної системи екрана}
ay:real; {кут повороту лінії 2 відносно осі Y в радіанах}
i:integer; {змінна циклу}
sx:real; {зсув локальної тривимірної системи координат відносно глобальної}
ns:byte; {номер графічної сторінки}
```

```
procedure KY; {процедура викреслювання осі Y}
begin
SetColor(15); {установка кольору для зображення осі}
SetLineStyle(0, 0, 1); {установка товщини ліній для зображення координатної осі}
Line(320, 200, 580, 316);
Line(580, 316, 574, 308);
Line(580, 316, 569, 316);
end;
procedure L1; {процедура викреслювання лінії 1}
begin
SetLineStyle(0, 0, 3); {установка товщини ліній для зображення переміщуючихся елементів}
SetColor(8); {установка кольору лінії 1}
Line(dx+Round(0.71*(y11-x11m)),      dy+Round(k*0.82*(-z11+0.5*(x11m+y11))),
  dx+Round(0.71*(y12-x12m)), dy+Round(k*0.82*(-z12+0.5*(x12m+y12))));
end;
procedure L2; {процедура викреслювання лінії 2}
begin
SetLineStyle(0, 0, 3); {установка товщини ліній дня зображення переміщуючихся елементів}
SetColor(13); {установка кольору лінії 2}
Line(dx+Round(0.71*(y21y-x21m)),   dy+Round(k*0.82*(-z21y+0.5*(x21m+y21y))),
dx+Round(0.71 *(y22y-x22m)), dy+Round(k*0.82*(-22y+0.5*(x22m+y22y))));
end;
begin
GraphDriver:=3;
GraphMode:=l;
InitGraph(GraphDriver, GraphMode,''); {запуск графічного режиму}
GetAspectRatio(mx, my); {визначення параметрів масштабування}
k;=mx/my; {визначення коефіцієнта масштабування}
dx:=320; {задання положення глобальної тривимірної системи координат відносно координатної системи екрана}
dy:=200;
sx:=50; {задания початкового положення локальної тривимірної системи координат відносно глобальної}
ns;=0; {установка номера активної графічної сторінки}
SetTexJustify(0,0); {установка стилю вирівнювання тексту}
SetTextStyle(0, 0, 1); {установка стилю тексту}
x11:=0; {координати лінії 1 в локальній системі координат}
```

```
y11:=0;
z11:=0;
x12:=0;
y12:= 100;
z12:=0;
x21:=-50; {координати лінії 2 в локальній системі координат}
y21:=0;
z21:=0;
x22:=-50;
y22:=100;
z22:=0;
for i:=0 to 360 do {цикл по куту повороту лінії 2 навколо лінії 1}
begin
ay:=3.14*i/180; {переведення кута повороту в радіани}
if ns=0 then {зміна активної і видимої графічних сторінок}
begin
SetActivePage(0);
SetVisualPage(1)
end
else
begin
SetActivePage(1);
SetVisualPage(0)
end;
SetBkColor(9); {установка кольору фону зображення}
ClearDevice; {очищення активної сторінки}
SetColor(15); {установка кольору для зображення координатних осей}
SetLineStyle(0, 0, 1); {установка товщини ліній для зображення координатних
осей}
Line(320, 200, 320, 10); {відображення координатних осей X и Z}
Line(320, 10, 315, 20);
Line(320, 10, 325, 20);
Line(320, 200, 60, 316);
Line(60, 316, 66, 308);
Line(60, 316, 71, 316);
SetColor(4); {установка кольору для відображення назв осей}
OutTextXY(587, 316, 'Y'); {відображення назв осей}
OutTextXY(310, 10, 'Z');
OutTextXY(50, 316, 'X');
x11m:=Round(x11+sx); {координати лінії 1 по осі X після чергового зсуву}
```

```
x12m:=Round(x12+sx);
x21y:=x21*cos(ay)+z21*sin(ay); {розрахунок координат лінії 2 в локальній тривимірній системі після повороту навколо осі Y}
y21y:=y21;
z21y:=-x21 *sin(ay)+z21*cos(ay);
x22y:=x22*cos(ay)+z22*sin(ay);
y22y:=y22;
z22y:=-x22*sin(ay)+z22*cos(ay);
x21m:=Round(x21y+sx); {координати лінії 2 після чергового зсуву по осі X}
x22m:=Round(x22y+sx);
if x21m>0 then {порядок викреслювання елементів при позитивних значеннях координати X лінії 2}
begin
KY;
L1;
L2;
end
else
begin {порядок викреслювання елементів при від'ємних значеннях координати X лінії 2}
L2;
L1;
KY;
end;
if ns=0 then {зміна номера активної графічної сторінки}
ns:=1
else
ns:=0;
sx:=sx+0.333; {перетворення зсуву локальної тривимірної системи відносно глобальної по осі X}
Delay(100); {затримка зображення на кожному кроці переміщення}
End;
Readln; {зупинка зображення до натиснення будь-якої кнопки}
Closegraph; {вихід із графічного режиму}
end.
