МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНЙ ЗАКЛАД

ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Федоров О. В.

Комп'ютерна графіка

КУРС ЛЕКЦІЙ

Донецьк – 2009

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

КОМП'ЮТЕРНА ГРАФІКА

КУРС ЛЕКЦІЙ

(для студентів інженерних спеціальностей)

Р о з г л я н у т о
на засіданні кафедри
«Енергомеханічні системи».

Протокол № 3 від 07.10.2009 р.

З а т в е р д ж е н о
на засіданні навчально-методичної
ради ДонНТУ.

Протокол № __ від __.__.2009 р.

Донецьк – 2009

УДК 681.3.06

Федоров О. В. Комп’ютерна графіка. Конспект лекцій (для студентів інженерних спеціальностей) — Донецьк: ДонНТУ, 2009. – 112 с.

Викладено основні поняття, принципи і методи комп’ютерної графіки. Висвітлено питання передачі кольору, кодування й обробки растрових зображень. Приведено основні методи й алгоритми двовимірної графіки. Розглянуто основи тривимірного моделювання і способи візуалізації тривимірних об’єктів.

Укладач: О. В. Федоров

Зміст

ВСТУП

Здатність людини створювати художні образи завжди вважалася однією з основних і дивних рис, що відрізняють її від тварин. Перші зображення, намальовані на скелях вохрою, вугіллям, глиною, з'явилися десятки тисяч років тому. Згодом техніка створення зображень мінялася, але незмінною залишалася тяга людини до творчості, той подив і замилування, що викликали в сучасників і нащадків витвори майстрів.

Створюючи прекрасне, людина експериментував з матеріалами. Що тільки не йшло в хід — полотно і фарби, папір і грифель, глина, мармур, бронза. Стрімкий технічний прогрес останнього століття не пройшов для мистецтва непоміченим. З'явилися нові способи і технології створення прекрасного, від гігантських скульптур зі сталевим каркасом і алюмінієвим обшиванням до голографічних зображень, отриманих за допомогою лазера.

Електронні обчислювальні машини, які вперше з'явилися в 40-х та отримали поширення в 50-х роках, здавалося б, не мали до мистецтва ніякого відношення. Скоріше навпаки, їхня суха математика і тверда логіка вважалися чимось, абсолютно протилежним художній творчості. До того ж, перші ЕОМ не мали ніяких засобів для створення зображень.

Проте, з розвитком комп'ютерної техніки, збільшенням продуктивності й обсягів пам'яті, обчислювальні машини стали залучати до рішення задач, близьких до художніх. За допомогою комп'ютерів стали проектувати корпуси автомобілів і літаків, будинки — з'явилися перші системи автоматизованого проектування. Їхня поява поставила перед інженерами і програмістами нову задачу — навчити комп'ютер створювати зображення, зробити їх найбільш схожими на дійсні, реалістичними. Для цього, по-перше, потрібно було створити пристрої, здатні відображати такі картинки — кольорові монітори і принтери з високою якістю печатки, а по-друге, розробити методи й алгоритми, які дозволяють створювати ці зображення.

Успіх у цій області був досягнутий не одразу. Потрібно було п'ять десятиліть пошуків, досліджень і розробок, щоб ми навчилися створювати художні фільми, у яких не знімався жоден живий актор, подорожувати по вигаданих світах комп'ютерних ігор, проектувати літаки і підвідні човни, жодного разу не доторкнувшись до олівця. Зараз те, що ми називаємо ***комп'ютерною графікою***, оточує нас усюди. У телепередачах і художніх фільмах, рекламних плакатах, книгах журналах і газетах — скрізь ми бачимо зображення і відео, створені за допомогою комп'ютера. Але мало хто уявляє собі, наскільки складні алгоритми сховані за цими «картинками», які обсяги обчислень необхідно виконати, щоб їх створити. Далі ми розглянемо ті ідеї, методи й алгоритми, що покладені в основу комп'ютерної графіки.

1 ОСНОВНІ ПОНЯТТЯ. ВІЗУАЛІЗАЦІЯ. ГРАФІЧНІ ПРИСТРОЇ

1.1 Сутність, види і задачі комп'ютерної графіки

Комп'ютерна графіка включає візуалізацію, обробку та розпізнавання зображень за допомогою ЕОМ.

Візуалізація — створення зображення. Візуалізація виконується, виходячи з опису (моделі) того, що потрібно відображати. Існує багато методів і алгоритмів візуалізації, що розрізняються між собою в залежності від того, що і як відображати. Наприклад, відображення того, що може бути тільки в уяві людини — графіки, діаграми, схеми, карти. Чи навпаки, імітація тривимірної реальності — зображення сцен у комп'ютерних іграх, художніх фільмах, тренажерах, у системах архітектурного проектування.

Обробка це перетворення готових зображень. Прикладами обробки зображень можуть служити: підвищення контрасту, чіткості, корекція кольорів, згладжування, усунення дефектів, ретушування, реставрація фотознімків, створення колажів і т. д. Як матеріал для обробки можуть використовуватися скановані зображення, цифрові фотографії, кінокадри, штучно синтезовані зображення.

Для ***розпізнавання*** зображень основна задача — одержання опису об'єктів, представлених зображенням (звичайно растровим). Методи й алгоритми розпізнавання розроблялися насамперед для забезпечення зору роботів і для систем спеціального призначення. Але останнім часом комп'ютерні системи розпізнавання зображень усі частіше з'являються в повсякденній практиці багатьох людей, наприклад, офісні системи *розпізнавання текстів*, програми *векторизації*.

Призначення комп'ютерної графіки дуже різноманітно. Це:

— ***художня графіка*** (оформлення видань, створення рекламних проспектів, інтернет-сайтів і т.д.);

— ***інженерна графіка*** (створення креслень і 3D-моделей);

— ***кінематографічна графіка*** (комп'ютерні спецефекти у фільмах, комп'ютерні мультфільми, відеоролики і заставки);

— ***інтерактивна анімована графіка*** (2D – і 3D – комп'ютерні ігри, тренажери і симулятори);

— ***графіка систем керування*** (відображення інформації на дисплеях диспетчерів, операторів, пультах керування машин, верстатів і т.д., графічні операційні системи й інтерактивні програми);

Досить популярним донедавна було словосполучення ***інтерактивна комп'ютерна графіка***, яке позначало здатність комп'ютерної системи створювати графіку в режимі нормального часу і вести діалог з людиною. Зараз будь-яку програму можна вважати інтерактивною системою комп'ютерної графіки.

Історично першими інтерактивними системами вважаються ***системи автоматизованого проектування*** (САПР — англійська абревіатура CAD — Computer Aided Design), що з'явилися в 60-х роках. Вони являють собою значний етап в еволюції комп'ютерів і програмного забезпечення. У системі інтерактивної комп'ютерної графіки користувач сприймає на дисплеї зображення, що представляє деякий складний об'єкт, і може вносити зміни в опис (модель) об'єкта. Такими змінами можуть бути як введення і редагування окремих елементів, так і завдання числових значень для будь-яких параметрів, а також інші операції по введенню інформації на основі сприйняття зображень.

Системи автоматизованого проектування активно використовуються в багатьох областях, наприклад, у машинобудуванні й електроніці. Одними з перших були створені САПР для проектування літаків, автомобілів, об'єктів архітектури, розробки мікроелектронних інтегральних схем і т.п. Такі системи спочатку функціонували на великих комп'ютерах і були доступні лише великим фірмам і проектним організаціям. Потім одержали поширення швидкодіючі комп'ютери середнього класу з розвитими графічними можливостями — графічні робочі станції. Зі зростанням потужностей персональних комп'ютерів усі частіше САПР почали використовувати на дешевих масових комп'ютерах, які зараз мають достатню швидкодію й обсяги пам'яті для рішення багатьох задач. Це привело до значного поширення систем САПР. Назвемо декілька популярних САПР: багатоцільова система для виконання проектних робіт у різних областях — *AutoCAD* і його російський аналог — система *KOMPAS-3D*, для архітекторів — *ArchiCAD*, для проектування електронних схем — *ORCAD, PCAD* і т.д.

Широко використовуються графічні системи в дизайні, рекламі. Крім традиційних ***2D-редакторів*** — *Adobe Photoshop, CorelDraw* і подібних їм — використовуються і ***3D-пакети***. Анімаційні ролики створюються засобами систем моделювання тривимірних сцен, таких як *Maya, 3D Studio Max, Light Wave, Bryce 3D* та інших. Один з напрямків досліджень і розробок для сучасної комп'ютерної графіки — анімація руху людей і тварин, вивчення міміки.

Важливим етапом розвитку систем комп'ютерної графіки стали так називані ***системи віртуальної реальності*** (*virtual reality*). Нарощування потужності комп'ютерів, підвищення реалістичності тривимірної графіки й удосконалення способів діалогу людини з комп'ютером дозволяють створювати ілюзію входження людини у віртуальний простір. Цей простір може бути моделлю існуючого простору або вигаданим. Спочатку такі системи використовувалися як тренажери для пілотів, однак найбільше поширення одержали різноманітні комп'ютерні ігри-3D. Слід зазначити, що в багатьох комп'ютерних іграх реалізовані ідеї і методи, що раніше були втілені в професійних дорогих системах.

Широко використовується комп'ютерна графіка в кіно. Донедавна технології комп'ютерної графіки застосовувалися для спецефектів, створення зображень екзотичних чудовиськ, імітації стихійних лих і інших елементів, що

є лише фоном для гри живих акторів. У 2001 році вийшов на екрани повнометражний кінофільм «Фінальна фантазія», у якому все, включаючи зображення людей, синтезовано комп'ютером — живі актори тільки озвучили ролі за кадром.

Як наука, комп'ютерна графіка містить у собі ***розділи***:

1. Способи кодування і відображення графічної інформації (растрова і векторна графіка, способи передачі кольору).

2. Алгоритми растрової двовимірної графіки (відображення відрізків, кривих, літер, заповнення кольором, обробка растрових зображень).

3. Алгоритми векторної двовимірної графіки (створення, відображення і редагування векторних зображень).

4. Алгоритми тривимірної графіки (створення і відображення 3D-моделей, візуалізація, відсікання невидимих частин, створення реалістичних зображень).

5. Способи анімації дво- і тривимірних зображень.

1.2 Способи візуалізації

Найбільш відомі два способи візуалізації: ***растровий*** і ***векторний***.

Растрова візуалізація ґрунтується на представленні зображення на екрані чи папері у виді сукупності окремих точок (пікселів). Разом пікселі утворюють растр.

Векторна візуалізація — це створення зображення на екрані або папері за допомогою малювання суцільних ліній (векторів) — прямих чи кривих.

Якість векторної візуалізації обумовлюється точністю виведення і номенклатурою ***базових графічних примітивів*** — ліній, дуг, еліпсів і інших. При цьому в пам'яті комп'ютера зберігається інформація про зображення у виді набору даних про ці примітиви — їхню форму, кольори, розміри, взаємне розташування і порядок відображення. Для виведення векторного зображення на екран чи друкувальний пристрій система щораз заново «малює» зображення по цьому наборі даних.

До *переваг* векторного способу візуалізації відносяться:

— можливість *редагувати* зображення шляхом додавання нових примітивів, їхнього видалення, а також зміни параметрів вже існуючих примітивів; при цьому інша графічна інформація залишається незмінною;

— можливість *масштабувати* зображення (збільшувати або зменшувати) без втрати якості, рис. 1.1 *а*;

— порівняно невеликі обсяги пам'яті, займані інформацією про зображення.

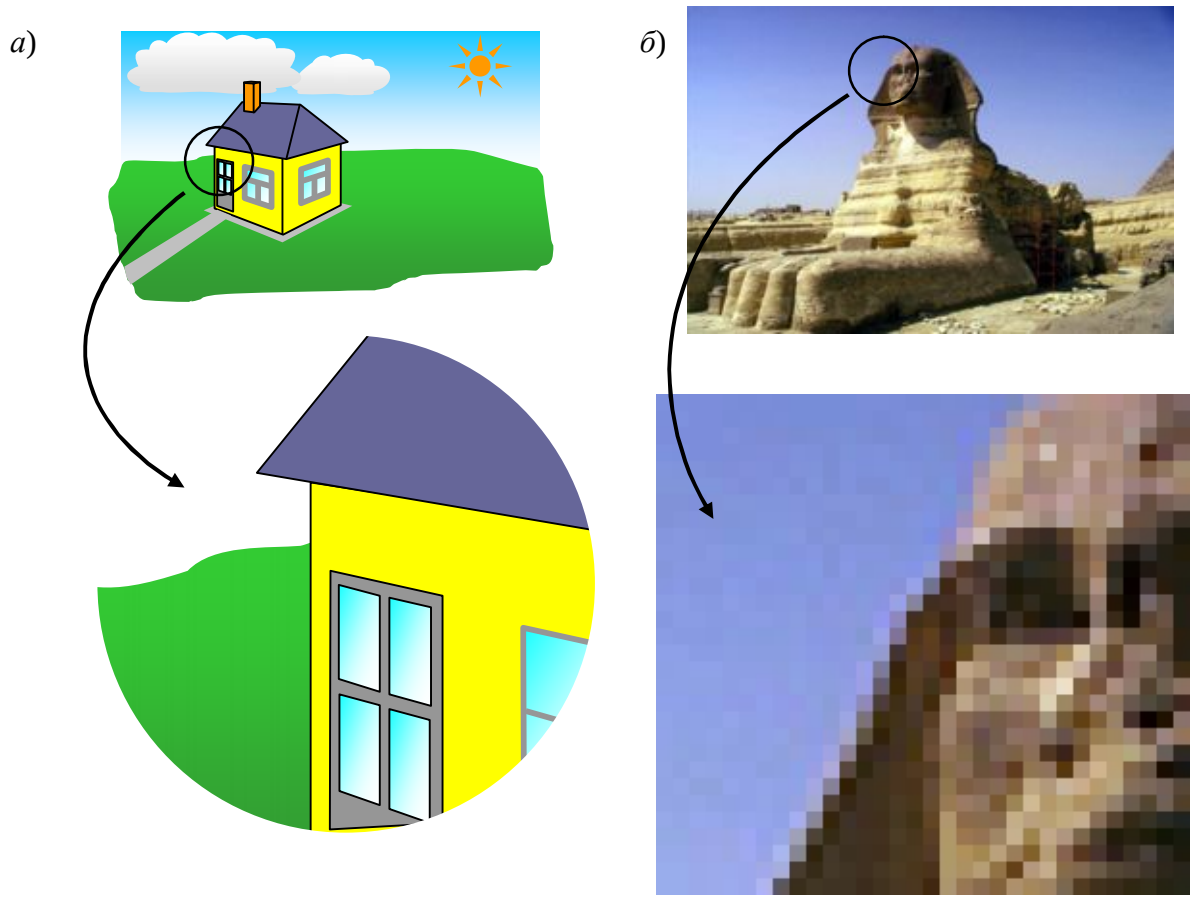


Рисунок 1.1 – Приклади векторного (*а*) і растрового (*б*) способів візуалізації

Недоліками векторного способу є:

— відносно мала швидкість виводу зображень на растрові пристрої відображення, що пов’язано з необхідністю виконання великих обсягів обчислень при ***растеризації*** векторного зображення (перетворенні його в растр);

— обмежені можливості у відображенні — векторний спосіб візуалізації гарний для схем, креслень, малюнків і інших «штучних» об’єктів, але практично непридатний для збереження реалістичних зображень, наприклад, фотографій;

— оскільки інформація про склад зображення закодована, зображення може бути розшифровано тільки пристроєм або програмою, що «розуміють» цей код; це викликає проблеми сумісності при переносі векторних зображень в інші програмні пакети.

В даний час домінує растровий спосіб візуалізації. Це обумовлено більшим поширенням растрових дисплеїв і принтерів. Растрове зображення значне швидше, ніж векторне, виводиться на дисплей, оскільки для цього не потрібно виконувати складні обчислення. Растровий спосіб візуалізації зручний

для збереження складних зображень, у тому числі і багатобарвних фотореалістичних, рис. 1.1 *б*, але має істотні недоліки:

— дискретність зображення (розкладання на окремі точки) погіршує його якість і обмежує можливість масштабування, див. рис. 1.1 *б*;

— растрові зображення значно складніше редагувати, оскільки в пам'яті комп'ютера зберігається інформація про окремі точки (пікселі), а не про об'єкти, які ми хочемо змінити;

— растрові зображення (особливо повнокольорові) займають великі обсяги пам'яті.

У сучасних комп'ютерних системах одержала поширення змішана технологія, при якій на растрових пристроях відображення (дисплеях) реалізується «віртуальна» векторна графіка. Користувач створює і редагує векторне зображення, яке виводиться на екран растрового дисплея за допомогою алгоритмів растеризації. Прикладами можуть бути графічний редактор *CorelDraw*, убудований графічний редактор *Microsoft Word*, а також CAD-пакети: *AutoCAD, KOMPAS-3D* і ін.

1.3 Типи графічних пристроїв

Існує багато різноманітних **пристроїв для виводу зображень**, побудованих за допомогою машинної графіки:

— пір'яні графобудівники;

— точечно-матричні, струминні, лазерні, термічні та ін. друкувальні пристрої;

— векторні дисплеї на запам'ятовуючій трубці або з регенерацією зображення;

— растрові дисплеї на електронно-променевій трубці;

— рідкокристалічні дисплеї та ін.

Друкувальні пристрої використовуються для виведення на папір результатів побудови або обробки зображень. Для безпосередньої роботи з зображенням використовуються різні пристрої на електронно-променевих трубках і рідкокристалічні дисплеї (РК-дисплеї).

Найбільш часто як пристрій відображення виступає ***відеомонітор***, основу якого складає ***електронно-променева трубка*** (***ЕПТ***).

На рис. 1.2 схематично показана ***ЕПТ***, використовувана у відеомоніторах. ***Катод*** (негативно заряджений) нагрівають доти, поки збуджені електрони не створять хмару, що розширюється, (електрони відштовхуються друг від друга, оскільки мають однаковий заряд). Ці електрони притягаються до сильно зарядженого позитивного ***анода***, для чого між катодом і анодом створюється різниця потенціалів у кілька десятків кіловольтів.

На внутрішню сторону розширеного кінця ***ЕПТ*** нанесений ***люмінофор*** — речовина, здатна випускати видиме світло при опроміненні пучком електронів.

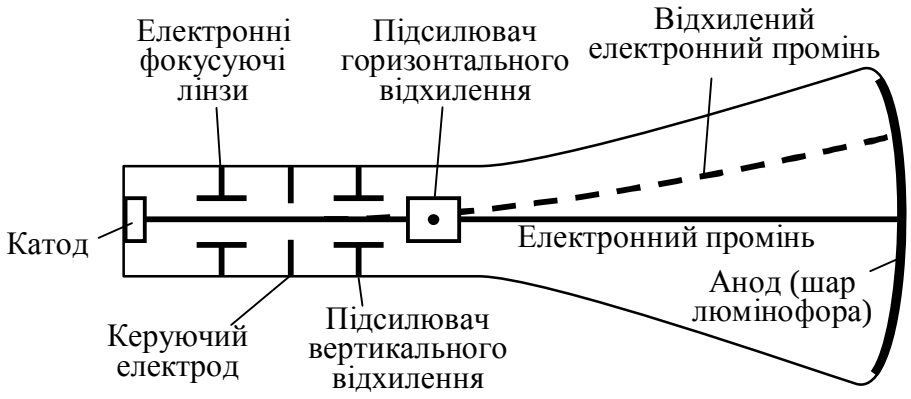


Рисунок 1.2 – Схема будови електронно-променевої трубки

Хмара електронів за допомогою ***електронних лінз*** фокусується у вузький пучок, який дає яскраву пляму в центрі ЕПТ. Промінь відхиляється уліво чи вправо, вище або нижче центра за допомогою ***підсилювачів горизонтального і вертикального відхилення***. Переміщаючись по екрані, промінь залишає слід — лінію.

Для відхилення променя можуть використовуватися як ***електроди***, на які подається напруга (ЕПТ осцилографів), так і ***електромагнітні котушки*** (ЕПТ телевізорів). У другому випадку довжина ЕПТ значно зменшується, але ускладнюється система керування променем.

Для регулювання яскравості лінії використовується керуючий електрод, на який подається негативний заряд. Чим вище напруга на керуючому електроді, тим менше інтенсивність пучка електронів і яскравість світлової плями.

У кольоровій ЕПТ знаходяться ***три електронні пушки***, по одній на кожен основний колір: ***червоний***, ***зелений*** і ***синій***. Електронні пушки об'єднані в блок, що відповідає подібному блоку точок червоного, зеленого і синього люмінофорів на екрані ЕПТ, рис. 1.3.

Для того, щоб електронні пушки збуджували тільки відповідні їм точки люмінофора (наприклад, червона пушка збуджувала тільки точку червоного люмінофора), між електронними пушками і поверхнею екрана поміщена перфорована металева сітка — ***тіньова маска***.

Колірні пушки розташовані таким чином, що їхні промені сходяться і перетинаються в площині тіньової маски, рис. 1.4. Після проходження через отвір червоний промінь, наприклад, захищений від влучення на зелену чи синю точки люмінофора. Він може потрапити лише на червону точку. Змінюючи інтенсивність електронного променя для кожного основного кольору, можна одержати різні відтінки. Комбінація цих відтінків дає велику кількість кольорів для кожної точки.

З усіх дисплеїв на ЕПТ найпростіше побудований дисплей на запам'ятовуючій ЕПТ із прямим копіюванням зображення. ***Запам'ятовуюча ЕПТ*** це ЕПТ, покрита люмінофором із тривалим часом післясвітіння. Лінія або літера залишаються на ній видимими протягом тривалого часу, перш ніж стануть остаточно нерозрізненними, рис. 1.5 *а*. Щоб ***намалювати*** відрізок на дисплеї,

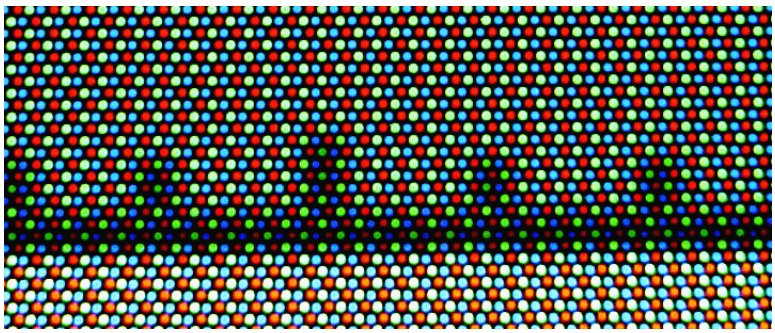


Рисунок 1.3 – Точковий люмінофорний растр для ЕЛТ з тіньовою маскою

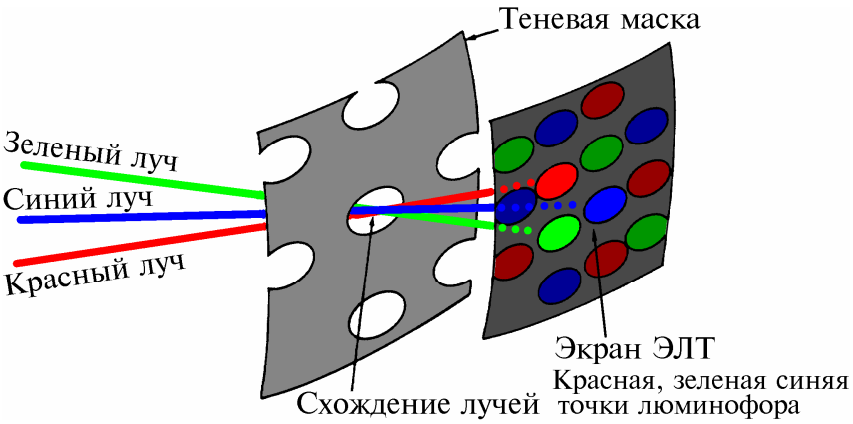
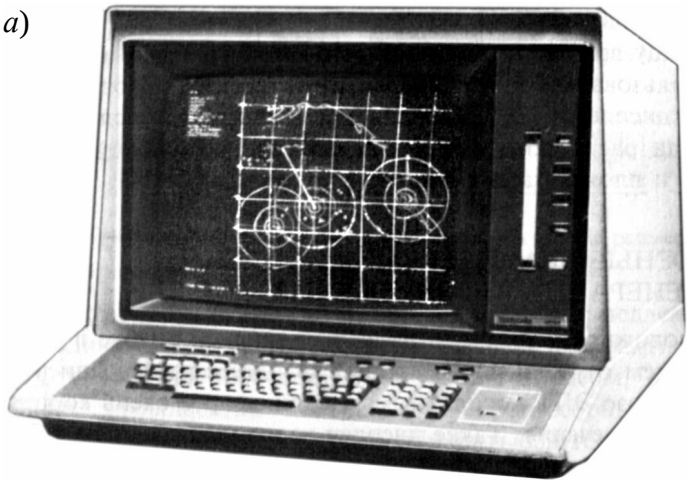


Рисунок 1.4 – Розташування електронної пушки та тіньової маски кольорової ЕЛТ

інтенсивність електронного променя збільшують до такої величини, що викликає запам’ятовування сліду променя на люмінофорі.

Для ***стирання*** зображення на всю трубку подають спеціальну напругу, що знімає світіння люмінофора. При цьому стираються усі відрізки і літери. Таким чином, стерти окремі лінії і літери неможливо, і зображення рухомих об’єктів або анімація неможливі.

а)



б)

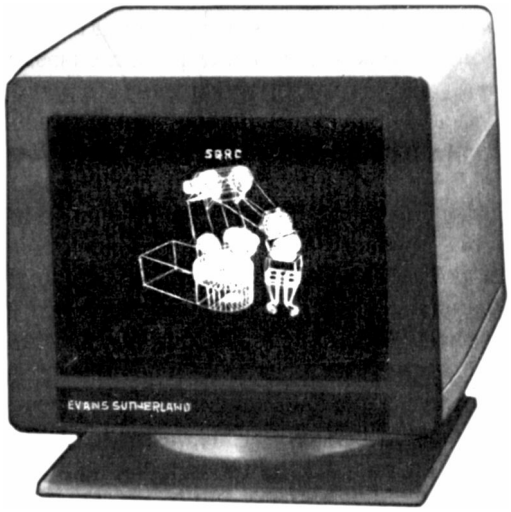


Рисунок 1.5 – Векторні дисплеї на запам’ятовуючій трубці (*а*) та з регенерацією (*б*)

Дисплей на запам'ятовуючій трубці — це ***векторний дисплей***, або дисплей з довільним скануванням. Це означає, що відрізок (вектор) може бути намальований безпосередньо з однієї адресованої точки в будь-яку іншу. ЕОМ обробляє зображення у виді набору ***координат вузлових точок***, по яких рухається електронний промінь.

На противагу дисплею на запам'ятовуючій трубці у ***векторному дисплеї з регенерацією зображення***, рис. 1.5 *б*, використовується люмінофор з дуже коротким часом післясвітіння. Зображення на ЕПТ за секунду має багаторазово перемальовуватися — ***регенеруватися***. Частота регенерації має складати не менш 30 Гц, у противному випадку зображення буде мерехтіти.

Для векторного дисплея з регенерацією потрібно крім ЕПТ ще два елементи: дисплейний буфер і дисплейний контролер. ***Дисплейний буфер*** — це безперервна ділянка пам'яті, що містить інформацію, необхідну для виведення зображення на ЕПТ (набір координат вузлових точок).

Функція ***дисплейного контролера*** полягає в тому, щоб циклічно обробляти цю інформацію зі швидкістю регенерації.

Складність малюнка (число зображених векторів) обмежується двома факторами — розміром дисплейного буфера і швидкістю дисплейного контролера.

Широко розповсюджені в 60-і і 70-і роки, векторні дисплеї не одержали подальшого розвитку в силу обмежених можливостей відображення — неможливості передачі складних зображень (типу фотографій).

Значно ширші можливості для створення й обробки зображень надали ***растрові графічні дисплеї*** на електронних променевих трубках, до яких відноситься більшість випущених у останні десятиріччя комп'ютерних дисплеїв.

Растровий пристрій можна розглядати як матрицю дискретних точок — ***пікселів***, кожна з яких може бути підсвічена. Інформація про стан кожної точки міститься в спеціальній пам'яті — ***буфері кадру***.

Процес перетворення растрової картинки, що зберігається в буфері кадру, в упорядкований набір точок на екрані називається ***растровим розгорненням***, рис. 1.6. Електронний промінь рухається по екрані горизонтально зліва направо, потім гаситься і швидко повертається вліво, переходячи при цьому на рядок, розташований нижче. Після проходу всього екрана промінь швидко повертається нагору.

При русі променя ***дисплейний контролер*** керує його інтенсивністю відповідно до вмісту буфера кадру. При цьому на екрані створюється зображення, що складається зі світлих чи темних точок у чорно-білих дисплеях, або з точок різних кольорів у кольорових.

Для усунення мерехтіння зображення частота його регенерації повинна бути дуже високою. У сучасних дисплеях вона сягає 80…100 Гц.

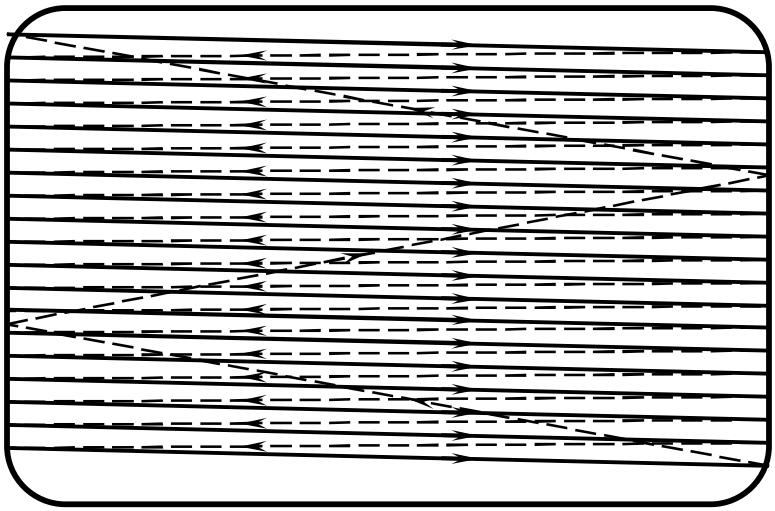


Рисунок 1.6 – Схема растрового розгорнення

Якість растрового зображення визначається його розміром **—** кількістю горизонтальних рядків і пікселів у кожному рядку. Можливість збільшення розміру растра обмежується обсягом пам'яті буфера кадру і максимальною частотою розгорнення дисплея.

Маючи низку переваг, дисплеї на ЕПТ мають недоліки — великі габарити (що особливо помітно при їхньому розташуванні на звичайному робочому столі) і високе енергоспоживання. Тому останнім часом усе більше поширення одержують ***рідкокристалічні дисплеї***.

Рідкі кристали — це органічні речовини, здатні під дією електричного потенціалу переходити з аморфного в упорядкований стан. Рідкі кристали виконують роль клапана, що пропускає або не проникне світло. Принцип роботи рідкокристалічного дисплею полягає у наступному.

Світло від джерела проходить через вертикальний поляризатор, рідкий кристал і горизонтальний поляризатор, рис. 1.7. Джерело випромінює звичайне неполяризоване біле світло. Як відомо, світло являє собою електромагнітну хвилю, де вектори електричних і магнітних полів спрямовані перпендикулярно до напрямку поширення хвилі. Після того, як світло пройде через поляризатор, вектор його електричного поля буде мати переважний напрямок (на рис. — вертикальний). Якщо світло потім потрапить на другий поляризатор, де вісь поляризації перпендикулярна першому (на рис. вона горизонтальна), світло ***не пройде*** другий поляризатор.

Рідкий кристал, який знаходиться між двома поляризаторами, ***повертає*** площину поляризації світлових хвиль на 90° (так званий "***твіст-ефект***"), що дозволяє їм пройти другий поляризатор, рис. 1.7 *а*. Для орієнтації молекул РК на панелі наноситься ***орієнтуюче покриття***, що «закручує» молекули. Подача на кристал електричного потенціалу переорієнтує молекули і «забороняє» поворот площини поляризації світла. Тоді світло ***не проходить*** другий поляризатор, рис. 1.7 *б*.

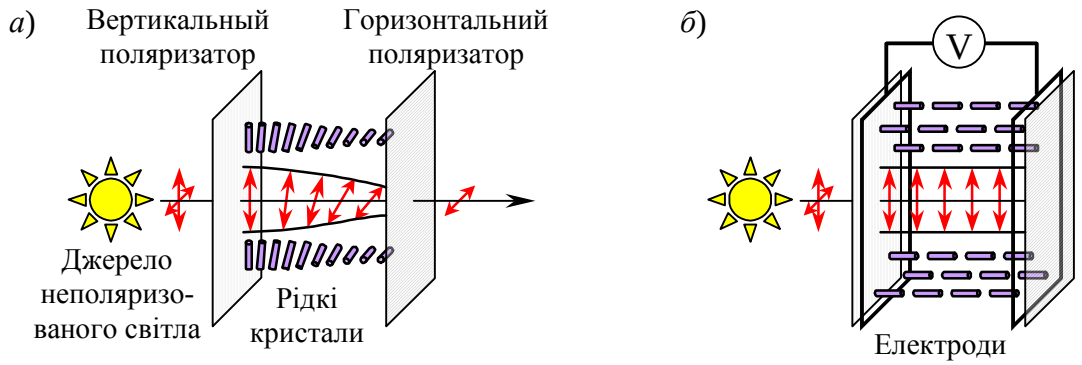


Рисунок 1.7 – Принцип роботи дисплею на рідких кристалах із застосуванням “твіст-ефекту”

Піксель РК-панели складається з трьох ***субпікселів*** основних кольорів. Біле світло, випромінюване підсвічуванням, проходить через рідкий кристал, а потім і офарблюється ***колірним фільтром***. Рідким кристалом кожного субпікселья можна керувати як клапаном. У залежності від кута повороту, через кристал проходить більше або менше світла, у результаті чого кожен піксель дає ту чи іншу кількість червоного, зеленого та синього кольорів.

Для керування пікселями на внутрішню поверхню обкладинок наносяться ***тонкошарові прозорі електроди***, рис. 1.8. На передні вертикальні електроди подається напруга, що відповідає потрібній якості пікселів деякого рядка. Задні горизонтальні провідники служать “землею”. У місцях перетинання вертикальних і горизонтальних провідників створюється різниця потенціалів, що змінює стан РК. Зображення виводиться шляхом почергового засвічування рядків пікселів.

Оскільки напруга подається на РК деякого пікселя не постійно, а короткочасно (у момент керування цим рядком), більшу частину часу РК є некерованим і прагне повернутися до вихідного стану, розмиваючи зображення. Такі матриці називаються «***пасивними***», дають нерізку картинку і «повільні».

Щоб усунути описані недоліки, виробники перейшли на технології активних матриць. У «***активних***» матрицях кожному пікселю додається ***транзистор***, що працює як перемикач, і ***запам’ятовуючий конденсатор***. Якщо транзистор відкритий (включений), то в запам’ятовуючий конденсатор можуть записуватися дані. Якщо транзистор закритий (виключений), то дані залишаються в конденсаторі, що виконує функцію “пам’яті”, тому підведення

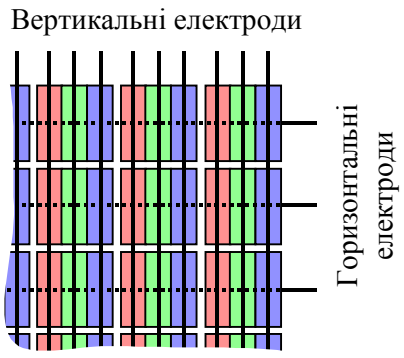


Рисунок 1.8 – Схема адресації субпікселів РК-матриці

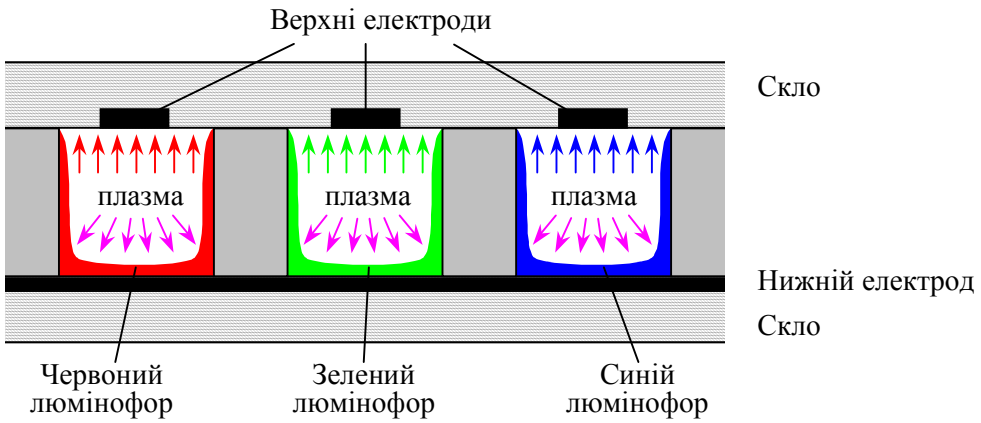


Рисунок 1.9 – Принцип роботи плазмової панелі

напруги до РК не припиниться, у той час як керуючі лінії будуть адресувати інший піксель. Тобто піксель не буде повертатися у вихідний стан, як відбувалося у випадку пасивної матриці. Крім того, час запису в конденсатор набагато менший, ніж час повороту кристала, тому можна швидше передавати на пікселі панелі дані. Технологія "***TFT***" (thin film transistors, тонкоплівочні транзистори), дозволяє одержати дисплей з малим часом відгуку і якісним чітким зображенням.

Іншою сучасною технологією створення засобів відображення є ***плазмові панелі***. Кожен піксель плазмової панелі складається з трьох порожнин (субпікселів), що містять інертний газ (ксенон) і та мають два електроди: зверху та знизу, рис. 1.9. Якщо до електродів прикласти напругу (близько 300 В), у порожнині утвориться ***плазма***, яка буде випромінювати ультрафіолетове світло (на рис. показане фіолетовим), що попадє на люмінофор в нижній частині порожнини. Під дією ультрафіолету люмінофори випромінюють світло видимого діапазону одного з основних кольорів: червоного, зеленого або синього.

Адресація субпікселів плазмової панелі робиться аналогічно адресації РК-панелі, за допомогою сітки вертикальних та горизонтальних електродів. Проблемою є неможливість регулювання інтенсивності випромінювання плазми. Субпіксель може або горіти, або не горіти, але проміжного стану немає. Тому для керування яскравістю світіння використовують метод ***імпульсно-кодової модуляції*** – субпіксель запалюють і гасять з частотою кілька тисяч разів за секунду. Щоб субпіксель горів яскраво, його потрібно часто запалювати, для одержання більш темного відтінку запалювати треба рідше. Око людини не помітить окремі спалахи й усереднить значення яскравості.

2 ПЕРЕДАЧА КОЛЬОРУ. КОЛІРНІ МОДЕЛІ

2.1 Фізична і психофізіологічна природа кольору

Сприйняття ***кольору*** людиною має як *фізичну*, так і *психофізіологічну*, природу. Воно залежить від фізичних властивостей ***світла***, тобто електромагнітної енергії, від його взаємодії з фізичними речовинами, а також від його інтерпретації зоровою системою людини.

Зорова система людини сприймає як ***видиме світло*** електромагнітну енергію з довжинами хвиль від 380 до 760 нм (1 нм = 10^{-9} м). Випромінювання з довжинами хвиль від 380 до 470 нм мають фіолетовий і синій колір, від 470 до 500 нм — синьо-зелений, від 500 до 560 нм — зелений, від 560 до 590 нм — жовто-помаранчевий, від 590 до 760 нм — червоний, рис. 2.1.

Система колірного зору людини включає два типи *світлочутливих фоторецепторів*, розташованих на *сітківці* ока: колбочки, чуттєві до деякого кольору, і палички, які не мають переважної чутливості до будь-якого кольору і граючі головну роль у створенні ахроматичних зорових образів. У кожному оці 6 … 8 млн. колбочок і 100 … 120 млн. паличок.

Існують три типи колбочок, піки чутливості яких приходяться приблизно на 420 нм, 534 нм і 564 нм, які називають відповідно «синіми», «зеленими» і «червоними». Терміни «червоний» і «зелений» стосовно до колбочок дуже

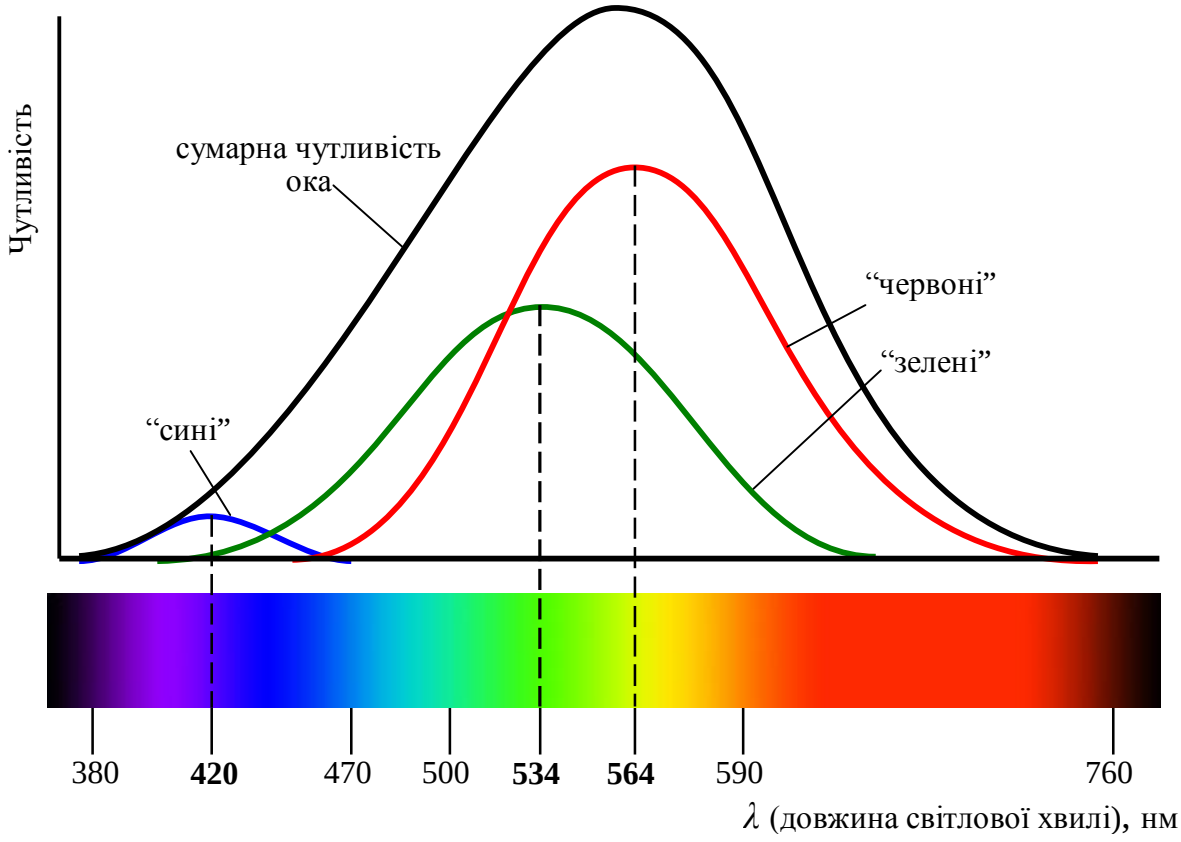


Рисунок 2.1 – Сприйняття світлових хвиль фоторецепторами ока

умовні, оскільки пікові значення 534 і 564 нм лежать у жовтому діапазоні. Чутливість ока до синього кольору істотно нижче, ніж до зеленого і червоного.

Сумарна чутливість ока максимальна при довжині хвилі порядку 550 нм, а на краях видимого діапазону спектра вона різко падає. Крива сумарної чутливості на рис. 2.1 називається *функцією спектральної чутливості ока*. Це міра світлової енергії або інтенсивності з урахуванням властивостей ока. Джерела випромінювання однакової інтенсивності але з різним спектральним складом будуть сприйматися оком як такі, що мають різну яскравість.

Світло сприймається або *безпосередньо від джерела*, наприклад електричної лампочки, екрана ЕПТ, або *побічно* при *віддзеркаленні* від поверхні об'єкта або після *переломлення* в ньому.

Розрізняють поняття *яскравості* і *світлоти*. ***Яскравість*** є властивістю самосвітних або *випромінюючих* об'єктів і визначає інтенсивність випромінювання. ***Світлота*** — властивість об'єктів, що віддзеркалюють, відбивати деяку частину падаючого на них світла.

Інтенсивність віддзеркаленого світла зручно розглядати в діапазоні від 0 до 1, де 0 відповідає чорному, 1 — білому, а проміжні значення — сірому кольору.

Джерело або об'єкт є ***ахроматичним***, якщо світло, що спостерігається, містить усі видимі довжини хвиль у приблизно рівних кількостях. Ахроматичне джерело здається *білим*, а відбите чи переломлене ахроматичне світло — *білим*, *чорним* або *сірим*. *Білими* виглядають об'єкти, що ахроматично відбивають більш 80% світла білого джерела, а *чорними* — менш 3%. Проміжні значення дають різні *відтінки сірого*.

Якщо сприймане світло містить довжини хвиль у *довільних нерівних кількостях*, то воно називається ***хроматичним***. Якщо довжини хвиль сконцентровані у верхнього краю видимого спектра, то світло здається *червоним* або червонуватим, тобто домінуюча довжина хвилі лежать у червоній області видимого спектра. Якщо довжини хвиль сконцентровані в нижній частині видимого спектра, то світло здається *синім* чи блакитнуватим, тобто домінуюча довжина хвилі лежать у синій частині спектра.

Однак сама по собі електромагнітна енергія визначеної довжини хвилі не має ніякого кольору. *Відчуття кольору* виникає в результаті перетворення фізичних явищ в оці і мозку людини. *Колір* об'єкта залежить від розподілу довжин хвиль джерела світла і від фізичних властивостей об'єкта. Об'єкт здається *кольоровим*, якщо він відбиває або пропускає світло лише у вузькому діапазоні довжин хвиль і поглинає всі інші. При взаємодії кольорів падаючого і віддзеркаленого або пропущеного світла можуть вийти несподівані результати. Наприклад, при віддзеркалені зеленого світла від білого об'єкта, об'єкт здається зеленим, а якщо зеленим світлом освітлюється червоний об'єкт, то він буде чорним, оскільки нього світло взагалі не відбивається.