```

Варіанти завдань. Проаналізувати результат виконання попередньої лабораторної роботи на предмет некоректності зображення. За відсутності останньої потрібно змінити завдання так, щоб з'явилась потреба в контролі видимості елементів. Визначити умову(и) коректного зображення моделюючого об'єкта.

Зміст звіту

1) Зміст завдання;
2) лістинг програми:
3) надруковане зображення.

Контрольні питання

1. Чи можна розв'язати завдання, розглянуту як приклад, без застосування апарату процедур?
2. Який з розглянутих алгоритмів контролю видимості елементів: Уоткінса чи Ньюелла, Ньюелла і Санча, − більш ефективний (потребує менших обчислювальних затрат) під час виконання даної лабораторної роботи?
3. Яка з тривимірних проекцій потребує найменших обчислювальних затрат під час контролю видимості елементів?
4. Які комп'ютери називаються графічними робочими станціями?
5. За яким математичним законом збільшується кількість взаємних співставлень моделюючих елементів на екрані об'єкта?

Список рекомендованої літератури

1. Блінова Т.О., Порєв В.М. Комп’ютерна графіка / За ред. В.М. Горєва. – К.: Видавництво „Юніор”, 2004. – 456 с.
2. Меженный О.А. Tyrbo Pascal: учитесь программировать. – М.: Издательский дом „Вильямс”, 2001. – 448 с.
3. Програмування в середовищі Tyrbo Pascal 7.0 / Марченко А.И., Марченко Л.А.: Под ред. Тарасенко В.П. – К.: ВЕК+ , М.: ДЕСС, 1999. – 496 С.
4. Шищук В.В. Основи програмування на алгоритмічній мові Pascal: Навчальний посібник. – К.: Кондор, 2006. – 224 с.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Аппаратные средства IBM: Энциклопедия / Сост. М. Гук – СПб.: Питер, 2000. – 816 с.
2. Блінова Т.О., Порєв В.М. Комп'ютерна графіка / За ред. В.М. Горєва. – К.: Видавництво «Юніор», 2004.
3. Васильев В.Е. Компьютерная графика: Учебное пособие. – Санкт-Петербург, 2004. – 96 с.
4. Веселовська Г.В., Ходаков В.Є., Веселовський В.М. Комп'ютерна графіка: Навч. посібник для студентів вищих навчальних закладів / Під ред. В.Є. Ходакові. – Херсон: ОЛДІ-плюс, 2004. – С.112– 133.
5. Глушаков С.В., Крабе Г.А. Компьютерная графика. – Харьков: Фолио, 2002. – 500 с.
6. Горобець С.М. Основи комп'ютерної графіки: Навч.пос. /За ред. М.В.Лемківського. – К.: Центр навчальної літератури, 2006. – С.66– 9
7. Комп'ютерна графіка. Конспект лекцій для студентів спеціальності /6.092501/ «Автоматизоване управління технологічними процесами» денної та заочної форм навчання / Федік Л. Ю. – Луцьк: ЛДТУ, 2007. – С.25
8. Комп'ютерна графіка. Методичні вказівки до виконання лабораторних робіт з дисципліни «Комп'ютерна графіка» для студентів спеціальності /6092501/ «Автоматизоване управління технологічними процесами» денної та заочної форми навчання. Федік Л.Ю., Решетило О.М. – Луцьк: ЛНТУ, 2007. – 54 с.
9. Компьютерная графика: Учебник для ву зов / Под ред. Петров М.Н., Молочков В.П. – СПб.: Питер, 2003. – 736 с.
10. Компьютерная графика: Учебник для вузов / Под ред. Петров М.Н., Молочков В.П. – СПб.: Питер, 2003. – 736 с.
11. Костелов С.С., Решетило О.М., Смолянкін О.О. Программно-апаратний комплекс для підбору та приготування автомобільної фарби : Восточно-Европейский журнал передовых технологий. – Харків, 2009. – с 12-14.
12. Кукарин А.Б., Саган С.А., Шестаков В.Л. Термины по автоматизации производственных процессов. – К.: Высшая школа, 1992. – 156 с.
13. Л.Ю.Федік, Л.О.Гуменюк Застосування комп'ютерної графіки та її класифікація // Наукові нотатки. Міжвузівський збірник (за напрямом «Інженерна механіка»), Випуск 20 (травень, 2007). – Луцьк, 2007. – С. 519– 521
14. Мартиневич М.С., Федік Л.Ю. Деякі пакети для створення тривимірної комп'ютерної графіки // Студентський науковий вісник. Серія «Технічні науки». Науковий збірник. Випуск 3. – Луцьк: РВВ ЛНТУ, 2011. – с. 111– 115.
15. Мартиневич М.С., Федік Л.Ю. Найбільш поширені формати 3D графіки // Студентський науковий вісник. Серія «Технічні науки». Науковий збірник. Випуск 3. – Луцьк: РВВ ЛНТУ, 2011. – с. 105– 111.
16. Мельниченко В.В., Легейда В.В. Настоящий САМОУЧИТЕЛЬ компьютерной графики. – К.: Век+, СПб.: КОРОНА принт, К.: НТИ, 2005. – 560.