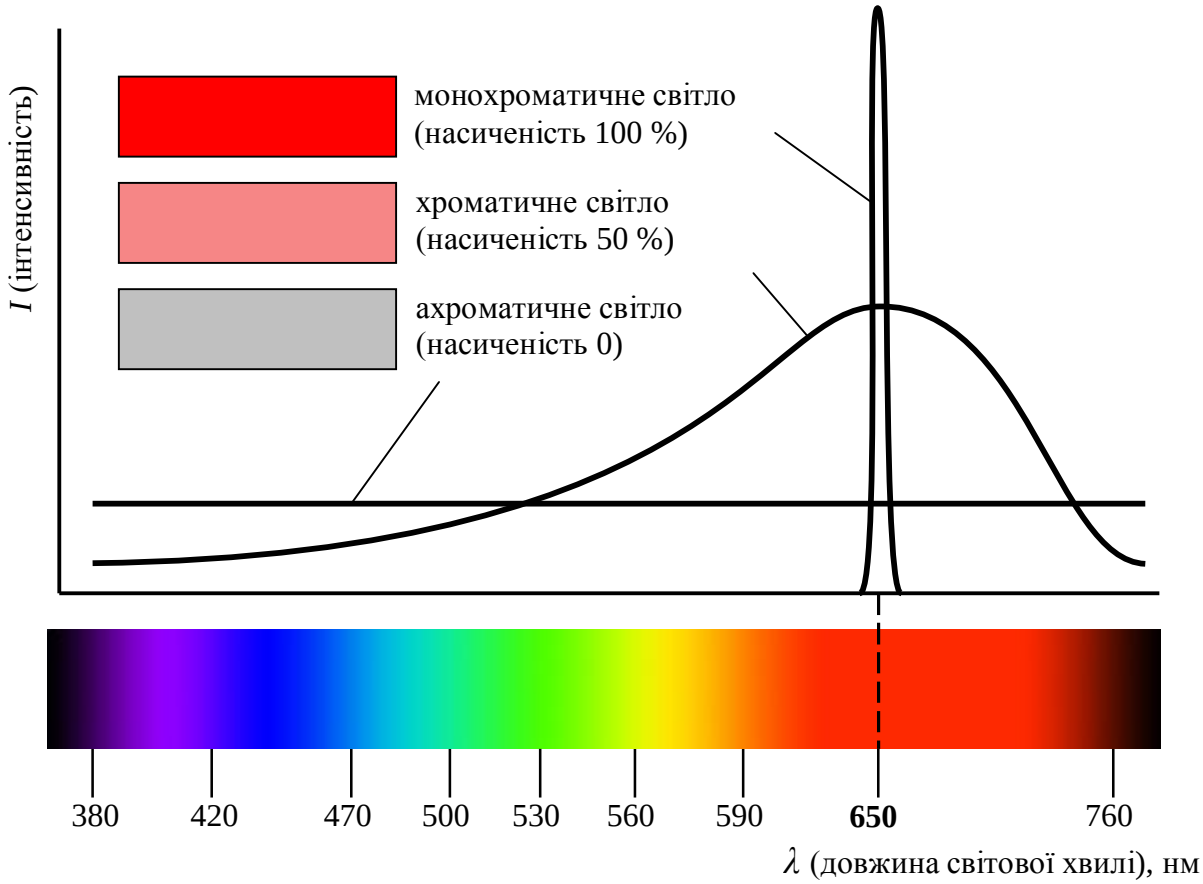


Рисунок 2.2 – Хроматичне и ахроматичне світло

Психофізіологічне представлення світла визначається *колірним тоном*, *насиченістю* і *світлотою*. ***Колірний тон*** дозволяє розрізняти кольори, а ***насиченість*** — визначати ступінь ослаблення (розведення) даного кольору білим кольором. Для чистого кольору вона дорівнює 100% і зменшується в міру додавання білого. Насиченість ахроматичного кольору складає 0%, а його світлота дорівнює інтенсивності цього світла.

Фізичними еквівалентами колірного тону, насиченості та світлоти є *домінуюча довжина хвилі*, *чистота* і *яскравість*. Електромагнітна енергія однієї довжини хвилі у видимому спектрі дає ***монохроматичний*** колір. На рис. 2.2 зображений розподіл енергії ахроматичного («білого») світла, хроматичного світла з домінуючою довжиною хвилі 650 нм і насиченістю 50 %, і монохроматичного світла з такою же довжиною хвилі.

Яскравість пропорційна енергії світла і розглядається як сумарна енергія хвиль усіх довжин. Графічно яскравість світла визначається площею під кривою спектрального складу, рис. 2.2. Однак суб'єктивне відчуття яскравості при сприйнятті світла людським оком залежить від його спектральної чутливості. Світло зеленого чи жовтого тону буде здаватися значно яскравіше, ніж світло з такою же енергією, але синього тону.

2.2 Трикомпонентна теорія кольору

Звичайно зустрічаються не чисті монохроматичні кольори, а їхні суміші. Основою ***трикомпонентної теорії кольору*** є припущення про те, що в центральній частині сітківки знаходяться три типи чуттєвих до кольору колбочок. Перший тип сприймає довжини хвиль, що лежать у середині видимого спектра, тобто зелений колір; другий — довжини хвиль у верхнього краю видимого спектра, тобто червоний колір; третій — короткі хвилі нижньої частини спектра, синій колір.

Якщо на всі три типи колбочок впливають хвилі однакової інтенсивності (яскравості), то світло здається *білим*. Природне біле світло містить усі довжини хвиль видимого спектра, однак *відчуття білого світла* можна одержати, змішуючи будь-які три кольори, якщо жоден з них не є *лінійною комбінацією* двох інших. Такі три кольори називаються ***основними***.

У машинній графіці застосовуються дві системи змішування основних кольорів, рис. 2.3:

— ***адитивна*** — червоний, зелений, синій (*RGB*);

— ***субтрактивна*** — блакитний, пурпурний, жовтий (*CMY*).

Адитивна колірна система RGB зручна для *світних поверхонь*, наприклад екранів ЕПТ. Поверхня екрана ЕПТ, РК-дисплею, світлової панелі та інших пристроїв відображення кольорової графічної інформації складається з ділянок, що випромінюють світло трьох основних кольорів — звичайно червоного, зеленого і синього, див. п. 1.3. Оскільки ці ділянки розташовані близько і дуже малі, при спостереженні такого екрана з деякої відстані вони зливаються, тобто в один світлочутливий рецептор ока спостерігача попадає світло з декількох ділянок різних кольорів. Око «*підсумовує*» ці випромінювання і сприймає їхній як єдиний світловий потік з деяким спектральним складом — сумою спектральних складів випромінювань розташованих поруч джерел світла, рис. 2.3 *а*.

а)

б)

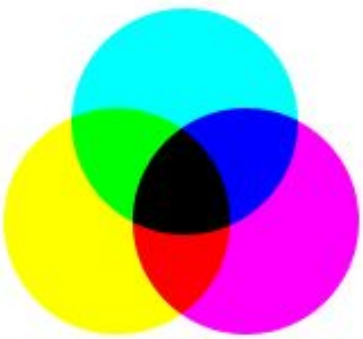


Рисунок 2.3 – Кольорові системи: адитивна *RGB* (*а*) і субтрактивна *CMY* (*б*)

Субтрактивна система CMY застосовується для *поверхонь, що відбивають світло*, наприклад типографських фарб, кольорових фотозображень, а також прозорих плівок і несвітлових екранів. У субтрактивних системах зі спектра білого світла, що падає на зображення або проходить крізь плівку, поглинаються (*віднімаються*) хвилі деякого кольору, рис. 2.3 *б*. Наприклад, щоб одержати червоний колір, з білого потрібно відняти голубий (суму синього і зеленого). При віддзеркаленні або проходженні світла крізь пурпурний об'єкт поглинається зелена частина спектра. Такі кольори називаються *додатковими*.

Додатковий колір — це різниця білого і даного кольору: голубий це білий мінус червоний, пурпурний — білий мінус зелений, жовтий — білий мінус синій. Кольори однієї системи є додатковими до іншої: голубий — до червоного, пурпурний — до зеленого, жовтий — до синього. Цікаво, що в спектрі веселки пурпурного кольору немає, тобто він породжується зоровою системою людини.

Передача довільного кольору за допомогою трьох основних кольорів робиться в такий спосіб. Нехай на білий екран падає довільне контрольне світло. Спостерігач намагається дослідним шляхом зрівняти на екрані поруч з контрольним світлом його колірний тон, насиченість і світлоту за допомогою монохроматичних потоків світла різної інтенсивності, рис. 2.4.

Якщо використовується тільки один інструментальний колір, то довжина хвилі в нього повинна бути такою ж, як у контрольного. У противному випадку, якщо не взяти до уваги *колірний тон* і *насиченість* контрольного світла, можна зрівняти кольори тільки за *світлотою*. Ця процедура називається ***фотометрією***. Таким способом створюються монохроматичні репродукції кольорових зображень.

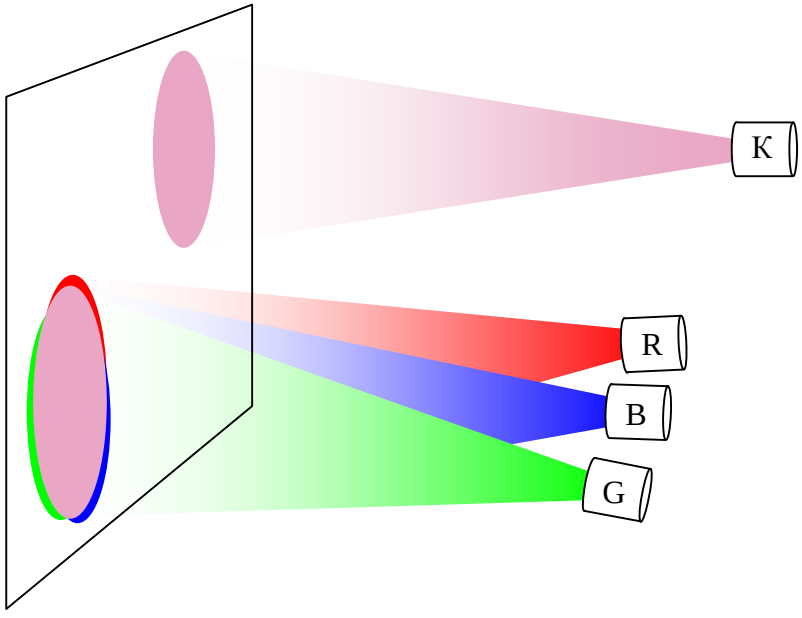


Рисунок 2.4 – Схема врівноваження кольорів: R, G, B — джерела червоного, зеленого й синього монохроматичного світла; К — контрольне джерело

Якщо в розпорядженні спостерігача є два монохроматичних джерела, то він може зрівняти обмежену кількість контрольних зразків. Додаючи третій інструментальний колір, можна одержати майже всі контрольні варіанти, за умови, що ці три кольори досить широко розподілені по спектрі і жоден з них не є лінійною комбінацією інших, тобто що це ***основні кольори***. Бажано, щоб перший колір лежав в області спектра з великими довжинами хвиль (*червоний*), другий — із середніми (*зелений*) і третій — з малими довжинами хвиль (*синій*). Об'єднання цих трьох кольорів для зрівняння контрольного кольору математично виражається як

$$C = r + g + b$$

де C — колір контрольного світла; R, G, B — червоний, зелений і синій інструментальні потоки світла; r, g, b — відносні кількості відповідних потоків світла. Вивченням питань передачі кольору займається ***колориметрія***.

Результати колориметричних досліджень узагальнюються в ***законах Грасмана***:

1. Око реагує на три різних стимули, що підтверджує ***тривимірність*** природи кольору. Як стимули можна розглядати, наприклад, довжину хвилі, що домінує (колірний тон), насиченість і яскравість або червоний, зелений і синій кольори.

2. Кольори завжди ***лінійно залежні***, тобто $C = r + g + b$. Отже, для суміші двох кольорів C_1, і C_2 має місце рівність

$$C_1 + C_2 = r_1R+g_1G+b_1B + r_2R +g_2G+b_2B = (r_1+r_2)R + (g_1+g_2)G + (b_1+b_2)B$$

При цьому структура спектрів кольорів C_1 і C_2 значення не має.

3. Якщо в суміші трьох кольорів один безперервно змінюється, а інші залишаються постійними, то колір суміші буде мінятися безперервно , тобто ***тривимірний колірний простір безперервний***.

Відомо, що зорова система людини здатна розрізняти приблизно 350 000 кольорів. Якщо кольори розрізняються тільки по тонах, то в центральній синьо-жовтій частині спектра різними виявляються кольори, у яких домінуючі довжини хвиль відрізняються на 1 нм, у той час як біля країв спектра — на 10 нм. Чітко помітні приблизно 130 колірних тонів. Якщо міняється тільки насиченість, то зорова система здатна виділити вже не так багато кольорів. Існує 16 степенів насиченості жовтого і 23 — червоно-фіолетового кольору.

2.3 Колірні моделі

Тривимірна природа світла дозволяє представити сукупність кольорів у виді деякого простору, причому кожний з компонентів буде представлений координатою цього простору. Таке представлення кольору називається ***колірною моделлю***.

Найпростішими колірними моделями є системи ***RGB*** і ***CMY***, представлені у виді «колірних кубів». Будь-який колір *C* можна представити як вектор у тривимірному просторі, у якому координатні осі відповідають інтенсивності основних кольорів *RGB* чи *CMY*. Проекціями цього вектора на осі будуть інтенсивності його складових *r*, *g* і *b*, рис. 2.5.

На малюнку максимальна інтенсивність основного кольору дорівнює 1, а менші значення інтенсивності виражаються дробовими значеннями в діапазоні 0…1 Таке представлення є зручним для розрахунків і аналізу, але на практиці, при передачі кольорових цифрових зображень не застосовується, оскільки збереження чисел з комою, що плаває, вимагає більших обсягів пам'яті, чим цілих чисел. Тому компоненти задаються або у відсотках (0…100),або в діапазоні 0…255,що відповідає одному байту інформації.

Початком координат у колірному кубі *RGB* служить чорний колір, а в *CMY*— білий. *Основні* кольори в обох випадках розташовані по осях, а *додаткові* кольори лежать у протилежних вершинах.

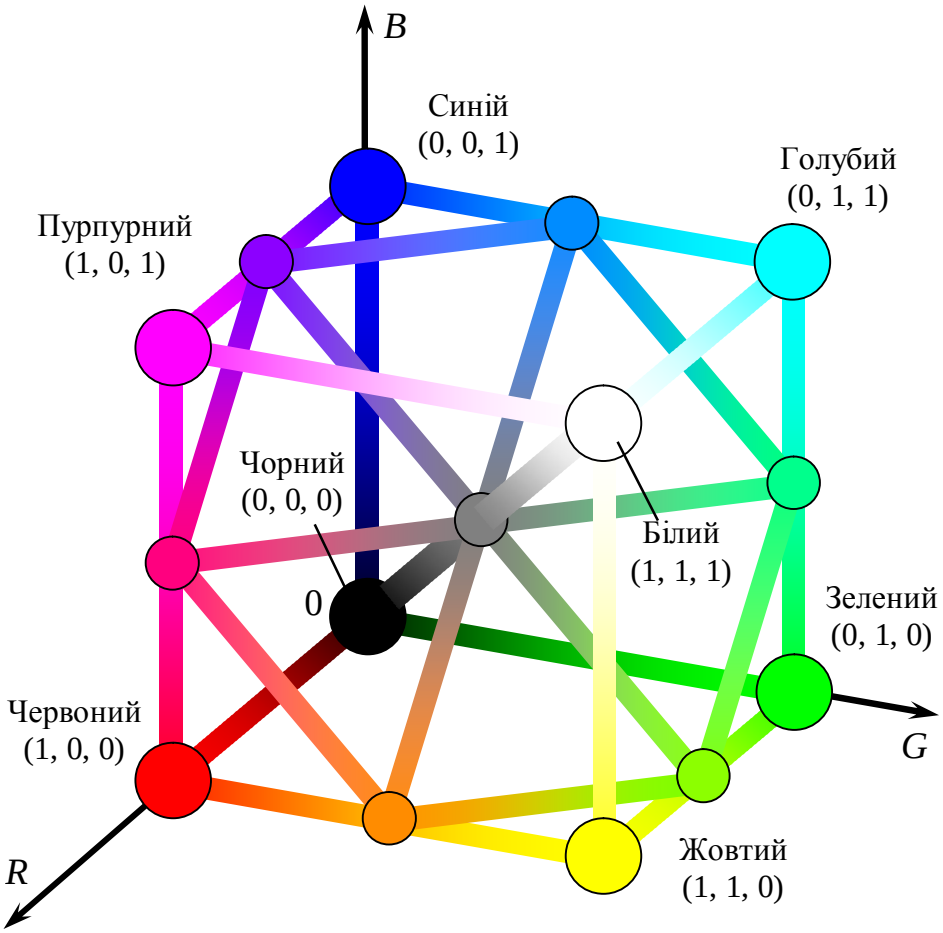


Рисунок 3.5 – Колірний куб *RGB*

Перетворення між просторами *RGB* і *CMY* виражається в такий спосіб:

$$[R\ G\ B] = [1\ 1\ 1] - [C\ M\ Y].$$

Для того, щоб виконати це перетворення, потрібно «перевернути» куб, зображений на рис. 2.5.

Ахроматичні, тобто сірі кольори в обох моделях розташовані уздовж головної діагоналі куба — від чорного до білого, — оскільки для них має виконуватися умова рівності інтенсивностей трьох складових:

$$r = g = b.$$

Площини, проведені перпендикулярно головній діагоналі куба, називаються *площинами рівної інтенсивності*. Для всіх точок, що лежать на такій площині, виконується рівність

$$r + g + b = Br,$$

де *Br* — яскравість (світлота), однакова для всіх точок площини. На рис. 2.5 показана площина, проведена через середину головної діагоналі. Така площина перетинає ребра куба в їхніх серединах і утворює *медіанний перетин*.

Медіанний перетин має дуже важливі властивості:

1. *Яскравість* світла в будь-якій точці медіанного перетину дорівнює половині яскравості білого світла.

2. У центрі такого перетину (точці, що лежить на головній діагоналі) *насиченість* кольору дорівнює нулю — колір ахроматичний. В міру віддалення від центра перетину *насиченість* кольору зростає і стає максимальною на його гранях.

3. На медіанному перетині присутні всі три основних і три додаткових кольори, розташованих по його окружності в такій послідовності: червоний, жовтий, зелений, блакитний, синій, пурпурний.

Моделі *RGB* і *CMY* зручні для машинної обробки, відображення і збереження зображень, оскільки в них явно задані інтенсивності компонентів. Однак описувати суб’єктивне сприйняття кольору людьми в цих системах незручно. Наприклад, як у позначеннях *RGB* чи *CMY* задати пастельний червонясто-помаранчовий колір?

Художники характеризують колір за допомогою таких понять, як *розбіли*, *відтінки*, *тони*. Розбіли одержують, додаючи в чистий колір білий, відтінки — чорний, тони — додаючи обидві ці фарби. Це можна представити у виді трикутника, рис. 2.6.

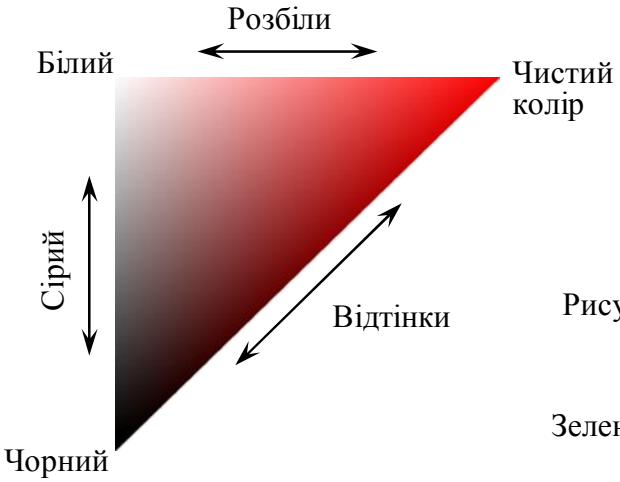


Рисунок 2.6 – Розбіли та відтінки чистого кольору

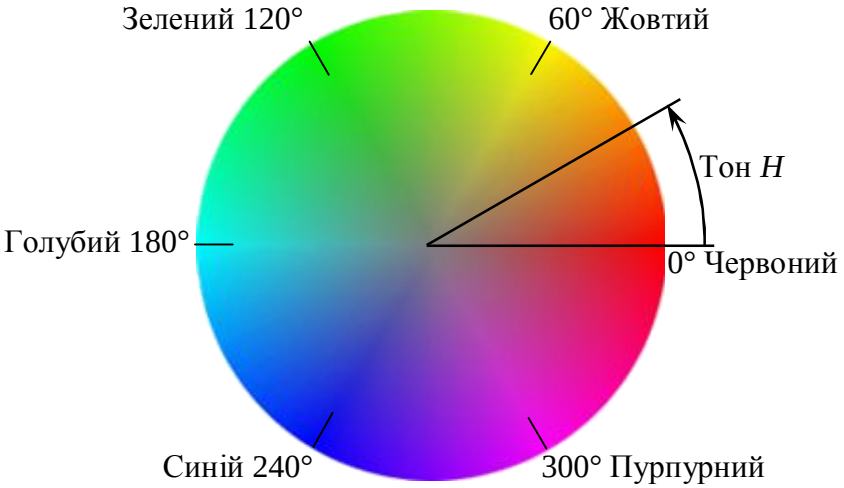


Рисунок 2.7 – Колірні тони

Тонову шкалу можна одержати на основі медіанного перетину колірного куба. Останнє являє собою правильний шестикутник, вершини якого — основні кольори і додаткові до них, а ребра — перехідні тони, одержувані змішуванням основних у різних пропорціях. Замінивши ребра шестикутника дугами, одержимо колірне коло, рис. 2.7. Прийнявши один з основних кольорів за 0 (звичайно приймають червоний), одержимо шкалу, у якій колірний тон задається кутом у діапазоні 0 … 360°...

Якщо зібрати трикутники розбілів і відтінків для всіх чистих кольорів навколо центральної чорно-білої осі, одержимо тривимірну модель суб'єктивного представлення кольору — **колірний конус *HSB***, рис. 2.8 *а*. У цій моделі колірний тон *H* задається в градусах у відповідності зі шкалою рис. 2.7, насиченість *S* визначається відстанню до осі конуса, а світлота *B* — відстанню до вершини; величини *S* і *B* звичайно задаються в діапазоні 0 … 100 %. Вершина конуса відповідає чорному кольору (світлота *B* = 0), вісь конуса — ахроматичним кольорам від чорного до білого (насиченість *S* = 0). Поверхня конуса — чисті кольори (*S* = 100 %).

Модель *HSB* відповідає тому, як складають кольори художники. Чистим пігментам відповідають значення *S* = 100%, *B* = 100%; розбілам — кольори зі збільшеним змістом білого, тобто з меншим *S*; відтінкам — кольори зі зменшеним *B*, що виходять при додаванні чорного. Тому модель *HSB* доцільно застосовувати для предметів, що відбивають світло, наприклад типографських зображень.

а)

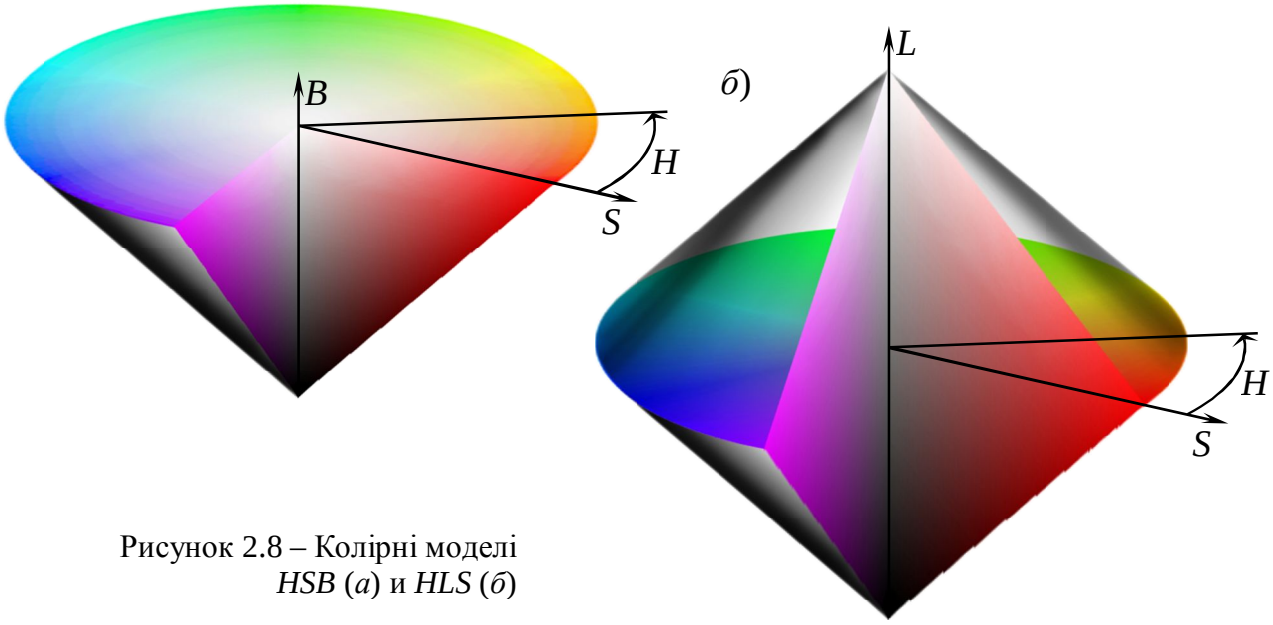


Рисунок 2.8 – Колірні моделі
HSB (*а*) и *HLS* (*б*)

Колірна модель ***HLS*** у виді подвійного конуса є розширенням одиночного конуса *HSB* і застосовується для *самосвітних* об'єктів, рис. 2.8 *б*. Світлота тут позначає яскравість *L* джерела світла. Нижня вершина подвійного конуса відповідає нульової яскравості (*L* = 0), а верхня — максимальної яскравості джерела (*L* = 100 %). Тон *H* і насиченість *S* задаються так само, як і в *HSB*. Основна відмінність *HLS* від *HSB* полягає в тому, що чистий колір у *HSB* виходить при максимальній світлоті (*B* = 100 %), а в *HLS* — при середній яскравості (*L* = 50 %). У цьому модель *HLS* ближче до колірного куба *RGB*.

Існують інші колірні моделі, розраховані на застосування в різних областях техніки. У телебаченні системи NTSC застосовується колірна модель ***YIQ***, що забезпечує можливість передачі кольору за допомогою частотної модуляції і сумісна з чорно-білим телебаченням. Сигнал *Y* містить інформацію про рівень яскравості і цілком аналогічний сигналу чорно-білого телебачення. Сигнали *I* і *Q* несуть інформацію про колірний тон і насиченість.

При підготовці зображень для кольорового друку використовується чотирикомпонентна система *CMYK*, що відрізняється від *CMY* тим, що параметри *C*, *M* і *Y* задають відносні вмісти блакитного, пурпурного і жовтого кольорів, а параметр *K* — світлоту зображення. Така система відповідає принципу кольорового друку за допомогою барвників чотирьох кольорів: блакитного, пурпурного, жовтого і чорного.

3 КОДУВАННЯ РАСТРОВОГО ЗОБРАЖЕННЯ. ОСОБЛИВОСТІ РАСТРА

3.1 Растрові зображення і їхні основні характеристики

Растр — це матриця точок (пікселів). Будь-який ***піксель*** має свій колір. Сукупність пікселів різного кольору утворить ***зображення***.

Для опису розташування пікселів використовують різноманітні *системи координат*. Найчастіше використовується система цілих координат — номерів пікселів з початком (0, 0) у лівому верхньому куті.

Растр має такі *характеристики*:

Тип растра — розташування пікселів у просторі. Розрізняють квадратний, прямокутний, гексагональний і ін. типи растра, рис. 3.1. У комп'ютерній техніці найбільш розповсюджений квадратний растр. Прямокутний растр зустрічався в старих дисплейних системах (CGA, VGA і ін.). Гексагональний растр застосовується в поліграфії.

Розмір растра звичайно вимірюється кількістю пікселів по горизонталі і вертикалі, або їх загальною кількістю, рис. 3.2.

Форма пікселів растра визначається особливостями пристрою графічного висновку (рис. 3.3). Наприклад, пікселі можуть мати форму чи прямокутника квадрата, що по розмірах рівні кроку растра (дисплей на рідких кристалах); пікселі можуть мати круглу форму і по розмірах можуть не дорівнювати кроку растра (принтери).

Роздільна здатність растра характеризує відстань між сусідніми пікселями — крок дискретної сітки растра. Роздільну здатність вимірюють кількістю пікселів на одиницю довжини. Найбільш популярною є одиниця виміру dpi (dots per inch) кількість пікселів в одному дюймі довжини (2,54 см).

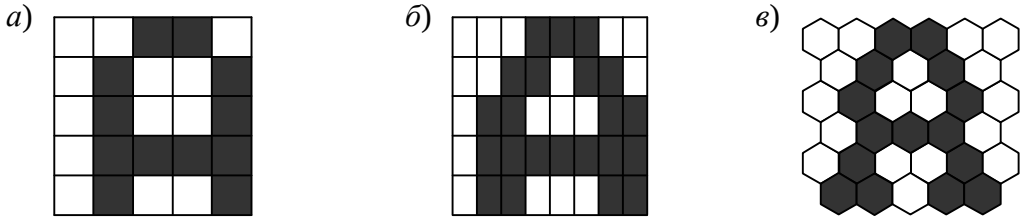


Рисунок 3.1 – Приклади квадратного (*а*), прямокутного (*б*) и гексагонального (в) растрів

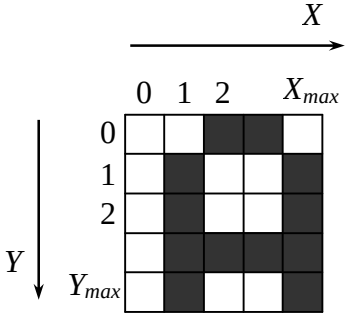


Рисунок 3.2 – Система координат та розмір растра

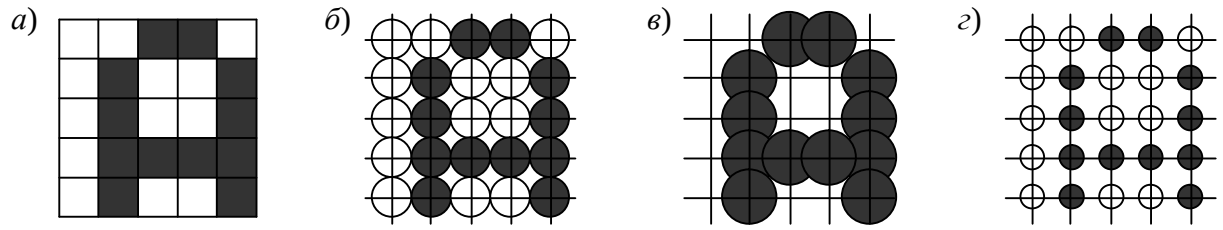


Рисунок 3.3 – Різні форми і розміри пікселів

Око людини з нормальним зором здатне розрізняти об’єкти з кутовим розміром біля однієї хвилини. Якщо відстань між окремими точками (пікселями) мало, то ці точки вже не сприймаються як окремі. Якщо вважати відстань, з якої людина звичайно розглядає друковані документи, рівною 30 см, то можна оцінити мінімальну роздільну здатність, при якій вже не помітні окремі пікселі, — приблизно 300 dpi. Лазерні чорно-білі принтери цілком задовольняють такій вимозі.

Дисплеї звичайно рекомендується розглядати з відстані не ближче 50 см. Відповідно до приведеної вище оцінки мінімальної роздільної здатності, відстані 0,5 м відповідають приблизно 200 dpi. У сучасних дисплеях роздільна здатність складає 80…120 dpi — це замало; наприклад, дисплей розміром 17" по діагоналі має забезпечувати не 1280x1024 пікселів, а вдвічі більше. Але на сучасному рівні розвитку техніки це поки що неможливо.

Кількість кольорів (*глибина кольору*) — важлива характеристика будь-якого зображення, не тільки растрового. Відповідно до психофізіологічних досліджень, око людини здатне розрізняти 350 000 кольорів. Класифікують зображення в такий спосіб:

— *двоколірні* (*бінарні*) — піксель може приймати один із двох кольорів (звичайно чорний або білий), при цьому на кожен піксель приходиться 1 біт пам’яті; двоколірні зображення звичайно використовуються при друку на лазерному або іншому некольоровому принтері.

— *напівтонові* — градації яскравості сірого або іншого кольору. Для передачі реалістичних чорно-білих зображень (наприклад, фотографій) звичайно використовують 256 градацій яскравості — при цьому для кодування інформації про піксель необхідно 1 байт (8 біт) пам’яті.

— *кольорові зображення* — піксель може приймати різні кольори і відтінки; у залежності від кількості біт пам’яті, що приходяться на 1 піксель, глибина кольору може складати 8 (3 біти), 16 (4 біти), 256 (8 біт), 65 536 (16 біт) 16 777 216 (24 біта) і 4 294 967 296 (32 біти). Перші три з зазначених використовуються для створення спрощених, а останні два — фотореалістичних зображень. Глибина в 65 536 кольорів дозволяє створити й обробляти зображення, близьке до фотореалістичного, при менших витратах пам’яті і машинного часу.

3.2 Способи растрового розгорнення

Для виведення на відеомонітор розкладений у ***растр*** образ необхідно представити у виді того шаблона, який вимагає дисплей. Це перетворення називається ***растровим розгорненням***.

На відміну від *дисплейного списку* для векторного дисплея, що містить інформацію тільки про відрізки чи літери, у даному випадку дисплейний список повинен містити інформацію про кожний *піксель* на екрані. Необхідно, крім того, щоб ця інформація організовувалася і виводилася зі швидкістю *відеогенерації* в порядку сканування рядків, тобто зверху вниз і зліва направо.

Існує чотири способи досягнення такого результату — *растрове розгорнення в реальному часі*, *групове кодування*, *клітинне кодування* і *буфер кадру*.

1) Растрове розгорнення в реальному часі

При розгорненні в реальному часі, або «на льоту», сцена зберігається в пам'яті у виді *дисплейного списку*, аналогічно векторним дисплеям. Під час відтворення кожного кадру процесор сканує цю інформацію й обчислює інтенсивність кожного пікселя на екрані, рис. 3.4.

При такому розгорненні *не потрібні великі кількості пам'яті*. Вимоги на пам'ять звичайно обмежуються необхідністю зберігати дисплейний список плюс один рядок, що сканується. Більш того, оскільки інформація про сцену зберігається в довільно організованому дисплейному списку, *додавання* або *видалення* інформації зі списку здійснюється легко, а це зручно для динамічних застосувань. Однак *складність* виведеного зображення обмежується швидкістю дисплейного процесора. Звичайно це означає, що обмеженим є число відрізків та багатокутників у картині, кількість перетинань зі рядком, що сканує, число кольорів або напівтонів сірого.

Для одержання перетинань (якщо вони є) кожного відрізка дисплейного списку зі рядком, що сканує, у найпростішій реалізації методу всякий раз при

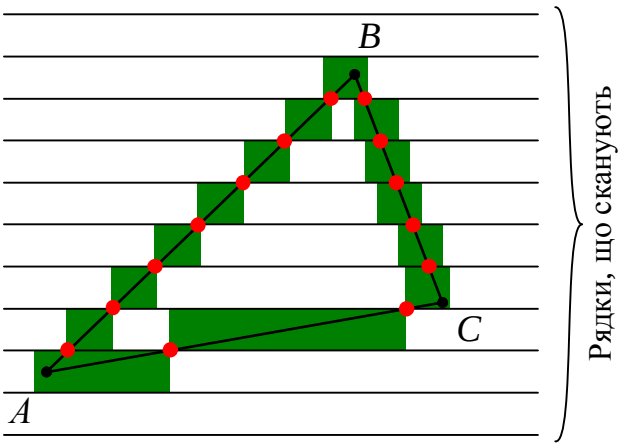


Рисунок 3.4 – Растрове розгорнення в реальному часі

зображенні рядка обробляється весь дисплейний список. При регенерації відеозображення на кожний рядок, що сканує, а виходить, і на обробку всього списку приходиться тільки 63,5 мікросекунди. Настільки малий час дозволяє використовувати даний метод тільки для малювання нескладних креслень.

2) Групове кодування

Метод групового кодування застосовується у випадках, коли великі області зображення мають однакову інтенсивність або колір. При груповому кодуванні визначається тільки інтенсивність і кількість послідовних пікселів з цією інтенсивністю на даного рядка, що сканує, рис.3.5.

Дані, що кодують, слід розглядати *групами по два числа*. Перше число — колір або *інтенсивність* пікселя, друге — *кількість послідовних пікселів* на рядку, що сканує.

Таким чином, у рядку 1 на рис. 3.2, мається 8 пікселів кольору 0, тобто чорних чи фонових, 1 піксель кольору 1 (червоний) і потім ще 1 кольору 0.

Переваги групового кодування: невеликий обсяг займаної пам’яті (у порівнянні з буфером кадру) і можливість відображення областей, заповнених кольором.

Недоліки групового кодування:

— додавання чи видалення відрізків тексту з зображення є трудомісткою операцією і займає багато часу через послідовне збереження довжин ділянок;

— для зображень, що містять велику кількість дрібних деталей, може знадобитися більше пам’яті, чим при попіксельному збереженні.

Групове кодування застосовується у деяких графічних форматах для зменшення обсягу файлів.

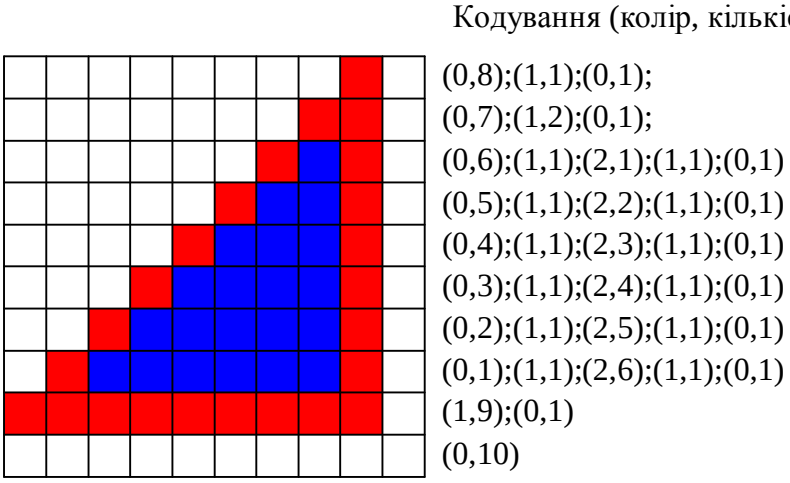


Рисунок 3.5 – Групове кодування

3) Клітинне кодування

У методі клітинного кодування за допомогою мінімуму інформації (один або два байти) кодуються цілі *області зображення*, тобто *клітки*, рис. 3.6.

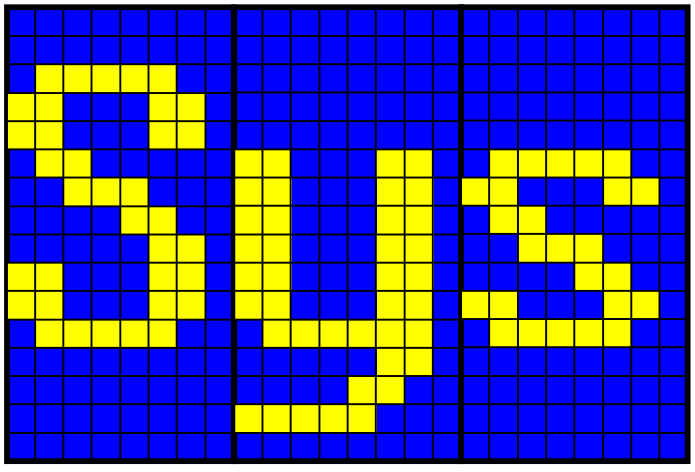
У такому дисплеї область екрана розбивається на клітки або області, досить великі, щоб містити *одну літеру*. Наприклад, екран можна розбити на області розміром 8×8 чи 8×16 пікселів. Для дисплея 640 × 400 пікселів вийде 50 чи 25 рядків, що містять по 80 кліток.

Для виведення зображення на екран дисплейний контролер використовує дві області пам'яті: *дисплейний буфер* і *область знакогенератора*. У ***дисплейному буфері*** зберігаються коди символів, які виводяться на екран. Кожен символ кодується одним чи двома байтами (в останньому випадку другий байт кодує колір символу і фона, інші характеристики символу, наприклад, миготіння).

В області ***знакогенератора*** зберігаються ***маски символів*** — інформація про їхнє накреслення. Для символів розміром 8×8 чи 8×16 пікселів на кожен символ приходиться по 8 чи 16 байт відповідно.

Окрім літер і розділових знаків знакогенератор містить набір *графічних символів*: горизонтальних і вертикальних ліній, сполучень, стрілок і т.д., що дозволяють виводити на екран найпростіші зображення — таблиці, схеми. Виведення більш складних графічних зображень при такому способі кодування є неможливим.

В ЕОМ 60…80 років описаний вище режим виведення інформації був основним або єдиним можливим. Дисплейні контролери сучасних персональних комп'ютерів підтримують такий режим, який звичайно називають ***текстовим***, для сумісності зі старим програмним забезпеченням.



Код символу: 83 121 115

Рисунок 3.6 – Клітинне кодування

4) Буфер кадру

Буфер кадру являє собою велику безперервну ділянку пам'яті, у якій для кожної точки (*пікселя*) приділяється як мінімум один біт пам'яті. Ця пам'ять називається ***бітовою площиною***. Для растра розміром 800×600 пікселів потрібно 800×600 = 480000 біт або 60 кілобайт пам'яті в одній бітовій площині.

Зображення в буфері кадру будується побітно. Через те, що біт пам'яті має тільки *два стани* (0 чи 1), маючи одну бітову площину можна одержати лише двоколірне (чорно-біле) зображення.

Бітова площина є цифровим пристроєм, тоді як растровий дисплей — аналоговий пристрій, для роботи якого потрібна електрична напруга. Тому при зчитуванні інформації з буфера кадру і її виведенні на графічний пристрій з растровою ЕПТ має відбуватися перетворення з цифрового представлення в аналоговий сигнал. Таке перетворення виконує *цифро-аналоговий перетворювач* (ЦАП).

Кожен піксель буфера кадру має бути зчитаний і перетворений, перш ніж він буде відображений на растровій ЕПТ. На рис. 3.7 приведена схема графічного пристрою з чорно-білою растровою ЕПТ, побудованого на основі буфера кадру з однією бітовою площиною. Інформація з буфера кадру надходить у ***регістр***, а потім ЦАП перетворює вміст регістра в аналоговий електричний сигнал, що подається на дисплей.

3.3 Кодування кольорів і напівтонів

Кольори або *напівтони* сірого кольору можуть бути введені в буфер кадру шляхом використання *додаткових бітових площин*.

На рис. 3.8 показана схема буфера кадру з N бітовими площинами для ***градацій сірого кольору***. *Інтенсивність* кожного пікселя на ЕПТ керується вмістом комірок пам'яті в кожній з N бітових площин.

У відповідну позицію *регістра* завантажується бінарна величина (0 чи 1) з кожної площини. Двоїчне число, що вийшло в результаті, інтерпретується як *рівень інтенсивності* між 0 (темний екран) і 2^N–1 (максимальна інтенсивність світіння), тобто усього можна одержати 2^N рівнів інтенсивності. За допомогою ЦАП це число перетворюється в напругу.

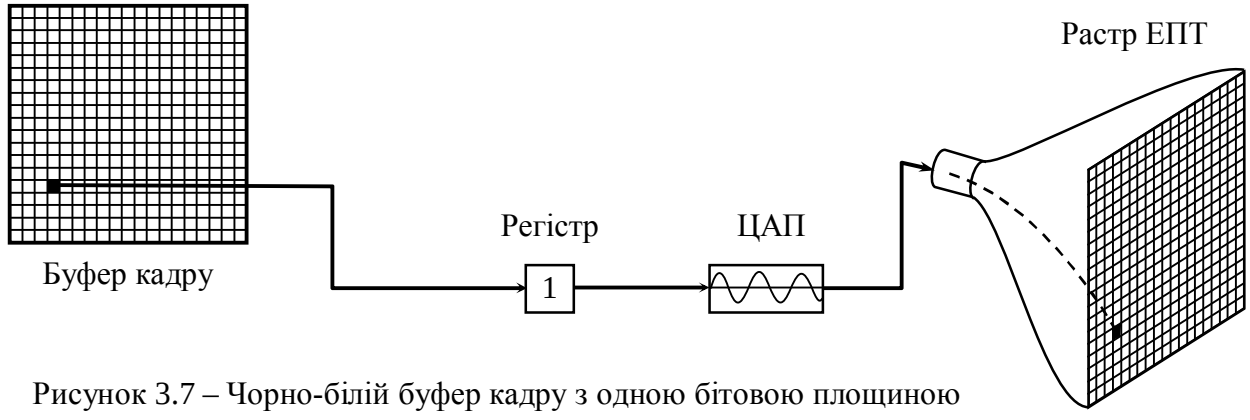


Рисунок 3.7 – Чорно-білий буфер кадру з одною бітовою площиною

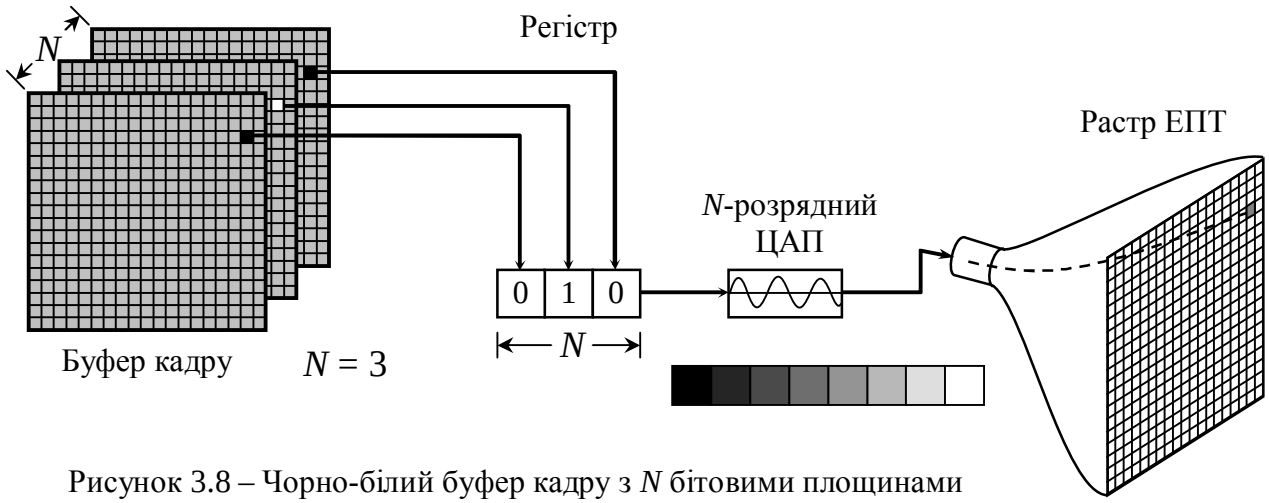


Рисунок 3.8 – Чорно-білий буфер кадру з N бітовими площинами

Рис. 3.8 ілюструє систему з трьома бітовими площинами для $2^3 = 8$ *рівнів інтенсивності*. Для передачі реалістичного чорно-білого зображення використовується більша кількість бітових площин — до 8. При цьому число рівнів інтенсивності складає $2^8 = 256$.

Для кожної бітової площини потрібний повний обсяг пам'яті при даному розмірі растра: наприклад, буфер кадру з трьома бітовими площинами для растра 800×600 займає 800×600×3 = 1440000 чи біт 180 кілобайт пам'яті, а при 8 бітових площинах — 480 кілобайт.

Оскільки існує три основних кольори, найбільш простий спосіб реалізації ***кольорового зображення*** — кольоровий буфер кадру з трьома бітовими площинами, по одній для кожного з основних кольорів, рис. 3.9 *а*. Кожна бітова площина керує індивідуальною електронною пушкою для кожного з трьох основних кольорів, використовуваних у відеотехніці. Три основних кольори, комбінуючись на ЕПТ, дають вісім кольорів. Ці кольори і відповідні їм двоїчні коди приведені на рис. 3.9 *б*. Зображення, одержуване при такому способі кодування, має дуже низьку колірну роздільну здатність.

Число доступних кольорів можна збільшити, скориставшись ***таблицею кольорів***, рис. 3.10. Після зчитування з буфера кадру число, що утворилося, розглядається як індекс у таблиці кольорів. У цій таблиці міститься 2^N елементів, сукупність яких називається ***палітрою***. Кожен її елемент може містити W біт кодування кольору, причому $W > N$. У такому випадку можна одержати 2^W значень кольорів, але *одночасно* можуть бути доступні лише 2^N з них. Для одержання інших значень таблицю кольорів необхідно змінити — завантажити в неї іншу палітру. Найчастіше кольори в палітрі задаються по моделі RGB, тоді розрядність таблиці кольорів $W = 3K$, де K – розрядність кодування інтенсивності кожного каналу (звичайно $W = 3 \times 8 = 24$).

Для відображення ***реалістичних кольорових зображень*** необхідно мати можливість регулювати інтенсивність кожного з трьох основних кольорів: червоного, синього і зеленого. Для цього потрібно, щоб кожній з трьох колірних пушок відповідало кілька бітових площин.

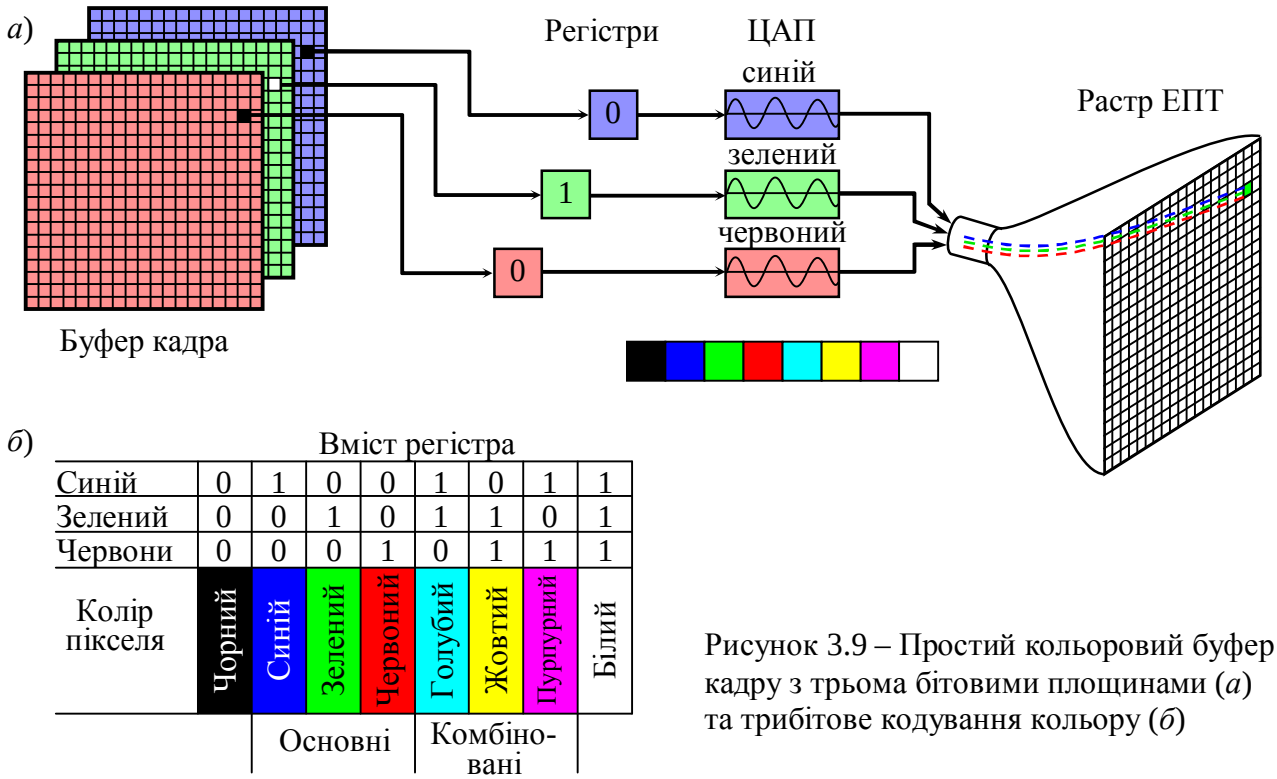


Рисунок 3.9 – Простий кольоровий буфер кадру з трьома бітовими площинами (*а*) та трибітове кодування кольору (*б*)

На рис. 3.11 показаний кольоровий буфер кадру з $K = 8$ бітовими площинами на кожен колір, тобто з $N = 8 \times 3 = 24$ бітовими площинами. Кожна група бітових площин керує 8-розрядним ЦАП і може генерувати $2^8 = 256$ рівнів інтенсивностей червоного, зеленого та синього кольорів. Їх можна скомбінувати в $2^{24} = 256^3 = 16\ 777\ 216$ кольорів. Це «*повнокольоровий*» буфер кадру.

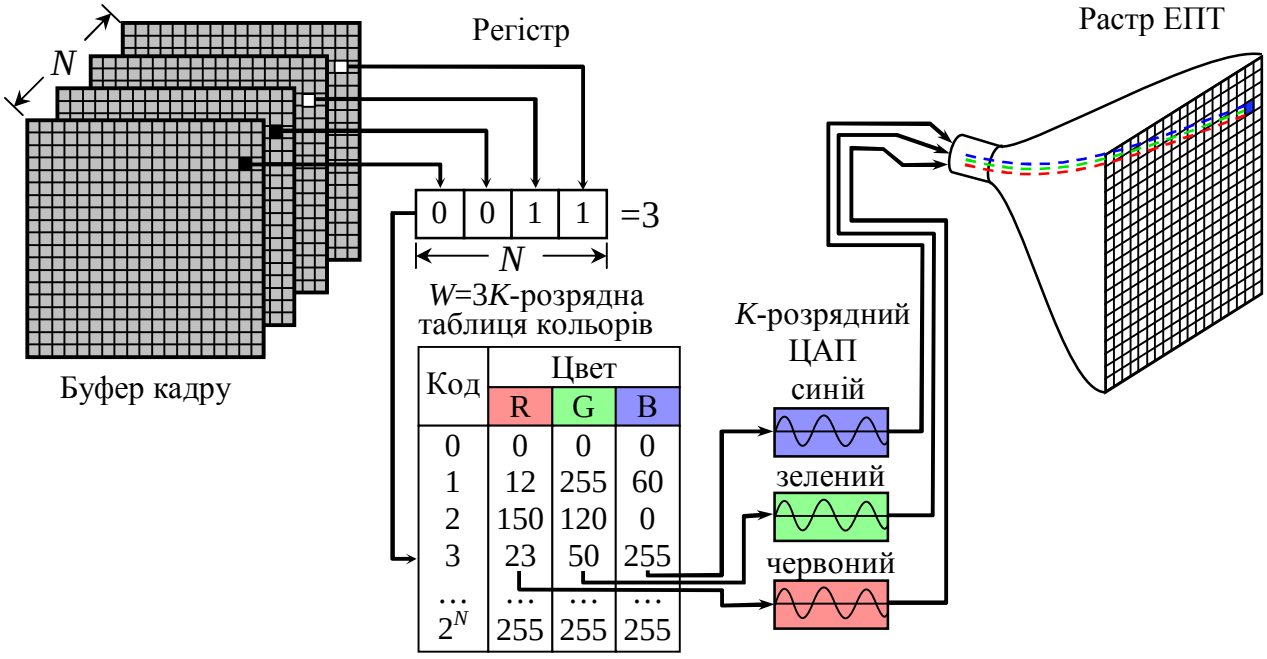


Рисунок 3.10 – Буфер кадру з N бітовими площинами та таблицею кольорів

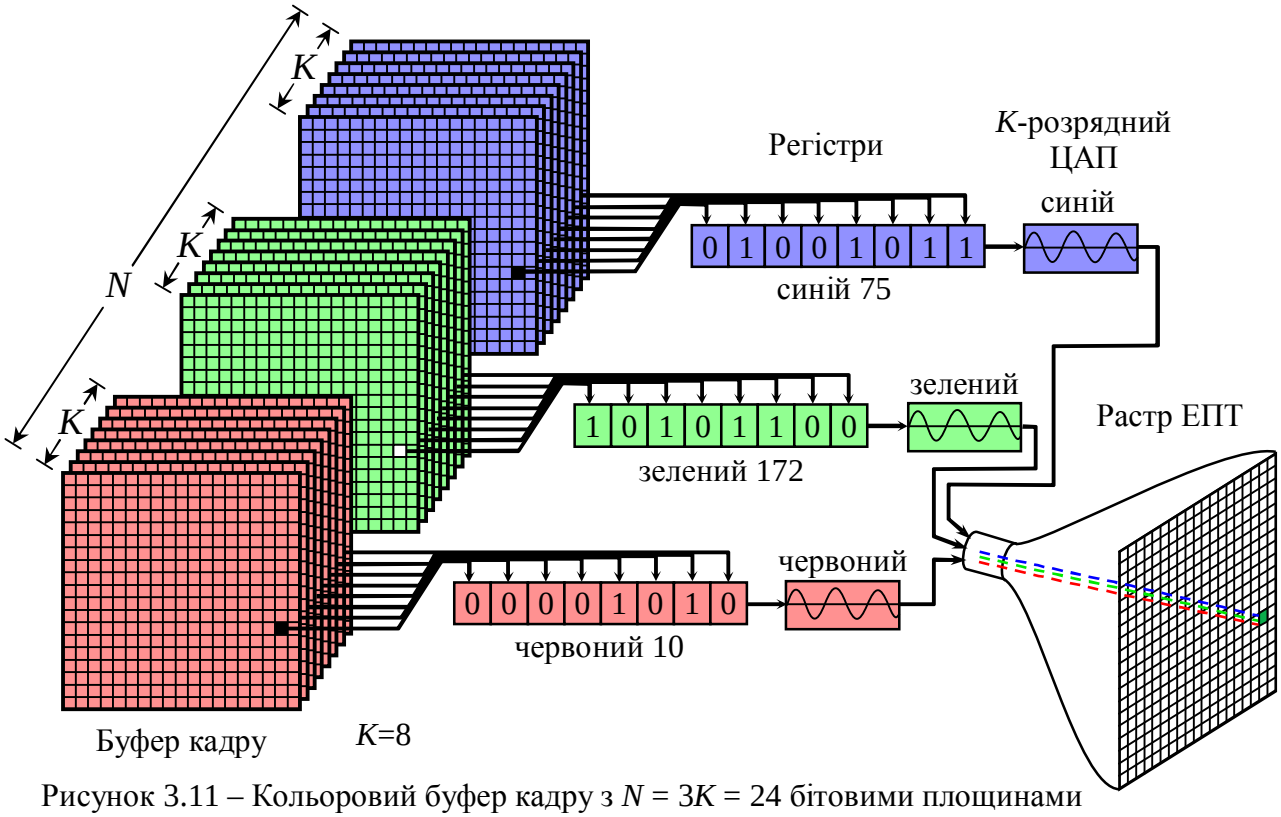


Рисунок 3.11 – Кольоровий буфер кадру з $N = 3K = 24$ бітовими площинами

3.4 Основні формати растрових графічних файлів

Для збереження і передачі растрових зображень існує багато форматів, що розрізняються структурою запису інформації про зображення, максимальним розміром растра і глибиною кольору, можливістю запису декількох зображень та додаткової інформації і т.д. Одна з основних відмінностей різних форматів, що визначає їхнє призначення й область застосування — можливість і спосіб ***стиснення інформації***. За цією ознакою растрові формати можна розділити на три групи:

— *без стиснення* інформації про зображення — растр записується у файл у такому ж виді, у якому він міститься в буфері кадру;

— зі *стисненням без утрат* — використовуються алгоритми стиснення, що не приводять до втрати інформації і дозволяють відновити зображення точно таким же, як до стиснення;

— зі *стисненням із утратами* — використовуються алгоритми стиснення, що приводять до втрати частини інформації, при цьому зображення після відновлення трохи спотворюється.

До першої групи відноситься формат ***BMP*** (bit map image), що є базовим графічним форматом *OS Windows*. Файли формату *BMP* можуть зберігати інформацію про зображення з глибиною кольору від 1 до 24 біт на піксель — двоколірних, напівтонових, кольорових з таблицею кольорів і повнокольорових. Окрім самого растра, файл має заголовок, у якому міститься інформація про розмір растра, кольоровий режим і роздільну здатність. Остання використо-

вується для виведення зображення на друк. Також для неповнокольорових зображень файл *BMP* містить таблицю кольорів — палітру.

Оскільки у форматі *BMP* відсутнє стиснення, файли обробляються з високою швидкістю і розпізнаються без помилок різними програмними пакетами, однак займають значні обсяги пам'яті — для великих повнокольорових зображень це десятки мегабайт. Це утруднює збереження і робить практично неможливу передачу по каналах зв'язку, наприклад Internet.

До другої групи відносяться формати, що використовують відносно прості методи стиснення. Прикладом може служити розглянутий вище метод *групового кодування*, який використовується у форматах ***PCX***, ***TGA*** і ***TIFF*** та дозволяє в кілька разів зменшити обсяг файлу. Такий метод стиснення найбільш ефективний для зображень типу креслень, схем — які мають великі області, зафарбовані одним кольором. Стиснення таким методом складних повнокольорових зображень (наприклад, фотографій), в яких кожен піксель відмінний від сусідніх, неефективно і може привести не до зменшення, а до збільшення обсягу файлу.

Існують інші способи стиснення без утрат, наприклад *LZ-алгоритми*, які використовують *словниковий метод стиску*. Створюється словник, що містить повторювані послідовності значень — *фрази*, які знаходяться у вихідному масиві даних. Кожна фраза одержує код (індекс) у словнику. Кодування масиву здійснюється заміною фраз відповідними індексами. Алгоритм забезпечує більший ступінь стиснення, ніж групове кодування, але працює повільніше.

LZ-алгоритм стиснення використовується у форматі ***GIF***, розробленому для обміну растровими зображеннями в мережі *Internet*. Формат підтримує зображення з глибиною кольору до 256 (8 біт на піксель) і може зберігати кілька зображень для створення анімації — послідовної зміни зображень через деякий інтервал часу.

Для стиснення зображень типу фотографій розроблені *алгоритми стиснення інформації з утратами*, що дозволяють знайти і видалити ту частину інформації, що істотно не впливає на сприйняття зображення. При цьому після відновлення зображення трохи відрізняється від вихідного, але досить «схоже» на нього, щоб людське око не знаходило різниці.

В даний момент найбільш розповсюдженим є метод ***JPEG*** (*Joint Photographic Experts Group*), розроблений у 1991 році. Сутність методу полягає в тім, що зображення:

1) перетворюється з колірної моделі *RGB* у колірну модель *YCbCr*, що використовується в кольоровому телебаченні; канал яскравості *Y* несе значно більше інформації, чим колірні канали *Cb* і *Cr*, тому останні можна піддавати найбільшому стисненню;

2) розбивається на блоки розміром 8×8 пікселів і кожен блок піддається *дискретному косинусному перетворенню* (*ДКП*), що перетворює просторовий розподіл у частотний; у результаті одержуємо блоки 8×8, кожен елемент яких є частотним коефіцієнтом спектра;

а) *б*)

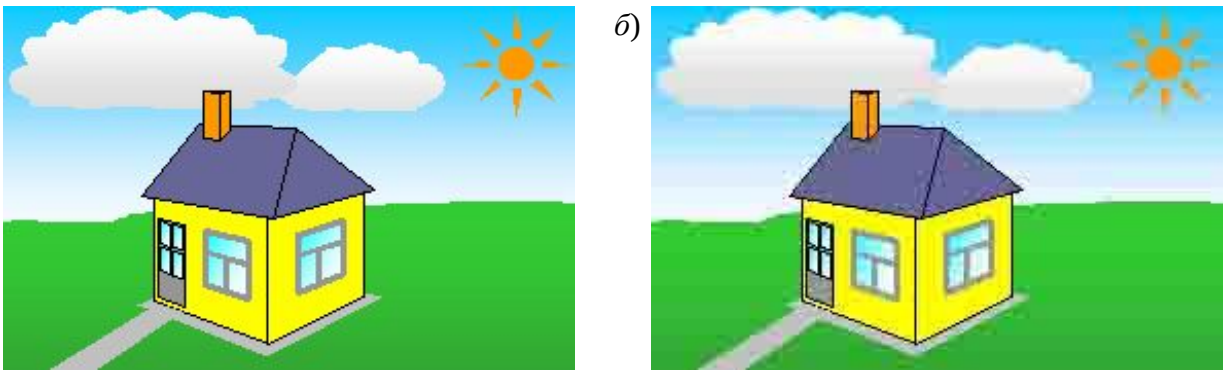


Рисунок 3.27 – Спотворення при *JPEG*-стисненні: вихідне (*а*) и стиснуте (*б*) зображення

3) інформація в блоках фільтрується шляхом ділення на матрицю квантування і наступного округлення частотних коефіцієнтів до цілого; при цьому інформація про високі частоти в спектрі обнуляється, і, таким чином, зображення «спрощується» — залишаються тільки низькі частоти, що несуть основну інформацію; більшість же елементів блоку 8×8 будуть дорівнювати нулю;

4) інформація в блоках кодується методом групового стиснення.

Метод стиснення *JPEG* дозволяє зменшити обсяг файлу в 10…30 разів, при цьому ступінь стиснення можна задавати — чим більше ступінь стиснення, тим меншим буде отриманий файл, але великими — спотворення. Недолік методу — спотворення картинки при великих степенях стиснення, що виявляються у виді помітних оку блоків — «квадратиків», а також розмитих границь і ореолів навколо контурів об'єктів, рис. 3.27. Спотворення найбільш помітні при стисненні зображень з чіткими границями — креслень, схем, тому для таких зображень використовувати метод *JPEG* не рекомендується.

Модифікований метод *JPEG* використовується також для стиснення кадрів відеозображення.

В останні часи усе більше поширення одержує метод рекурсивного стиснення зображень, називаний також хвильовим алгоритмом (*wavelet-перетворенням*). Метод вимагає більших, ніж *JPEG*, обсягів пам'яті і витрат процесорного часу на обробку, але дозволяє одержати більш якісне зображення при тому ж ступені стиснення, що і *JPEG*, оскільки не розбиває картинку на блоки. Зображення стає небагато «розмитим», але спотворення не так помітно. Хвильовий алгоритм реалізований у новому стандарті ***JPEG* 2000**, а також у деяких інших.

Дуже популярним стає формат ***DjVu***, розроблений для розміщення в *Internet* документів, сканованих у кольорі з високою роздільною здатністю. Основний принцип *DjVu* — поділ зображення на фон і передній план. Формат використовує хвильовий алгоритм для стиснення фону і зображень типу фотографій, а для тексту і графіки використовується ефективний алгоритм стиснення без утрат. Формат *DjVu* забезпечує дуже високий ступінь стиску як чорно-білих, так і кольорових зображень, при цьому спотворення практично відсутні. У файл *DjVu* може бути записано кілька зображень, що дозволяє зберігати в електронному вигляді журнали, книги.

4 МЕТОДИ Й АЛГОРИТМИ ДВОВИМІРНОЇ ГРАФІКИ

4.1 Координатний метод. Перетворення координат

Фундаментом комп'ютерної графіки є ***аналітична геометрія***, в основі якої лежить ***координатний метод*** — завдання положення точок і об'єктів на площині або в просторі за допомогою ***системи координат***. Найчастіше використовується *прямокутна* чи *Декартова* система координат, що пов'язано як з її широким застосуванням в аналітичній геометрії, так і з тим, що координати пікселів у дисплеях і друкувальних пристроях також задаються в прямокутній системі координат.

Розглянемо особливості координатного методу на площині. Положення точки в цьому випадку задається двома числами — координатами x і y, рис. 4.1 *а*. Положення деякого об'єкта може бути задано координатами деякої базової точки, що є центром його власної (*локальної*) системи координат, і кутом повороту цієї системи координат щодо *глобальної системи координат*, зв'язаної з екраном дисплея або листом папера в принтері. Перерахування координат точок з однієї в іншу систему називається ***перетворенням координат***.

Нехай існує деякий об'єкт, який має свою систему координат (x, y), рис. 4.1 *б*. Форма об'єкта визначається набором точок, положення яких задане координатами в системі (x, y). Для того, щоб відобразити його в деякім положенні на екрані, листі чи папера в якомусь більш складному зображенні, необхідно перерахувати координати цих точок у глобальну систему координат (X, Y). Нехай положення об'єкта задане координатами центра його локальної системи координат (X_0, Y_0) і кутом повороту α. Розглянемо три окремі випадки перетворення координат.

1. ***Зсув*** — центр локальної системи координат зміщений в точку (X_0, Y_0), кут повороту дорівнює нулю, рис. 4.1 *в*. Координати деякої точки A об'єкта в новій системі координат складуть:

$$\begin{cases} X = x + X_0, \\ Y = y + Y_0, \end{cases} \tag{4.1}$$

де x, y – координати точки A у системі координат об'єкта.

2. ***Поворот*** навколо центра координат локальної системи на кут α, рис. 4.1 *г*:

$$\begin{cases} X = x\cos\alpha - y\sin\alpha, \\ Y = x\sin\alpha + y\cos\alpha. \end{cases} \tag{4.2}$$

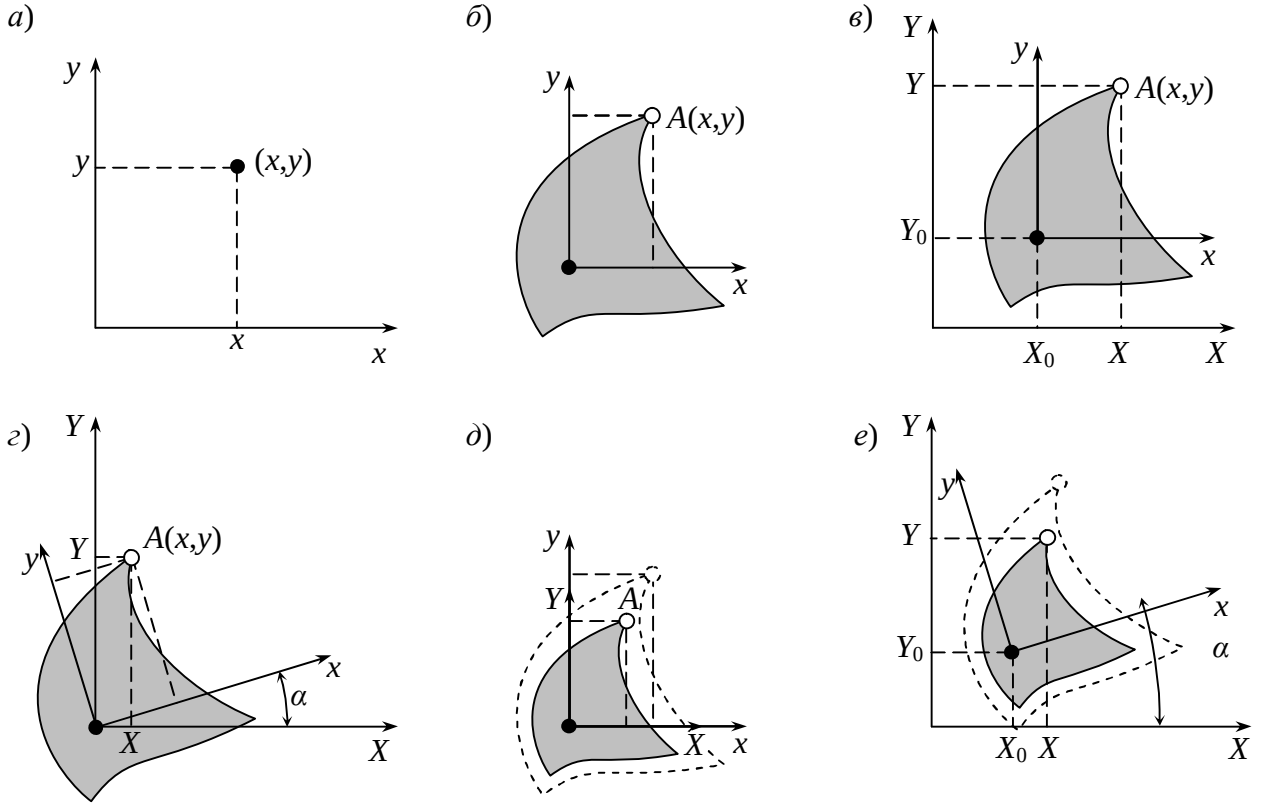


Рисунок 4.1 – Координатний метод та перетворення координат

3. ***Масштабування*** (*стиск/розтягання*) з центром на початку координат — збільшення або зменшення об’єкта, обумовлене коефіцієнтами масштабування k_x і k_y, рис. 4.1 *д*:

$$\begin{cases} X = x\,k_x, \\ Y = y\,k_y. \end{cases} \tag{4.3}$$

Якщо $k_x = k_y$ — об’єкт масштабуєтсья пропорційно; у протилежному випадку відбувається деформація об’єкта — неоднакове розтягання або стиск по різних напрямках.

Як видно з рис. 4.1 *е*, послідовно застосовуючи масштабування, поворот і зсув координат, можна розташувати деякий об’єкт (геометричний примітив або фрагмент зображення) у довільному положенні. При цьому формули для перерахування координат матимуть вигляд:

$$\begin{cases} X = x\,k_x \cos\alpha - y\,k_y \sin\alpha + X_0, \\ Y = x\,k_x \sin\alpha + y\,k_y \cos\alpha + Y_0. \end{cases} \tag{4.4}$$

Подібні перетворення координат в аналітичній геометрії називають *афінними*. У загальному виді вони можуть бути записані у виді системи рівнянь

$$\begin{cases} X = A\,x + B\,y + C, \\ Y = D\,x + E\,y + F. \end{cases} \tag{4.5}$$

Перетворення координат зручно виконувати в матричному виді. Константи *A*, *B*, …, *F* утворять матрицю перетворення, яка, будучи помноженою на матрицю-стовпець координат (*x*, *y*), дає матрицю-стовпець (*X*, *Y*). Щоб врахувати константи *C* і *F*, необхідно перейти до так званих *однорідних координат* — додати ще один рядок зі значенням, рівним 1. Тоді (4.5) можемо записати у виді

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} A & B & C \\ D & E & F \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{4.6}$$

або

$$\boldsymbol{W} = \boldsymbol{A}\,\boldsymbol{w}, \tag{4.7}$$

де $\boldsymbol{w}$ і $\boldsymbol{W}$ – матриці координат у вихідній і новій системах; $\boldsymbol{A}$ – матриця перетворення. Наприклад для операцій зсуву, повороту і масштабування одержимо матриці відповідно

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & X_0 \\ 0 & 1 & Y_0 \\ 0 & 0 & 1 \end{bmatrix}; \tag{4.8}$$

$$\boldsymbol{B} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}; \tag{4.9}$$

$$\boldsymbol{C} = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.10}$$

Тоді перетворення (4.4) запишемо як

$$\boldsymbol{W} = \boldsymbol{C}\,\boldsymbol{B}\,\boldsymbol{A}\,\boldsymbol{w}. \tag{4.11}$$

4.2 Алгоритми виведення ліній

Багато сучасних програм (векторні графічні редактори, САПР, математичні й інші спеціальні програми) працюють із зображеннями, представленими деякими *математичними моделями*. Найчастіше такі зображення складені зі стандартного набору простих елементів, називаних ***графічними примітивами***. Це можуть бути прямі і криві лінії, фігури (прямокутники, окружності, еліпси) і т.п. Для того, щоб описати зображення, яке складається з примітивів, достатньо перелічити ці примітиви, указати їхні параметри (колір, товщину і стиль ліній, спосіб заливання фігур і ін.) і взаємне розташування.

Оскільки практично всі сучасні засоби відображення графіки (дисплеї, принтери) растрові, для відображення картинки, заданої у векторному виді, необхідно перетворити її в *растр*. Таке перетворення звичайно виконується тією програмою, у якій зображення було створено, з використанням алгоритмів ***растеризації***.

Для того, щоб перетворити в растр лінію, необхідно змінити колір пікселів, по яких ця лінія проходить, рис. 4.2 *а*. Щоб це зробити, необхідно одержати рівняння лінії. Очевидно, найбільш простим графічним примітивом є ***відрізок прямої***, який можна задати координатами двох точок: початку (x_1, y_1) і кінця (x_2, y_2). Для довільної точки (x, y), що лежить на відрізку прямої, має виконуватися умова

$$\frac{x - x_1}{y - y_1} = \frac{x_2 - x_1}{y_2 - y_1}. \tag{4.12}$$

Звідси можемо одержати рівняння прямої як функцію y від x або навпаки:

$$y = y_1 + (x - x_1)\frac{y_2 - y_1}{x_2 - x_1}. \tag{4.13}$$

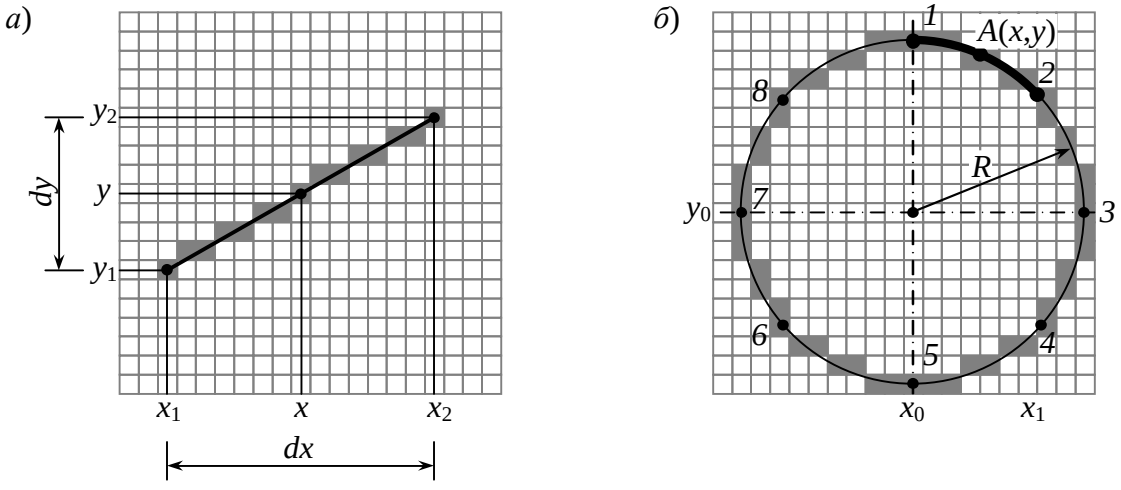


Рисунок 4.2 – Растеризація відрізка прямої (*а*) та окружності (*б*)

$$x = x_1 + (y - y_1)\frac{x_2 - x_1}{y_2 - y_1}. \qquad (4.14)$$

Очевидно, що дріб у рівнянні являє собою кутовий коефіцієнт і при побудові може бути обчислена один раз. Для растеризації відрізка слід з'ясувати, яка з величин $dx = x_2 - x_1$ та $dy = y_2 - y_1$ є більшою, і побудувати «графік» функції $y = f_1(x)$ у першому або $x = f_2(y)$ у другому випадку. Виконується простий цикл по відповідній координаті від початкової до кінцевої точки з кроком в один піксель. Інша координата визначається розрахунком по (4.13) або (4.14) і округляється до цілого значення; піксель з отриманими координатами (x, y) зафарбовується заданим кольором.

Для точок ***окружності*** виконується рівняння

$$(x - x_0)^2 + (y - y_0)^2 = R^2, \qquad (4.15)$$

де R – радіус окружності, а (x_0, y_0) – координати її центра, рис. 4.2 *б*. Перетворивши цей вираз, одержимо залежність

$$y = y_0 \pm \sqrt{R^2 - (x - x_0)^2}, \qquad (4.16)$$

Оскільки окружність симетрична щодо вертикальної і горизонтальної осей, які проходять через її центр, необов'язково розраховувати всі точки окружності. Досить виконати розрахунок для дуги *1-2* — октанта (1/8) окружності, — для чого виконується цикл по координаті x від x_0 до x_1 із кроком в один піксель, обчислюються й округляються до цілого відповідні значення y. Потім зафарбовується піксель $A(x, y)$ і ще сім пікселів, розташованих на дугах *2-3*, *3-4*, …, *8-1*.

Для побудови ***еліпса*** існує два способи. Перший — аналогічний вищеописаному, при цьому використовується рівняння еліпса

$$\frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} = 1, \qquad (4.17)$$

де a і b – половини довжин осей еліпса. Розрахунок виконується не для октанта, а для всього квадранта, інші 3 квадранти будуються симетрично.

Другий спосіб — використання алгоритму побудови окружності з наступним масштабуванням — «розтягуванням» окружності пропорційно довжинам осей еліпса.

Описані вище способи побудови ліній прості для розуміння, але їхня реалізація пов'язана з використанням операцій із комою, що плаває, множення і ділення, добування квадратного кореня. Такі операції виконуються процесором відносно повільно, що приводить до низької швидкості відображення, особливо на малопотужних ЕОМ.

Існують ***інкрементні алгоритми растеризації***, називані алгоритмами ***Брезенхема***, що дозволяють будувати відрізки прямих, окружностей, еліпсів і інших кривих, використовуючи тільки операції цілочисельного додавання і віднімання. Сутність алгоритму Брезенхема проілюструємо на прикладі побудови відрізка, рис. 4.3:

1. Визначаємо величини збільшень: $dx = x_2 - x_1$; $dy = y_2 - y_1$.

2. Вибираємо більше зі збільшень: $dx > dy$, значить $d = dx$.

3. Уводимо поточні координати x і y, а також допоміжні перемінні ex і ey. До початку циклу приймаємо $x = x_1$; $y = y_1$; $ex = 0$; $ey = 0$. Зафарбовуємо піксель з координатами (x, y).

4. Виконуємо d раз цикл, у якому перемінні ex і ey збільшуємо відповідно на dx і dy та перевіряємо:

— якщо $ex > d$ — виконуємо $ex = ex - d$ і $x = x + 1$;

— якщо $ey > d$ — $ey = ey - d$ і $y = y + 1$;

одержуємо координати (x, y) наступного пікселя; зафарбовуємо піксель (x, y).

Аналогічні, але дещо складніші алгоритми використовуються для побудови окружностей, еліпсів, дуг. Як видно, у приведеному алгоритмі застосовані лише операції додавання, віднімання і порівняння. Оскільки останні виконуються у багато разів швидше операцій із комою, що плаває, такі алгоритми забезпечують високу швидкість відображення навіть на малопотужних ЕОМ.

До недоліків алгоритму можна віднести деяку «неточність» засвітлення пікселів, що видно з рис. 4.3, а також те, що опорні точки відрізків і інших фігур можуть мати лише цілочисельні координати (середини відповідних пікселів).

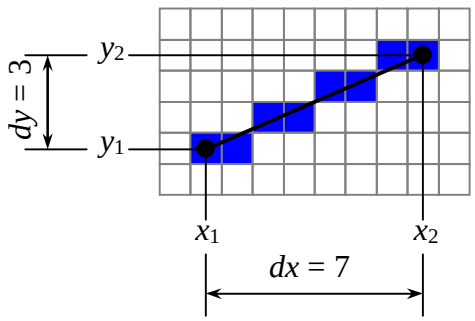


Рисунок 4.3 – Растеризація відрізку прямої алгоритмом Брезенхема

4.3 Криві Безьє і NURBS

Розроблені математиком *П'єром Безьє*, ***криві*** і ***поверхні Безьє*** були використані в 60-х роках компанією «Рено» для комп'ютерного проектування форми кузовів автомобілів. Зараз вони широко використовуються в комп'ютерній графіці та моделюванні.

Криві Безьє описуються в *параметричній формі*: $\{x = f_x(t),\ y = f_y(t)\}$. Значення t виступає як параметр, якому відповідають координати окремої точки лінії. Параметрична форма опису може бути зручніше для деяких кривих, ніж завдання у виді функції $в = f(x)$, оскільки функція $f(x)$ може бути набагато складніше, ніж $f_x(t)$ і $f_y(t)$. Крім того $f(x)$ може бути неоднозначною. Наприклад рівняння окружності в параметричному виді:$\{x = x_0 + R\cos\varphi;\ y = y_0 + R\sin\varphi\}$, де φ – деякий кут. Порівняєте з вираженням (4.16).

Багаточлени Безьє взагалі мають такий вид:

$$\begin{cases} x(t) = \sum_{i=0}^{m} C_i t^i (1-t)^{m-i} x_i; \\ y(t) = \sum_{i=0}^{m} C_i t^i (1-t)^{m-i} y_i, \end{cases} \tag{4.18}$$

де x_i і y_i – координати точок-орієнтирів P_i, а величини C_i – коефіцієнти бінома Ньютона. Значення m можна розглядати і як ступінь полінома, і як значення, що на одиницю менше кількості точок-орієнтирів. Параметр t задає точку на кривій — у початковій точці кривої $t = 0$, у кінцевій $t = 1$, кожному значенню t у діапазоні від 0 до 1 відповідає деяка точка на кривій.

Як видно з виражень, вид кривої Безьє при заданому ступені m однозначно визначається розташуванням точок-орієнтирів P_i. Саме ця особливість робить криві Безьє універсальним і надзвичайно зручним графічним примітивом.

Розглянемо криві Безьє, класифікуючи їх за значеннями m і кількістю точок-орієнтирів.

m = 1 (по двох точках):

$$\begin{cases} x(t) = (1-t)\, x_0 + t\, x_1; \\ y(t) = (1-t)\, y_0 + t\, y_1. \end{cases} \tag{4.19}$$

Крива вироджується у відрізок прямої лінії, що визначається кінцевими точками P_0 і P_1, як показано на рис. 4.4 *а*.

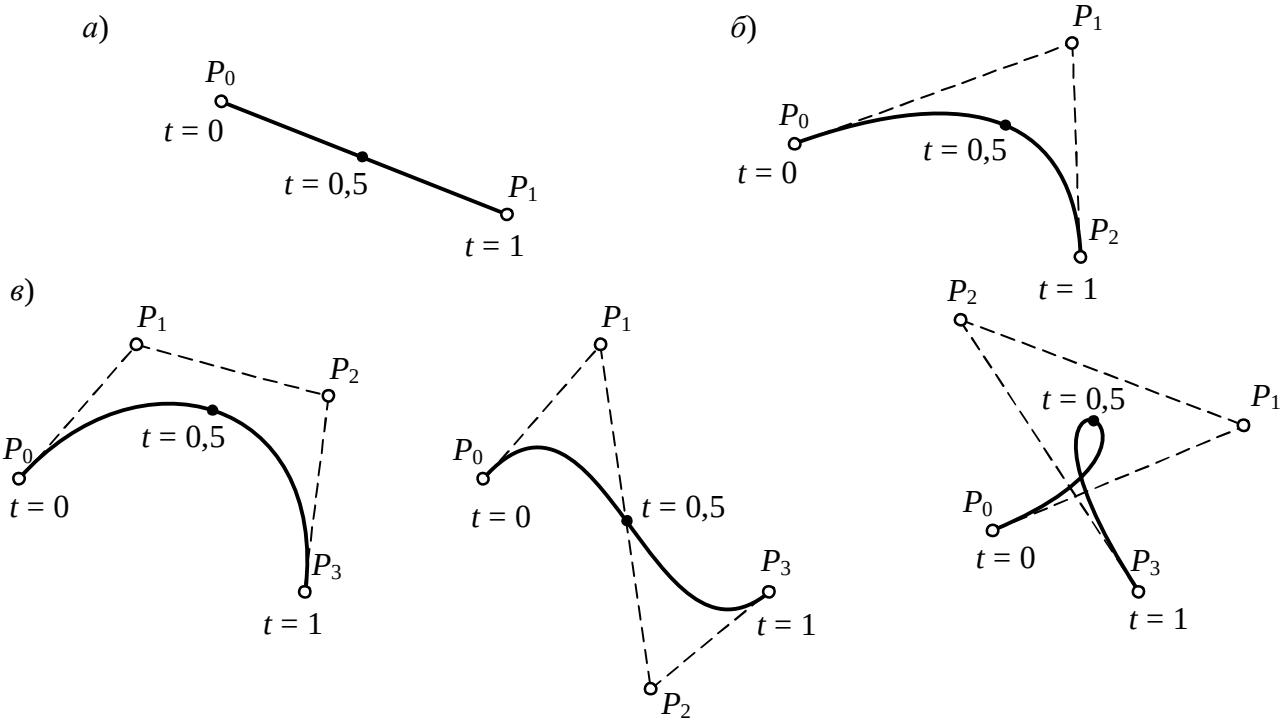


Рисунок 4.4 – Криві Безьє першого (*а*), другого (*б*) і третього (*в*) порядків

m = 2 (по трьох точках):

$$\begin{cases} x(t) = (1-t)^2 x_0 + 2t(1-t) x_1 + t^2 x_2; \\ y(t) = (1-t)^2 y_0 + 2t(1-t) y_1 + t^2 y_2. \end{cases} \tag{4.20}$$

Одержуємо квадратичну криву, показану на рис. 4.4 *б*.

m = 3 (по чотирьох точках):

$$\begin{cases} x(t) = (1-t)^3 x_0 + 3t(1-t)^2 x_1 + 3t^2(1-t) x_2 + t^3 x_3; \\ y(t) = (1-t)^3 y_0 + 3t(1-t)^2 y_1 + 3t^2(1-t) y_2 + t^3 y_3. \end{cases} \tag{4.21}$$

Крива Безьє 3-го порядку може мати різноманітну форму, рис. 4.4 *у*, що дозволяє використовувати її для побудови різних ліній, фігур, контурів довільної форми. Криві Безьє є основними графічними примітивами більшості векторних графічних редакторів «художнього» призначення (наприклад, Corel Draw, Adobe Illusttrator), а також використовуються в інженерних і архітектурних пакетах (AutoCAD, ArchiCAD, КОМПАС-3D) для моделювання об'єктів складної форми.

Для побудови кривої Безьє можна використовувати залежності (4.20) і (4.21), розрахувавши координати для набору значень *t* від 0 до 1 з деяким кроком. Однак існує алгоритм побудови кривих Безьє, не потребуючий виконання складних обчислень, зведення в ступінь. Це *геометричний алгоритм*, який дозволяє знайти координати (*x*, *y)* точки кривої Безьє за значенням параметра *t*:

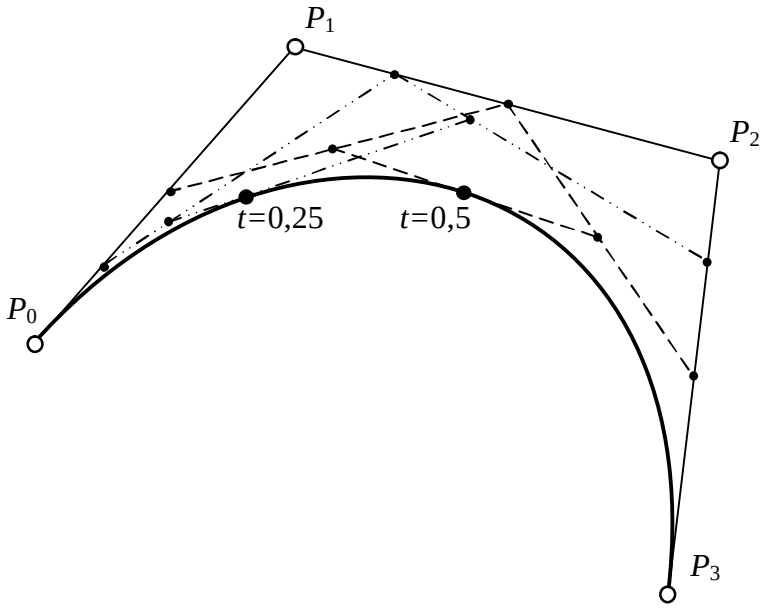


Рисунок 4.5 – Геометричний алгоритм побудови кривих Безьє

1. Кожне ребро ламаної лінії, що проходить по точках-орієнтирах, поділяється пропорційно значенню t.

2. Точки розподілу з'єднуються відрізками прямих і утворюють нову ламану лінію. Кількість вузлів нової ламаної на одиницю менше, ніж кількість вузлів попередньої.

3. Сторони нової ламаної лінії знову поділяються пропорційно значенню t. Так продовжують доти, доки не буде отримана єдина точка розподілу. Ця точка і буде точкою кривої Безьє.

На рис. 4.5 показані побудови для t = 0,5 (пунктирна лінія) і для t = 0,25 (штрих-штрих-пунктирна лінія). Алгоритм знаходження середини відрізка дуже простий, виконується за допомогою простих цілочисельних операцій, тому описаний метод може бути легко реалізований на ЕОМ і забезпечує високу швидкість побудови.

Для створення складних кривих ліній використовуються ***полілінії Безьє*** — декілька кривих Безьє, що зістиковані разом і утворюють єдиний графічний примітив, рис. 4.6. Полілінії Безьє можуть бути як *розімкнутими*, так і *замкнутими*. В останньому випадку одержуємо ***контур*** із криволінійними границями, який може бути, наприклад, заповнений деяким кольором за допомогою алгоритму заповнення.

Точки, у яких виконується стикування окремих кривих Безьє (точки P_0, P_3, P_6 і P_9 на рис. 4.6.) називаються ***вузлами*** полілінії. У залежності від способу стикування сусідніх кривих вузли полілінії Безьє можуть бути:

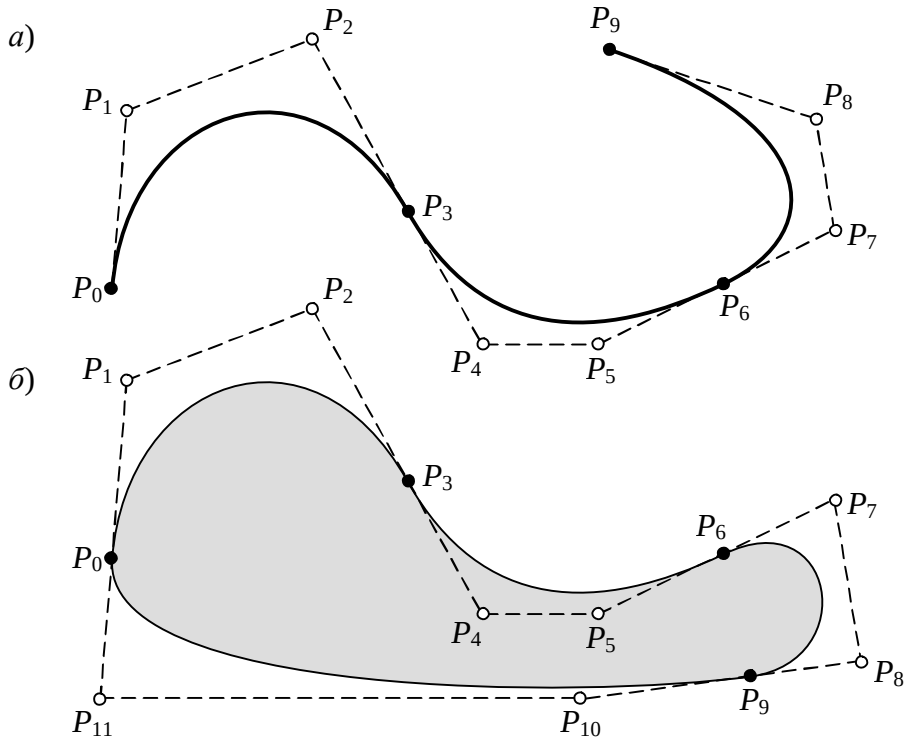


Рисунок 4.6 – Розімкнена (*а*) і замкнена (*б*) полілінії Безьє

— ***гладкими***, (стикування кривих з вирівнюванням по першій і другій похідних) — у таких вузлах виконуються умови збігу напрямків дотичних до обох кривих і однакових радіусів кривизни обох кривих у точці стикування, рис. 4.7 *а* ;

— ***прямими*** (стикування з вирівнюванням по першій похідній) — виконується умова збігу напрямків дотичних до обох кривих у точці стикування, радіуси кривизни можуть бути різними, рис. 4.7 *б*;

— ***кутовими*** (довільне стикування) — і напрямки дотичних до обох кривих і радіуси кривизни можуть бути різними, рис. 4.7 *в*.

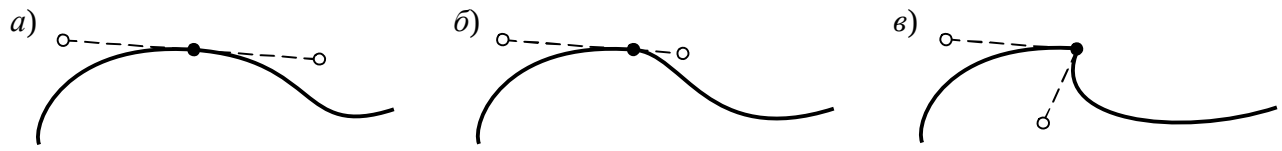


Рисунок 4.7 – Гладкий (*а*), прямий (*б*) та кутовий (*в*) вузли полілінії Безьє

Узагальнення методів Безьє і B-сплайнів на початку 70-х років дозволило одержати один з найпотужніших і універсальних засобів геометричного моделювання криволінійних контурів — ***NURBS-технологію***.

Абревіатура ***NURBS*** позначає ***Non-Uniform Rational B-Splines***, тобто ***неоднорідні раціональні B-сплайни***. Це математичні об'єкти для завдання двовимірних кривих і гладких поверхонь у тривимірному просторі.

На відміну від кривих Безьє, *NURBS-криві* можуть бути задані ***будь-яким числом точок-орієнтирів***, можуть бути як *розімкненими*, так і *замкненими*, рис. 4.8. Точки-орієнтири кривих або поверхонь NURBS можуть мати різні ***ваги***, тобто в різному ступені «притягати» до себе криву (на що вказує слово «*неоднорідні*» у назві кривих).