17. Методичні вказівки до виконання самостійної роботи з дисципліни «Комп'ютерна графіка» для студентів спеціальності /6092501/ «Автоматизоване

управління технологічними процесами» денної форми навчання. Федік Л.Ю. – Луцьк: ЛНТУ, 2009. – 40 с.

18. Методичні вказівки до контрольних робіт з дисципліни «Комп'ютерна графіка» для студентів спеціальності /6092501/ «Автоматизоване управління технологічними процесами» заочної форми навчання. Федік Л.Ю. – Луцьк: ЛНТУ, 2010. – 16 с.

19. Основы компьютерной графики. Часть 1: Математический аппарат компьютерной графики (электронная версия) А.В. Казанцев. – Казань, 2001. – 62 с.

20. Петров М.Н., Молочков В.П. Компьютерная графика: Учебник для вузов. 2–е изд. – СПб.: Питер, 2005.

21. Пономаренко С.И. Пиксел и вектор. Принципы цифровой графики.– СПб.: БХВ-Петербург, 2002. – 496 с.

22. Ренди Дж. Рост. OpenGl. Трехмерная графика и язык программирования шейдеров. Для профессионалов. – Питер, 2005

23. Роджерс Д. Алгоритмические основы машинной графики: Пер. с анг. – М.: Мир, 1989. – 512 с.

24. Сучасний словник іншомовних слів: Близько 20 тис. слів і словосполучень / Уклали: О.І. Скопненко, Т.В. Цимбалюк. – К.: Довіра, 2006. – 789 с. – (Словники України).

25. Федік Л.Ю. Аналіз деяких колірних моделей комп'ютерної графіки // Тези XXV-ої науково-технічної конференції професорсько-викладацького складу «Актуальні проблеми та перспективи науки і виробництва. – Луцьк, Навчально-науковий відділ ЛНТУ, 2010 – с. 4– 5.

26. Федік Л.Ю. Аналіз колірних моделей HSV/HSB, HSL, Lab // Науковий журнал «Комп'ютерно-інтегровані технології. Освіта, наука, виробництво». – Луцьк: Видавництво ЛНТУ, 2010 – №2. – с. 103– 107.

27. Федік Л.Ю. Аналіз колірних моделей RGB, CMY, CMYK // Міжвузівський збірник (за галузями знань «Машинобудування та металообробка», «Інженерна механіка», «Металургія та металообробка», «Металургія та матеріалознавство») Наукові нотатки. – Луцьк, 2010. Вип. 29 (жовтень, 2010) – с. 217– 221.

28. Федік Л.Ю. Деякі види форматів графічних файлів // Наукові нотатки. ЛНТУ. Міжвузівський збірник (за напрямом «Інженерна механіка»). Випуск 25, частина II (червень, 2009) – Луцьк, – с. 286– 290.

29. Федік Л.Ю. Найпоширені редактори растрової графіки // Наукові нотатки. Міжвузівський збірник за напрямком «Інженерна механіка». Випуск 20 (2) (жовтень, 2007)– Луцьк, с. 233– 238.

30. Федік Л.Ю. Основні формати графічних даних у мережі World Wide Web // Наукові нотатки. Міжвузівський збірник (за напрямом «Інженерна механіка»). – Луцьк, 2008 – с. 306– 315.

31. Федік Л.Ю. Растрові формати графічних файлів // Наукові нотатки. ЛНТУ. Міжвузівський збірник (за напрямом «Інженерна механіка»). Випуск 25, частина I (червень, 2009) – Луцьк, – с. 378– 385.

32. Федік Л.Ю. Редактори векторної графіки // Наукові нотатки. ЛНТУ. Міжвузівський збірник (за напрямом «Інженерна механіка»). Випуск 24, (березень, 2008) – Луцьк, – с. 306– 315.
33. Федік Л.Ю. Формати графічних даних у мережі World Wide Web // Тези XXII-ої науково-технічної конференції професорсько-викладацького складу (технічний напрямок). – Луцьк, Навчально-науковий відділ ЛДТУ, 2007 – с. 28–30.
34. Федік Л.Ю., Мартиневич М.С. Редактори 3-D графіки // Тези XXXIV-ої університетської студентської науково-технічної конференції «Україна сьогодні, інтеграція освіти і науки (технічний напрямок. – Луцьк, 2011 – с. 35–36.
35. Федік Л.Ю., Сус А.Р. Найбільш поширені редактори фрактальної графіки // Студентський науковий вісник. Серія «Технічні науки». Науковий збірник. Випуск 3. – Луцьк: РВВ ЛНТУ, 2011. – с. 172– 177.
36. Шушан Р., Райт Д., Льюис Л. Дизайн и компьютер – М.: Издательский отдел «Русская Редакция», ТОО «Channel Trading Ltd», 1997. – 544 с.
37. http://allsoft.ru/program_page.php?grp=27286
38. http://bezobeda.net
39. http://sergeyk.kiev.ua/conspect/comp-ech/AdvancedGrapher.shtml
40. http://wiki.auditory.ru
41. http://www.geol.univ.kiev.ua/lib/zhukov_n_n/Tema_10_Surfer.pdf
42. http://www.osp.ru/os/1996/02/178854/