Через свою гнучкість і точність ***NURBS-моделі*** можуть використовуватися в будь-якому процесі ілюстрації, анімації і промислового дизайну. Більшість сучасних САПР, систем комп'ютерної графіки й анімації підтримують моделювання з використанням *NURBS-кривих* і поверхонь, оскільки:

— за допомогою NURBS простіше імітувати поверхні природних об'єктів або об'єктів, поверхні яких мають складним образом скривлені профілі (наприклад, обведення корпуса автомобіля);

— NURBS-моделі забезпечують кращу якість візуалізації об'єктів завдяки розбивці на грані, виконуваній з використанням аналітичних виражень (слово «*раціональні*» означає, що об'єкт *NURBS* може бути описаний за допомогою математичних формул).

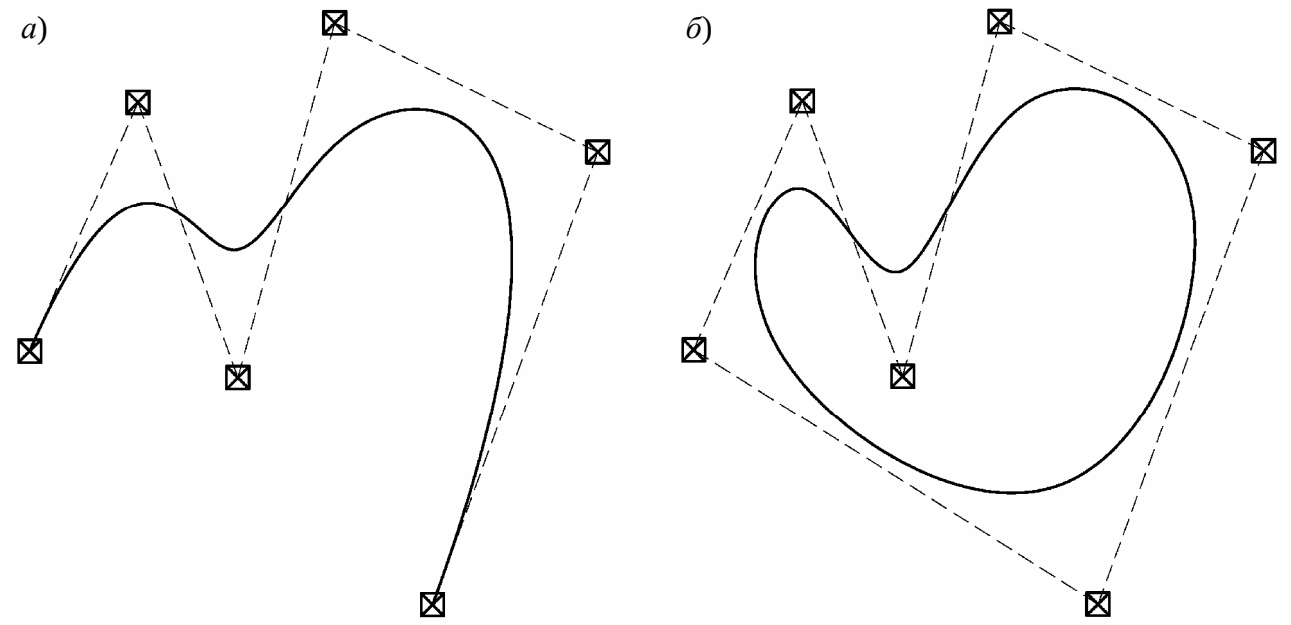


Рисунок 4.8 – Розімкнена (*а*) і замкнена (*б*) *NURBS*-криві

4.4 Алгоритми зафарбовування фігур

Фігурою будемо вважати плоский геометричний об'єкт, що складається з ліній ***контуру*** і точок ***заповнення***, що містяться усередині контуру. Контурів може бути декілька — наприклад, якщо об'єкт має усередині порожнечі.

Графічне виведення фігур поділяється на дві задачі: виведення контуру і виведення точок заповнення. Оскільки контур являє собою лінію, то виведення контуру проводиться на основі алгоритмів виведення ліній. У залежності від складності контуру, це можуть бути відрізки прямих, кривих або довільна послідовність сусідніх пікселів.

Для виведення точок заповнення відомі методи, що розділяються в залежності від використання контуру на два типи:

— алгоритми зафарбовування від внутрішньої точки до *границь довільного контуру*;

— алгоритми, що використовують *математичний опис контуру*.

Розглянемо ***алгоритми зафарбовування довільного контуру***, який вже намальовано у растрі. Спочатку визначаються координати довільного пікселя, що знаходиться усередині обкресленого контуру фігури. Колір цього пікселя змінюємо на потрібний колір заповнення. Потім виконується аналіз кольорів усіх сусідніх пікселів. Якщо колір деякого сусіднього пікселя не дорівнює кольору границі або кольору заповнення, то колір цього пікселя змінюється на колір заповнення. Потім аналізується колір пікселів, сусідніх з попередніми. І так далі, поки всередині контуру всі пікселі не перефарбуються в колір заповнення. Пікселі контуру утворюють границю, за яку не можна виходити в ході послідовного перебору всіх сусідніх пікселів.

Алгоритм зафарбовування лініями одержав широке поширення в комп'ютерній графіці. Сутність його полягає в тому, що на кожному кроці зафарбовування малюється горизонтальна лінія, що розміщається між пікселями, які мають колір контуру. Потім проводиться аналіз кольору пікселів на сусідніх рядках, і якщо колір пікселів не дорівнює кольору пікселів контуру, процедура заповнення лінії викликається знову для наступної рядка. Зафарбовування починається в деякій точці *A*, координати якої потрібно задати при виклику процедури, і закінчується, коли всередині контуру не залишається жодного незафарбованого пікселя, рис. 4.9.

Основна область застосування алгоритмів зафарбовування довільного контуру — растрові графічні редактори (*Paint, Adobe Photoshop* і ін.), де процес обробки зображення (зокрема, завдання початкової точки), контролюється людиною. Для використання в програмах, які виконують растеризацію векторних зображень, такі алгоритми занадто повільні і не забезпечують стовідсоткової гарантії правильного зафарбовування всього контуру. Алгоритми заповнення, застосовувані для таких цілей, використовують ***математичний опис контуру***.

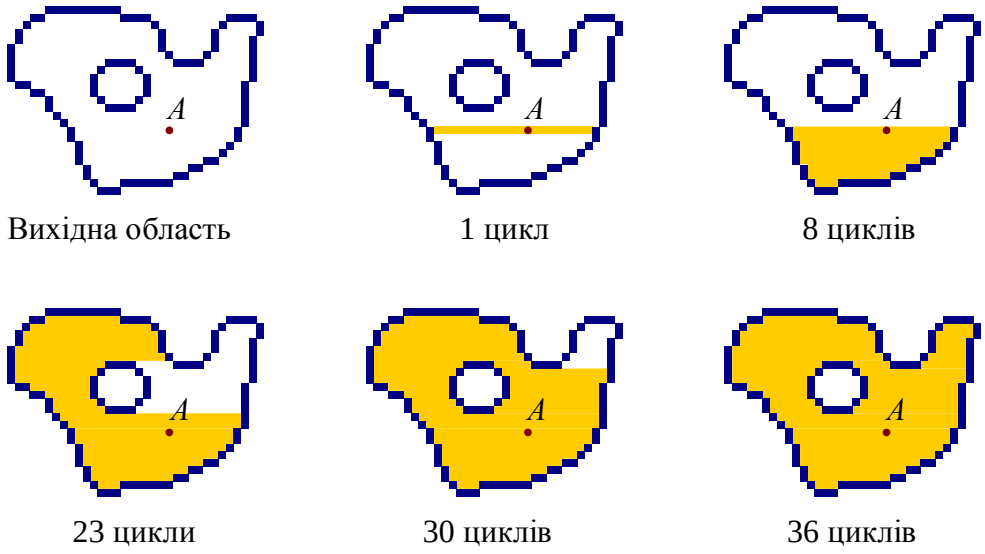


Рисунок 4.9 – Зафарбовування області за допомогою алгоритму зафарбовування линями

Математичним описом контуру фігури може служити рівняння $y = f(x)$ для контуру окружності, еліпса або іншої кривої. Для багатокутника (*полігона*) контур задається множиною координат вершин (x_i, y_i). Можливі й інші форми опису контуру. Загальним для розглянутих нижче алгоритмів є те, що для генерації точок заповнення не потрібні попередньо сформовані в растрі пікселі границі контуру фігури. Контур може взагалі не малюватися в растрі ні до, ні після заповнення.

Серед усіх фігур найпростіше заповнювати ***прямокутник***. Якщо прямокутник заданий координатами протилежних кутів, наприклад, лівого верхнього (x_1, y_1) і правого нижнього (x_2, y_2), тоді алгоритм може складатися в послідовному малюванні горизонтальних ліній заданого кольору рис. 4.10 *а*.

Для заповнення ***кола*** можна використовувати алгоритм виведення контуру, розглянутий вище. У процесі виконання цього алгоритму послідовно обчислюються координати пікселів контуру в границях одного октанта. Для заповнення слід виводити горизонталі, що з'єднують пари точок на контурі, розташовані симетрично щодо осі *y*, рис. 4.10 *б*. Так само можна створити й алгоритм заповнення еліпса.

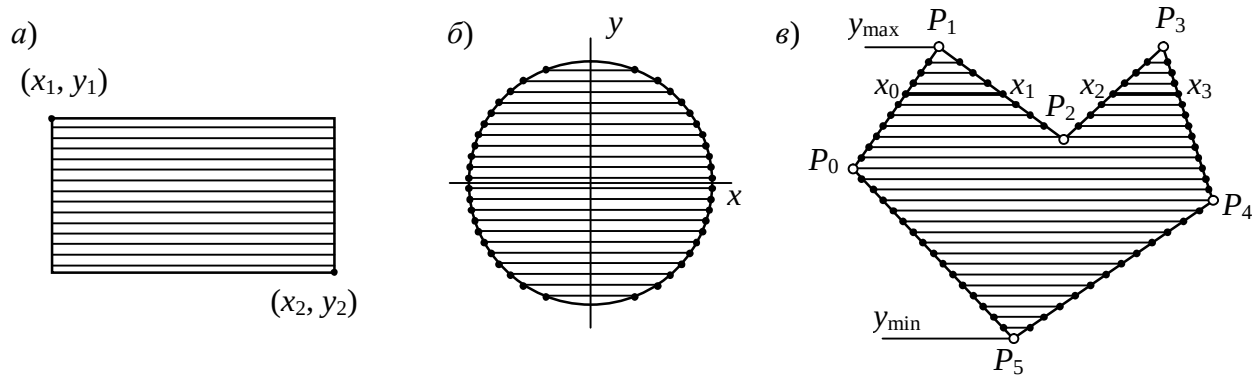


Рисунок 4.10 – Заповнення прямокутника (*а*), кола (*б*) та полігона (*в*)

Контур ***полігона*** визначається вершинами, що з'єднані відрізками прямих — ребрами, рис. 4.10 *в*. Розглянемо один з найбільш популярних алгоритмів заповнення полігона. Його основна ідея — зафарбовування фігури відрізками горизонтальних ліній. Алгоритм представляє собою цикл уздовж осі *y*. Протягом цього циклу виконується пошук точок перетинання лінії контуру з відповідними горизонталями. В алгоритмі використана топологічна властивість контуру фігури, яка полягає в тому, що будь-яка пряма лінія перетинає будь-який замкнений контур парну кількість разів.

Для визначення координат точок перетинання для кожної горизонталі необхідно проаналізувати всі ребра, вершини яких лежать по різні боки цієї горизонталі. Координата x перетинання ребра $P_i - P_{i+1}$ з горизонталлю y дорівнює

$$x = x_i + (y_{i+1} - y)\frac{x_{i+1} - x_i}{y_{i+1} - y_i}. \qquad (4.22)$$

Отриманий у результаті перебору ребер масив точок перетинання упорядковується по зростанню, (x_0, x_1, x_2, x_3, рис. 4.10 *в*.) і розбивається на пари, які з'єднуються відрізками горизонтальних ліній.

Приведені вище алгоритми заповнення можуть бути використані не тільки для малювання фігур. На основі алгоритмів заповнення можуть бути розроблені алгоритми для інших цілей. Наприклад, для визначення площі фігури (якщо вважати площу пропорційною кількості пікселів заповнення). Або, наприклад, алгоритм для пошуку об'єктів по внутрішній точці — ця операція часто використовується у векторних графічних редакторах.

4.5 Стиль лінії. Перо. Алгоритми виведення товстої лінії

Можливості алгоритмів, описаних у п. 4.2, обмежені виведенням лінії товщиною в 1 піксель. Очевидно, що для побудови якісних зображень цього недостатньо. Необхідно мати можливість будувати лінії довільної ширини.

Для опису різних по виду зображень на основі ліній використовують терміни ***стиль ліній*** і ***перо***. Термін *перо* іноді робить більш зрозумілою суть алгоритму виведення ліній для деяких стилів — особливо для товстих ліній. Наприклад, якщо для тонкої безупинної лінії перо відповідає одному пікселю, то для товстих ліній перо можна уявити собі як фігуру або відрізок лінії, що сковзає уздовж осі, залишаючи за собою слід, рис. 4.11.

Узявши за основу будь-який алгоритм виведення звичайних тонких ліній (наприклад, алгоритм Брезенхема), його можна модифікувати для виведення товстої лінії, якщо замість малювання окремого пікселя з координатами (x, y) виконувати виведення у цьому місці фігури або лінії, що відповідають формі пера — прямокутника, кола, відрізка прямої, іншої фігури.

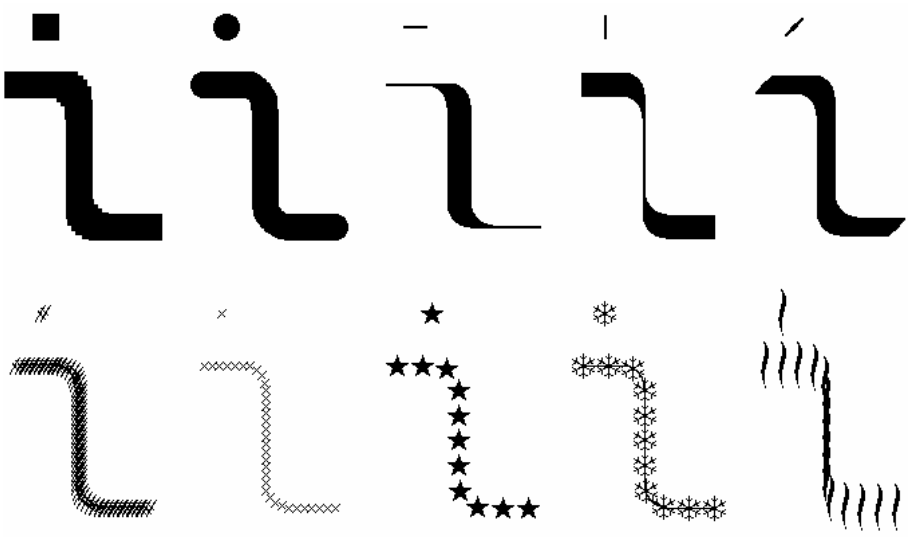


Рисунок 4.11 – Лінія, що нарисована з різними формами пера

Такий підхід до розробки алгоритмів виведення товстих ліній має переваги і недоліки. Перевага в простоті — можна майже без змін використовувати уже відомі ефективні алгоритми для обчислення координат точок лінії осі, наприклад, алгоритми Брезенхема. Недолік — неефективність для деяких форм пера. Для пер, що відповідають фігурам із заповненням, кількість тактів роботи алгоритму пропорційна квадрату товщини лінії. При цьому робиться багато зайвої роботи — більшість пікселів багаторазово зафарбовується в тих самих точках рис. 4.12 *а*, *б*.

Такі алгоритми більш ефективні для пер у вигляді відрізків ліній. У цьому випадку будь-який піксель малюється тільки один раз. Але тут важливий нахил зображуваної лінії. Ширина пера залежить від нахилу рис. 4.12 *в*, *г*, *д*. Зауважимо наприклад, що горизонтальне перо не може малювати товсту горизонтальну лінію.

Для виведення товстих ліній за допомогою пера — відрізка лінії найчастіше використовуються відрізки горизонтальної або вертикальної лінії, рідше — діагональні відрізки під кутом 45 градусів. Доцільність використання такого способу визначається великою швидкістю виведення горизонтальних і

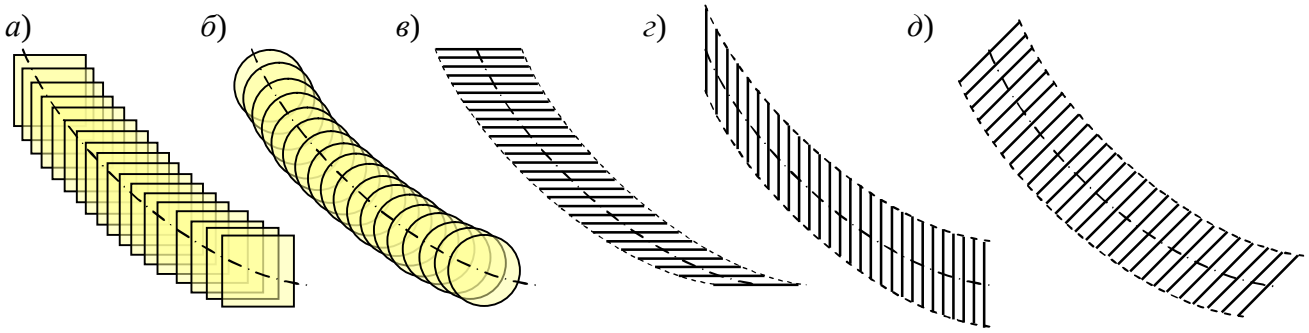


Рисунок 4.12 – Перья прямоугольной (*а*) и круглой (*б*) формы, перья в виде горизонтального (*в*), вертикального (*г*) и наклонного (*д*) отрезков

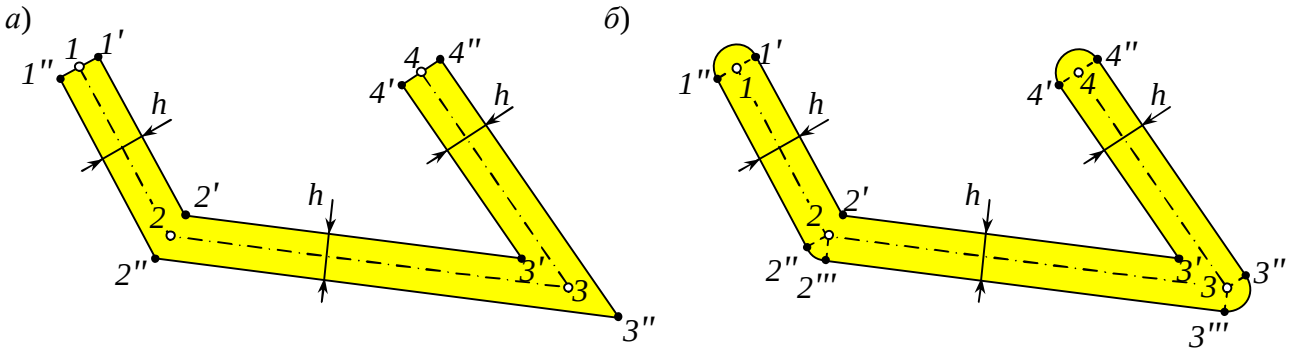


Рисунок 4.13 – Малювання товстої лінії методом заповнення полігона (*а*) та фігури з криволінійними елементами (*б*)

вертикальних відрізків прямої. При цьому щоб мінімізувати кількість тактів виведення, товсті лінії, які ближчі за нахилом до вертикальних, малюють горизонтальним пером, а лінії, що ближче за нахилом до горизонталі, малюють вертикальним пером.

У сучасних програмах, які здійснюють растеризацію векторних зображень, для малювання товстих ліній використовують *алгоритми заповнення фігур*. Наприклад товсту ламану лінію можна представити у виді полігона, рис. 4.13 *а*. Для цього обчислюються координати точок перетинання ліній, що відстоять від осьової лінії ланок ламаної на величину $h/2$ (половину товщини лінії) — точки *2'*, *3'*, *2''*, *3''*, а також ліній, перпендикулярних осі, які проходять через кінці ламаної — точки *1'*, *1''*, *4'* і *4''*. Усі ці точки утворять контур полігона, який зафарбовується стандартним алгоритмом.

Основний недолік такого підходу — гострі кути, особливо помітні, якщо ланки ламаної розташовані під малим кутом друг до друга. У багатьох програмах векторної графіки використовують удосконалені алгоритми, що забезпечують згладжування гострих кутів дугами окружностей. Для додання лінії більш природного виду, її кінці також можуть округляться, рис. 4.13 *б*.

Незважаючи на те, що описаний спосіб побудови ліній вимагає значно більшого обсягу обчислень (необхідно обчислити координати точок полігона і виконати алгоритм його заповнення), і, отже, має меншу швидкість виведення зображення, він одержав велике поширення в сучасних графічних пакетах, оскільки значно більш гнучкий, дозволяє легко масштабувати лінії, змінювати їхню товщину, змінювати роздільну здатність растра (наприклад, при виведенні зображення на друк). При цьому якість відображення лінії, її «гладкість», якість стикування ланок лінії не зменшується.

5 ОБРОБКА РАСТРОВИХ ЗОБРАЖЕНЬ

Елементами растрових зображень є не графічні примітиви, а окремі пікселі. Це обумовлює специфічні методи роботи з такими зображеннями. Існує кілька основних операцій, виконуваних над растрами.

5.1 Масштабування

Масштабування растрових зображень, на відміну від векторних, пов'язано зі *зміною роздільної здатності растра*. Таке масштабування виконується в двох ситуаціях.

По-перше, таке масштабування треба виконувати, якщо зображення має бути відображене за допомогою деякого растрового графічного пристрою, роздільна здатність якого відрізняється від роздільної здатності самого зображення. Наприклад, ми поміщаємо растрове зображення на сторінку текстового редактора MS Word. При відображенні цієї сторінки на екрані роздільна здатність малюнка має бути приведена до роздільної здатності екрана, а при печатці сторінки — до роздільної здатності принтера. У загальному випадку коефіцієнти масштабування растра по осях x і y складуть

$$k_{x.\text{р}} = k_{x.\text{з}}\, m\, \frac{R_\text{е}}{R_\text{з}} \qquad \text{і} \qquad k_{y.\text{р}} = k_{y.\text{з}}\, m\, \frac{R_\text{е}}{R_\text{з}}, \tag{5.1}$$

де $k_{x.\text{з}}$ і $k_{y.\text{з}}$ – коефіцієнти масштабування при розміщенні зображення на сторінці; m – масштаб відображення сторінки на екрані (або принтері); $R_\text{з}$ і $R_\text{е}$ – роздільна здатність растрів зображення й екрана (принтера). Зауважимо, що саме вихідне зображення (інформація про нього, що зберігається у пам'яті) не змінюється, воно тільки відображається на екрані або папері.

По-друге, роздільна здатність растра самого зображення може бути змінена за допомогою спеціальних програм обробки растрових зображень. Наприклад, роздільну здатність растра можна зменшити, щоб скоротити обсяг пам'яті, займаної зображенням. Інший приклад — сполучення за розмірами двох растрових зображень для створення фотоколажу.

Випадки, коли необхідно зменшити роздільну здатність растра і коли її потрібно збільшити, принципово різні. У випадку ***зменшення роздільної здатності*** має місце *надмірність інформації*, оскільки вихідне зображення містить більше пікселів, чим зображення, що одержується в результаті.

Існують два методи зменшення роздільної здатності. Найбільш простий і швидкий — присвоєння пікселю створюваного зображення кольору *одного* пікселя вихідного. Звичайно беруть піксель, розташований ближче усього до центра нового пікселя, рис. 5.1 *а*. Інформація з інших пікселів губиться, тому якість зображення при такому способі істотно погіршується, рис. 5.1 *г*, — метод використовується тільки для швидкого перегляду зображень.

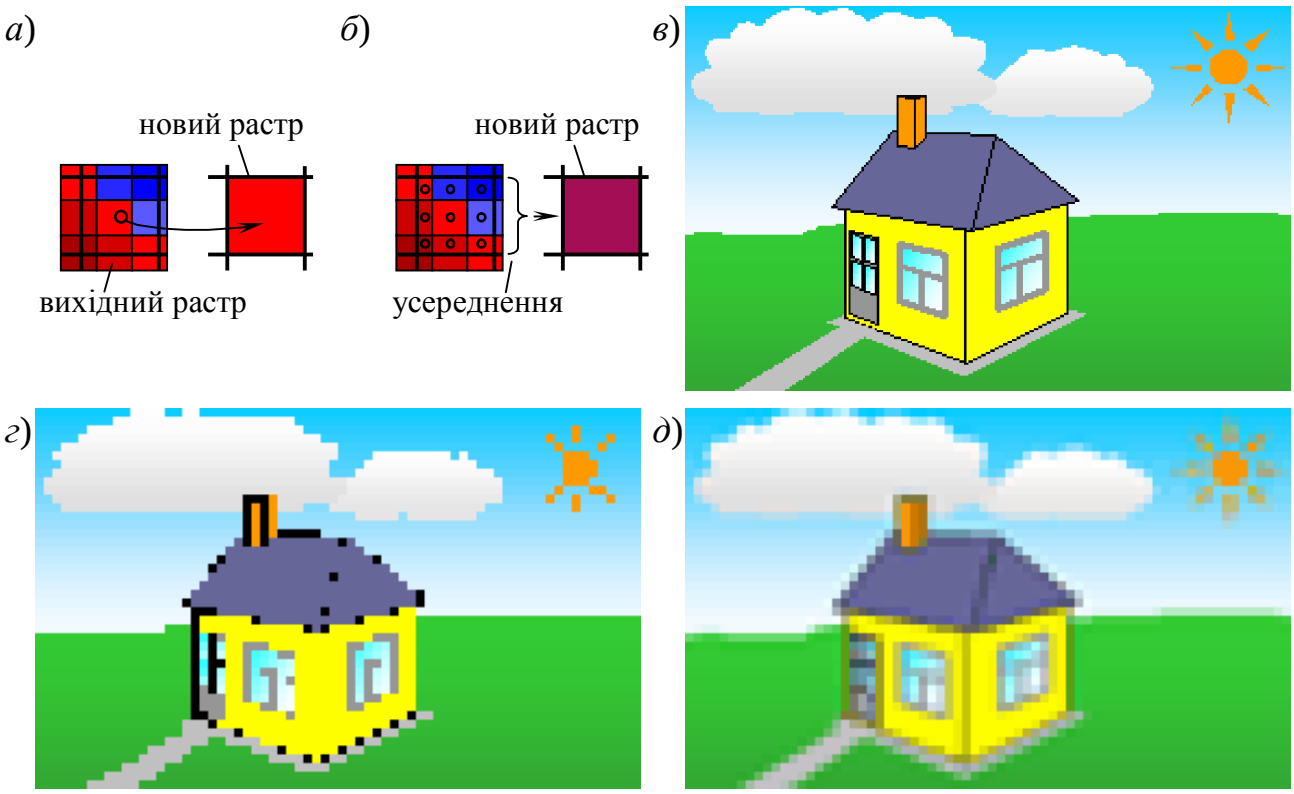


Рисунок 5.1 – Зменшення роздільної здатності растра без усереднення (*а*, *г*) і з усередненням (*б*, *д*); роздільна здатність вихідного зображення (*в*) зменшена в 4 рази (*г*, *д*)

Більш якісне зображення дає метод ***усереднення кольорів*** усіх пікселів вихідного зображення, що попадають в область пікселя нового, рис. 5.1 *б*, *д*. При дробових значеннях коефіцієнтів масштабування в таку область потраплятимуть як цілі пікселі, так і частини пікселів, тому усереднення необхідно виконувати з урахуванням площ пікселів:

$$C_{\text{м}} = \frac{1}{S_{\text{м}}} \sum_i S_i \, C_i \,, \qquad (5.2)$$

де $C_{\text{м}}$ і $S_{\text{м}}$ – колір і площа пікселя масштабованого зображення; C_i і S_i – кольори пікселів вихідного зображення і площі ділянок цих пікселів, що потрапляють в область пікселя масштабованого зображення.

Збільшення роздільної здатності растра можна представити як задачу, зворотну розглянутій. При цьому маємо недолік інформації — в одержуваному зображенні має бути *більше* пікселів, ніж у вихідному. Найпростіший спосіб збільшення роздільної здатності — *заповнення* кольором кожного пікселя тієї області, що відповідає цьому пікселю в збільшеному зображенні. У результаті одержимо зображення, що складається з прямокутників, рис. 5.2 *а* та *б*.

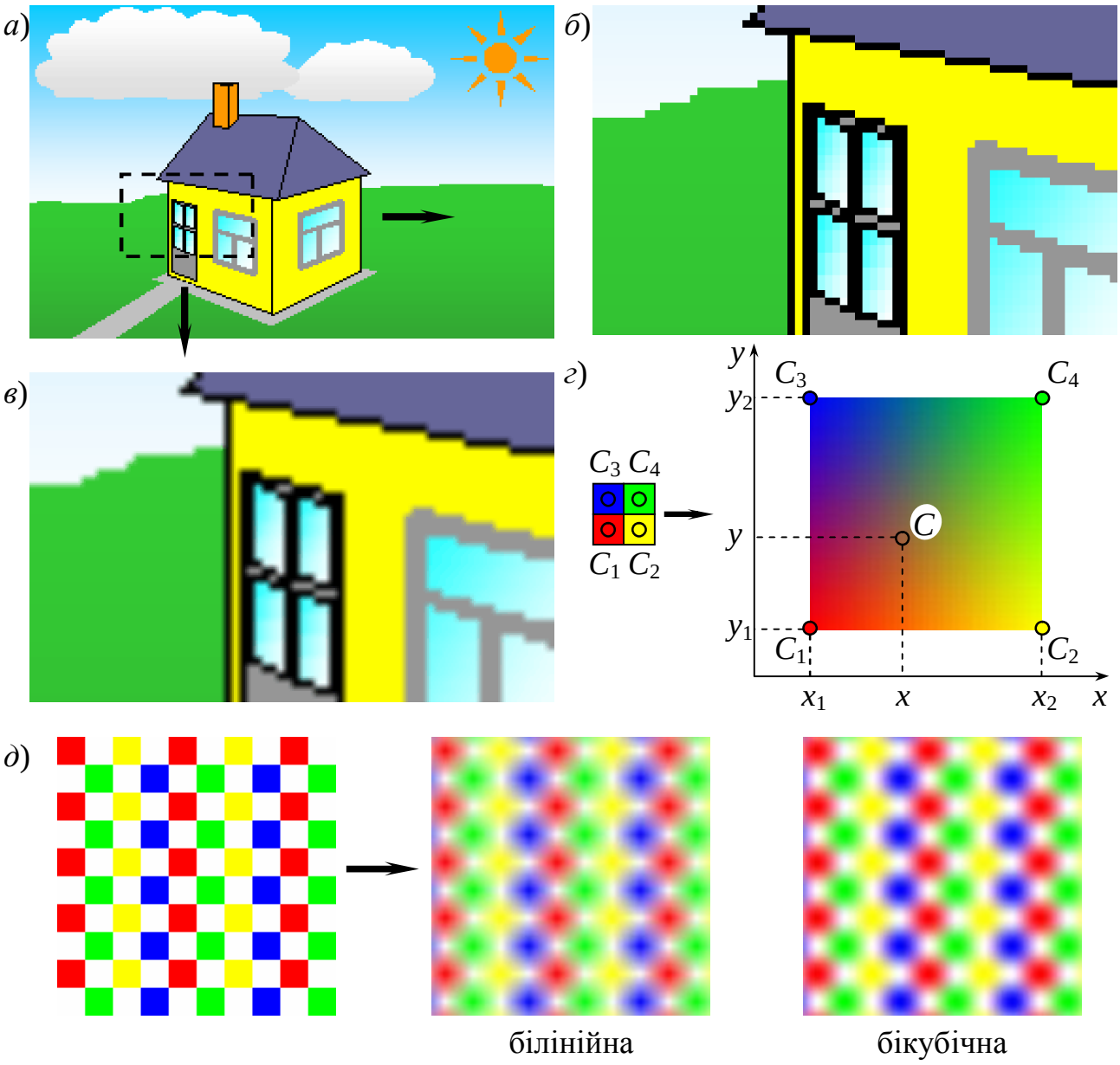


Рисунок 5.2 – Збільшення роздільної здатності растра (*а*) в 4 рази без інтерполяції (*б*) та з білінійною інтерполяцією кольору пікселів (*в*); схема білінійної інтерполяції (*г*); порівняння білінійної и бікубічної інтерполяцій (*д*)

Більш якісний результат дають методи ***інтерполяції*** кольору пікселів, рис. 5.2 *в*. Найпростіший її варіант — ***білінійна інтерполяція***. Сутність її полягає в тому, що при масштабуванні зображення визначаються нові положення *центрів пікселів*. При збільшенні фрагмента зображення, рис. 5.2 *м*, центри пікселів C_1, C_2, C_3 і C_4 розташувалися в точках з координатами (x_1, y_1), (x_2, y_1), (x_1, y_2) і (x_2, y_2) відповідно. Прямокутна область з вершинами в цих точках має бути заповнена колірним переходом, у якому колір пікселя C з координатами центра (x, y) визначається шляхом усереднення кольорів вершин по залежності

$$C = \frac{1}{(x_2 - x_1)(y_2 - y_1)} \begin{pmatrix} C_1(x_2 - x)(y_2 - y) + C_2(x - x_1)(y_2 - y) + \\ + C_3(x_2 - x)(y - y_1) + C_4(x - x_1)(y - y_1) \end{pmatrix}, \qquad (5.3)$$

Знаменник дробу — площа прямокутника. Уздовж ребер прямокутника колір пікселів визначається тільки двома вершинами, що забезпечує непомітне стикування сусідніх областей інтерполяції

Існують інші, більш складні залежності, що використовуються для інтерполяції при збільшенні масштабу растра, наприклад ***бікубічна***. Зображення в такому випадку виходить більш якісним, рис. 5.2 *д*, але збільшуються обсяги обчислень.

5.2 Колірна корекція

Колірна корекція растрових зображень — зміна кольору, яскравості, колірної насиченості пікселів усього зображення або його частини за деяким законом.

Найчастіше використовуються три види колірної корекції: змішування каналів, корекція рівнів та HLS-корекція.

Змішування каналів — найбільш простий вид колірної корекції. Перетворення кольору для RGB-зображення виконується по залежностях:

$$\begin{cases} R' = k_{r-r}R + k_{g-r}G + k_{b-r}B \\ G' = k_{r-g}R + k_{g-g}G + k_{b-g}B, \\ B' = k_{r-b}R + k_{g-b}G + k_{b-b}B \end{cases} \qquad (5.4)$$

де R, G і B – вихідні інтенсивності червоного, зеленого і синього каналів; R', G' і B' – інтенсивності після перетворення; $k_{r\text{-}r}$, $k_{g\text{-}r}$ … $k_{b\text{-}b}$ – коефіцієнти, що задають взаємовплив каналів. Коефіцієнти можуть бути як позитивними, так і негативними — в останньому випадку інтенсивність відповідного каналу віднімається.

На рис. 5.3 приведений приклад змішання каналів — до червоного каналу додано 50% інтенсивності синього каналу.

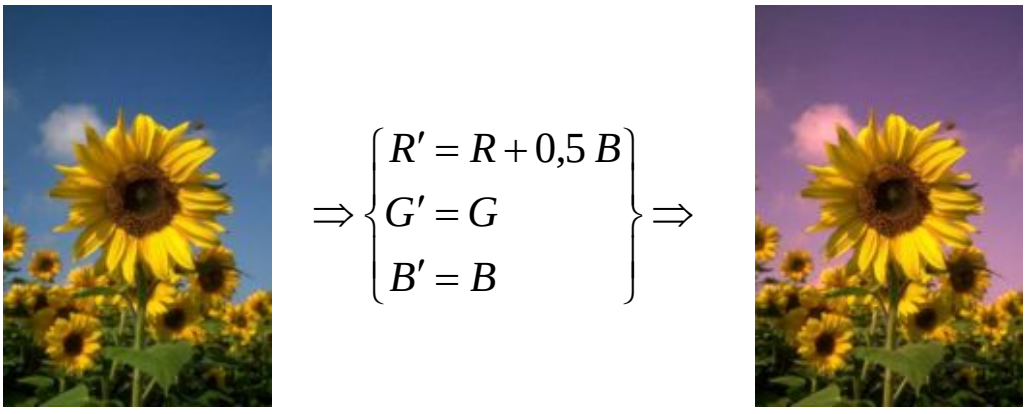


Рисунок 5.3 – Змішування каналів

Корекція рівнів полягає в перетворенні за визначеним законом інформації про яскравість пікселя або інтенсивність колірних каналів. Для RGB-зображення запишемо:

$$\begin{cases} R' = f_R(R), \\ G' = f_G(G), \\ B' = f_B(B), \end{cases} \tag{5.5}$$

де f_R, f_G і f_B – функції перетворення. На відміну від перетворення (5.4) корекція рівнів припускає незалежне перетворення кожного з колірних каналів. У випадку, якщо $f_R = f_G = f_B = f$ (усі канали перетворюються за єдиним законом) — виконується ***корекція яскравостей*** пікселів:

$$I' = f(I), \tag{5.6}$$

де I та I' – яскравість пікселя до та після перетворення.

Як відомо, для RGB-зображення інтенсивність кожного з каналів (червоного, зеленого і синього) кодується числом у діапазоні від 0 до $(2^n–1)$, де n – розрядність (кількість біт інформації на канал). У повнокольоровому зображенні на кожен канал приходиться 8 біт, тобто маємо діапазон кодування інтенсивностей від 0 до 255. Надалі будемо задавати інтенсивність у відсотках — найбільшій інтенсивності буде відповідати 100 %.

Корекцію рівнів зручно представити на графіку, рис. 5.4, де по шкалі абсцис відкладені інтенсивності каналів (яскравості пікселів) вихідного зображення, а по шкалі ординат — перетвореного. Крива *1* графіка функції перетворення для кожного значення інтенсивності вихідного зображення задає значення інтенсивності перетвореного. Діагональна лінія 2 відповідає відсутності перетворення (зображення не міняється). Чим більше відхилення кривої *1* від лінії *2*, тим істотніше зміна інтенсивностей.

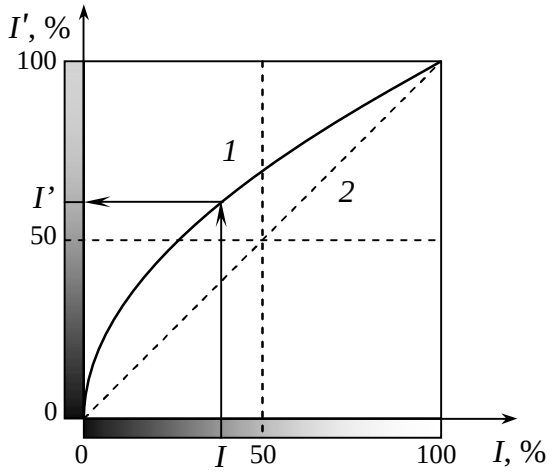


Рисунок 5.4 – Корекція рівнів

На рис. 5.5 приведені приклади корекції рівнів. Для обробки зображень особливе значення мають зміна яскравості, контрастності та вирівнювання рівнів.

Зміна яскравості, рис. 5.5 *д*, *е*, виконується як збільшення або зменшення інтенсивності всіх пікселів на величину $\Delta_{я}$.

$$I' = I \pm \Delta_{я}. \qquad (5.7)$$

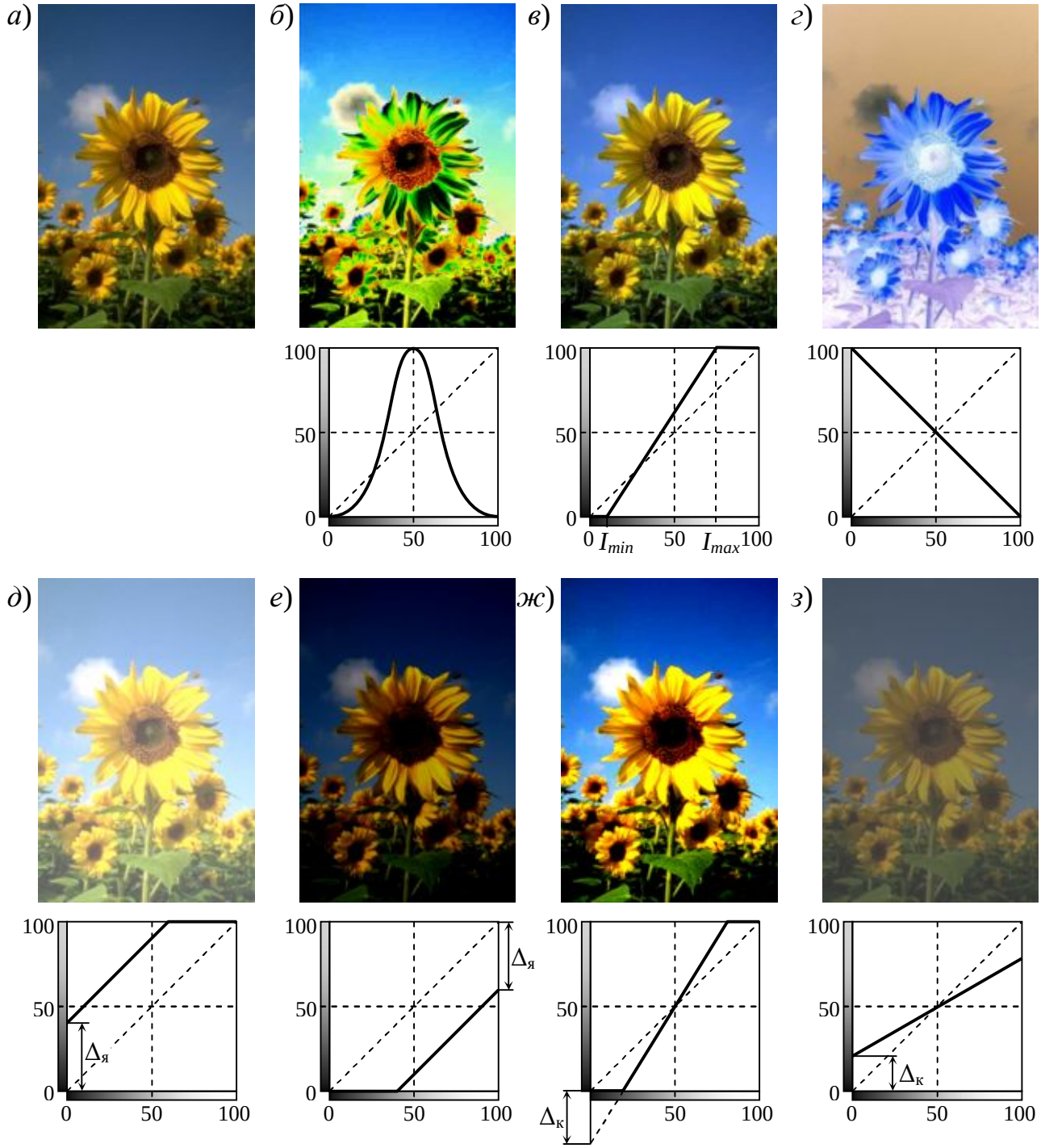


Рисунок 5.5 – Приклади корекції рівнів: вихідне зображення (*а*); довільна корекція (*б*); вирівнювання рівнів (*в*); інверсія (*г*); збільшення яскравості (*д*); зменшення яскравості (*е*); збільшення контрастності (*ж*); зменшення контрастності (*з*)

Зміна контрастності зображення, рис. 5.5 *ж*, *з*, на величину $\Delta_к$ виконується за залежністю

$$I' = I \pm \Delta_к \left(\frac{I}{I_{0,5}} - 1 \right), \tag{5.8}$$

де $I_{0,5}$ – інтенсивність, що дорівнює 50 % максимальної.

Оскільки інтенсивність колірних каналів обмежена інтервалом 0…(2^n–1), зміна яскравості і збільшення контрастності приводять до *втрати частини інформації* (горизонтальні ділянки відповідних графіків). Виконання зворотної операції (наприклад, зменшення яскравості після її збільшення) не дозволить відновити зображення у вихідному виді.

Операція ***вирівнювання рівнів***, рис. 5.5 *в*, потребує аналізу всіх пікселів зображення для пошуку найбільшої I_{max} і найменшої I_{min} інтенсивностей колірних каналів (або каналу яскравості). Перетворення виконується таким чином, щоб інтенсивності колірних каналів зайняли весь діапазон інтенсивностей 0...(2^n–1):

$$I' = \frac{I - I_{\min}}{I_{\max} - I_{\min}} I_{1,0}, \tag{5.9}$$

де $I_{1,0}$ — максимально можливе значення інтенсивності (2^n–1). Вирівнювання рівнів виконується автоматично програмами для обробки растрових зображень і дозволяє поліпшити вид зображення, отриманого зі сканера або за допомогою цифрового фотоапарата.

До розглянутої групи операцій відноситься також ***гамма-корекція***. При виведенні на монітор RGB-зображення дисплейний контролер виконує перетворення *числових значень* інтенсивностей колірних каналів в *аналоговий сигнал*, у якому рівні напруги задають яскравість світлових плям на екрані електронно-променевої трубки (ЕПТ). Однак, залежність між напругою *U*, поданою на ЕПТ, і яскравістю *I* світіння люмінофора нелінійна:

$$I = c\, U^{\gamma}, \tag{5.10}$$

де γ – показник ступеня, для кольорових моніторів рівний 2,3…2,8; *c* – константа. Для того, щоб компенсувати нелінійність передачі яскравості й одержати на екрані зображення з коректною передачею кольорів (що необхідно, наприклад, у видавничій справі), виконується *калібрування монітора*, щоб визначити значення γ та *c*, і потім, при виведенні зображення, виконується корекція рівнів за залежністю, зворотною (5.10):

$$I' = \left(\frac{I}{c} \right)^{\frac{1}{\gamma}}, \tag{5.11}$$

Приблизний вигляд графіка гамма-корекції показаний на рис. 5.4.

HLS-корекція полягає в перетворенні колірного кодування з моделі RGB у модель HLS (колірний тон, світлота і насиченість), зміні цих параметрів і зворотному перетворенні з HLS у RGB.

У моделі HLS колірний тон задається як кут в інтервалі 0…360° по круговій шкалі кольорів. Зміну колірного тону можна представити як поворот колірного кола на заданий кут δ_H, рис. 5.6. На рис. 5.7 *б*, *в* та *г* показано приклади такого перетворення. Зміна світлоти і насиченості виконується додаванням чи відніманням деякого значення, аналогічно зміні яскравості (5.7), рис. 5.7 *д*, *е*, *ж* и *з*.

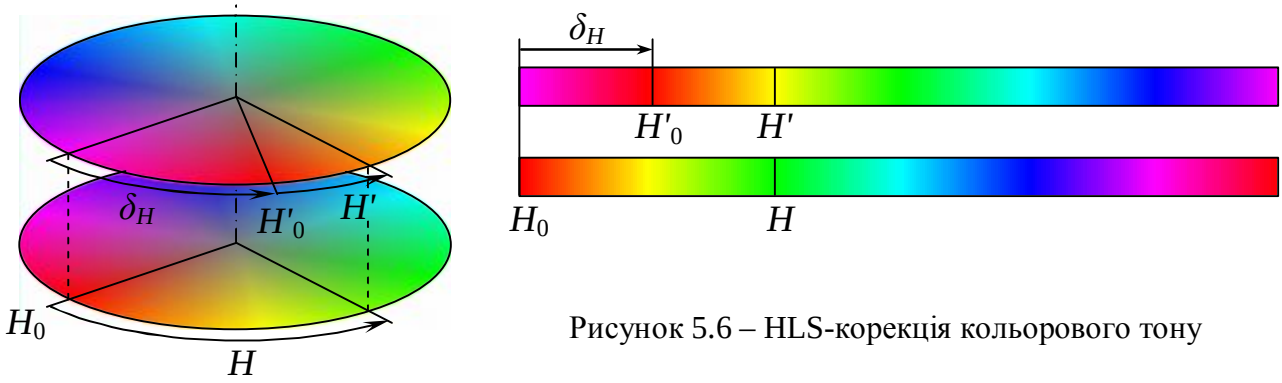


Рисунок 5.6 – HLS-корекція кольорового тону

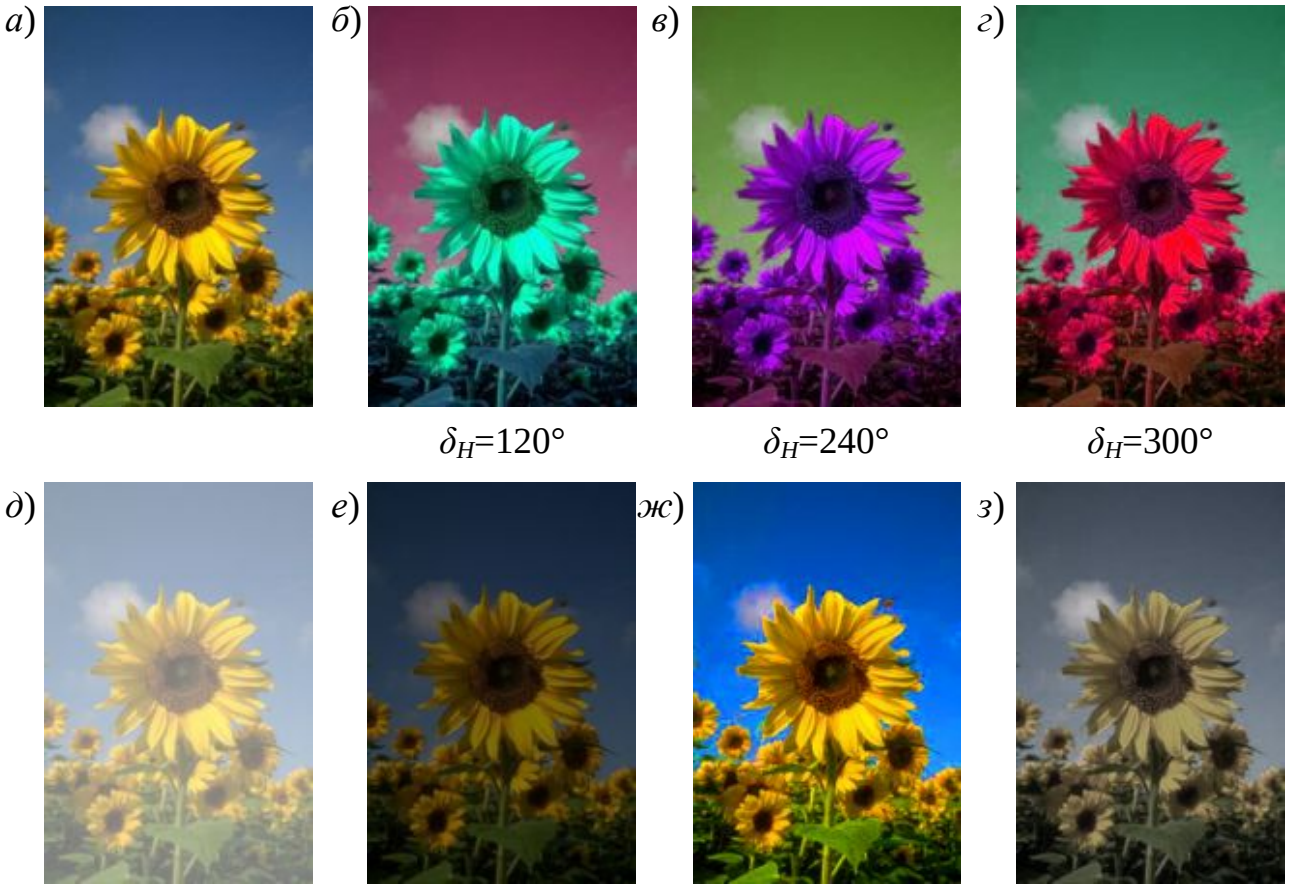


Рисунок 5.7 – Приклади HLS-корекції: вихідне зображення (*а*); корекція кольорового тону (*б*, *в* и *г*); збільшення світлоти (*д*); зменшення світлоти (*е*); збільшення насиченості (*ж*); зменшення насиченості (*з*)

5.3 Фільтрація

Фільтрацією растрового зображення називається перетворення, при якому колір кожного пікселя одержуваного зображення визначається як функція кольорів декількох пікселів вихідного зображення (звичайно розташованих в околиці даного пікселя).

Математична операція, що виконується при фільтрації зображення, називається ***згорткою***, а функція, використовувана для фільтрації — ***функцією згортки***. Область зображення, задіяна в обчисленні кольору одного пікеля, назвемо ***областю згортки***, а сам цей піксель — ***ядром згортки***, рис. 5.8.

У загальному виді процес фільтрації можна описати вираженням:

$$C'_{i,j} = \sum_{(k,p)\subset\Delta} C_{i+k,j+p} \cdot F(k,p), \qquad (5.12)$$

де C і C' – кольори пікселів вихідного і фільтрованого зображень; i, j – координати пікселя, колір якого обчислюється; $F(k, p)$ – функція згортки; Δ – область згортки; k, p – перемінні функції згортки (координати в системі координат області згортки).

У процесі фільтрації зображення ядро згортки переміщується, проходячи крізь кожен піксель зображення. Для кожного пікселя виконується підсумовування яскравостей усіх пікселів, які попадають в область згортки, помножених на значення функції згортки для цих пікселів.

Найпростіший спосіб завдання функції згортки — *матричний*. Значення функції згортки для кожного пікселя області згортки задається відповідним значенням у матриці. Для зручності і прискорення обчислень ці коефіцієнти звичайно задаються цілочисельними, а вихідний рівень інтенсивності нормується масштабним коефіцієнтом 1/S:

$$F(k,p) = \frac{1}{S}\begin{Vmatrix} \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & f_{-1,1} & f_{0,1} & f_{1,1} & \cdots \\ \cdots & f_{-1,0} & f_{0,0} & f_{1,0} & \cdots \\ \cdots & f_{-1,-1} & f_{0,-1} & f_{1,-1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{Vmatrix}, \qquad (5.13)$$

де $f_{k,p}$ – коефіцієнти.

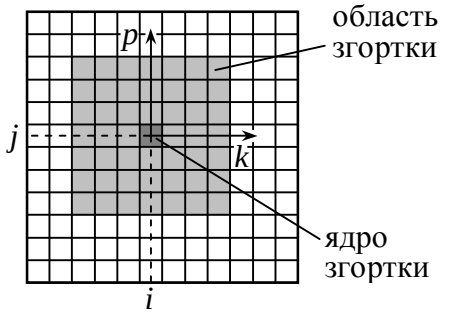


Рисунок 5.8 – Схема фільтрації зображення

Для того, щоб яскравість зображення після фільтрації залишалася незмінною, має виконуватися *умова нормування*

$$\sum_{(k,p)\subset\Delta} F(k,p) = 1, \tag{5.14}$$

тобто сума значень функції згортки для всієї області згортки має дорівнювати одиниці. Для цього масштабний коефіцієнт має дорівнювати

$$S = \frac{1}{\sum_{(k,p)\subset\Delta} F'(k,p)}, \tag{5.15}$$

де $F'(k, p)$ — ненормована матриця коефіцієнтів згортки.

Найчастіше використовуваними фільтрами, для яких виконується умова (5.14), є фільтри ***розмиття*** і ***підвищення різкості***, приклади яких приведені на рис. 5.9 *б* та *в*. Фільтри розмиття додають до кольору пікселя — ядра згортки кольори сусідніх пікселів. У результаті зображення стає менш різким, «розмитим». Фільтри підвищення різкості, навпаки, віднімають кольори пікселів, сусідніх з ядром згортки. Це приводить до підвищення чіткості границь областей, зафарбованих різними кольорами.

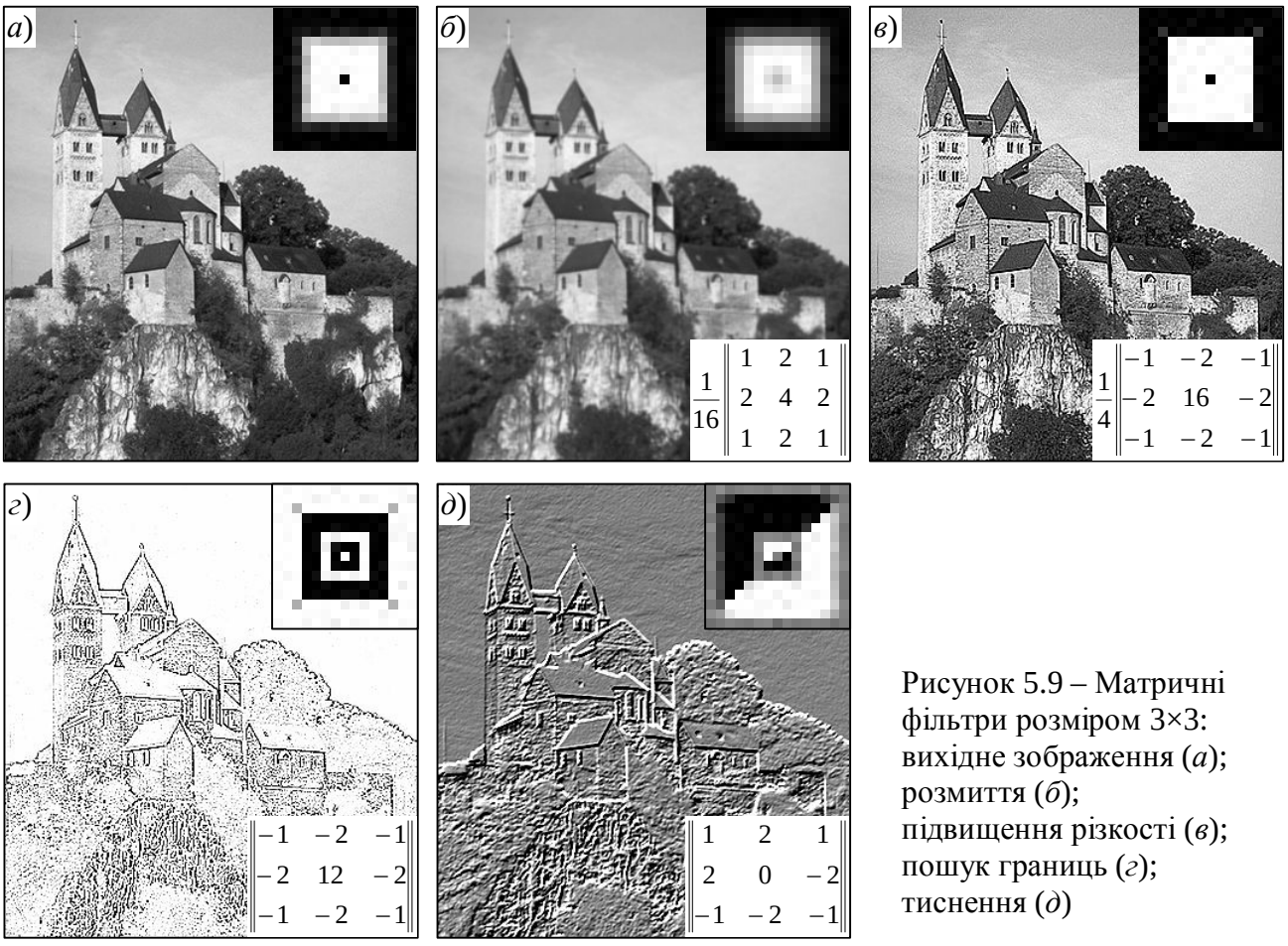


Рисунок 5.9 – Матричні фільтри розміром 3×3: вихідне зображення (*а*); розмиття (*б*); підвищення різкості (*в*); пошук границь (*г*); тиснення (*д*)

Якщо сума значень функції згортки для всієї області згортки дорівнює нулю, такий фільтр перетворить однорідні області зображення (заповнені одним кольором) у чорні. Відрізнятися від чорного фона будуть тільки ті пікселі, що знаходяться на границях областей, заповнених різними кольорами, або в області колірного переходу. Такі фільтри використовують для ***пошуку границь*** об'єктів на зображенні, рис. 5.9 *г*, і для одержання «рельєфних» зображень — фільтр «***тиснення***», рис. 5.9 *д*.

Фільтр *пошуку границь* дає зображення, що виглядає як білий малюнок на чорному фоні. Зображення, представлене на рис. 5.9 *г*, було інвертовано, щоб одержати ефект малюнка на білому папері.

При створенні рельєфного зображення, рис. 5.9 *д*, для перетворення чорних областей у сірі в процесі фільтрації до інтенсивності пікселя додавалася величина, що дорівнює половині максимальної інтенсивності.

Більш універсальними є фільтри, у яких функцію згортки задано *аналітично*. Прикладом може бути ***фільтр розмиття Гауса*** з функцією згортки

$$F(k,p)=\frac{1}{2\pi\,\sigma^2}\cdot e^{-\frac{k^2+p^2}{2\sigma^2}}, \qquad (5.16)$$

де σ – параметр функції Гауса, що визначає розмір області згортки і ступінь розмиття зображення, рис. 5.10. Область згортки являє собою окружність радіусом 3σ і може мати будь-який, у тому числі дуже великий, розмір.

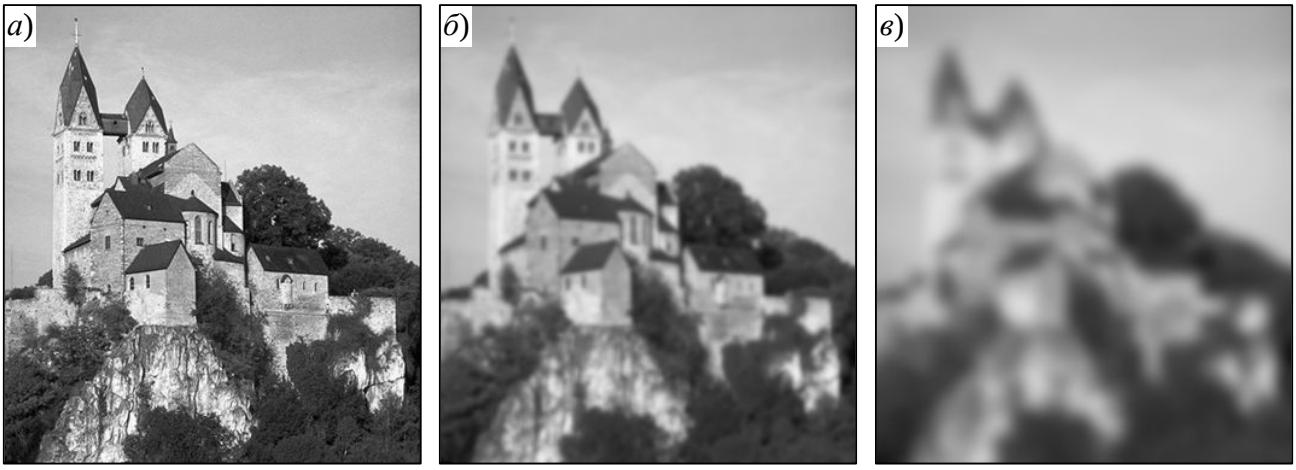


Рисунок 5.10 – Фільтр розмиття Гауса: вихідне зображення (*а*), $\sigma = 2$ (*б*) та $\sigma = 6$ (*в*)

5.4 Палітризація і псевдотонування

Для повноцінної передачі кольорових або напівтонових фотографічних зображень необхідно мати можливість кодування інтенсивності колірних каналів або каналу яскравості з досить малим кроком квантування. У сучасних графічних системах звичайно використовують 8-бітове кодування, що дозволяє задати $2^8 = 256$ рівнів інтенсивності (див. п. 3.3). Однак, можливі ситуації, коли використовувати таку кількість кольорів неможливо або нераціонально з таких причин:

1) Відеоадаптер не має достатніх апаратних засобів для відображення повнокольорового зображення (недостатньо пам'яті для збереження 24 бітових площин, мала швидкість обробки інформації, низька розрядність ЦАП). Зараз ця проблема не так актуальна, але недорогі комп'ютери 10 … 15-літньої давнини в кращому випадку забезпечували відображення 16 кольорів, комп'ютери 80-х років — 4 кольори або взагалі 2 — чорний і білий. В наші часи подібна проблема виникає для малопотужних цифрових пристроїв, наприклад, контролерів різного устаткування.

2) При друку зображення. На відміну від дисплеїв, пристрої, що друкують на папері (струминні, лазерні та ін. принтери), мають лише декілька кольорів — по кількості кольорів фарби (тонера) плюс білий колір — чистий папір. Звичайно в кольоровий принтер заправляють фарбу 4 кольорів, рідше (у дорогих моделях) — 7 кольорів. Некольорові принтери заправлені однією чорною фарбою і дають двоколірне зображення.

3) При необхідності зменшити розмір файлу зображення, наприклад при передачі його по мережі. Для деяких типів зображень (схеми, креслення, малюнки) велика кількість півтонів непотрібна і навіть шкідлива — при друку такі зображення можуть вийти нечіткими, «брудними».

Звичайно використовуються два типи зображень з низькою колірною роздільною здатністю: ***двоколірні*** (один біт на піксель) і зображення з ***таблицею кольорів*** (***палітрові***), див. п. 3.3.

Найпростішим способом перетворення напівтонового зображення в дворівневе є ***граничний метод***, при якому ділянки вихідного зображення, інтенсивність яких вище деякого *граничного значення*, відображаються пікселями білого кольору, а ділянки, інтенсивність яких нижче — чорними пікселями, рис. 5.11. Граничну величину звичайно встановлюють приблизно рівною половині максимальної інтенсивності.

Метод ефективний для креслень, схематичних малюнків та інших зображень, де немає необхідності у півтонах — навпаки, потрібно зробити таке зображення максимально чітким і контрастним. Більш складні зображення (фотографії) стають малореалістичними і погано сприймаються. Через великі помилки виведення інтенсивності для кожного пікселя відбувається утрата великої кількості неконтрастних деталей.

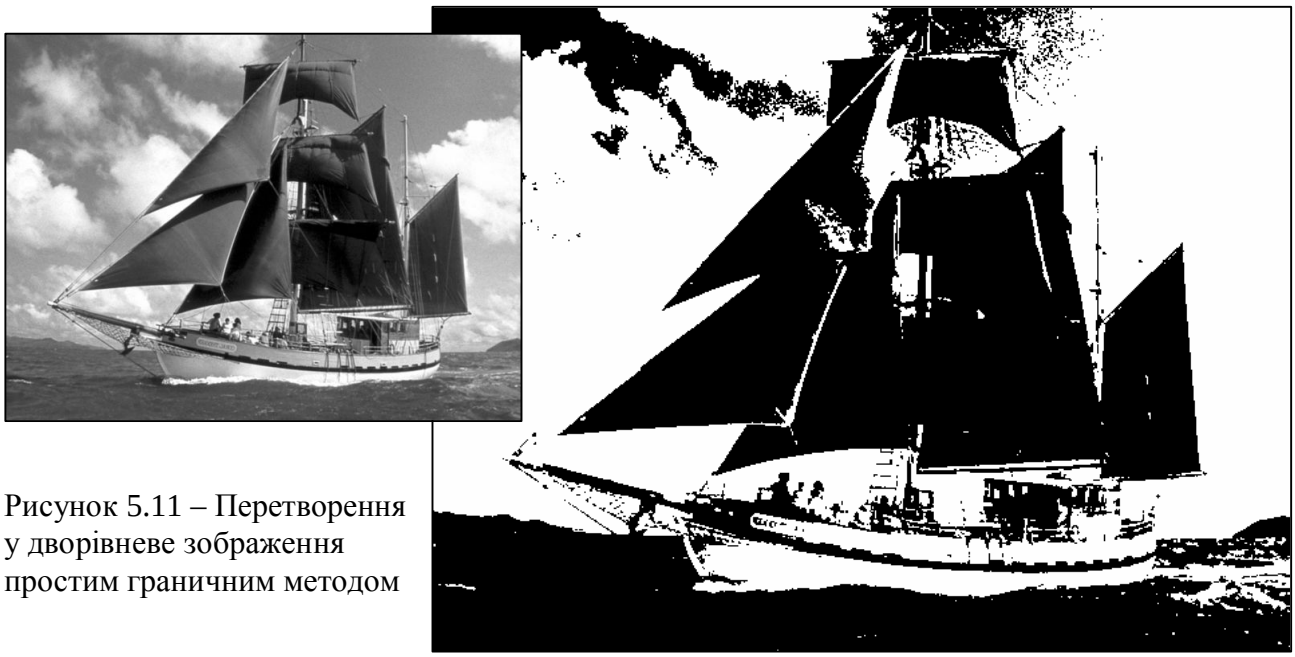


Рисунок 5.11 – Перетворення у дворівневе зображення простим граничним методом

Перетворення повнокольорового зображення в кольорове з малою кількістю кольорів може виконуватися подібним образом — довільний колір пікселя замінюється найближчим кольором з палітри. Таку процедуру називають ***палітризацією*** способом ***округлення***. Ступінь розходження вихідного і нового кольорів визначається сумою різностей інтенсивностей для кожного з колірних каналів, узятих за модулем:

$$\Delta_i = \left|R - R_{\Pi i}\right| + \left|G - G_{\Pi i}\right| + \left|B - B_{\Pi i}\right|, \qquad (5.17)$$

де R, G, B – інтенсивності колірних каналів деякого пікселя вихідного зображення; $R_{\Pi i}$, $G_{\Pi i}$, $B_{\Pi i}$ – інтенсивності колірних каналів *і-го* кольору з палітри. У ході перетворення колір пікселя заміняється індексом кольору палітри, для якого різниця Δ буде найменшою.

Якість одержуваного таким способом зображення залежить від двох факторів — *кількості кольорів* палітри і *вмісту палітри*. Кількість кольорів палітри обмежена розрядністю кодування інформації і дорівнює 2^N, де N – кількість бітових площин (число біт на один піксель). На рис. 5.12 *а* показаний результат округлення кольорової фотографії при 8-и кольорах палітри, що відповідають основним RGB-кольорам.

Використання стандартних палітр надає дуже обмежені можливості передачі кольору. Тому основна задача палітризації — підібрати кольори палітри так, щоб невеликою кількістю кольорів найякісніше передати зображення. Такий підбір може бути виконаний як вручну, так і автоматично. Палітри, отримані *автоматично* шляхом пошуку найбільш придатного набору кольорів, називають ***адаптивними***, рис. 5.12 *б*. Один з найбільш розповсюджених алгоритмів пошуку називається *кластеризацією*:

1. Вибирається *початкове наближення* з 2^N кольорів, розташованих деяким чином усередині колірного куба.

2. Обсяг колірного куба розбивається на 2^N *кластерів*. Кожен кластер містить кольори, для яких помилка Δ_i (5.17) щодо центра кластера (*i*-го обраного кольору) менше, ніж помилка щодо всіх інших обраних кольорів, рис. 5.12 *в*. Простіше — це область, найближча до обраного кольору.

4. Виконується *статистичний аналіз розподілу кольорів* у зображенні і для кожного кластера знаходиться *центроїда* — усереднення всіх кольорів, що потрапили в кластер.

5. Центроїди кластерів призначаються новими кольорами палітри і процедура кластеризації повторюється починаючи з кроку 2. Пошук виконується доти, доки центроїди всіх кластерів не збіжаться зі вже обраними кольорами палітри.

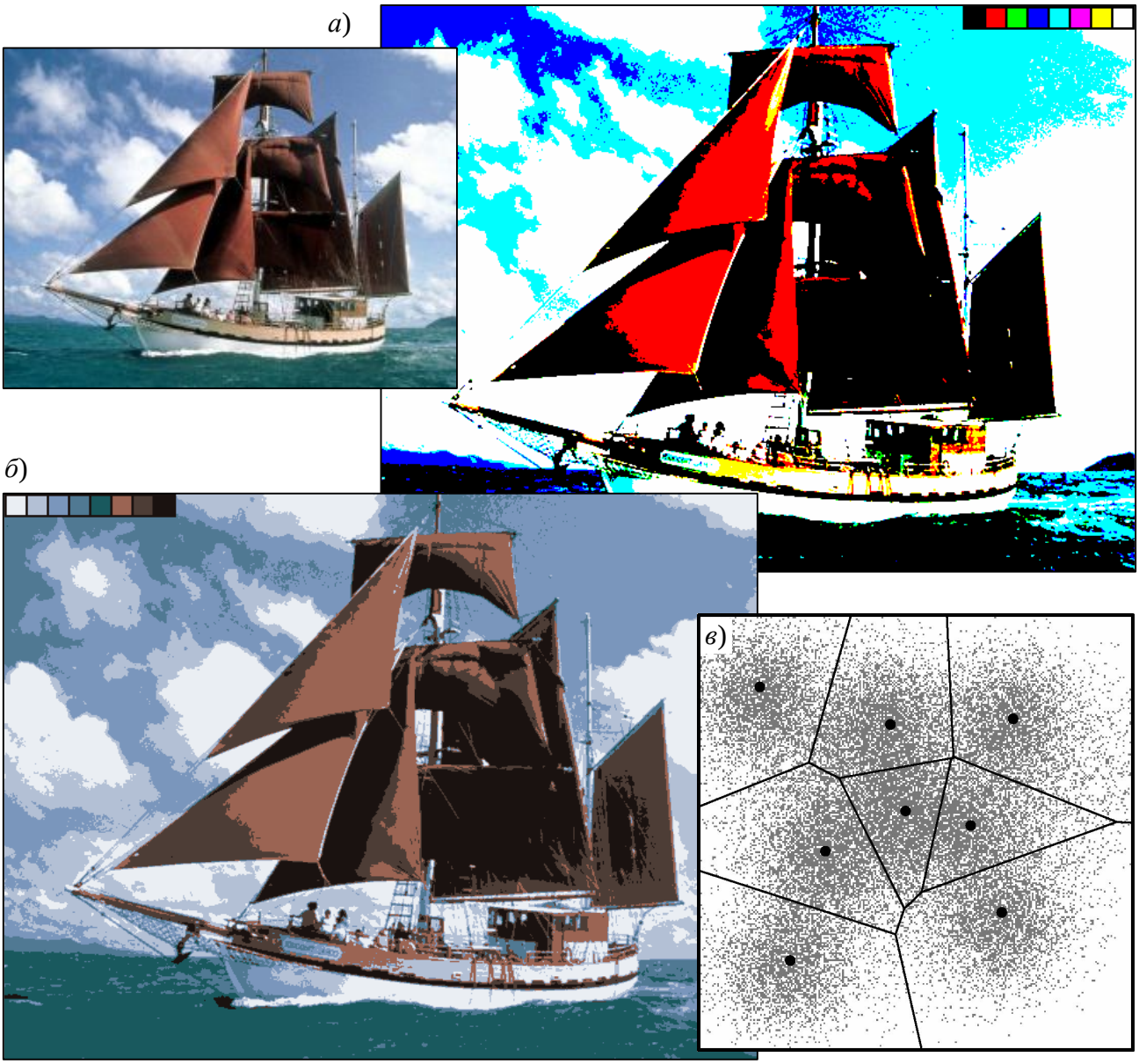


Рисунок 5.12 – Палітризація способом округлення при 8-колірній стандартній (*а*) та адаптивній (*б*) палітрах, кластеризація простору кольорів (*в*)

Для поліпшення якості двоколірних зображень та зображень з малим числом кольорів використовується ***метод півтонів*** або ***псевдотонування***. Успіх методу півтонів залежить від властивості зорової системи людини поєднувати (згладжувати) дискретну інформацію.

Метод півтонів відомий здавна. Спочатку він використовувався при виготовленні гравюр, шовкових картин, текстильних виробів. У *гравюрах*, матриці яких виготовляються вручну, півтони передаються штрихами, розташованими більш-менш густо, рис. 5.13 *а*.

У 1880 р. Стефаном Хагеном було винайдено сучасний півтоновий друк. Зображення за допомогою спеціальної решітки розкладається на точки, причому розмір кожної точки залежить від інтенсивності відображуваної ділянки зображення. Цим методом можна одержати велику кількість фотографічних півтонів сірого, використовуючи дворівневе середовище: чорну фарбу на білому папері, рис. 5.13 *б*. При використанні 4-х фарб (чорної, жовтої, голубої та пурпурної) одержують кольорові зображення, рис. 5.13 *в*. Якість передачі зображення залежить від кроку решітки. Для газетних фотографій застосовуються решітки від 50 до 90 точок на дюйм. Папір більш високої якості дозволяє використовувати решітки із кількістю точок до 300 на дюйм.

Розмір точки цифрового зображення (пікселя екрана або принтера) звичайно фіксований. Тому в комп'ютерній графіці використовуються методи, дещо відмінні від поліграфічних.

Для передачі півтонів використовується ***метод конфігурацій***. Ділянки зображення (звичайно квадратної форми), що містять декілька пікселів, поєднуються в клітки — ***конфігурації***, — які заповнюються за визначеною схемою в залежності від рівня інтенсивності ділянки зображення. Кількість переданих півтонів визначається кількістю пікселів у конфігурації — на одиницю більше числа пікселів.

На рис. 5.14 *а* показана одна з можливих груп конфігурацій для дворівневого чорно-білого дисплея. Для кожної клітки використовується *чотири* пікселі (2×2). При такій організації виходить *п'ять* можливих тонів сірого. Число доступних рівнів інтенсивності можна збільшити за допомогою збільшення розміру клітки. Конфігурації для клітки 3×3 пікселів приведені на рис. 5.14 *б*. Вони дають десять рівнів інтенсивності.

Малюнок конфігурацій може бути різним, але має задовольняти таким вимогам:

1) Область, заповнена однаковими конфігураціями, має казатися якомога більш однорідною.

2) Кожна наступна конфігурація має отримуватися з попередньої шляхом додавання одного зафарбованого пікселя (інші пікселі повинні залишатися такими ж).

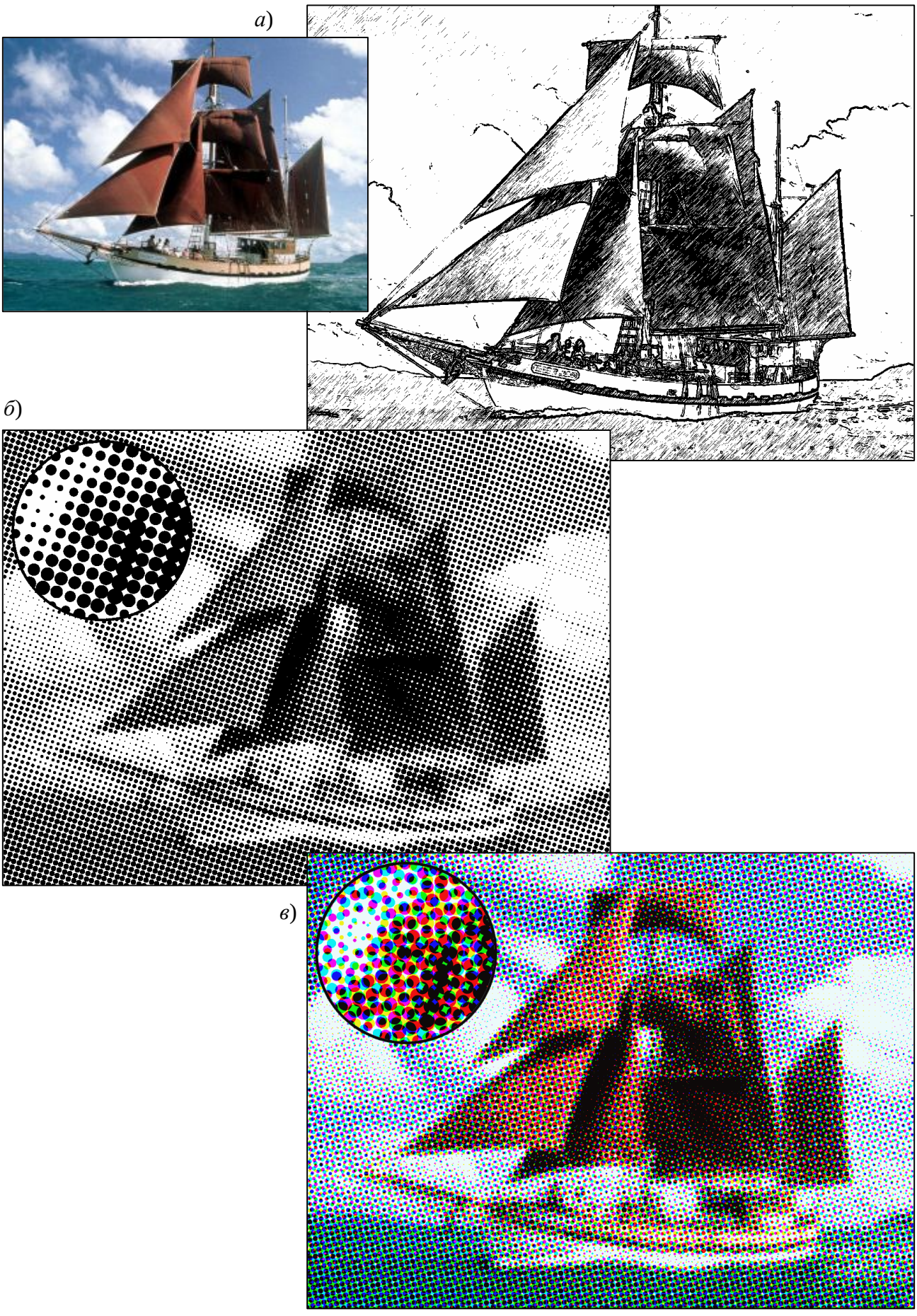


Рисунок 5.13 – Метод півтонів: гравюра (*а*), півтоновий (*б*) та кольоровий (*в*) друк

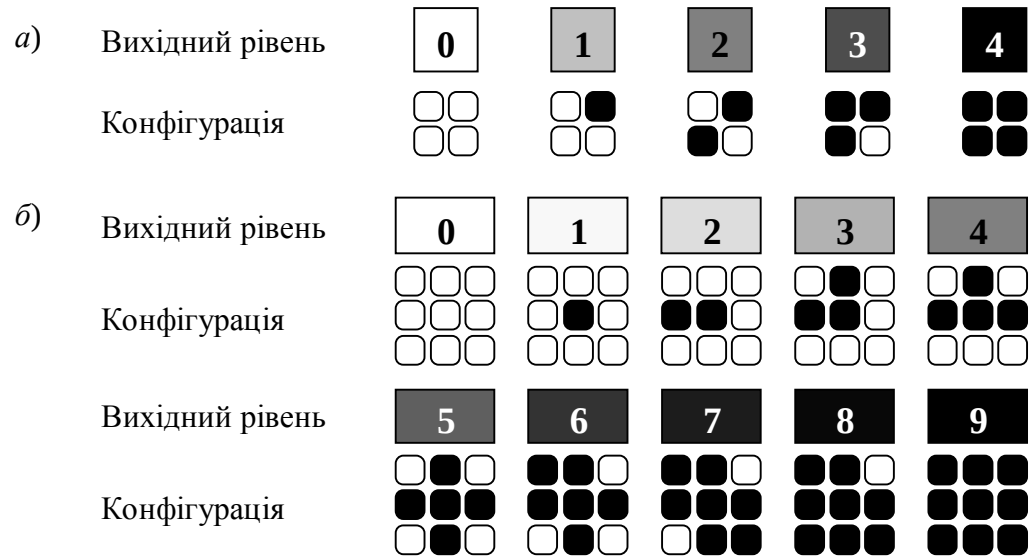


Рисунок 5.14 – Дворівневі конфігурації 2×2 (*а*) та 3×3 (*б*)

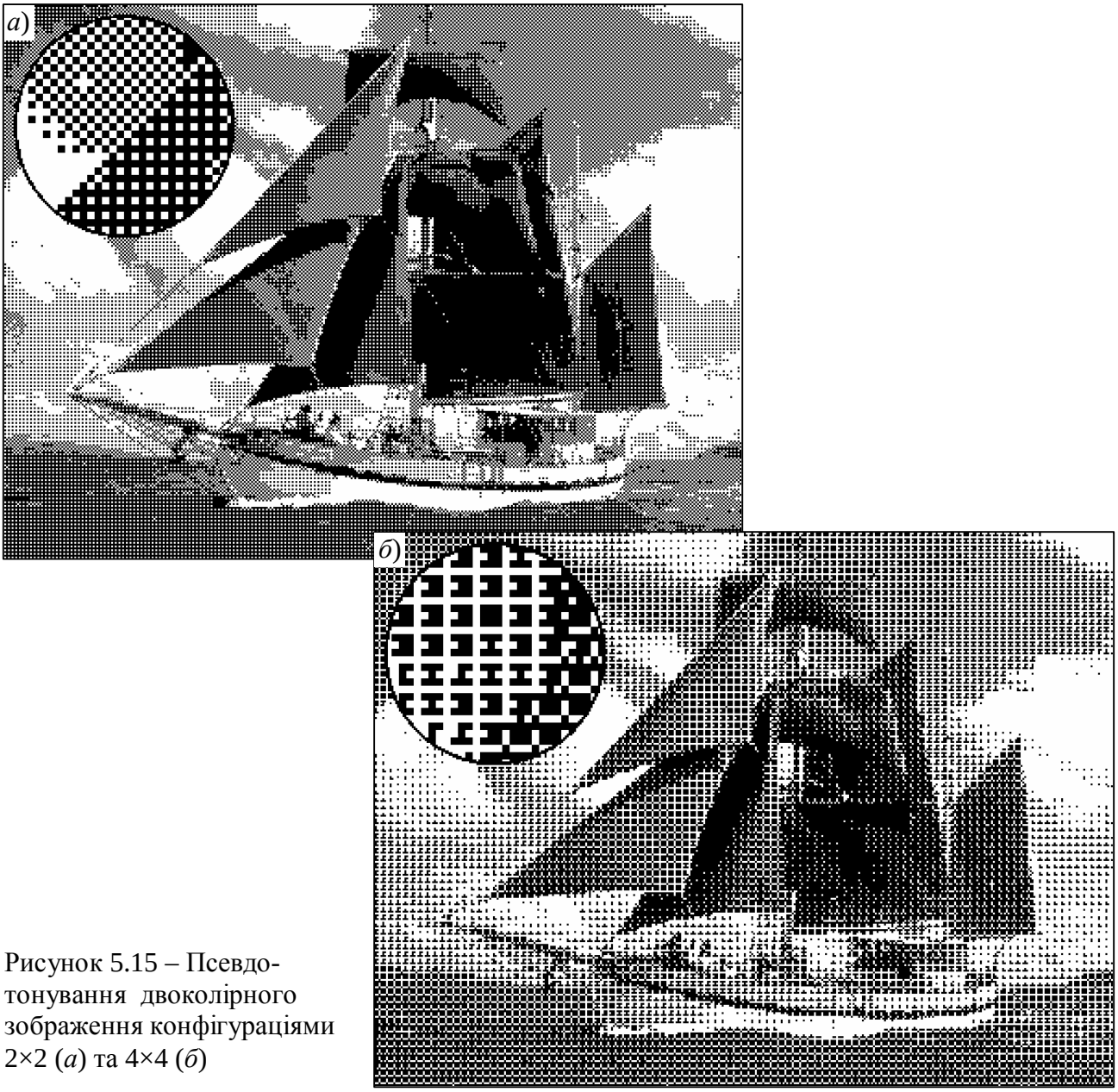


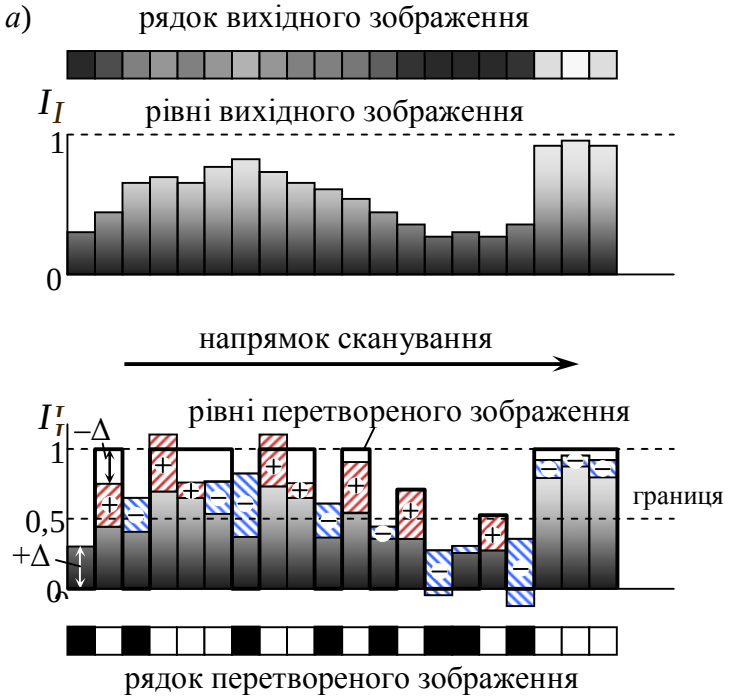
Рисунок 5.15 – Псевдо-тонування двоколірного зображення конфігураціями 2×2 (*а*) та 4×4 (*б*)

Приклади зображень, апроксимованих конфігураціями (2×2) і (4×4) приведені на рис. 5.15. Помітно, що поліпшення *колірної роздільної здатності* (кількості півтонів) при збільшенні розміру клітки супроводжується погіршенням просторової роздільної здатності, що приводить до втрати дрібних деталей. Окрім того, конфігурації, що повторюються, приводять до появи ефекту *фактури* — регулярних візерунків, які погіршують сприйняття зображення.

Метод конфігурацій дає кращий ефект, якщо при перетворенні півтонового зображення в двоколірне можна збільшити роздільну здатність растра. Найкращий результат дає варіант, при якому кожному пікселю вихідного зображення буде відповідати одна клітка (конфігурація) — потрібно буде збільшити растр у стільки разів, яким є розмір конфігурацій. Зрозуміло, це можливо тільки в тому випадку, якщо графічний пристрій може відобразити картинку з такою роздільною здатністю.

Існує метод поліпшення візуального дозволу для дворівневих зображень практично без зменшення просторового дозволу — ***метод дифузії похибки*** (***метод Флойда-Стейнберга***). Його можна розглядати як модифікований граничний метод.

Зображення сканується порядково; інтенсивність кожного пікселя порівнюється з граничною величиною (звичайно половиною максимальної інтенсивності), і виконується заміна кольору на чорний або білий. При цьому обчислюється *похибка* Δ (різниця вихідної і кінцевої яскравостей), яка може бути як позитивною, так і негативною, рис. 5.16 *а*. Похибка в заданій пропорції, рис. 5.16 *б*, розподіляється по сусіднім пікселям до їхнього порівняння з граничною величиною.



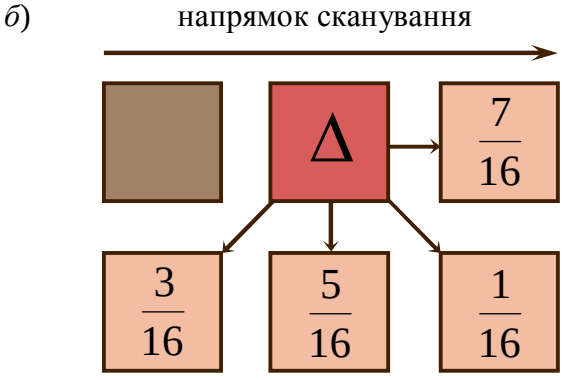


Рисунок 5.16 – Псевдотонування методом дифузії похибки (Флойда-Стейнберга): принцип перетворення (*а*) та матриця переносу похибки (*б*)

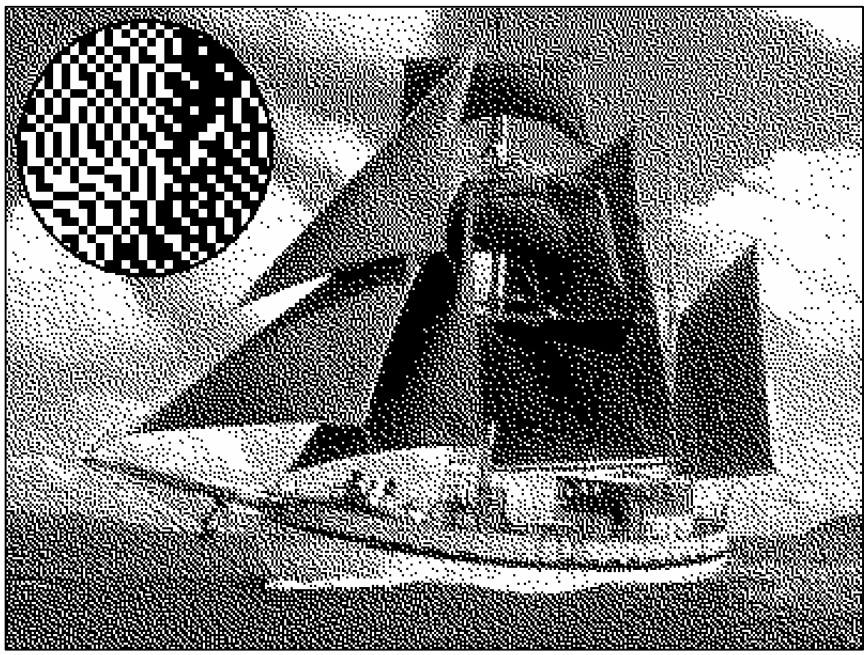


Рисунок 5.17 – Зображення, отримане методом дифузії похибки

У результаті ділянка з деяким рівнем інтенсивності заповнюється хаотичним набором світлих і темних точок, причому співвідношення між кількістю тих та інших відповідає середньої інтенсивності ділянки, рис. 5.17. Розподіл похибки на сусідні пікселі поліпшує вид деталей зображення, оскільки інформація, що міститься в зображенні, не губиться.

Метод *дифузії похибки* дає більш якісне псевдо-півтонове зображення, ніж *метод конфігурацій*, оскільки не знижує просторового дозволу і виключає появу ефекту фактури.

Методи конфігурацій і дифузії похибки застосовуються і для поліпшення якості *кольорових зображень* з малим числом кольорів, у тому числі палітрових, рис. 5.18.

Методи конфігурацій і дифузії похибки широко використовувалися на ЕОМ, що мали відеоадаптери з малою глибиною кольору. Зараз основна область застосування цих методів — друк фотографічних і інших зображень на матричних, струминних і лазерних принтерах, оскільки для цих пристроїв псевдотонування — єдиний спосіб передачі півтонів.

Рисунок 5.18 – Псевдотонування кольорового зображення методами конфігурацій (*а*) та дифузії похибки (*б* и *в*)

5.5 Дефекти растрових зображень та їхнє усунення

Дискретна структура растрового зображення приводить до деяких ***спотворень*** машинно-згенерованих зображень. Можна виділити два типи таких спотворень: ***ступінчастість*** ребер (*сходовий ефект*) і некоректна візуалізація тонких деталей або фактури.

Основна причина появи *сходового ефекту* полягає в тому, що відрізки, ребра багатокутника, колірні границі і т.д. мають безупинну природу, тоді як растрове зображення дискретно. Для представлення відрізка, ребра багатокутника і т.д. на растровому пристрої необхідно накреслити їх у *дискретних координатах*, рис. 5.19 *а*.

В алгоритмах розкладання відрізка в растр і заповнення багатокутника, інтенсивність або колір пікселя визначаються інтенсивністю або кольором єдиної точки усередині області пікселя — його центра. Якщо центр знаходиться усередині, то активується весь піксель. Якщо центр знаходився поза границею, то вся область пікселя ігнорується. У результаті виходить характерне східчасте, зазубрене ребро багатокутника або відрізок, рис. 5.19 *б*. Існує два способи усунення спотворень зображення такого роду.

Перший спосіб пов'язаний зі *збільшенням роздільної здатності растра*. Таким чином враховуються більш дрібні деталі, а "сходини" стають нерозрізненно малі. Однак, роздільна здатність сучасних дисплеїв рідко перевищує 100 пікселів на дюйм, що робить окремі пікселі, а отже і "сходини" видимими. На даний час спосіб є застосовним для високоякісних принтерів (наприклад, лазерних), з роздільною здатністю 600…1200 точок на дюйм.

Другий метод усунення спотворень зображення складається в *розмиванні країв*. При цьому піксель трактується не як точка, а як кінцева область. Інтенсивність або відтінок пікселя визначається способом *усереднення* відтінків всіх елементів, що попадають у цю область.

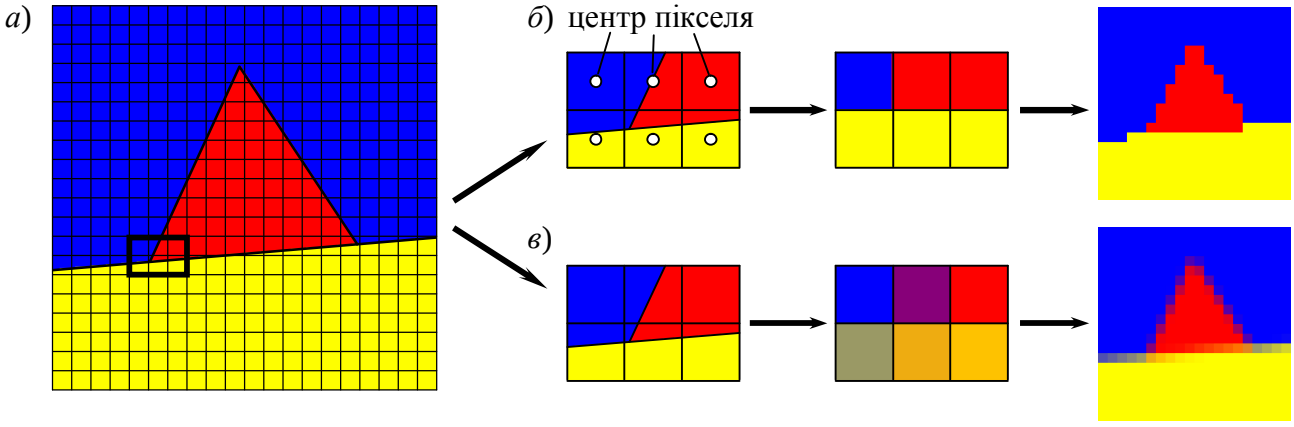


Рисунок 5.19 – Сходовий ефект: накладення безперервних елементів на растр (*а*), растеризація по центрам пікселів (*б*) та усередненням кольорів усередині пікселя (*в*)

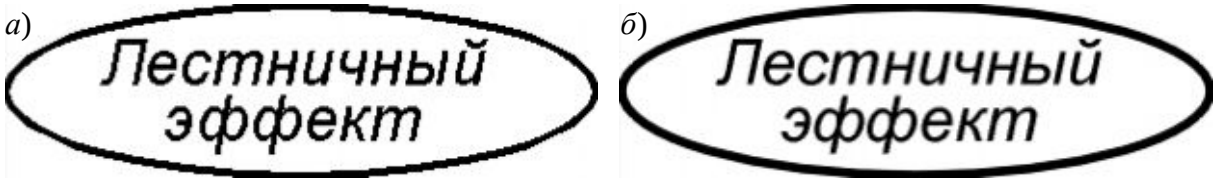


Рисунок 5.20 – Приклад зображення до (*а*) та після (*б*) усунення сходового ефекту

Сторони області пікселя утворюють *вікно, що відсікає*. Для модулювання інтенсивності пікселя використовується відношення площі отриманого в результаті відсікання багатокутника до площі пікселя. Якщо усередині пікселя знаходиться кілька багатокутників, то використовується середнє зважене їхніх атрибутів для модулювання атрибутів пікселя, рис. 5.19 *в.* У результаті на границі зафарбованої і незафарбованої зон створюється плавний півтоновий перехід, що візуально згладжує ступінчастість — границя зорово сприймається гладкою. Приклад зображення до і після усунення ступінчастості приведений на рис. 5.20 *а* і *б*.

Другий тип дефектів растра — *некоректна візуалізація тонких деталей та фактури* — найпомітніше виявляється при відображенні структур, що складаються з тонких ліній, розташованих паралельно або під деяким кутом. Якщо товщина ліній і відстань між ними менше або близькі до розміру пікселя, лінії зливаються, утворюючи деякі зафарбовані та незафарбовані зони складної форми, т.зв. *муарові візерунки*, рис. 5.21 *а*. Ефективного методу боротьби з такими ефектами практично не існує, оскільки і метод усереднення, і збільшення роздільної здатності растра можуть не дати необхідного ефекту, рис. 5.21 *б*, *в.* Тому при створенні растрових зображень слід уникати подібних регулярних структур.

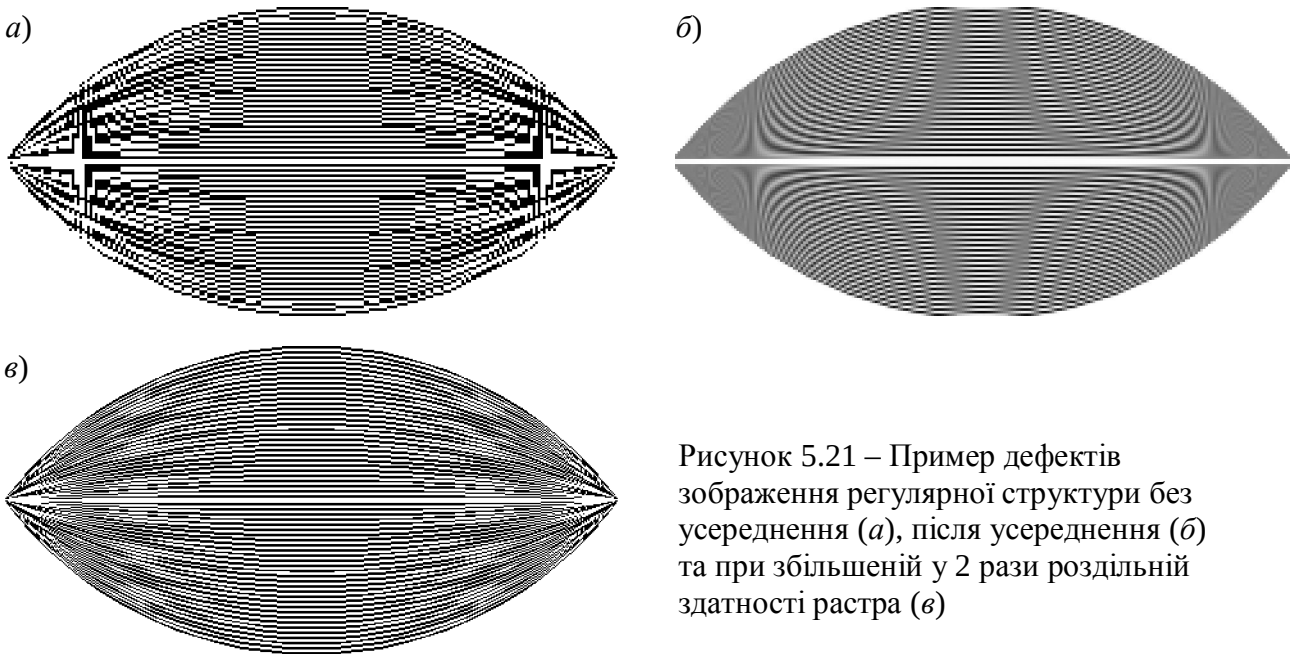


Рисунок 5.21 – Пример дефектів зображення регулярної структури без усереднення (*а*), після усереднення (*б*) та при збільшеній у 2 рази роздільній здатності растра (*в*)

6 ОСНОВИ ТРИВИМІРНОЇ ГРАФІКИ

6.1 Перетворення координат у просторі

В основі будь-якого методу побудови моделей у просторі, як і для площини, лежить ***координатний метод***. Положення точки в просторі визначається трьома координатами. У *прямокутній* чи ***Декартовій*** системі координат це *проекції вектора*, проведеного з початку координат у дану точку, на координатні осі — *x*, *y* і *z*.

Положення ***об'єкта*** в просторі може бути задано координатами деякої базової точки, що є центром його власної (*локальної*) системи координат, і *трьома кутами* повороту цієї системи координат щодо кожної з осей *глобальної системи координат*.

Перетворення координат у просторі виконується аналогічно перетворенню на площині і включає *масштабування*, *зсув* і *поворот*. У загальному виді *афінні перетворення в просторі* можуть бути представлені системою рівнянь:

$$\begin{cases} X = A\,x + B\,y + C\,z + D, \\ Y = E\,x + F\,y + G\,z + H, \\ Z = K\,x + L\,y + M\,z + N; \end{cases} \tag{6.1}$$

або в матричному виді:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ K & L & M & N \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{6.2}$$

де *x*, *y* і *z* – координати точки в локальній системі координат об'єкта; *X*, *Y* і *Z* – координати цієї ж точки в глобальній системі координат; *A*, *B* … *N* – константи перетворення. Розглянемо окремі випадки афінного перетворення.

1. ***Зсув*** центра локальної системи координат у точку (X_0, Y_0, Z_0), рис. 6.1 *а*:

$$\begin{cases} X = x + X_0, \\ Y = y + Y_0, \\ Z = z + Z_0, \end{cases} \qquad \text{або} \qquad \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{6.3}$$

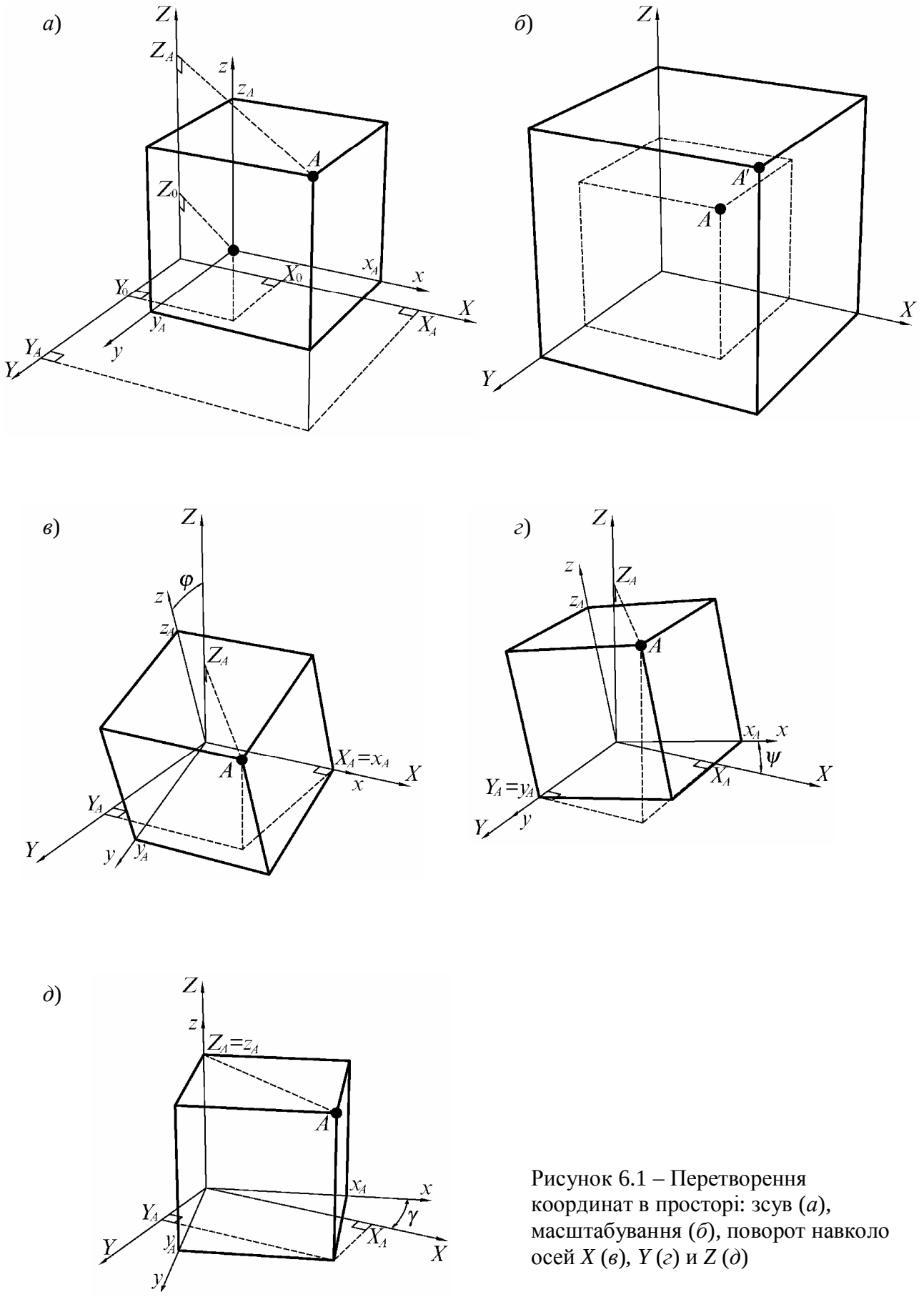


Рисунок 6.1 – Перетворення координат в просторі: зсув (*а*), масштабування (*б*), поворот навколо осей *X* (*в*), *Y* (*г*) и *Z* (*д*)

2. ***Масштабування*** (*стиск/розтягування*) з центром на початку координат і коефіцієнтами масштабування k_x, k_y і k_z, рис. 6.1 *б*:

$$\begin{cases} X = x\,k_x, \\ Y = y\,k_y, \\ Z = z\,k_z, \end{cases} \quad \text{чи} \quad \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} k_x & 0 & 0 & 0 \\ 0 & k_y & 0 & 0 \\ 0 & 0 & k_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{6.4}$$

3. ***Повороти***:

— навколо осі X на кут φ, рис. 6.1 *в*:

$$\begin{cases} X = x, \\ Y = y\cos\varphi - z\sin\varphi, \\ Z = y\sin\varphi + z\cos\varphi, \end{cases} \quad \text{чи} \quad \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi & 0 \\ 0 & \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{6.5}$$

— навколо осі Y на кут ψ, рис. 6.1 *г*:

$$\begin{cases} X = x\cos\psi - z\sin\psi, \\ Y = y, \\ Z = x\sin\psi + z\cos\psi, \end{cases} \quad \text{чи} \quad \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\psi & 0 & -\sin\psi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\psi & 0 & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{6.6}$$

— навколо осі Z на кут γ, рис. 6.1 *д*:

$$\begin{cases} X = x\cos\gamma - y\sin\gamma, \\ Y = x\sin\gamma + y\cos\gamma, \\ Z = z, \end{cases} \quad \text{чи} \quad \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi & 0 \\ 0 & \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{6.7}$$

Як можна бачити, при повороті навколо деякої осі глобальної системи координат координата точки по цій осі залишається незмінною, інші дві координати перетворяться аналогічно повороту на площині (5.2) і (5.9).

Послідовно виконуючи операції масштабування, зсуву і поворотів навколо осей X, Y і Z, можна розташувати деякий об’єкт у довільному положенні в деякій області простору.

6.2 Проекції

Переважна більшість сучасних засобів відображення комп'ютерної графіки (дисплеї, принтери) синтезують зображення *на площині*. При створенні зображень тривимірних об'єктів використовують метод ***проекцій***.

Проекція об'єкта — це зображення, отримане як результат перетинання *поверхні проектування* (площини, циліндричної, сферичної поверхонь) пучком паралельних або збіжних *променів, що проектують*, які йдуть від цього об'єкта.

У комп'ютерній графіці найбільш поширені ***паралельна*** і ***центральна*** проекції. *Паралельна проекція* створюється пучком променів, паралельних один одному, рис. 6.2 *а*. *Центральна проекція*, яку також називають ***перспективною***, виходить, що коли промені, що проектують виходять з однієї точки простору, рис. 6.2 *б*. *Перспективна проекція* дозволяє створити зображення, що ближче до тієї картинки, яку сприймає наше око, а тому виглядає реалістичніше.

При відображенні просторових об'єктів на екрані або листі паперу за допомогою принтера, необхідно знати координати проекцій точок об'єкта на екран або лист. Звичайно використовують дві системи координат — *світову* й *екранну*.

Світові координати описують дійсне положення об'єктів у просторі. У більшості випадків це *тривимірна* прямокутна система координат.

Екранні координати — це система координат (звичайно *двовимірна*) пристрою відображення, яке здійснює виведення зображень об'єктів у заданій проекції.

Для одержання зображення тривимірного об'єкта на пристрої відображення необхідно задати *перетворення координат* зі *світових* в *екранні*.

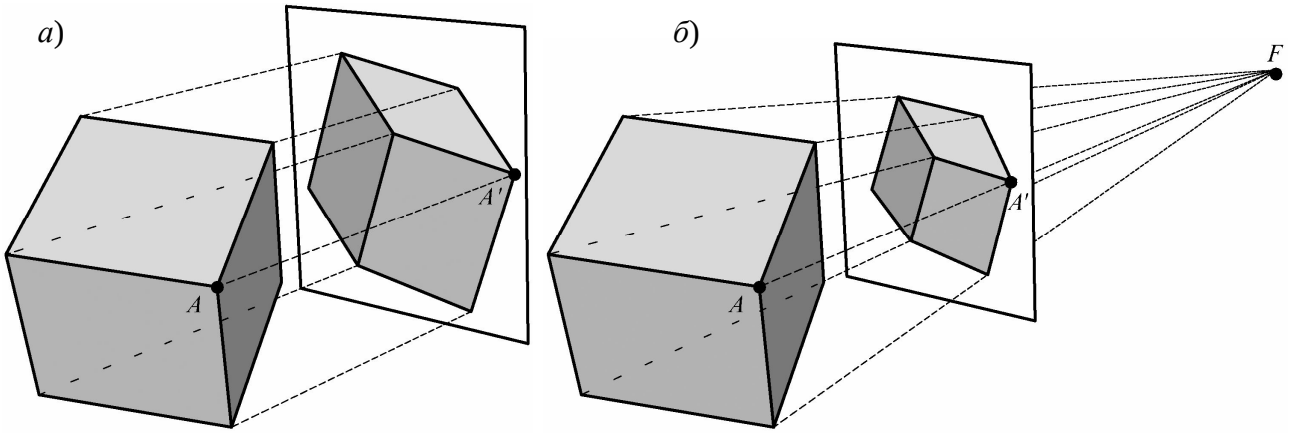


Рисунок 6.2 – Паралельна (*а*) та центральна (*б*) проекції

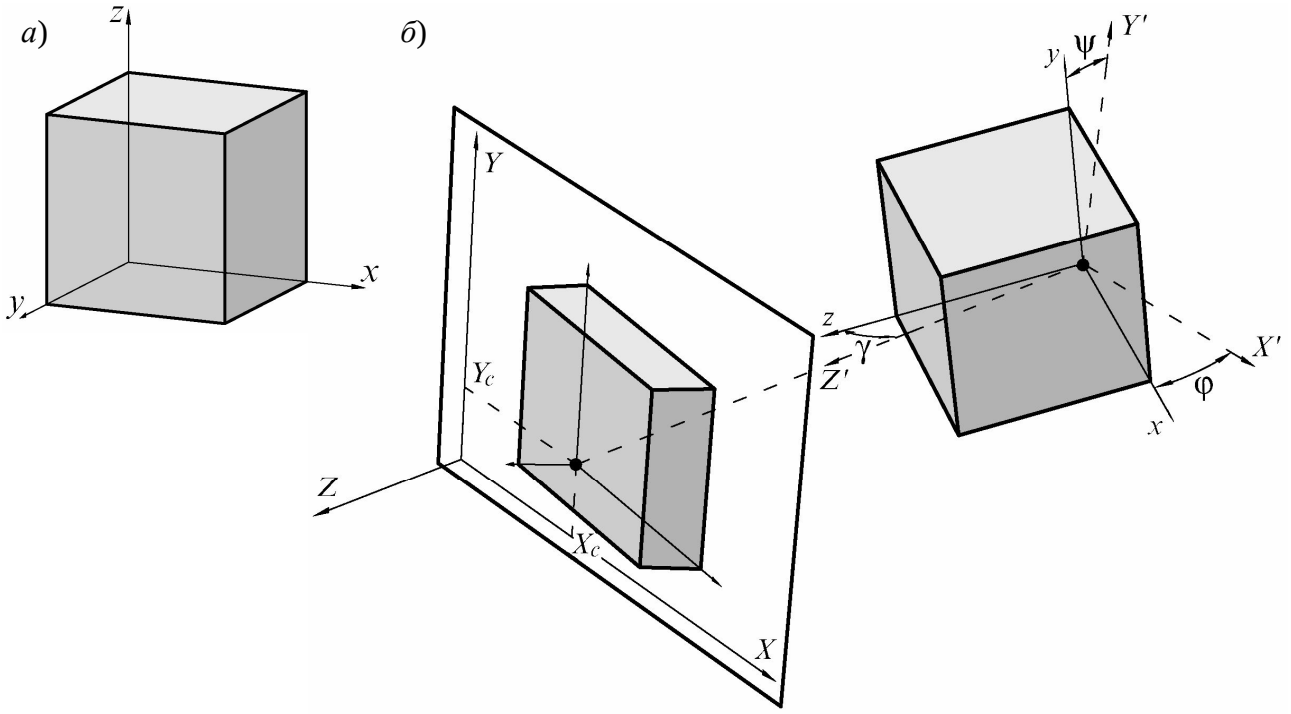


Рисунок 6.3 – Аксонометрична проекція куба (*а*) та перетворення координат (*б*)

Для створення спрощених зображень об’єктів (наприклад, у процесі тривимірного моделювання) використовують ***аксонометричну проекцію*** — різновид паралельної проекції, коли всі промені, що проектують, спрямовані під прямим кутом до площини проектування.

Аксонометрична проекція, рис. 6.3 *а*, має такі особливості:

— проекції відрізків паралельних прямих *завжди паралельні*;

— *довжини проекцій* однакових відрізків паралельних прямих *завжди однакові*, незалежно від їхнього віддалення від площини проектування.

Незважаючи на те, що аксонометрична проекція «точно» відтворює об’єкт, людина сприймає таке зображення як «плоске», позбавлене об’єму.

Розглянемо *перетворення координат* для одержання аксонометричної проекції. Нехай необхідно одержати аксонометричну проекцію деякого об’єкта (наприклад, куба), заданого у світовій системі координат (x, y, z), на розташовану довільно площину екрана, рис. 6.3 *б*. Маємо екранну систему координат (X, Y, Z), вісь Z спрямована перпендикулярно площині екрана. Щодо *екранної* системи координат *світова* система зміщена на величини X_C, Y_C і Z_C уздовж відповідних осей і повернена на кути φ, ψ і γ навколо осей X, Y і Z відповідно.

Уведемо допоміжну систему координат (X', Y', Z'), осі якої паралельні відповідним осям *екранної системи*, а центр збігається з центром *світової системи*. Таку систему координат називають ***видовою***. Вона визначає *ракурс* показу об’єкта.

Для перетворення координат об’єкта зі світової у видову систему, необхідно виконати множення матриці координат на матриці повороту навколо кожної з осей (6.5), (6.6) і (6.7). Умовно це можна записати так:

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} \text{Поворот} \\ \text{вокруг} \\ \text{оси } Z' \\ \text{на } \gamma \end{bmatrix} \begin{bmatrix} \text{Поворот} \\ \text{вокруг} \\ \text{оси } Y' \\ \text{на } \psi \end{bmatrix} \begin{bmatrix} \text{Поворот} \\ \text{вокруг} \\ \text{оси } X' \\ \text{на } \varphi \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{6.8}$$

Щоб одержати екранні координати, потрібно виконати зсув об’єкта на величини X_C, Y_C і Z_C уздовж відповідних осей. Зсув уздовж координати Z виконувати необов’язково, оскільки для побудови проекції достатньо координат X і Y:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \text{Сдвиг} \\ \text{на} \\ X_C \text{ и } Y_C \\ \end{bmatrix} \begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix}. \tag{6.9}$$

Повна формула перетворення координат об’єкта виглядає в такий спосіб:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \text{Сдвиг} \\ \text{на} \\ X_C \text{ и } Y_C \\ \end{bmatrix} \begin{bmatrix} \text{Поворот} \\ \text{вокруг} \\ \text{оси } Z' \\ \text{на } \gamma \end{bmatrix} \begin{bmatrix} \text{Поворот} \\ \text{вокруг} \\ \text{оси } Y' \\ \text{на } \psi \end{bmatrix} \begin{bmatrix} \text{Поворот} \\ \text{вокруг} \\ \text{оси } X' \\ \text{на } \varphi \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{6.10}$$

Для одержання проекції об’єкта на площину екрана тепер достатньо дорівняти координату Z кожної його точки до нуля. Тоді дві координати X і Y, що залишилися, зададуть положення проекції цієї точки на площині екрана.

Добуток усіх матриць зсуву і поворотів, необхідних для перетворення координат об’єкта зі світових в екранні, звичайно називають ***матрицею видового афінного перетворення***. Тоді (6.10) можемо записати:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \text{Матриця} \\ \text{видового} \\ \text{афінного} \\ \text{перетворення} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{6.11}$$

Оскільки відображуваний об’єкт може бути набагато більше, чи навпаки, набагато менше області екрана, у матрицю видового перетворення звичайно вводять *матрицю масштабування* (6.4).

Зображення, одержувані за допомогою фото- і кінокамер, так само, як і зображення, що сприймається оком людини, відрізняються від аксонометричних, оскільки відносяться до *центральних* або ***перспективних проекцій***. Тому для одержання реалістичного зображення тривимірних об’єктів необхідно генерувати їхню ***перспективну проекцію*** на площину екрана. Таку проекцію можна уявити собі як зображення на склі, через яке дивиться спостерігач, розташований у точці *F*, рис. 6.4. Розглянемо методику одержання такої проекції.

Нехай деякий об’єкт заданий у *світових* координатах (*x*, *y*, *z*). Розташуємо *видову систему координат* (*X*, *Y*, *Z*) так, щоб точка *F* лежала на осі *Z* і мала координати (0, 0, Z_F). Площина екрана перетинає вісь *Z* у точці *3* з координатою $Z_{пл}$.

Перетворимо координати об’єкта у *видові*, помноживши на матрицю видового афінного перетворення (6.11). Довільна точка *A* об’єкта, що має видові координати (X_A, Y_A, Z_A), проектується променем *AF* у точку *A′* на площині екрана, яка має екранні координати ($X_{пA}$, $Y_{пA}$). Виходячи з подоби трикутників *1F3* та *1′F3′* а також *2F3* і *2′F3′* запишемо пропорції:

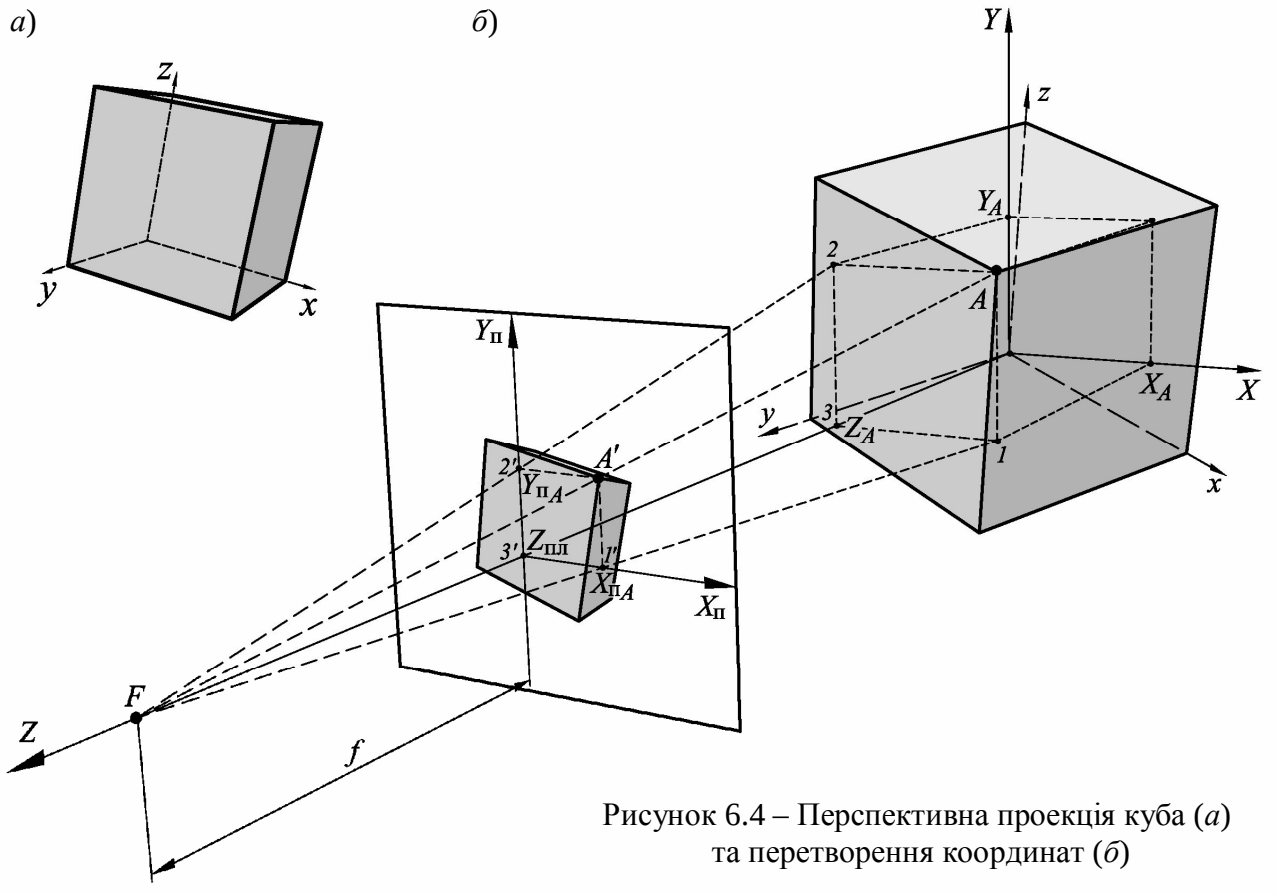


Рисунок 6.4 – Перспективна проекція куба (*а*) та перетворення координат (*б*)

$$\begin{cases} \dfrac{X_{\text{п}A}}{Z_F - Z_{\text{пл}}} = \dfrac{X_A}{Z_F - Z_A}, \\ \dfrac{Y_{\text{п}A}}{Z_F - Z_{\text{пл}}} = \dfrac{Y_A}{Z_F - Z_A}. \end{cases} \tag{6.12}$$

Знайдемо координати проекції:

$$\begin{cases} X_{\text{п}A} = X_A \dfrac{Z_F - Z_{\text{пл}}}{Z_F - Z_A}, \\ Y_{\text{п}A} = Y_A \dfrac{Z_F - Z_{\text{пл}}}{Z_F - Z_A}. \end{cases} \tag{6.13}$$

Відстань $f = Z_F - Z_{\text{пл}}$ від точки сходу променів (***фокуса***) до площини екрана називають ***фокусною відстанню***. Тоді можемо записати

$$\begin{cases} X_{\text{п}A} = \dfrac{f\ X_A}{Z_F - Z_A}, \\ Y_{\text{п}A} = \dfrac{f\ Y_A}{Z_F - Z_A}, \end{cases} \quad \text{чи} \quad \begin{cases} X_{\text{п}A} = \boldsymbol{\Pi}_x\left(f, X_A, Z_A\right), \\ Y_{\text{п}A} = \boldsymbol{\Pi}_y\left(f, Y_A, Z_A\right), \end{cases} \tag{6.14}$$

де $\boldsymbol{\Pi}$ – функція перспективного перетворення координат. На відміну від видового перетворення, перспективне перетворення — *нелінійне*. Значення коефіцієнтів перетворення залежать від координати *Z*, що знаходиться в знаменнику.

Перспективна проекція має наступні особливості, див. рис. 6.4 *а*:

— *прямі лінії* зображуються прямими лініями;

— *паралельні прямі*, які паралельні площини проектування, зображуються паралельними прямими;

— *паралельні прямі*, непаралельні площини проектування, зображуються прямими, що сходяться в одній точці;

— *довжини проекцій* однакових відрізків паралельних прямих будуть різними в залежності від їхньої віддаленості від площини проектування.

Вигляд перспективної проекції визначається взаємним розташуванням просторового об’єкта, площини проектування і точки фокуса. Для того, щоб перспективне зображення об’єкта сприймалося найбільше природно, необхідно, щоб кут α, під яким людина розглядає зображення (на екрані, листі папера й ін.) відповідав тому куту α′, під яким ми бачили б зображуваний тривимірний об’єкт, якби він існував у дійсності, рис. 6.5.

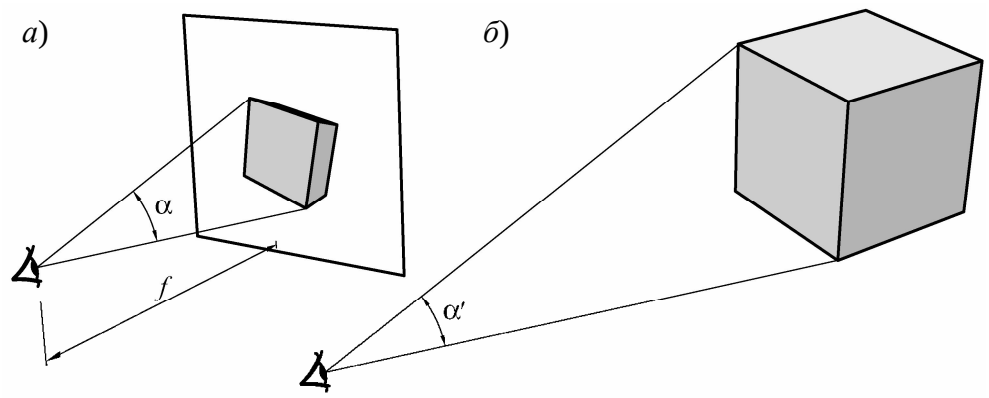


Рисунок 6.5 – Кут огляду зображення (*а*) та об’єкта (*б*)

Якщо фокусна відстань занадто мала і кут α′ огляду об’єкта з точки фокуса значно більше кута α огляду зображення спостерігачем, одержимо спотворене зображення з перебільшеною, надмірною перспективністю, рис. 6.6 *а*. Якщо ж фокусна відстань занадто велика а кут α′ дуже малий, зображення буде близьким до аксонометричного і не буде передавати відчуття глибини простору, рис. 6.6 *б*. За умови рівності кутів α′ і α зображення буде найбільш природним, рис. 6.6 *в*.

Проектування на площину зображень з великим кутом огляду приводить до значних спотворень, що видно на рис. 6.6 *а*. Тому для одержання зображень, що охоплюють значну область простору навколо глядача (*панорамних*) використовують проектування на криволінійні поверхні — *циліндричну* і *сферичну*.

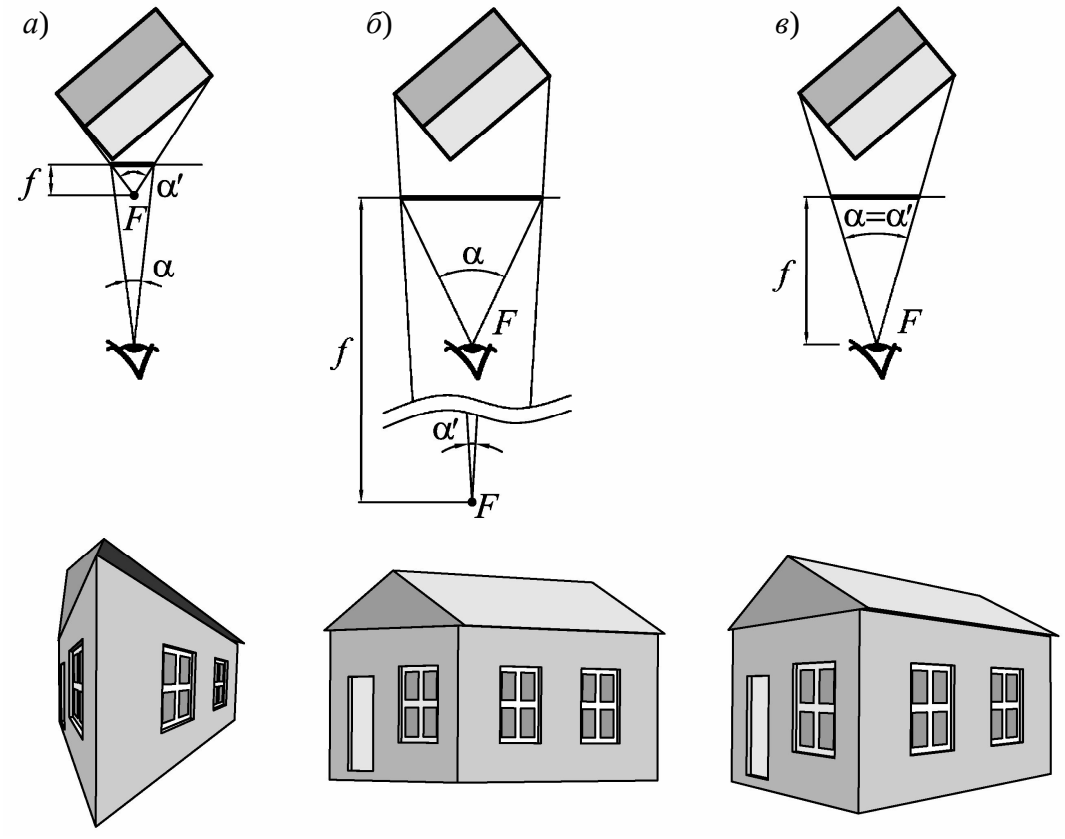


Рисунок 6.6 – Перспективне зображення, побудоване при α′ > α (*а*), α′ < α (*б*) и α′ = α (*в*)

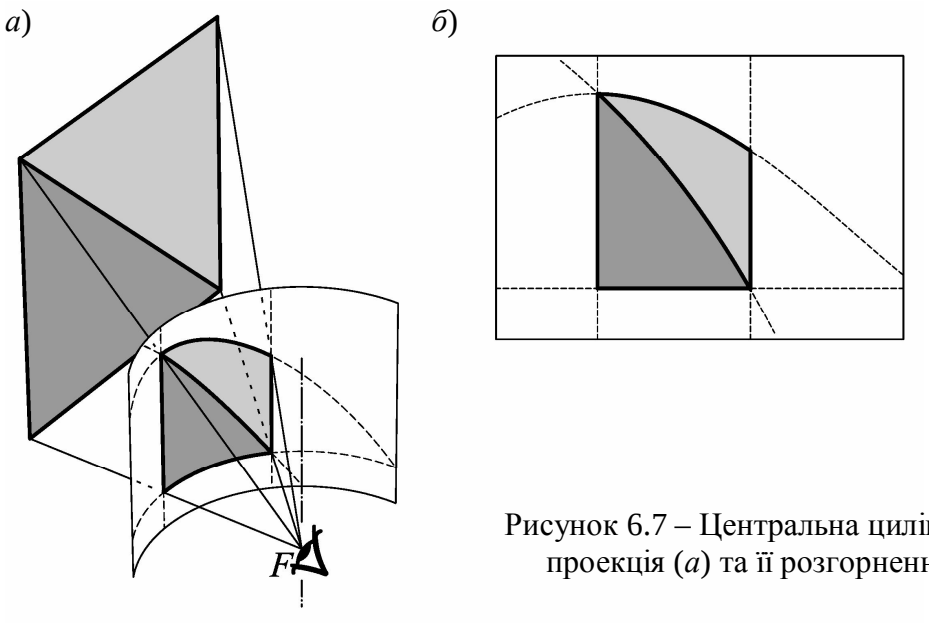


Рисунок 6.7 – Центральна циліндрична проекція (*а*) та її розгорнення (*б*)

У ***циліндричній проекції*** точка фокуса розташовується на осі циліндра (звичайно вертикальній), а зображення будується по точках перетинання з поверхнею циліндра променів, які йдуть із точки фокуса до об'єктів, що проектуються, рис. 6.7. Приклад розгорнутого зображення, отриманого циліндричною проекцією, показаний на рис. 6.8.

Характерним для циліндричної проекції є те, що:

— *розміри* зображень об'єктів зменшуються в міру їхнього віддалення від осі циліндра *незалежно від напрямку*, при цьому невеликі об'єкти зображуються без значних спотворень, навіть якщо вони розташовані під великим кутом по горизонталі до осі погляду;

— *прямими* на плоскому розгорненні проекції відображаються тільки відрізки прямих, що лежать у площинах, які проходять через вісь циліндра або через точку фокуса перпендикулярно осі;

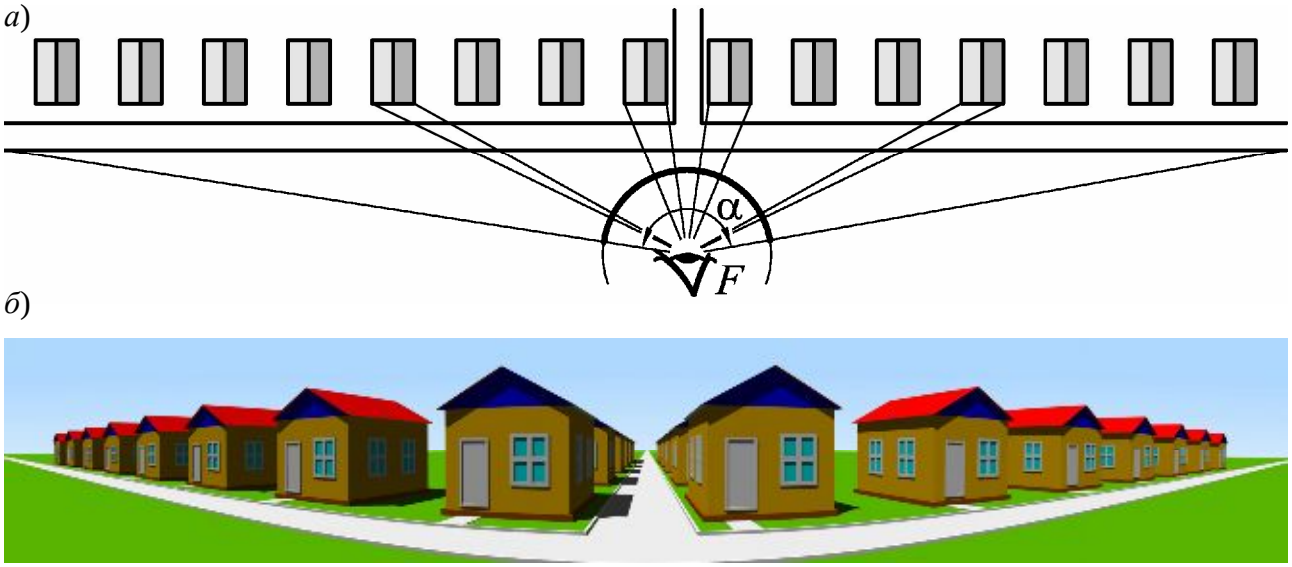


Рисунок 6.8 – Отримання панорамного зображення в центральній циліндричній проекції (*а*) та його розгорнення (*б*)

— відрізки будь-яких інших прямих на поверхні циліндра утворять *дуги еліпса*, а після розгортання зображення на площині відображаються ділянками *синусоїди*.

Основною перевагою циліндричної проекції перед проекцією на площину є великий кут зйомки, що може досягати 360° — у такому випадку одержуємо *кругову панораму*. Слід також зазначити, що скривлення прямих виявляється лише на плоскому розгорненні. Для глядача, який бачить зображення на циліндричному екрані, розташувавшись в точці фокуса (або досить близько до нього), будь-які спотворення непомітні. Тому циліндрична проекція використовується в широкоекранних «панорамних» кінотеатрах, де зображення проектується на вигнутий екран — глядач бачить події, що відбуваються не тільки перед ним, але і ліворуч, праворуч і навіть позаду.

У ***сферичній проекції*** поверхнею проектування служить *сфера*, а точка фокуса розташовується в її центрі. Сферична проекція багато в чому подібна до циліндричної, але дозволяє одержати великий кут зйомки як по горизонталі, так і по вертикалі — аж до відображення всього навколишнього простору. Сферична проекція дозволяє найбільш точно і повно відобразити те, що бачить людина, і, можливо, буде використовуватися в панорамних кінотеатрах майбутнього. Зараз сферична проекція використовується в планетаріях для імітації зоряного неба на напівсферичному екрані.

6.3 Моделі опису поверхонь

Для того, щоб синтезувати на комп'ютері зображення деякого тривимірного об'єкта (або групи об'єктів), необхідно мати опис цього об'єкта — ***математичну модель***. Математичне моделювання тривимірних об'єктів використовується в САПР, художній та кінематографічній графіці, 3D-іграх і симуляторах, при цьому підходи до моделювання поверхонь об'єктів можуть бути різними. Відомі наступні методи:

1. ***Аналітична модель*** — опис поверхні математичними формулами. Поверхня може бути задана у вигляді *функції двох аргументів*:

$$z = f\left(x, y\right), \tag{6.15}$$

або у вигляді *рівняння*:

$$F\left(x, y, z\right) = 0, \tag{6.16}$$

але найчастіше зустрічається *параметрична форма* опису поверхні:

$$\begin{cases} x = f_x(s,t), \\ y = f_y(s,t), \\ z = f_z(s,t), \end{cases} \tag{6.17}$$

де f_x, f_y, f_z – функції, що визначають форму поверхні; s і t – параметри цих функцій. Параметрична форма дозволяє простіше описувати складні поверхні, у тому числі замкнені та поверхні, що відповідають неоднозначним функціям. Наприклад, сферу можна описати як функцію двох аргументів, рівнянням і в параметричній формі в такий спосіб:

$$z = \pm\sqrt{R^2 - x^2 - y^2}\,; \tag{6.18}$$

$$x^2 + y^2 + z^2 - R^2 = 0; \tag{6.19}$$

$$\begin{cases} x = R\sin s\cos t, \\ y = R\sin s\sin t, \\ z = R\cos t. \end{cases} \tag{6.20}$$

Для опису складних поверхонь часто використовують ***сплайни*** — спеціальні функції, що апроксимують окремі фрагменти складної поверхні. При цьому модель складної поверхні утвориться декількома сплайнами.

Звичайно використовують *кубічні* сплайни, задані в параметричній формі, оскільки третій ступінь є найменшим, що дозволяє описувати будь-яку форму і при стикуванні сплайнів забезпечувати безупинну другу похідну — одержувати гладку поверхню без зламів. У САПР і 3D-графіці одержали поширення *сплайни Безьє* і *NURBS*.

Поверхня Безьє в загальному виді описується залежністю

$$\begin{cases} x(s,t) = \sum\limits_{i=0}^{m}\sum\limits_{j=0}^{n} C_i s^i (1-s)^{m-i} C_j t^j (1-t)^{n-i} x_{ij}, \\ y(s,t) = \sum\limits_{i=0}^{m}\sum\limits_{j=0}^{n} C_i s^i (1-s)^{m-i} C_j t^j (1-t)^{n-i} y_{ij}, \\ z(s,t) = \sum\limits_{i=0}^{m}\sum\limits_{j=0}^{n} C_i s^i (1-s)^{m-i} C_j t^j (1-t)^{n-i} z_{ij}, \end{cases} \tag{6.21}$$

де x_{ij}, y_{ij} і z_{ij}– координати точок-орієнтирів P_{ij}; C_i і C_j – коефіцієнти бінома Ньютона. Залежність має степінь $m \times n$ і однозначно описується сіткою з $(m + 1)(n + 1)$ опорних точок. Параметри s і t змінюються в діапазоні від 0 до 1; кожній парі їхніх значень відповідає деяка точка на поверхні.

Кубічний сплайн Безьє відповідає $m = n = 3$:

$$\begin{cases} x(s,t) = (1-s)^3(1-t)^3 x_{0,0} + 3(1-s)^3 t\,(1-t)^2 x_{0,1} + 3(1-s)^3 t^2(1-t)\,x_{0,2} + \\ \quad + (1-s)^3 t^3 x_{0,3} + 3s(1-s)^2(1-t)^3 x_{1,0} + 9st(1-s)^2(1-t)^2 x_{1,1} + \\ \quad + 9st^2(1-s)^2(1-t)\,x_{1,2} + 3st^3(1-s)^2 x_{1,3} + 3s^2(1-s)(1-t)^3 x_{2,0} + \\ \quad + 9s^2 t(1-s)(1-t)^2 x_{2,1} + 9s^2 t^2(1-s)(1-t)\,x_{2,2} + 3s^2 t^3(1-s)x_{2,3} + \\ \quad + s^3(1-t)^3 x_{3,0} + 3s^3 t\,(1-t)^2 x_{3,1} + 3s^3 t^2(1-t)\,x_{3,2} + s^3 t^3 x_{3,3}, \\ y(s,t) = (1-s)^3(1-t)^3 y_{0,0} + 3(1-s)^3 t\,(1-t)^2 y_{0,1} + \ \ldots + 3s^3 t^2(1-t)\,y_{3,2} + s^3 t^3 y_{3,3}, \\ z(s,t) = (1-s)^3(1-t)^3 z_{0,0} + 3(1-s)^3 t\,(1-t)^2 z_{0,1} \ + \ldots + 3s^3 t^2(1-t)\,z_{3,2} + s^3 t^3 z_{3,3}, \end{cases} \tag{6.22}$$

і може бути заданий сіткою з 4 × 4 = 16 опорних точок, рис. 6.9.

Кубічна поверхня Безьє може приймати дуже різноманітні форми, що дозволяє використовувати її для моделювання складних поверхонь.

Поверхні ***NURBS*** подібні з поверхнями Безьє, але задаються сіткою з довільного, у т.ч. дуже великого числа опорних точок, можуть бути замкненими. Сплайни *NURBS* дозволяють апроксимувати складні поверхні (наприклад, поверхні живих об’єктів) простіше, ніж сплайни Безьє.

Існують кілька варіацій аналітичного методу опису поверхонь і тіл, які використовують у різних областях моделювання. У САПР для створення моделей тіл типу деталей машин, що характеризуються відносно простими

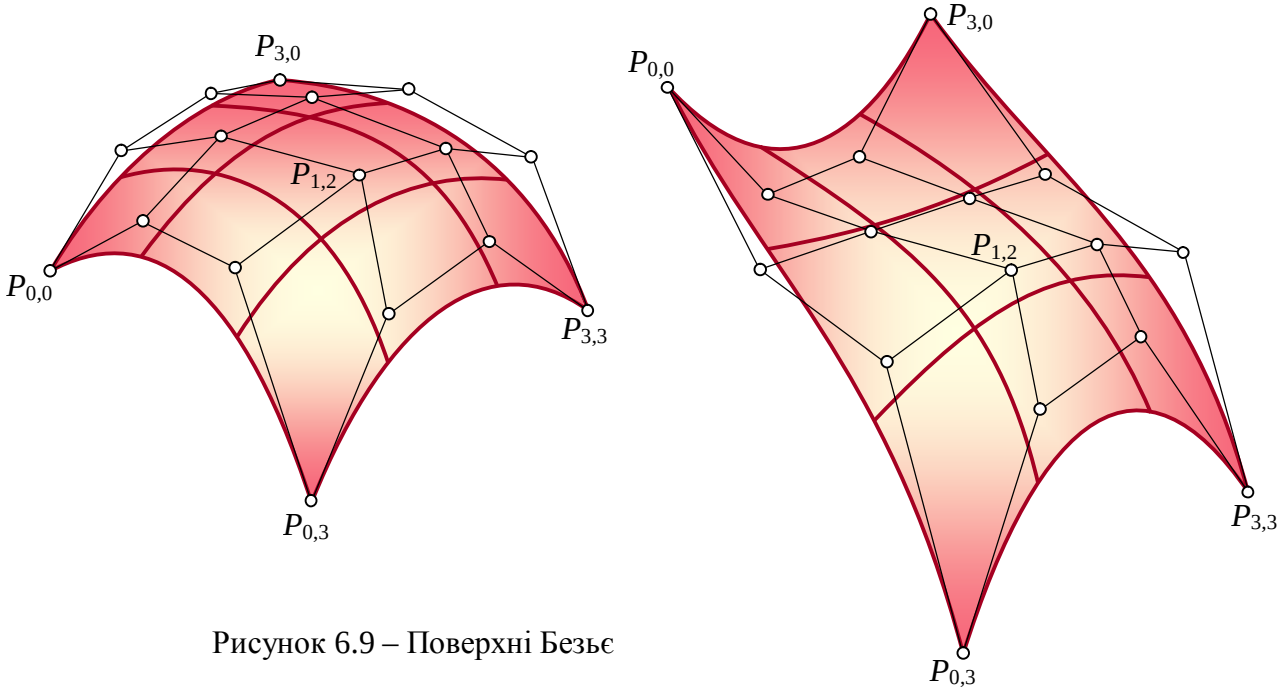


Рисунок 6.9 – Поверхні Безьє

формами, застосовується ***твердотілове моделювання***. Відомі два методи твердотілового моделювання: *метод геометричних примітивів* і *кінематичний метод*.

Метод геометричних примітивів полягає у створенні моделі об'єкта з набору стандартних об'ємних фігур, які досить просто описуються аналітично. Звичайно це прямокутний паралелепіпед, циліндр, конус, сфера, тороїд, рис. 6.10 *а*. Модель об'єкта створюється шляхом виконання топологічних операцій над примітивами: додаванням, відніманням, перетинанням, усіканням площиною, рис. 6.10 *б*, *в*.

Кінематичний метод дозволяє описати поверхні як слід у просторі від руху деякої лінії (утворюючої) по заданій траєкторії. Звичайно використовують рух уздовж відрізка прямої (*видавлювання*), *обертання*, а також рух уздовж *довільної траєкторії*, рис. 6.10 *г*.

Твердотілові моделі подають інформацію не тільки про поверхню об'єкта, але і про його обсяг, масу. Крім того, твердотілові аналітичні моделі дозволяють виконувати *топологічні операції* над об'єктами: одержувати розрізи, перетини, складати і віднімати об'єкти. Наприклад, щоб одержати

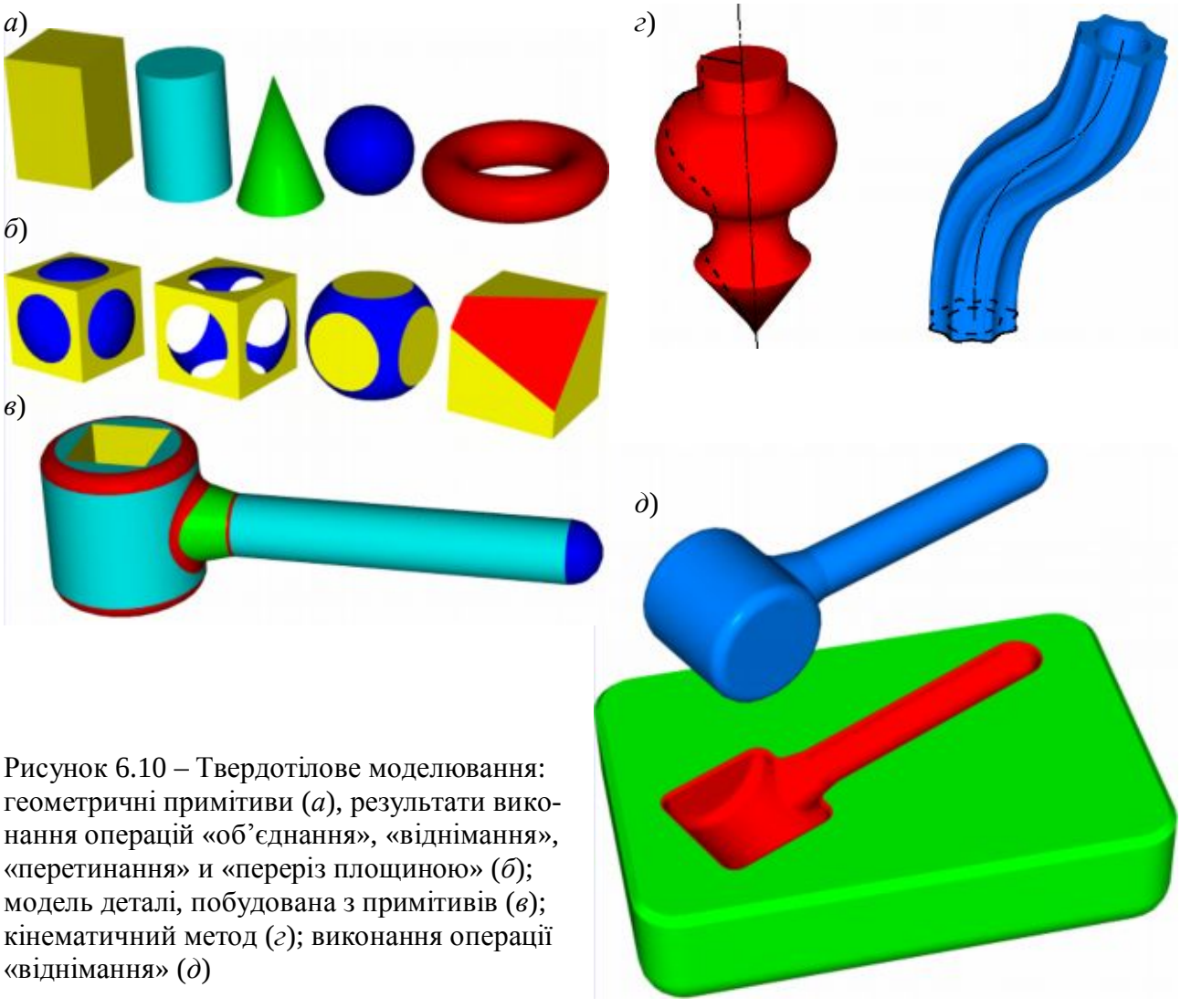


Рисунок 6.10 – Твердотілове моделювання: геометричні примітиви (*а*), результати виконання операцій «об'єднання», «віднімання», «перетинання» и «переріз площиною» (*б*); модель деталі, побудована з примітивів (*в*); кінематичний метод (*г*); виконання операції «віднімання» (*д*)

модель пресформи, достатньо з її заготовки відняти модель готової деталі, рис. 6.10 *г*. Твердотілові моделі дають можливість знаходити області взаємного перетинання об'єктів, наприклад, для аналізу помилок взаємного розташування деталей у проектованому вузлі.

У цілому *аналітичні моделі* найбільш придатні для операцій аналізу поверхонь, дозволяють досить просто розрахувати координати вектора нормалі до кожної точки поверхні. Недоліки аналітичних моделей — складність формул опису, необхідність виконання великих обсягів обчислень для побудови реалістичних зображень поверхонь. Тому в інтерактивній 3D-анімації (іграх, симуляторах) аналітичні моделі поверхонь не використовуються. Основна область застосування аналітичних моделей — САПР і високоякісна 3D-анімація (художні фільми).

Для прискорення відображення об'єктів, заданих аналітичними моделями, більшість програм виконують їхнє попереднє перетворення в *полігональну модель*, яка візуалізується за допомогою більш простих і швидкодіючих алгоритмів.

2. ***Полігональні моделі*** апроксимують поверхні *ділянками площин*, обмеженими прямими лініями — ***полігонами***.

Найчастіше полігон являє собою *трикутник*, заданий координатами трьох його ***вершин***, рис. 6.11 *а*. Представлення поверхонь трикутними гранями широко використовується в різних областях — комп'ютерних іграх, художній графіці, САПР. Трикутник — базовий елемент для сучасних відеоадаптерів з 3D-акселераторами. Пов'язано це, насамперед, з простотою алгоритмів побудови зображення трикутника, визначення вектору нормалі, освітленості, відсікання невидимих ділянок і т.д.

Щоб створити *полігональну модель* деякої криволінійної поверхні (її називають ще *полігональною сіткою*), необхідно задати в просторі набір полігонів, що стикаються один з одним по гранях і загальних вершинах. Найпростіше це зробити, задавши ***масив координат вершин*** розміром $m \times n$, при цьому вершинами деякого полігона будуть точки $P_{i,j}$, $P_{i+1,j}$ і $P_{i,j+1}$, рис. 6.11 *б*. Така модель найбільш економна по займаній пам'яті, оскільки координати вершин для всіх суміжних полігонів записуються один раз. Але при цьому необхідність завдання прямокутного масиву накладає деякі обмеження на можливості моделювання: ускладнює стикування з іншими поверхнями, не дозволяє одержувати розриви, «отвори» у поверхні, на деяких ділянках приводить до надлишку вершин.

Більш гнучкою є ***векторна полігональна модель***, у якій кожен полігон являє собою окремий елемент, рис. 6.11 *в*. Щоб уникнути багаторазового перезапису координат вершин для суміжних полігонів, спочатку в пам'яті формується *масив координат вершин*, кожній з який привласнюється номер. Тепер, щоб задати полігон, досить указати номера його вершин. Крім того, такий спосіб полегшує редагування полігональної моделі — змінивши координату деякої вершини, ми автоматично змінимо всі суміжні з нею полігони.

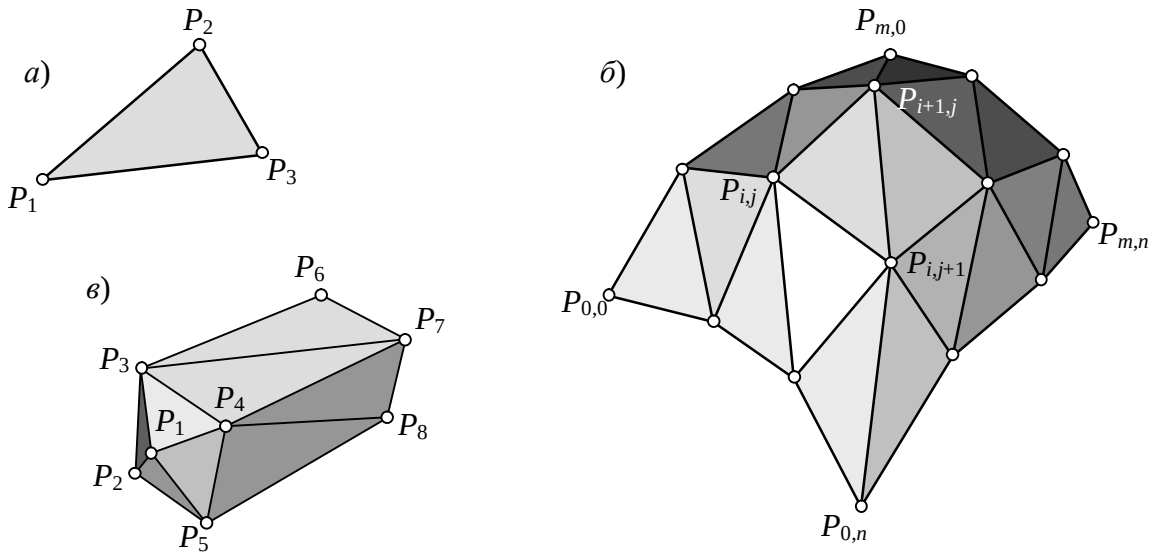


Рисунок 6.11 – Полігон (*а*), полігональна сітка (*б*) та лінійно-вузлова модель (*в*)

Часто використовують більш складну ***лінійно-вузлову модель***, у якій крім масиву вершин створюється масив ребер — кожне ребро задається номерами двох вершин. Полігон задається зазначенням номерів ребер, що його обмежують. Така модель дозволяє задавати властивості ребер, необхідні для побудови реалістичних зображень поверхонь (наприклад, чи потрібно це ребро «згладжувати», чи воно має бути гострим), а також виключає багаторазове промальовування ребер при відображенні «дротової моделі» поверхні.

Позитивні риси полігональних моделей наступні:

— зручність переміщення, обертання і масштабування об’єктів — достатньо для усіх вершин виконати відповідне перетворення координат;

— прості і швидкодіючі алгоритми візуалізації, апаратна підтримка цих алгоритмів у сучасних графічних системах, що дозволяє реалізувати інтерактивну анімовану 3D-графіку — створювати до 20 і більше зображень (кадрів) у секунду.

Поряд з цим варто виділити недоліки полігональних моделей:

— погрішності моделювання при апроксимації плоскими гранями криволінійних поверхонь; для одержання достатньо точних моделей необхідно використовувати дуже велике число полігонів, що збільшує обсяг пам’яті, займаний описом об’єкта і знижує швидкість побудови зображення;

— для створення реалістичних зображень об’єктів необхідно використовувати спеціальні алгоритми, що «приховують» грані;

— істотно ускладнюється виконання топологічних операцій над об’єктами: одержання розрізів і перетинів, знаходження областей перетинання об’єктів і т.п.

3. ***Воксельна модель*** — тривимірний растр.

Воксель — елемент обсягу (*voxel — volume element*). За аналогією з пікселями звичайного растра вокселі заповнюють об’єм у тривимірному растрі, розташовуючись у вузлах рівномірної сітки. Будь-який воксель може мати свій колір і, крім того, прозорість. Повна прозорість вокселя означає порожнечу відповідної точки обсягу.

Основна характеристика воксельної моделі — *роздільна здатність*, — кількість вокселів у визначеному об’ємі. Вона визначає точність моделювання тривимірних об’єктів.

Представлення об’ємних об’єктів у вигляді воксельних моделей використовують у комп’ютерних системах, призначених для медичних цілей. Наприклад, при скануванні томографом виходять зображення зрізів об’єкта, що потім поєднуються у виді об’ємної моделі для подальшого аналізу. Воксельний метод використовується також для графічних пристроїв, що створюють дійсно об’ємні зображення.

Воксельна модель забезпечує простий опис складних об’єктів і сцен, спрощує процедуру відображення (для відображення воксельної моделі досить зобразити «кубики» відповідного кольору у всіх точках сітки растра, крім «прозорих» точок), не вимагає виконання процедур відсікання невидимих частин об’єктів (досить відображати шари вокселів у порядку від далекого до ближнього, при цьому ближні вокселі перекриють далекі). Крім того, спрощується виконання топологічних операцій, наприклад — виконання розрізів, рис. 6.11, — треба просто зробити «невидимими» визначені групи вокселів.

Основний недолік воксельної моделі, що обмежує область її застосування — надзвичайно велика кількість інформації, необхідна для представлення об’ємних даних. Наприклад, об’єм 256×256×256 має не занадто велику роздільну здатність, але містить понад 16 мільйонів вокселів. Значні витрати пам’яті обмежують роздільну здатність, точність моделювання, швидкість виведення зображення. Крім того, як і звичайний растр, воксельна модель утруднює редагування, масштабування об’єктів.

Незважаючи на зазначені недоліки, з ростом можливостей комп’ютерної техніки воксельні моделі можуть знайти застосування в різних областях.

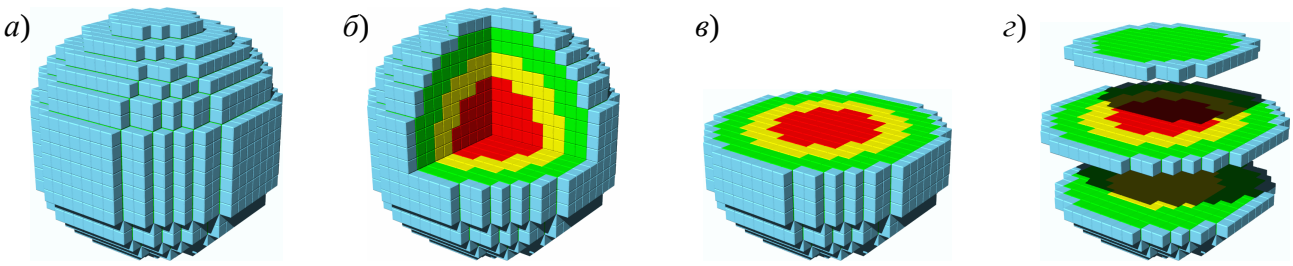


Рисунок 6.12 – Воксельна модель розміром 20×20×20 (*а*) та її розрізи (*б*, *в*, *г*)

7 ВІЗУАЛІЗАЦІЯ ТРИВИМІРНИХ ОБ'ЄКТІВ

7.1 Способи та рівні візуалізації

Будь-який тривимірний об'єкт може бути зображений по-різному, залежно від призначення створюваного зображення і використовуваного при цьому алгоритму. ***Способи візуалізації*** за характером зображень, що одержуються, і степенем складності алгоритмів можна розділити на такі рівні:

1. ***Каркасна візуалізація*** («*дротова модель*») полягає у рисуванні всіх елементів каркаса моделі — ребер полігонів, дуг еліпсів, сплайнових кривих і т.п. При цьому поверхні моделі «прозорі» та видні всі елементи об'єктів, рис. 7.13 *а*.

Для побудови зображення необхідно перетворити координати вершин полігонів і базових точок кривих зі світових в екранні координати, після чого використовуються прості растрові алгоритми виведення ліній. Каркасна візуалізація — найбільш швидка і використовується для відображення моделей об'єктів у процесі редагування.

2. ***Візуалізація з видаленням невидимих точок*** передбачає рисування тільки тих елементів об'єкта, що не приховані від спостерігача іншими об'єктами, рис. 7.13 *б*. Існує кілька методик видалення невидимих елементів.

Найбільш простий метод, застосовуваний для полігональних моделей — ***сортування граней по глибині***. Полігони малюються в порядку від самих далеких до найближчих; при заливанні ближніх полігонів стираються невидимі частини далеких. Метод найбільш ефективний при побудові поверхонь, заданих функціями $z = f(x, y)$, рис. 7.14 *а*. При рисуванні складних об'єктів, грані яких можуть розташовуватися довільним образом, перетинатися, рис. 7.14 *б*, метод дає похибки. Окрім того, метод потребує обов'язкового заповнення полігонів, що сповільнює процес відображення.

У ***методі горизонту***, ***що плаває***, грані виводяться в зворотному порядку — від ближніх до далеких. На кожному кроці границі граней утворюють дві лінії — *верхній* і *нижній горизонти*, що зберігаються у виді двох одномірних масивів. Кожному значенню координати *y* растра відповідають значення верхньої і нижньої границі. При виведенні наступної грані малюється тільки те, що вище верхнього або нижче нижнього горизонтів. Кожна наступна грань піднімає верхній і опускає нижній горизонти відповідно своїм границям. Метод має ті ж обмеження і таку ж область застосування, але більш швидкодіючий, оскільки не вимагає заливання граней (можна малювати тільки ребра) і дозвоює не витрачати час на рисування невидимих елементів. Метод застосовується при зображенні математичних поверхонь, рис. 7.14 *в*.

Аналітичний метод полягає у визначенні невидимих відрізків ліній розрахунковим шляхом. Метод заснований на аналітичній геометрії, потребує значних обсягів обчислень, має малу швидкодію та використовується у САПР для побудови видів на кресленнях за 3D-моделями.

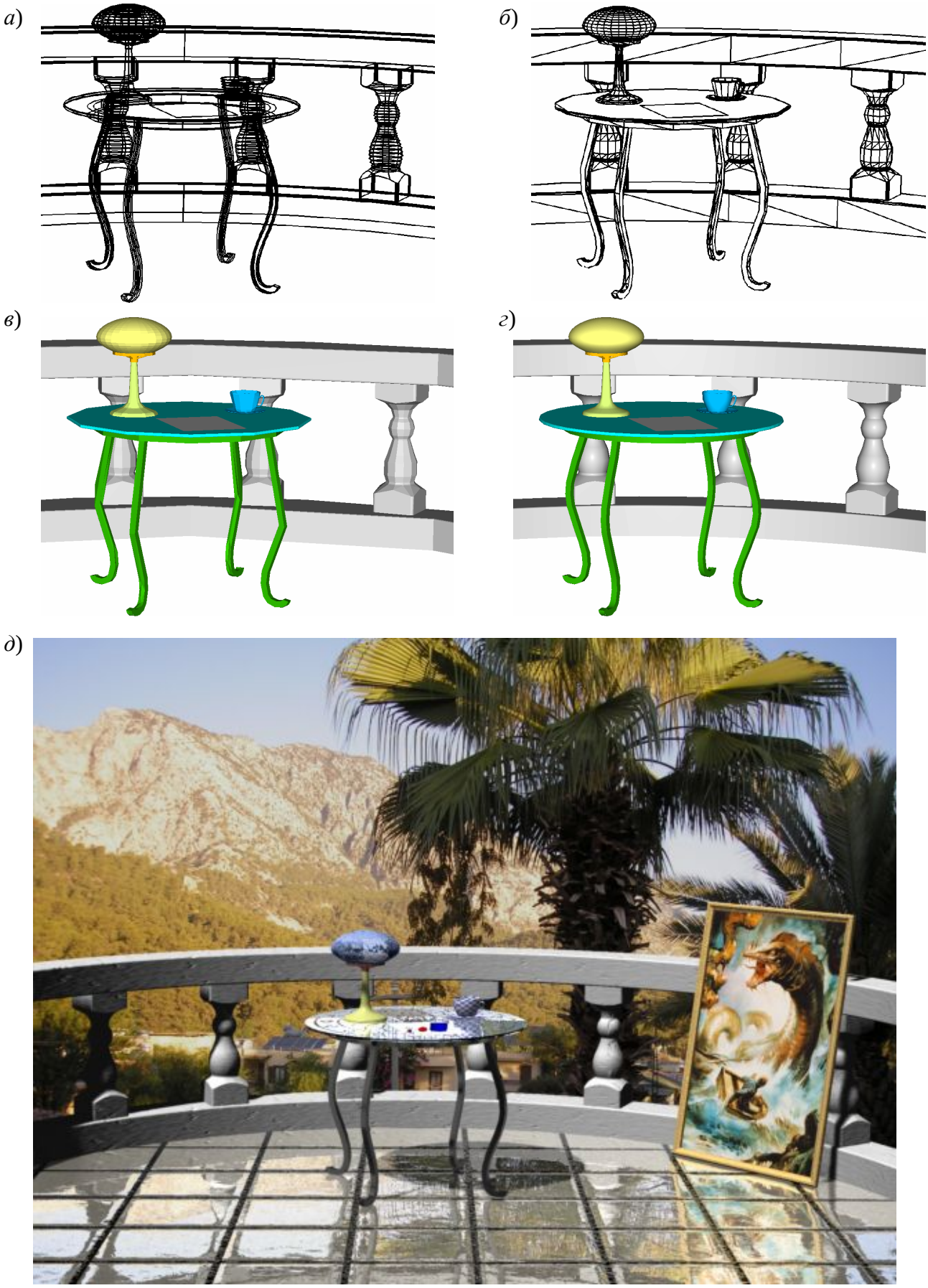


Рисунок 7.13 Рівні візуалізації 3D-моделі: каркасна (*а*); з видаленням невидимих точок (*б*); зафарбовування граней з урахуванням освітлення (*в*); імітація гладких поверхонь (*г*); накладення текстури, імітація тіней, відбиття і переломлення, трасування променів (*д*)

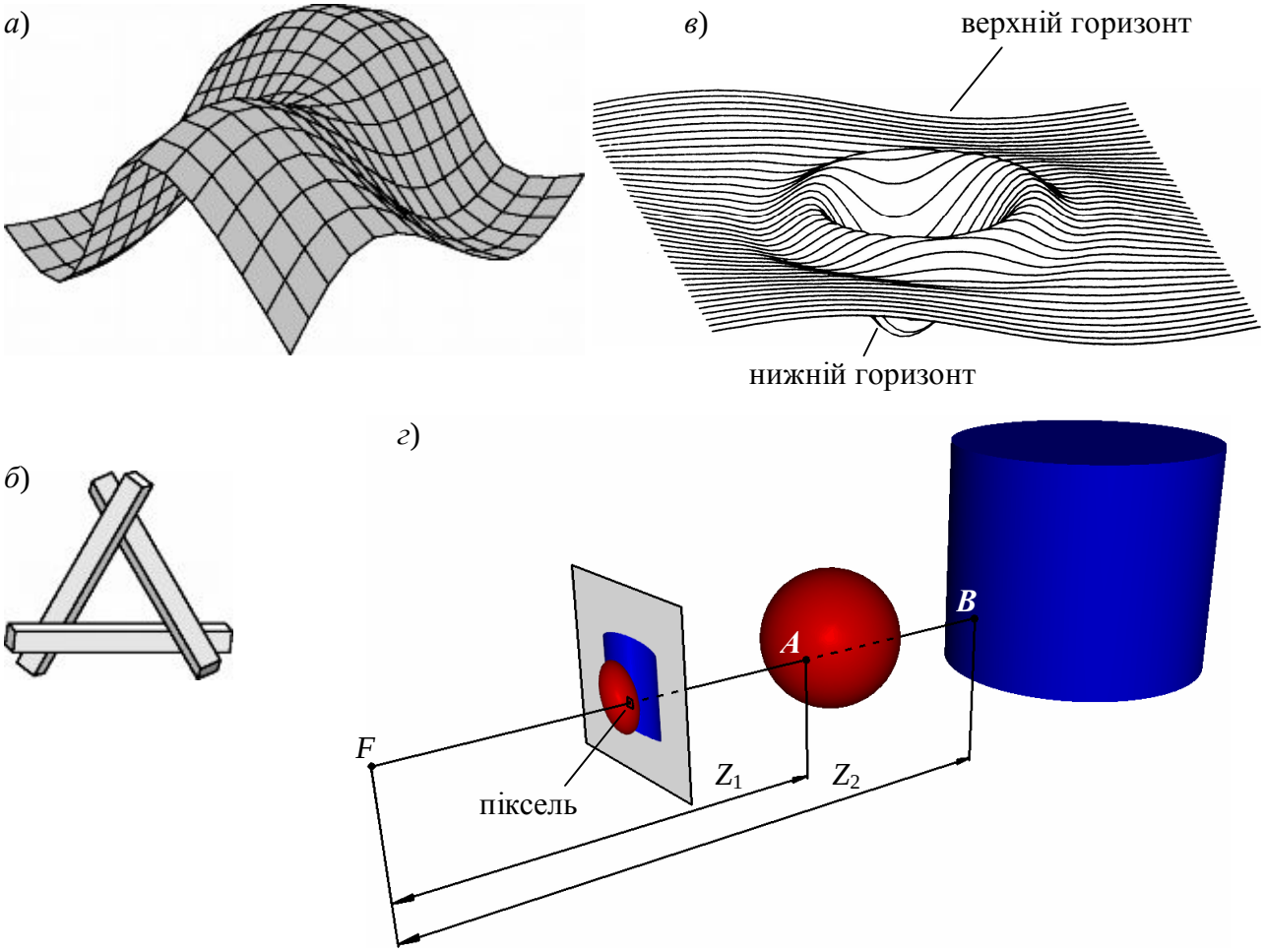


Рисунок 7.14 – Видалення невидимих елементів методом сортування граней за глибиною (*а*) і об'єкт, який неможливо відобразити цим методом (*б*); методи плаваючого горизонту (*в*) та Z-буфера (*г*)

Метод Z-буфера полягає у використанні спеціальної ділянки пам'яті, в котрій для кожного пікселя зображення записується значення координати *z* (у видових координатах) точки поверхні тривимірного об'єкта, що проектується в область цього пікселя.

Розглянемо приклад, зображений на рис. 7.14 *г*. Перед побудовою зображення *Z-буфер* ініціалізується — заповнюється максимальними значеннями координати *Z*. Проекційний промінь, що проходить через деякий піксель растра, перетинає об'єкти *сфера* і *циліндр*. Нехай першим зображується сфера. У *Z-буфер* записується координата Z_1 точки перетинання проекційного променя з поверхнею сфери (точка *A*). При побудові зображення циліндра у той же піксель буде проектуватися точка *B* з координатою Z_2, більшою, ніж Z_1. Отже, ця точка буде проігнорована.

Метод *Z-буфера* є основним в сучасній 3D-графіці, оскільки дозволяє правильно зображати поверхні будь-якої форми, що пересікаються. Метод підтримується апаратно сучасними графічними системами, що дозволяє реалізувати велику швидкість відпрацювання кадрів (понад 20 за секунду), та уможливлює інтерактивну графіку 3D-ігор та симуляторів.

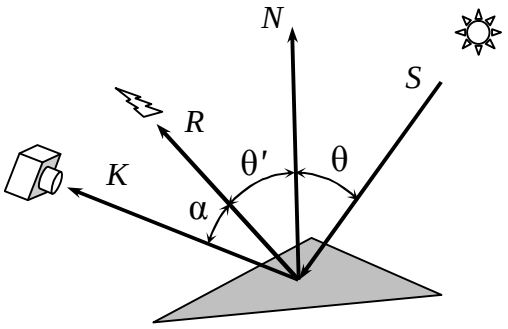


Рисунок 7.15 – Розташування векторів нормалі, джерела світла, віддзеркаленого променя та напрямку зору

3. ***Зафарбовування поверхонь*** припускає використання алгоритмів, що імітують відбиття світла реальними поверхнями тіл для створення більш-менш реалістичних зображень.

При обчисленні кольору точок поверхні використовується ***алгебра векторів***. Розташування поверхні характеризується ***вектором нормалі*** *N* до поверхні в даній точці, рис. 7.15. Напрямок променів падаючого і відбитого світла, а також напрямок погляду (променя проектування) також задається векторами *S*, *R* і *K* відповідно.

Поверхні реальних об’єктів мають низку характеристик, які впливають на характер відбиття ними світла: *світлота* і *колір* об’єкта, *шорсткість* поверхні, *прозорість* і ін. У комп’ютерній графіці для імітації цих властивостей використовуються кілька ***моделей відбиття світла***.

Дзеркальне відбиття світла характерно для *гладких поверхонь*. Кут θ між променем, що падає, і нормаллю дорівнює куту θ′ між нормаллю і відбитим променем. Промінь, що падає, відбитий промінь і нормаль розташовані в одній площині.

Поверхня вважається ***ідеально гладкою***, якщо розмір шорсткостей на ній набагато менше довжини світлової хвилі. У такому випадку уся світлова енергія що віддзеркалюється від поверхні, спрямована уздовж ***лінії відбитого променя***, будь-яке розсіювання в сторони від цієї лінії відсутнє, рис. 7.16 *а*.

Промінь світла, що попадає на поверхню ***неідеального дзеркала***, породжує не один, а безліч відбитих променів, розсіяних навколо напрямку ідеально відбитого променя, рис. 7.16 *б*. Інтенсивність променя, відбитого в деякому напрямку (наприклад, у напрямку камери) залежить від кута α, див. рис. 7.15. Ця залежність описується емпіричною ***моделлю Фонга***:

$$I_{\text{дз}} = I \cos^{p} \alpha , \qquad (7.23)$$

де I і $I_{\text{дз}}$ – інтенсивність падаючого і відбитого світла; p – показник, що залежить від якості полірування поверхні, рис. 7.17 *а*, *б*.

Дифузійне відбиття властиве матовим поверхням, що мають шорсткість більшу, ніж довжина світлової хвилі. Прикладами можуть служити папір, гіпс, пісок. Падаючий на таку поверхню промінь світла розсіюється рівномірно в усі сторони, рис. 7.16 *в*. Дифузійне відбиття описується ***законом Ламберта***,

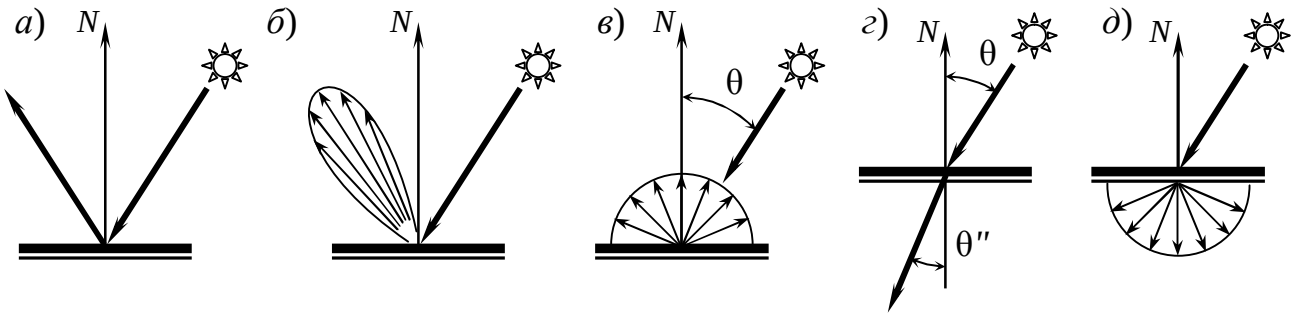


Рисунок 7.16 – Оптичні моделі: ідеальне дзеркало (*а*), неідеальне деркало Фонга (*б*); дифузне відбиття (*в*), ідеальне переломлення (*г*) та дифузне переломлення (*д*)

відповідно до якого інтенсивність відбитого світла пропорційна косинусу кута θ між променем світла, що падає, і нормаллю:

$$I_{\text{д}} = I\cos\theta. \tag{7.24}$$

Приклад об’єкта з дифузійним відбиттям приведений на рис. 7.17 *в*.

Дзеркальна і дифузійна моделі при деяких кутах α і θ дають нульові значення інтенсивності. Це значить, що ділянки об’єкта будуть абсолютно чорними. Оскільки в реальних сценах крім прямого освітлення звичайно присутнє *розсіяне світло*, що йде від неба або відбите від інших об’єктів, у модель вводиться імітація ***розсіяного освітлення*** з деякою інтенсивністю, що не залежить від напрямку джерела освітлення — усі точки поверхні об’єкта освітлюються рівномірно, рис. 7.17 *р*.

Для прозорих об’єктів використовуються моделі ***ідеального*** і ***дифузійного переломлення***, рис. 7.16 *г* і *д*. Характеристикою прозорої поверхні з ідеальним переломленням є ***коефіцієнт переломлення*** *n*, що визначає відхилення переломленого променя

$$\cos\theta'' = \frac{\cos\theta}{n}. \tag{7.25}$$

де θ" – кут переломленого променя.

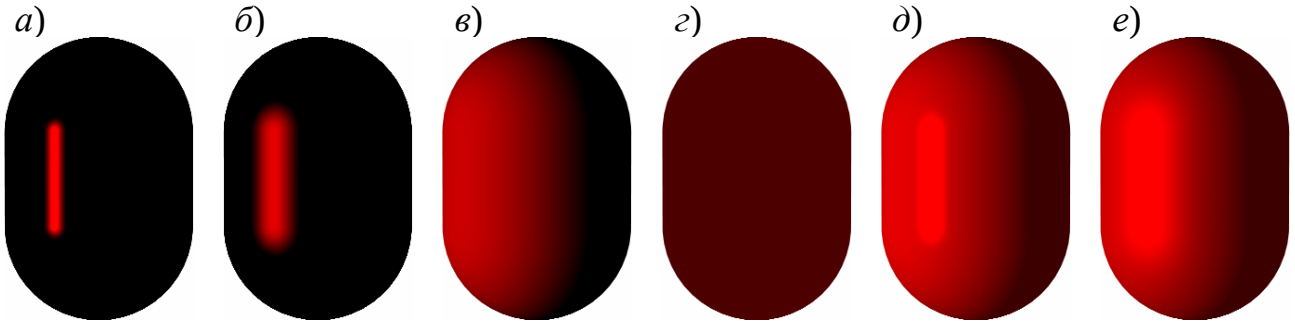


Рисунок 7.17 – Зафарбовування поверхонь з відбиттям за моделлю Фонга при різному степені шорсткості (*а*, *б*), за дифузійною моделлю (*в*); розсіяне освітлення (*г*); сполучення моделей відбиття Фонга, дифузійної та розсіяного освітлення з різними параметрами (*д*, *е*)

Параметри відбиття світла поверхнями об'єктів моделюються як сума інтенсивностей світла, розрахованих по різних моделях, з урахуванням вагових коефіцієнтів, що визначають ступінь впливу кожної моделі, рис. 7.17 *д*, *е*:

$$I_{\text{о}} = k_{\text{дз}} I_{\text{дз}} + k_{\text{д}} I_{\text{д}} + k_{\text{р}} I_{\text{р}} + k_{\text{п}} I_{\text{п}}, \qquad (7.26)$$

де $I_{\text{о}}$ – колір точки об'єкта; $I_{\text{р}}$ – інтенсивність розсіяного освітлення; $I_{\text{п}}$ – інтенсивність переломленого променя; $k_{\text{дз}}$, $k_{\text{д}}$, $k_{\text{р}}$, $k_{\text{п}}$ – коефіцієнти дзеркальності, шорсткості, розсіяного освітлення і прозорості об'єкта.

4. ***Імітація гладких поверхонь полігональних моделей***.

Оскільки полігональні моделі (що є найбільш розповсюдженим типом моделей 3D-поверхонь) являють собою *набір плоских граней*, найпростіший метод їхнього зафарбовування наступний. Для кожної грані визначаємо напрямок *вектора нормалі*, рис. 7.18 *а*, і відповідно до моделі оптичних властивостей матеріалу поверхні і напрямку променя освітлення обчислюємо *колір* цієї грані. Потім за допомогою простого алгоритму зафарбовування полігона (див п. 4.4) заповнюємо контур грані цим кольором.

Отримане таким методом зображення виходить «*гранованим*», кожну з граней полігональної сітки добре видно, рис. 7.13 *в* і 7.18 *д*. Тому метод використовується лише на етапі розробки моделі для її візуалізації в режимі реального часу — він дає найбільшу швидкість генерації кадрів зображення.

Для створення більш реалістичних зображень необхідно «згладити» нерівності полігональної апроксимації. Збільшення кількості граней при апроксимації криволінійних об'єктів дає ефект тільки в тому випадку, якщо видимий розмір полігонів наближається до розміру пікселя. При цьому модель буде занадто складною.

Існують методи, що дозволяють «згладити» границі між гранями, створивши ілюзію гладкої поверхні: методи *Гуро* і *Фонга*.

Метод Гуро передбачає наступну послідовність операцій:

— обчислюються нормалі до граней, рис. 7.18 *а*;

— обчислюються «нормалі» у вершинах полігональної сітки як усереднення векторів нормалей граней, що примикають до цієї вершини, рис. *7.18* б;

— спираючись на «нормалі» вершин, з урахуванням напрямку падаючого світла й оптичних властивостей матеріалу поверхні для кожної вершини обчислюється колір;

— полігон грані зафарбовується кольорами методом ***лінійної інтерполяції кольорів*** його вершин, рис. 7.18 *в*.

У результаті грань заповнюється рівномірним колірним переходом і здається опуклою. Колірний перехід уздовж ребра грані визначається тільки кольорами двох вершин, які з'єднує це ребро, і не залежить від кольорів інших вершин. Тому для двох сусідніх граней кольори уздовж загального ребра

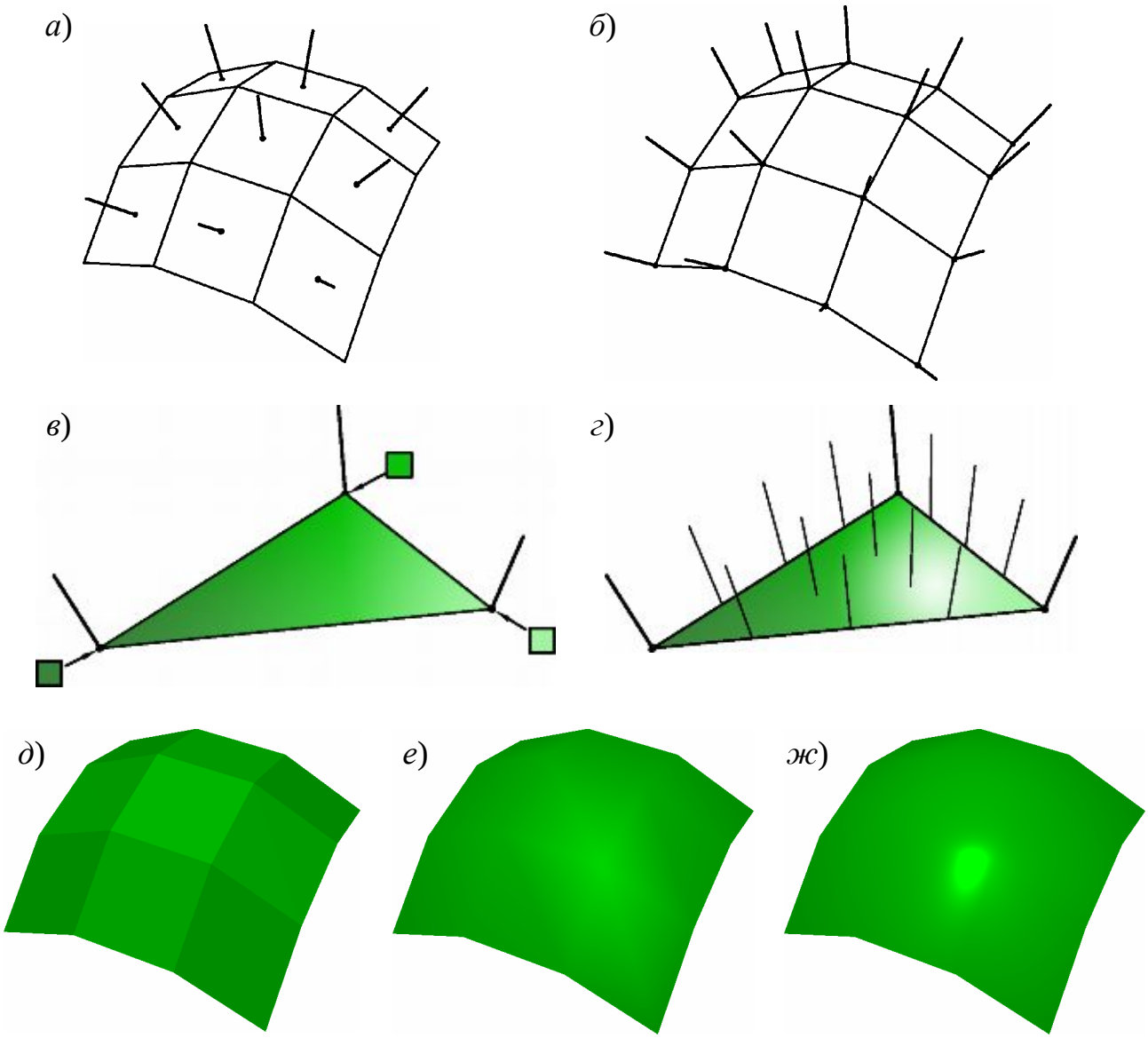


Рисунок 7.18 – Зафарбовування полігональних моделей: нормалі до граней (*а*) та усереднені нормалі в вершинах (*б*); зафарбовування полігона методом Гуро (*в*) та методом Фонга (*г*); поверхня, зафарбована за нормалями до граней (*д*), за методами Гуро (*е*) та Фонга (*ж*)

будуть однаковими — границя між гранями стає невидимою. Створюється ілюзія гладкої криволінійної поверхні, рис. 7.13 *г* та 7.18 *е*.

Зафарбовування за методом Гуро не потребує великої кількості обчислень і виконується за допомогою простого алгоритму заповнення полігона, у який додана процедура лінійної інтерполяції кольору. Тому метод дуже швидкодіючий і використовується при створенні зображень, що рухаються, у режимі реального часу. Метод Гуро підтримується на апаратному рівні сучасними графічними системами.

Разом з тим, оскільки моделювання процесу відбиття світла виконується тільки для деяких точок поверхні (вершин), метод Гуро дає достатньо реалістичне зображення тільки для матових поверхонь і не дозволяє одержати відблиски або дзеркальне відбиття.

Для створення більш реалістичних зображень поверхонь, у т.ч. блискучих, дзеркальних, прозорих, використовується ***метод Фонга***. За методом Фонга також обчислюються нормалі до граней і «нормалі» у вершинах як усереднення нормалей сусідніх граней. Але обчислення кольору (відповідно до моделі відбиття, інтенсивності і спрямованості висвітлення) виконується *окремо для кожного пікселя* зображення грані. При цьому виконується ***лінійна інтерполяція векторів нормалей*** — для точки поверхні грані, що проектується у даний піксель зображення, розраховується деякий вектор «нормалі» пропорційно відстані до векторів «нормалей» у вершинах. Тому грань зафарбовується так само, якби вона була справді опуклою або увігнутою, рис. 7.18 *г*.

Метод Фонга потребує значно більшої кількості обчислень і застосовується при остаточній візуалізації об'єктів, дозволяючи створити якісне реалістичне зображення, рис. 7.18 *ж* (див. також рис. 7.13 *д*).

Слід зазначити, що ефект «згладжування» граней як методом Гуро, так і Фонга проявляється тільки усередині контуру поверхні. Крайні, контурні грані об'єкта в будь-якому випадку будуть помітні — на рис. 7.18 *д*, *е* и *ж* контур поверхні складається з ламаних ліній. Єдиний спосіб зробити контур об'єкта більш плавним — збільшити кількість полігонів при апроксимації поверхні.

7.2 Імітація дрібних деталей і мікрорельєфу

У деяких областях застосування комп'ютерної графіки (наприклад, у САПР) цілком достатньо візуалізувати 3D-моделі поверхонь зафарбовуванням заданим кольором за методом Гуро, оскільки необхідно лише мати представлення про форму і зовнішній вигляд об'єкта проектування. Але в таких областях, як синтез зображень художнього характеру, створення спецефектів у кінофільмах, комп'ютерних мультфільмів і рекламних роликів, треба створювати зображення, максимально близькі до реальних. У цьому випадку необхідно вирішити задачі *трасування світлових променів* і *імітації дрібних деталей і мікрорельєфу*.

Реальні об'єкти рідко мають ідеально гладку одноколірну поверхню. Виключення складають лише об'єкти штучного походження (наприклад, кузов автомобіля, біла тарілка і т.п.). Більшість же об'єктів як природного, так і штучного походження (стовбур дерева, гора, шкіра людини, стіна будинку, дорога) мають поверхні надзвичайно складної будови через безліч *дрібних деталей* — нерівностей, областей різного кольору і т.п. Якщо спробувати передати всі дрібні деталі за допомогою полігональної моделі, така модель виявиться занадто складною. Однак дрібні деталі можна імітувати за допомогою *текстур* і *карт мікрорельєфу*.

Текстурою називається *растрове зображення*, що використовується для заповнення поверхні тривимірного об'єкта. У 3D-графіці звичайно використовуються ***проективні текстури*** — растри, що накладаються (проектуються) на поверхню об'єкта. Елемент растра текстури називається ***текселом***.

Текстура накладається на поверхню в такий спосіб. *Розташування растра текстури* задається кутами повороту її системи координат ($x_т$, $y_т$, $z_т$,) щодо системи координат об'єкта (x, y, z,), рис. 7.19 *а*. Також указується розмір растра і його зсув. У більшості програмних пакетів для зручності накладення текстури її розмір і зсув вказуються в частках габариту об'єкта. Якщо растр текстури менше розміру об'єкта, можливе *копіювання* текстури по всій його поверхні. Так, на рис. 7.13 *д* малюнок підлоги заданий растром для однієї плитки — інші отримані копіюванням текстури.

Кожен *тексел* трактується як прямокутник, зафарбований деяким кольором (прямокутник 1 2 3 4 на рис. 7.19 *а*). Накладення текстури виконується проектуванням областей текселів на поверхню. У результаті на поверхні утвориться деяка фігура $1'\,2'\,3'\,4'$. При проектуванні цієї ділянки поверхні на площину екрана йому привласнюється колір даного тексела.

Як видно з рис. 7.19 *а*, при великих кутах нахилу поверхні щодо площини текстури форма проекцій текселів сильно спотворюється. Існує два способи текстурування поверхонь з великими кутами охоплення. Перший — накладення

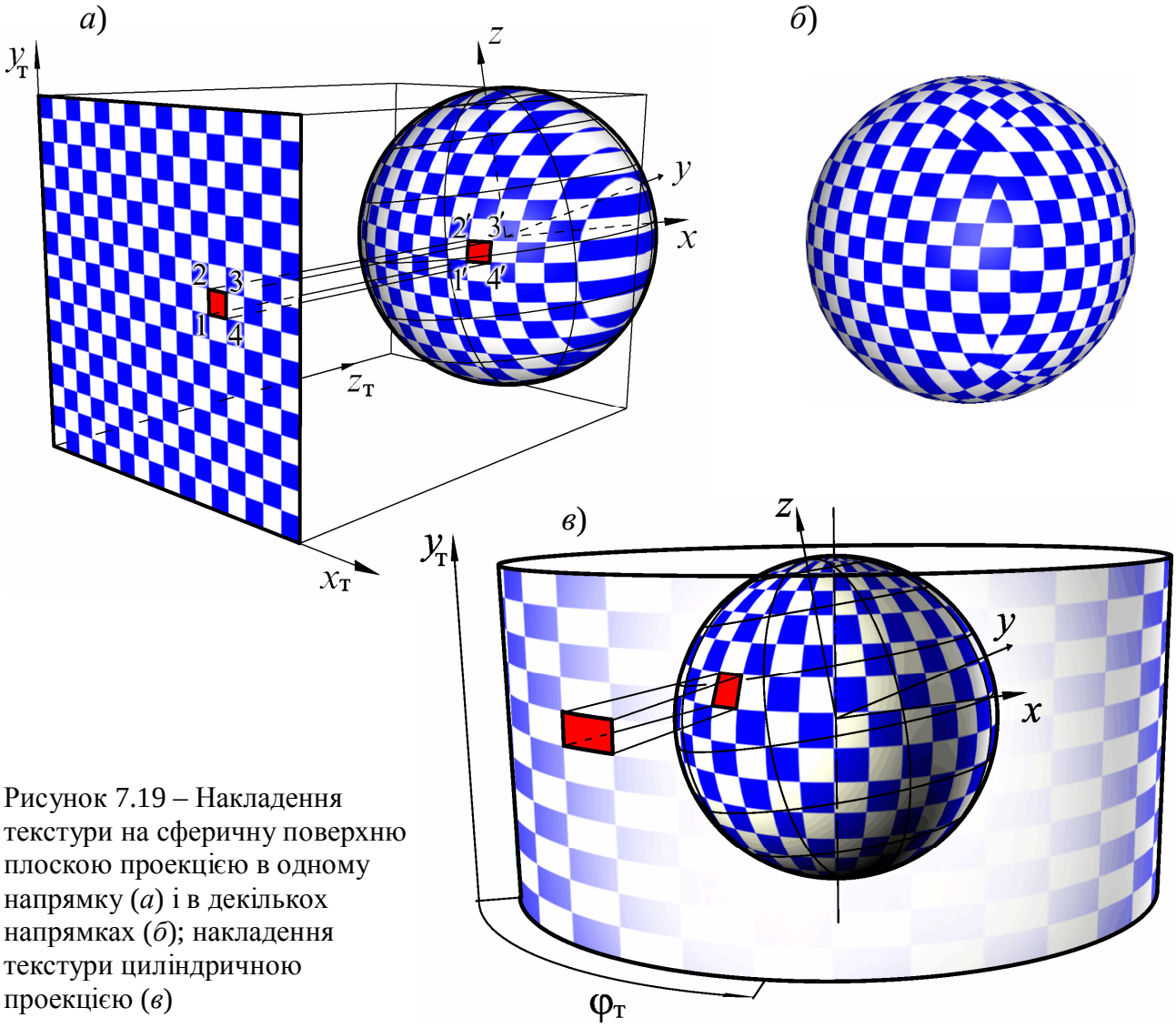


Рисунок 7.19 – Накладення текстури на сферичну поверхню плоскою проекцією в одному напрямку (*а*) і в декількох напрямках (*б*); накладення текстури циліндричною проекцією (*в*)

декількох однакових або різних текстур, кожна з яких розташована під невеликим кутом до тієї частини поверхні об'єкта, на яку проектується. На рис. 7.19 *б* різні ділянки сфери заповнені шістьма однаковими текстурами, розташованими у взаємно перпендикулярних площинах. Видно, що в місцях стикування текстур є дефекти.

Другий спосіб — накладення текстури з використанням *циліндричної* або *сферичної* проекцій. При ***циліндричній проекції*** текстури одна з координат растра трактується як кут повороту $\varphi_т$, рис. 7.19 *в*. Проектування робиться променями, що перетинають вісь циліндра і перпендикулярні цій осі. ***Сферична проекція*** відрізняється тим, що обидві координати растра розглядаються як кутові, а промені, що проектують, проходять через центр сфери. Циліндрична і сферична проекції дозволяють зафарбувати всю поверхню об'єкта.

При недостатньо великій роздільній здатності растра, а також при розгляді всього об'єкта або деякої його частини з занадто близької відстані стають помітні окремі тексели текстури, рис. 7.20 *а*. Для поліпшення виду текстурованої поверхні використовуються методи ***фільтрації*** (згладжування) растрів текстур, наприклад *білінійну фільтрацію*, рис. 7.20 *б*. Зображення стає злегка розмитим, але окремі тексели не помітні.

Застосування текстури для зафарбовування поверхні не виключає завдання для неї інших оптичних властивостей — дзеркального відбиття, прозорості й ін. При цьому текстура звичайно застосовується з дифузійною моделлю відображення; колірні ефекти від текстури й інших оптичних моделей складаються згідно (7.26).

Карти текстур застосовуються не тільки для фарбування поверхні, але і для моделювання інших її властивостей, наприклад ***мікрорельєфу*** — дрібних нерівностей на поверхні. Моделювання таких нерівностей за допомогою полігональної сітки вимагає значного ускладнення моделі. Використання *карт мікрорельєфу* дозволяє вирішити цю проблему з набагато меншими витратами часу.

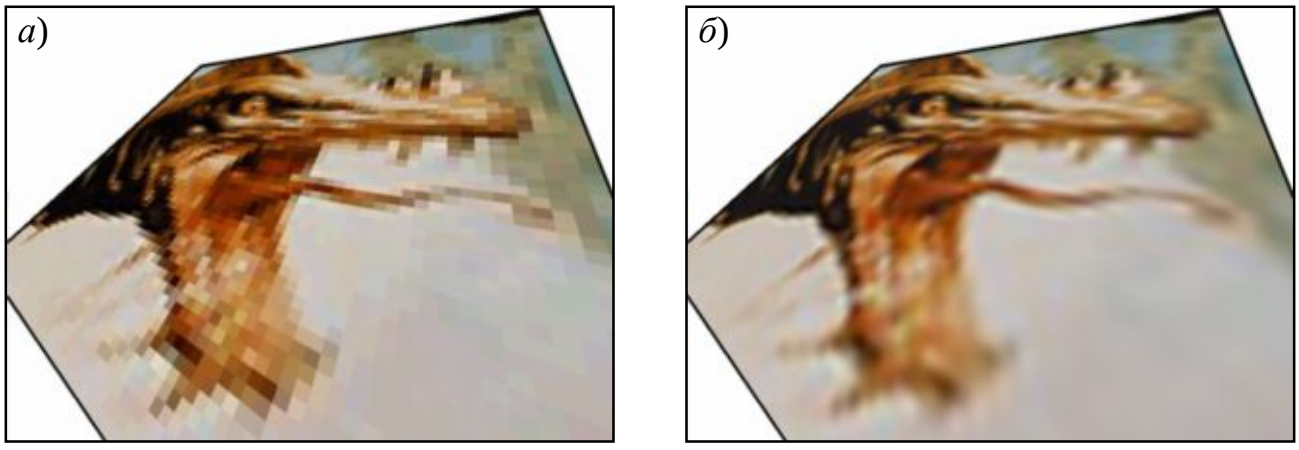


Рисунок 7.20 – Накладення текстури без фільтрації (*а*) та з білінійною фільтрацією (*б*)

Карта мікрорельєфу (*bump map*) — це растр, що дозволяє імітувати мікрорельєф. Як карти мікрорельєфу використовуються напівтонові або повнокольорові зображення (в останньому випадку використовується тільки інтенсивність, а колір ігнорується).

Мікрорельєф імітується в такий спосіб. Інтенсивність текселів растра мікрорельєфу трактується як *відхилення поверхні* мікрорельєфу в даній точці від поверхні грані, рис. 7.21. Для кожної точки поверхні обчислюється вектор «нормалі», відхилений від дійсної нормалі грані відповідно до карти мікрорельєфу. Подальший розрахунок параметрів відбиття і переломлення світла даною ділянкою поверхні виконується з використанням відхиленого вектора — у результаті створюється ілюзія «нахилу» ділянки поверхні. На поверхні з'являються «виступи» і «западини» — мікрорельєф, рис. 7.22 *а*.

Використання мікрорельєфу дозволяє створювати ефекти нерівної дзеркальної або прозорої поверхні, наприклад, поверхня плитки підлоги й абажура лампи на рис. 7.13 *д*, хвиль на рис. 7.22 *б*. Треба зауважити, що це не реальний рельєф, а лише його ілюзія, створена колірним малюнком на грані — у дійсності грань залишається плоскою.

Текстурні карти можуть використовуватися також для завдання інших властивостей поверхні: прозорості, дзеркальності, шорсткості й ін. Наприклад, на рис. 7.23 карта текстури задає прозорість поверхні.

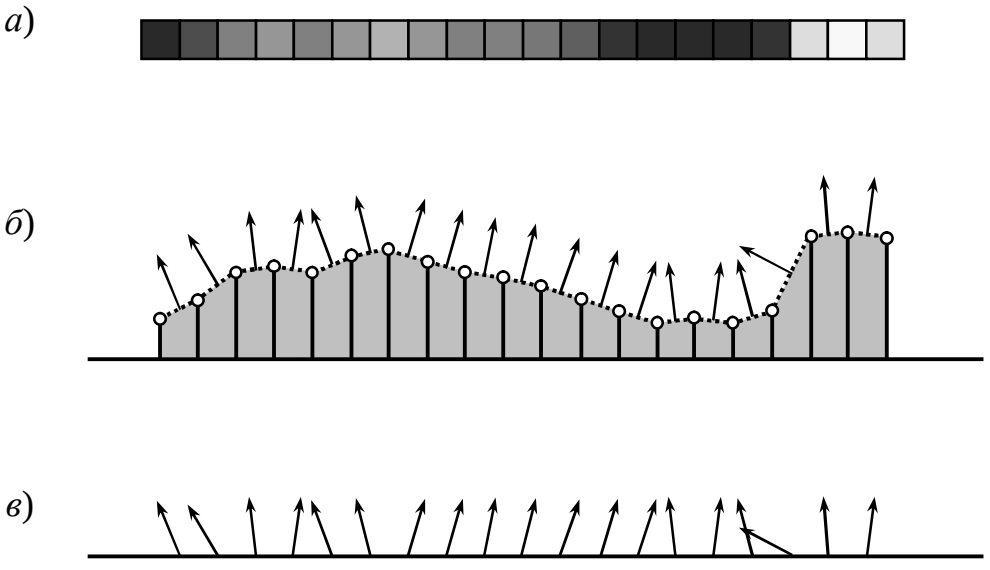
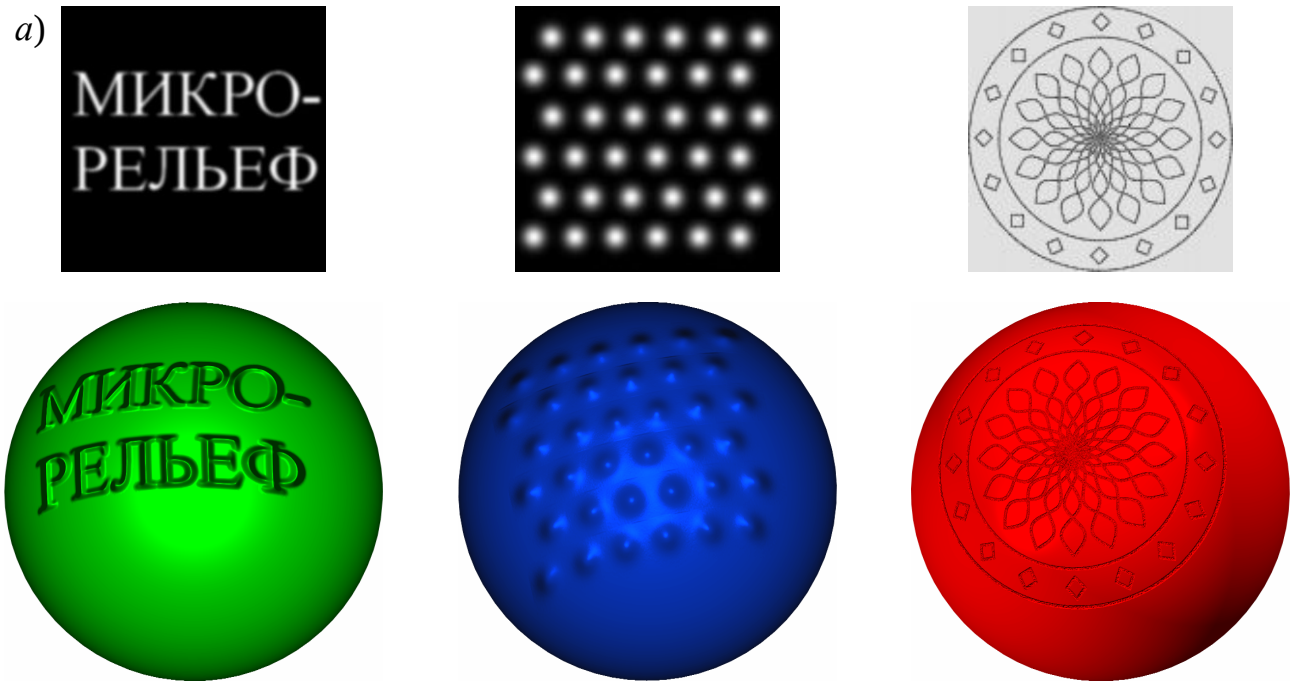


Рисунок 7.21 – Імітація мікрорельєфу: рядок растра (*а*), інтерпретація інтенсивностей текселів як висот мікрорельєфу (*б*), відхилення векторів нормалей на поверхні (*в*)

а)



б)

Рисунок 7.22– Накладення карт рельєфу на кулю (*а*) і на плоску дзеркальну поверхню (*б*)

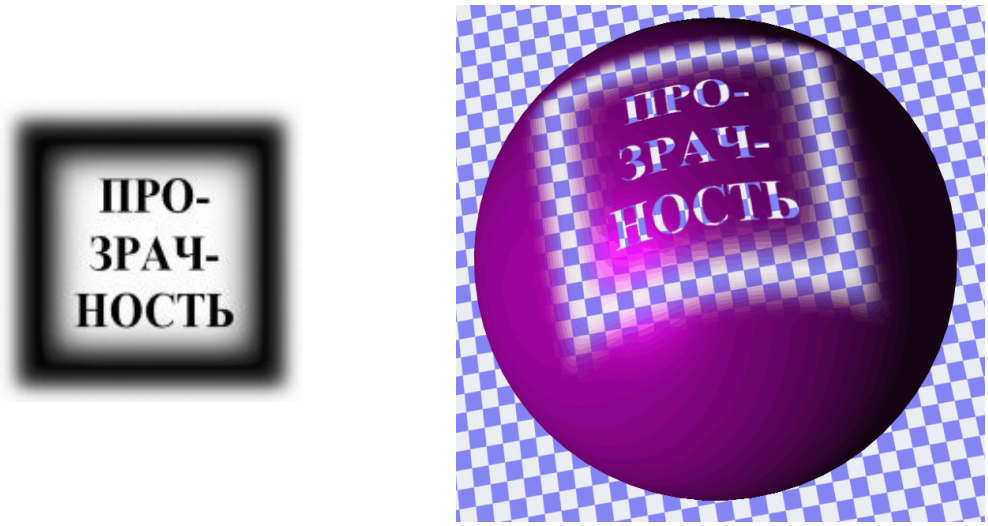


Рисунок 7.23 – Карта прозорості

7.3 Освітлення і тіні

Важливу роль у створенні реалістичних зображень грає імітація реальних умов ***освітлення*** світловідбиваючих об'єктів. При створенні спрощених зображень у процесі розробки моделі звичайно використовують джерело світла, спрямоване на об'єкт від глядача (*фронтальне освітлення*), рис. 7.24 *а*, що дозволяє підсвітити всі видимі елементи об'єкта.

При побудові більш складних сцен використовують різні ***моделі джерел світла***. Одержали поширення такі моделі:

1. ***Віддалене джерело***, рис. 7.24 *б*, імітує джерело світла, розташоване настільки далеко, що промені, які йдуть від його, взаємно паралельні. Таку властивість має світло, що йде від небесних світил — Сонця, Місяця. Інтенсивність світла з відстанню не міняється. Джерело задається *вектором*, що визначає напрямок променів світла. У деяких програмних пакетах передбачена спеціальна модель *сонячного освітлення*, яка є різновидом віддаленого джерела і дозволяє обчислити напрямок вектора освітлення в залежності від часу доби, широти і довготи місцевості.

2. ***Точкове джерело***, рис. 7.24 *в*, звичайно імітує штучні джерела світла (електролампу, свічу) і задається розташуванням у просторі однієї *точки*. Світлові промені виходять з цієї точки у всіх напрямках і освітлюють навколишній простір. З віддаленням від точкового джерела *інтенсивність світла зменшується*. Для моделювання цього ефекту використовуються зворотна лінійна або зворотна квадратична залежність:

$$I = \frac{I_0}{L}; \qquad I = \frac{I_0}{L^2}, \tag{7.27}$$

де I_0 – інтенсивність джерела; L – відстань від джерела до об'єкта.

Зворотна квадратична залежність відповідає дійсному оптичному закону зменшення інтенсивності світла, але часто не дає потрібного візуального ефекту, оскільки здатність графічних пристроїв передавати різні рівні яскравості значно уступає людському зору.

3. ***Спрямоване джерело***, рис. 7.24 *г*, є різновидом точкового, але його світловий потік обмежений нескінченним конусом з вершиною в точці розташування джерела. Спрямоване джерело задається в просторі положенням його *точки*, а також *вектором*, що визначає напрямок осі конуса і *кутом* конуса. Для моделювання зменшення інтенсивності використовуються закони (7.27).

Окрім розташування і напрямку, джерела світла характеризуються ***інтенсивністю*** і ***кольором*** — світло може бути *ахроматичним* (білим) і *хроматичним* (пофарбованим у деякий колір). В останньому випадку зображення поверхні не буде відповідати за кольором характеристиці самого об'єкта — об'єкт буде офарблюватися.

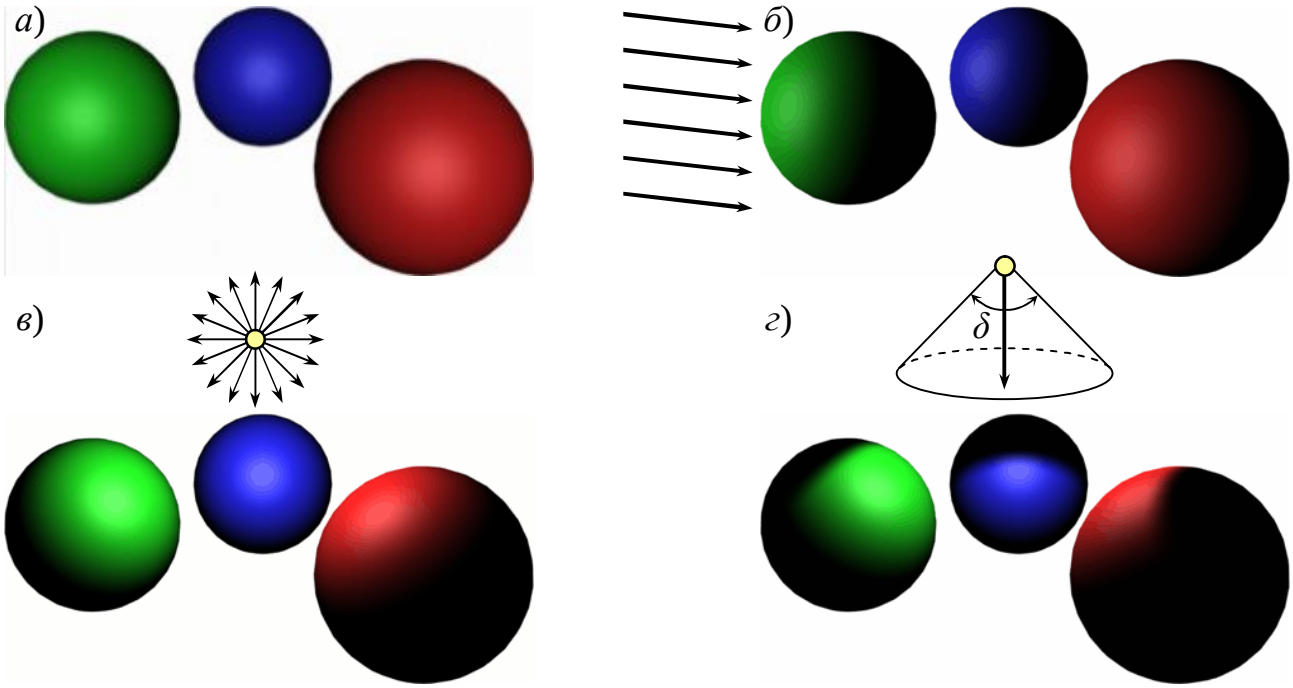


Рисунок 7.24 – Моделі освітлення об’єктів: фронтальне (*а*), віддалене джерело (*б*); точкове джерело (*в*) та спрямоване джерело (*г*)

Реальні об’єкти при освітленні їх деяким джерелом світла відкидають ***тінь*** — екранують інші об’єкти, розташовані далі від джерела, від улучання на них прямих світлових променів. Імітація явища *тіні* — дуже складна задача, рішення якої є необхідним при створенні дійсно реалістичних зображень.

Тіні, що відкидаються об’єктами, утворюють на поверхні інших об’єктів дуже складні фігури, які визначаються формою обох поверхонь і їхнім взаємним розташуванням, рис. 7.25. Задачу знаходження тіней можна розглядати як варіант задачі знаходження невидимих частин об’єктів, якщо розглядати їх із точки розташування джерела світла. Одержали поширення три методи моделювання тіней.

Перший метод припускає створення до візуалізації сцени ***карт тіней*** — двомірних масивів, що задають відстань від джерела світла до найближчої до нього поверхні. Цей метод трохи нагадує метод Z-буфера, але при проектуванні не на плоску, а на сферичну поверхню з центром у точці розташування джерела світла. Область простору, через яку проходить світловий потік, розбивається на елементи у виді нескінченних конусів із квадратним поперечним перерізом, кожний з який розглядається як окремий «промінь», рис. 7.25 *а*. Весь світловий потік являє собою сукупність таких «променів» — своєрідний растр. Для кожного «променя» методом перебору всіх об’єктів сцени визначаються ті об’єкти, які він перетинає. З цих об’єктів промінь висвітлює тільки той, що розташований ближче усього до джерела світла. Номер цього об’єкта записується в масив. При відрисовуванні інших об’єктів, розташованих уздовж даного променя далі від джерела, цей промінь не враховується.

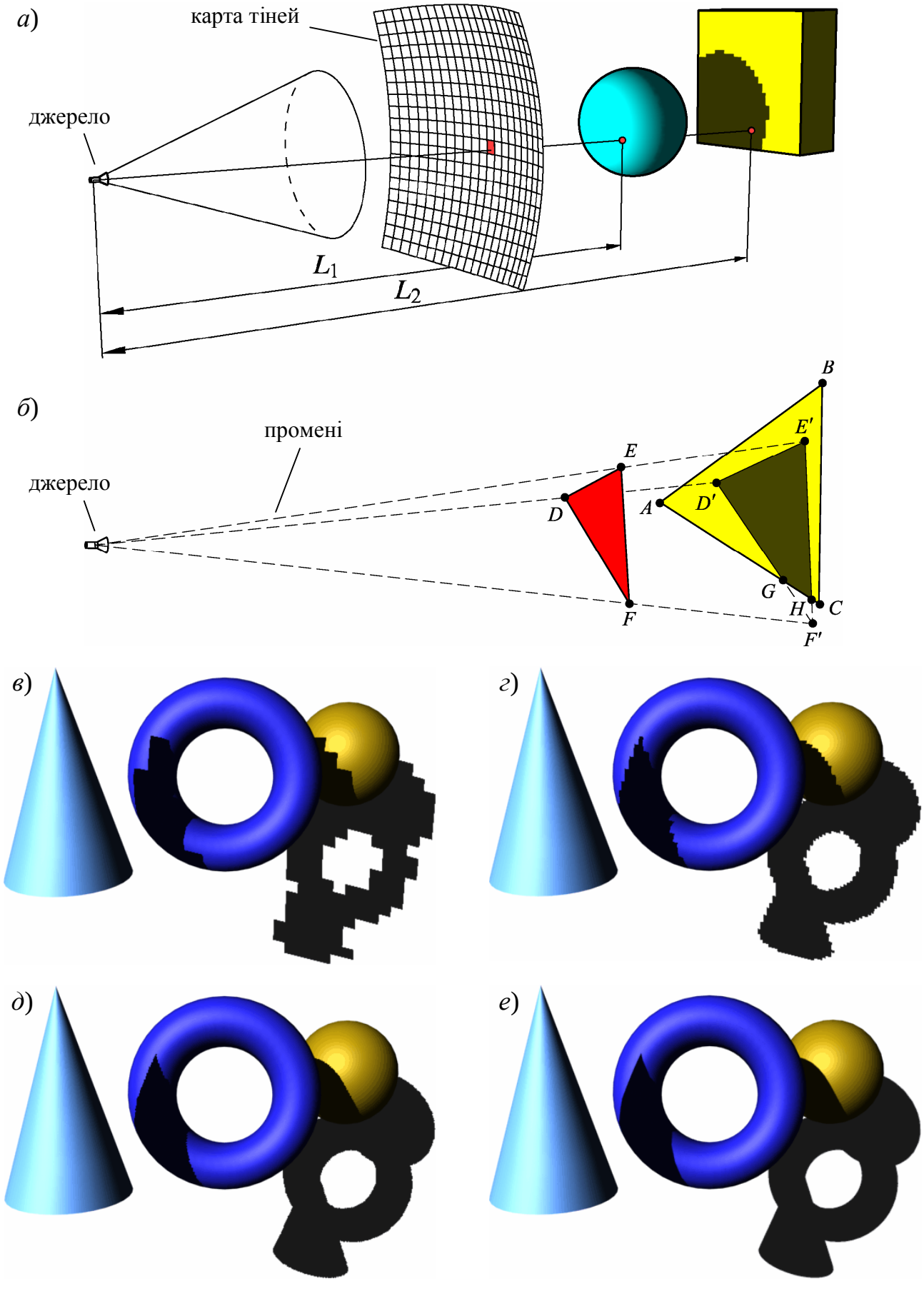


Рисунок 7.25 – Створення тіней за допомогою карти тіней (*а*) та методом проектування (*б*); тіни, створені при розмірі карти 32×32 (*в*), 128×128 (*г*), 1024×1024 (*д*) і трасуванням променів (*е*)

Для кожного джерела світла створюється своя карта тіней, тому метод потребує значних обсягів пам’яті і витрат процесорного часу на створення карт (особливо для складних сцен). Тіні на рис. 7.25 *в* створені за допомогою карти розміром 32×32 — при цьому чітко видна дискретна структура карти — кожна комірка дає тінь прямокутної форми. На рис. 7.25 *д* використана карта розміром 1024×1024 комірки.

Другий метод — визначення тіней ***розрахунковим методом*** безпосередньо в процесі візуалізації. Для кожного полігона поверхні виконується аналіз освітленості шляхом перебору всіх інших об’єктів сцени. Якщо деякий інший полігон знаходиться на шляху світлового променя від джерела до даного полігона, на останньому створюється тінь — проекція полігона, що екранує, на поверхню зображуваного. На рис. 7.25 *б* зображені два полігони. Полігон *DEF* розташований ближче до джерела світла, чим полігон *ABC* і проектується на площину останнього променями, що виходять із джерела світла, у виді фігури *D'E'F'*. Оскільки не уся фігура *D'E'F'* розташована усередині полігона *ABC*, необхідно знайти координати точок перетинання ребер фігури *D'E'F'* з ребрами полігона *ABC* (точки *G* і *H*). Одержуємо фігуру *D'E'HG*, що оконтурює тінь на грані *ABC*. Метод дозволяє одержати більш якісний, «гладкий» малюнок тіні, але вимагає виконання складних розрахунків і в деяких випадках може привести до помилок в обчисленні — неправильного рисування тіні.

Третій метод — побудова тіні ***трасуванням світлових променів*** — звичайно застосовується при побудові всього зображення методом *зворотного трасування* (див. нижче). Для кожної точки поверхні виконується розрахунок траси променя, що йде від джерела. Якщо промінь на своєму шляху перетинає інші поверхні, він вважається екранованим. Метод не потребує значних обсягів пам’яті, дозволяє одержати найбільш якісний малюнок тіні, рис. 7.25 *е*, дає можливість одержувати тіні від напівпрозорих об’єктів, але вимагає виконання великого обсягу обчислень.

7.4 Метод трасування променів

Зображення тривимірних сцен з відносно низьким рівнем реалістичності будуються методом *прямого розрахунку* — послідовно відрисовуються всі об'єкти та їхні елементи. При цьому можуть враховуватися освітленість об'єктів, їхнє взаємне перекриття і тіні. Однак такий метод не дозволяє моделювати складні оптичні ефекти, такі як *дзеркальне відбиття* одних об'єктів в інших, *переломлення* світлових променів при проходженні через прозорі об'єкти і т.п. Тому для створення якісних реалістичних зображень використовуються методи *трасування променів*.

Трасування променів — це метод одержання зображення, що полягає в *моделюванні поширення променів світла*, їхнього відбиття від об'єктів і переломлення в них.

Основним положенням трасування променів є те, що промінь світла у вільному просторі поширюється *по прямій* доти, доки не зустріне перешкоду — поверхню деякого об'єкта. Кожне джерело світла випускає незліченну безліч променів. Такі промені називають *первинними*. Частина з них іде в простір, частина — попадає на поверхні об'єктів. У залежності від своїх оптичних властивостей ці поверхні відбивають або переломлюють світло, породжуючи незліченну безліч *вторинних променів*. Вторинні промені також або ідуть у простір, або попадають на поверхні інших об'єктів, знову відбиваючись і переломлюючись — породжуючи нові вторинні промені і т.д. Якась (звичайно дуже мала) частина променів попадає в зіницю ока чи об'єктив камери і проектується на сітківку ока, фотоплівку або чуттєву матрицю. Так створюються реальні зображення.

Відомі два методи моделювання поширення світла — *пряме* і *зворотне трасування*.

Пряме трасування променів припускає відстеження руху променя від джерела світла до поверхонь об'єктів і далі — на площину екрана. Такий метод дозволив би найточніше відтворити реальні оптичні явища, але його реалізація на ЕОМ представляє серйозні утруднення — необхідно відслідковувати всі первинні і вторинні промені (але і тих, і інших *нескінченно багато*). Крім того, у створенні зображення беруть участь лише промені, що перетинають площину екрана і проходять через точку фокуса — знайти такі промені в загальній їхній масі дуже складно. Усі інші промені будуть розраховані марно. Значно простіше реалізувати алгоритм *зворотного трасування променів*.

Зворотне трасування променів полягає у відстеженні світлових променів у зворотному напрямку — від екрана до об'єктів, і далі — до інших об'єктів і джерел світла. У такому випадку завжди трасуються тільки «потрібні» промені — такі, що проходять через точку фокуса і пересікають площину екрана. Звичайно трасують промені, що проходять через центр пікселів растра зображення, що генерується, рис. 7.26, — результатом трасування буде *колір* даного пікселя.

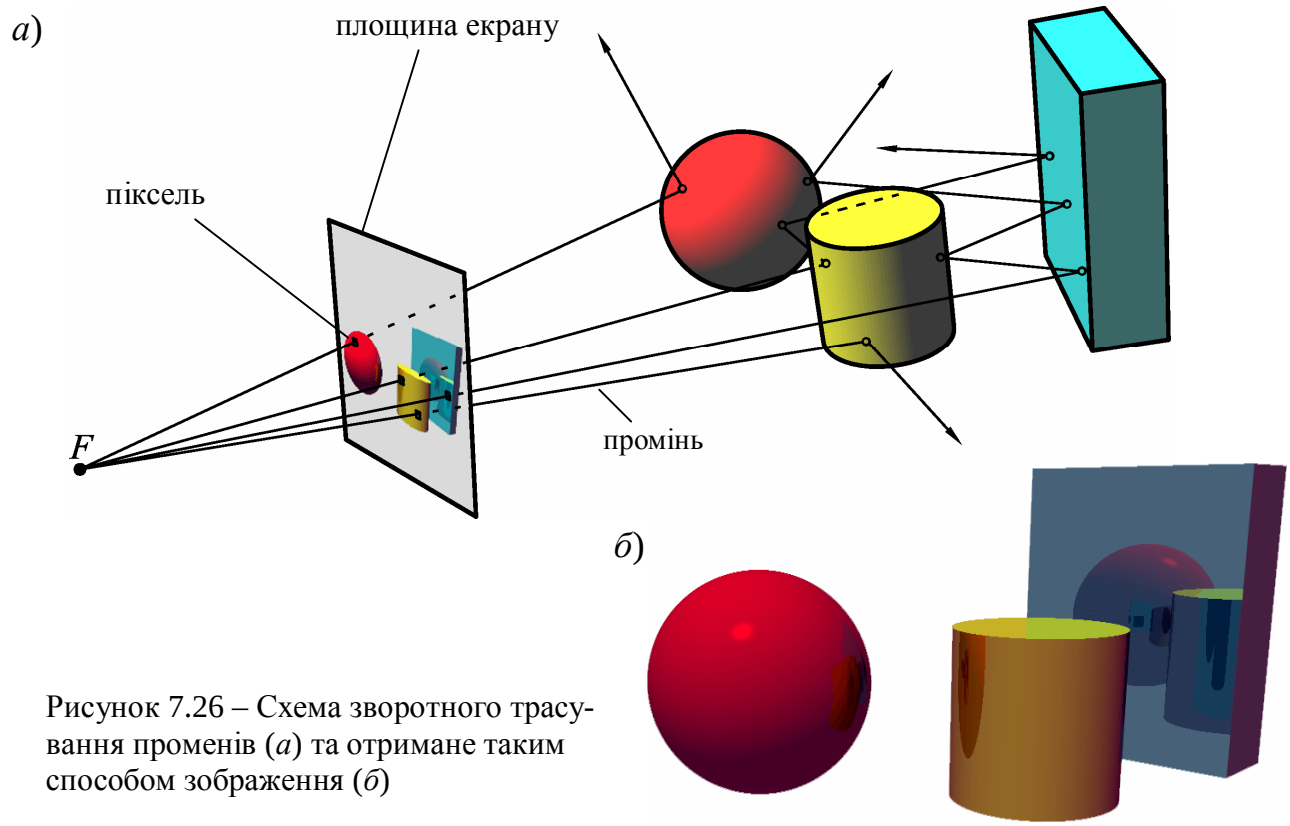


Рисунок 7.26 – Схема зворотного трасування променів (*а*) та отримане таким способом зображення (*б*)

Метод трасування вимагає пошуку *точок перетинання* променя з поверхнями об’єктів, який здійснюється переборними алгоритмами. Найближча зі знайдених точок перетинання використовується для подальших розрахунків.

Точка об’єкта, з яким перетинається промінь трасування, у свою чергу освітлюється *нескінченним числом* світлових променів, що йдуть від джерел світла й інших об’єктів. Тому для продовження трасування вводяться наступні обмеження:

1. Джерела світла можуть бути або *точковими*, або *нескінченно віддаленими* — вони не можуть мати розмір і форму. Для імітації розподілених джерел світла (світлових панелей, абажурів ламп, відкритого вогню) використовуються спеціальні методи.

2. Властивості поверхонь, що відбивають, описуються сумою п’яти компонентів — дифузійного відбиття, ідеального дзеркального відбиття, ідеального переломлення, дзеркального відбиття по моделі Фонга і освітлення розсіяним світлом, причому *трасування променів* виконується *тільки для моделей ідеального відбиття і переломлення*.

З точки перетинання вихідного променя з поверхнею трасується тільки один *ідеально відбитий* промінь, або два — відбитий і переломлений — для напівпрозорих об’єктів, рис. 7.27 (у дійсності цей промінь — падаючий, а відбивається вихідний для розрахунку, але відповідно до законів відображення і переломлення траєкторія світлового променя при зміні напрямку його руху не міняється, і його умовно можна назвати «відбитим»). Трасування цих променів

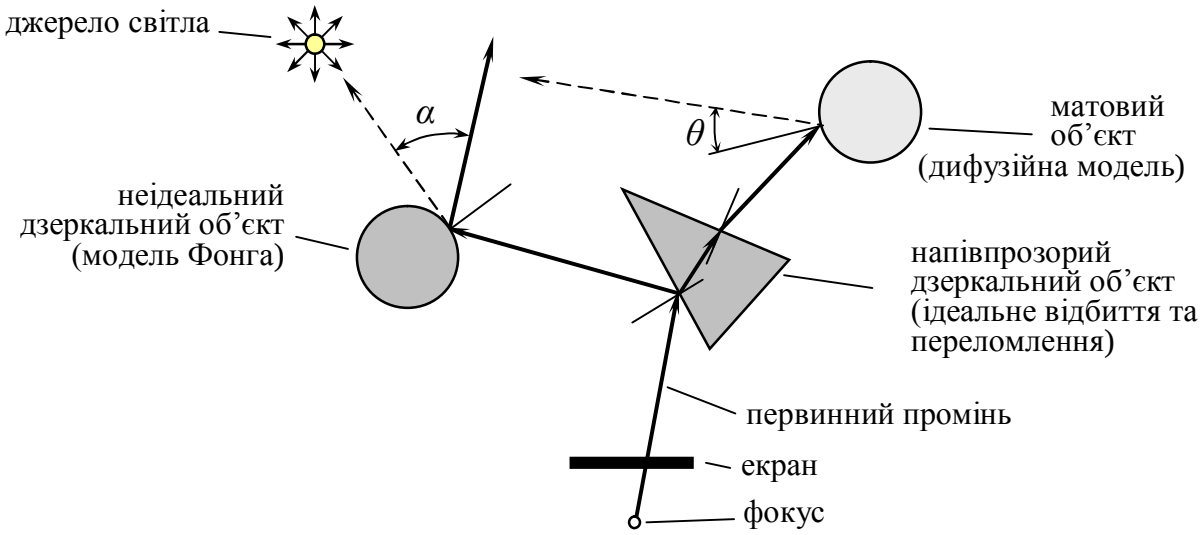
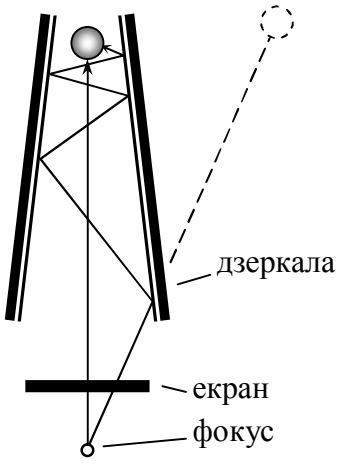


Рисунок 7.27 – Трасування променів для поверхонь з різними оптичними властивостями

дозволяє одержати результат ідеального відображення (або переломлення) світла від інших *об'єктів, що відбивають* (але не джерел світла).

Крім того, для розглянутої точки визначається напрямок на джерела світла і, відповідно до дифузійної моделі і моделі Фонга, обчислюються інтенсивності відбиття. Дифузійна модель дозволяє одержати рівномірну освітленість поверхні, а модель Фонга — *відблиски* від джерел світла. Отримані за всіма переліченими моделями яскравості і відтінки складаються і дають результат — колір і яскравість даної точки поверхні.

При трасуванні відбитого променя він може знову потрапити на дзеркальну поверхню, що викликає необхідність знову його трасувати. Можливі випадки, коли промінь буде відбиватися велике (і навіть нескінченно велике) число разів, рис. 7.28. Щоб не збільшувати надмірно час виконання трасування, звичайно вводять обмеження на глибину трасування — той самий промінь може відбиватися не більш *n* раз.



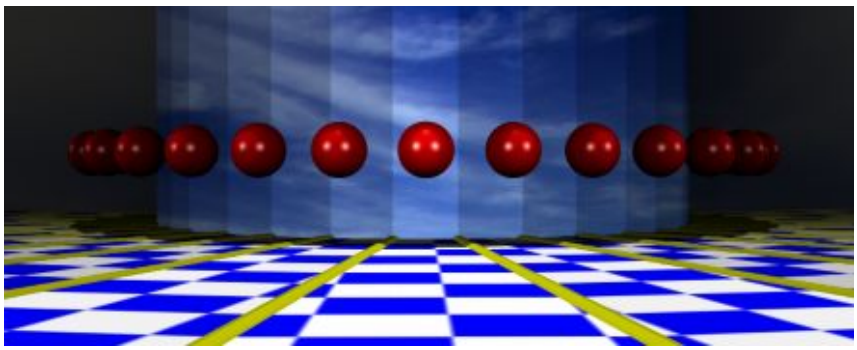


Рисунок 7.28 – Ефект багатократного відбиття

Розглянемо переваги і недоліки методу зворотного трасування світлових променів. Безумовною перевагою методу є створення *найбільш реалістичних зображень*. Жоден з відомих нині методів візуалізації 3D-сцен не дозволяє настільки ж точно врахувати закони оптики, відтворити складні оптичні ефекти, рис. 7.29 (див. також рис. 7.13 *д* і 7.29 *б*). Інша перевага методу — *автоматичне видалення невидимих елементів* — не потрібно використовувати ані Z-буфер, ані будь-який інший метод.

До недоліків методу слід віднести наступне:

1. Метод не дозволяє врахувати освітлення матових об’єктів і об’єктів – неідеальних дзеркал світлом, відбитим від інших об’єктів, оскільки при розрахунку освітленості по дифузійній моделі і моделі Фонга враховуються *тільки джерела світла*. Створення такого простого ефекту, як «сонячний зайчик» (висвітлення ділянки об’єкта світлом, відбитим від дзеркальної поверхні іншого об’єкта) за допомогою стандартних алгоритмів трасування променів неможливо. Такі ефекти імітують, уводячи додаткові джерела світла.

2. Метод трасування променів потребує виконання складних обчислень для кожного пікселя створюваного зображення, тому це — найповільніший із усіх методів візуалізації.

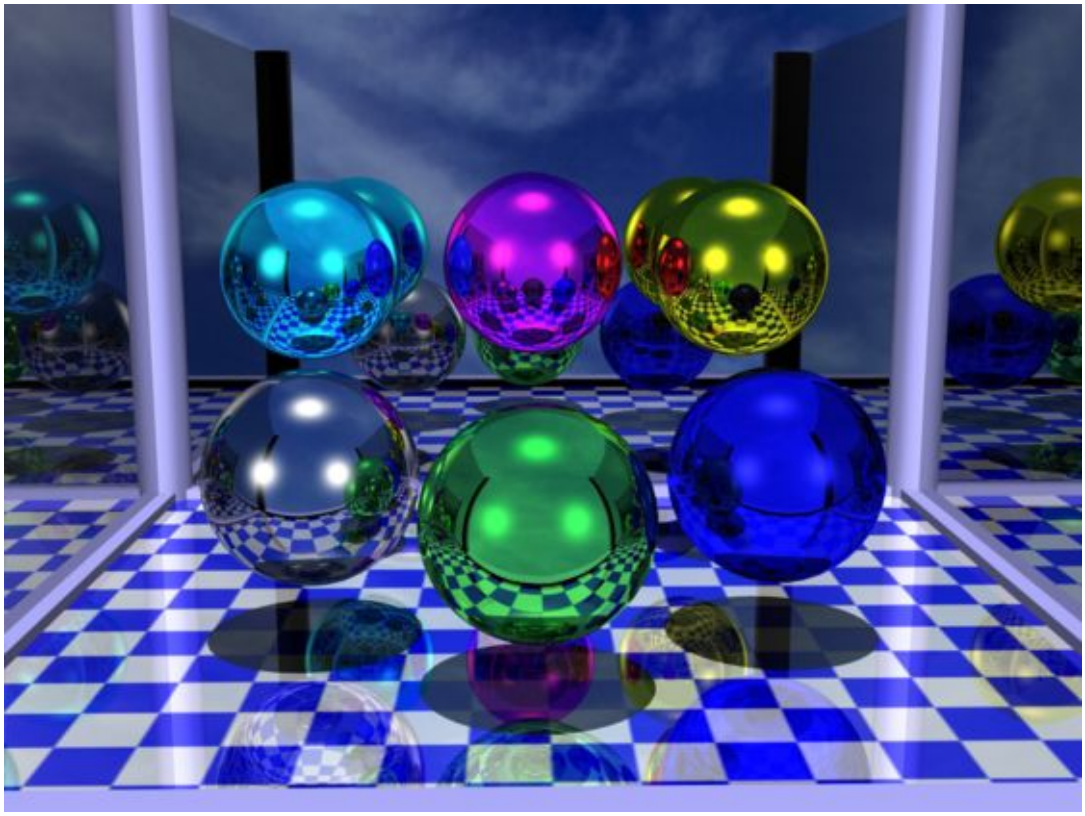


Рисунок 7.29 – Приклад зображення, отриманого методом зворотного трасування променів

СПИСОК ЛІТЕРАТУРИ

1. Ашкенази Г. И. Цвет в природе и технике. – М.: Энергоатомиздат, 1985.
2. Блинова Т. А., Порев В. Н. Компьютерная графика. – К. : Юниор, 2006.
3. Иванов В. П., Батраков А. С. Трехмерная компьютерная графика. – М: Радио и связь, 1995.
4. Павлидис Т. Алгоритмы машинной графики и обработки изображений. – М.: Радио и связь, 1986.
5. Порев В. Н. Компьютерная графика. – СПб.: БХВ-Петербург, 2002.
6. Роджерс Д. Алгоритмические основы машинной графики: Пер. с англ. – М: Мир, 1989.
7. Шикин Е. В., Боресков А. В. Компьютерная графика. – М.: «Диалог-МИФИ», 1995.
8. Шикин Е. В., Боресков А. В. Компьютерная графика. Полигональные модели – М.: «Диалог-МИФИ», 2000.

Основи комп’ютерної графіки
Курс лекцій (для студентів інженерних спеціальностей)

Укладач: Олег Васильович Федоров, к.т.н., доц.