НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ МІЖНАРОДНИХ ВІДНОСИН
КАФЕДРА КОМП'ЮТЕРНИХ МУЛЬТИМЕДІЙНИХ ТЕХНОЛОГІЙ

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни

«ІНЖЕНЕРНА І КОМП'ЮТЕРНА ГРАФІКА»

Укладачі: Веретільник Т.І., Родіонова О.В.

для студентів спеціальності 186 Видавництво та поліграфія

ОПП «Технології електронних мультимедійних видань»

Розділ «Векторна графіка. Adobe Illustrator»

Київ

2022

ЛЕКЦІЯ 1.
ТЕМА: ВСТУП. ОСНОВИ ГРАФІЧНОГО ПРЕДСТАВЛЕННЯ ІНФОРМАЦІЇ. ІНЖЕНЕРНА ГРАФІКА

Питання

1. Місце дисципліни «Інженерна і комп'ютерна графіка» в професійній діяльності.
2. Завдання інженерної графіки.
3. Види інженерної графіки.
4. Області застосування інженерної графіки.
5. Програмне забезпечення інженерної графіки.

ІНЖЕНЕРНА ГРАФІКА

Інженерна графіка - геометричне та проекційне креслення.

Креслення - виконання креслень за правилами, що визначаються комплексом державних стандартів (ДСТУ), або - за «Єдиною системою конструкторської документації» (ЄСКД), складеною за правилами та нормами міжнародних стандартів.

Історія інженерної графіки

Основними принципами розвитку уявлень про навколишній світ, з найдавніших часів і до сучасності, є геометризація і координатизація навколишнього простору та його об'єктів.

У процесі розширення знання та сфер людської діяльності відбувалася еволюція, від уміння переносити зорові образи на поверхню у вигляді контурів предметів до створення інженерної графіки.

Аналітична геометрія та декартова прямокутна система координат

Декартова тривимірна лівостороння система координат. Ox – вісь абцис, Oy – вісь ординат, Oz – вісь аплікат. *Декартовими прямокутними координатами* x, y та z точки M називаються величини спрямованих відрізків OM x, OM y та OM z.

В аналітичній геометрії кожна точка тривимірного простору описується як набір трьох величин — координат.

Задаються три взаємно перпендикулярні координатні осі, що перетинаються на початку координат. Положення точки задається щодо цих трьох осей завданням упорядкованої трійки чисел. Кожне з цих чисел задає відстань від початку відліку до точки, виміряне вздовж відповідної осі, що дорівнює відстані від точки до площини, утвореної двома вісями.

В основі цього методу лежить так званий метод координат, вперше сформульований П'єром Ферма в рукописному трактаті "Введення у вивчення плоских і тілесних місць" ("Ad locos planos et solidos"). Незалежно від Ферма, цей принцип був викладений Рене Декартом (René Descartes) у трьох книгах Геометрії в 1637. Кожному геометричному співвідношенню цей

метод ставить у відповідність деяке рівняння, що зв'язує координати фігури або тіла, і навпаки. Такий метод «алгебраїзації» геометричних властивостей довів свою універсальність і широко застосовується у багатьох природничих науках та у техніці.

Прямокутна система координат названа на честь Декарта, хоча в його творі «Геометрія» (1637) розглядалися косоугольна двовимірна система координат, в якій координати точок могли бути тільки позитивними. У виданні 1659—1661 років до «Геометрії» прикладено роботу голландського математика І. Гудде, у якій вперше допускаються як позитивні, і негативні значення двовимірних координат. Просторову (тривимірну) декартову систему координат ввів у 1679 році французький математик Ф. Лаїр. З усієї термінології, запропонованої Лаїром, прищепилося лише позначення O (з фр. *origine* - початок). На початку XVIII століття французьким геометром Жераром Дезаргом були введені позначення x, y, z.

Декартовими прямокутними координатами x, y та z точки M називаються величини спрямованих відрізків OM_x, OM_y та OM_z.

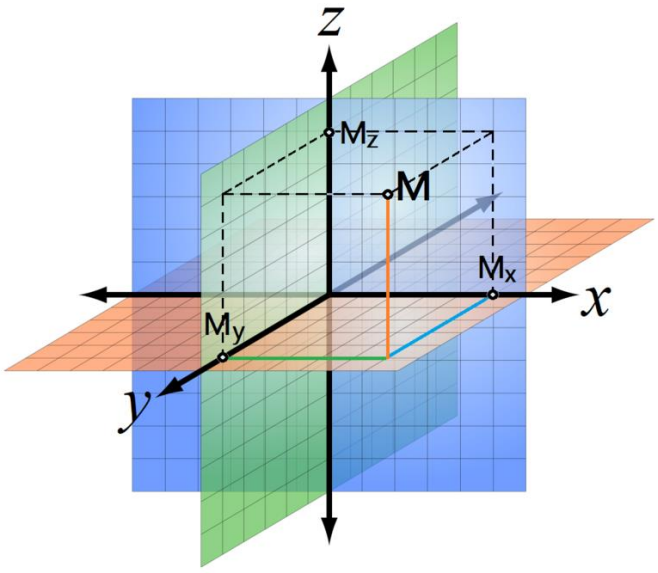


Рисунок 1 - Декартова тривимірна лівостороння система координат. Ox – вісь абцис, Oy – вісь ординат, Oz – вісь аплікат.

Розробка основ нарисної геометрії

Гаспар Монж (1746-1818), граф де Пелюз.

Механіки Стародавнього світу, Середньовіччя та Відродження складали креслення у процесі виготовлення та монтажу різних виробів. Більшість цих креслень не збереглося, оскільки вони були таємницею династій будівельників та механіків, а також гільдій ремісників.

Індустріальна революція і супутнє їй масове виробництво вимагали уніфікації та інформативності креслень, а також простоти їх виготовлення. Засновником нарисної геометрії вважають французького вченого Гаспара Монжу. У своїй книзі "Geometrie descriptive" ("Описова геометрія"), опублікованій в 1798 р., Гаспар Монж виклав загальну геометричну теорію, що дає можливість на плоскому аркуші, що містить ортогональні проекції тривимірного тіла, вирішувати різні стереометричні завдання.

Їм була створена абстрактна геометрична модель реального простору , згідно з якою кожній точці тривимірного простору ставиться у відповідність дві її ортогональні проекції на взаємно перпендикулярні площині. Згодом проекційний креслення, побудований за правилами накреслювальної геометрії, стає робочим інструментом інженерів, архітекторів і техніків усіх країн.

Монж використовував у своїй теорії терміни "горизонталь", "горизонтальна лінія проекції" та "горизонтальна площина проекцій", а також "вертикаль", "вертикальна лінія проекції" та "вертикальна площина проекцій". Наявність усталених термінів у професійному середовищі, на думку Монжа, є достатньою підставою для відмови від введення в обіг більш загальної абстрактної термінології.

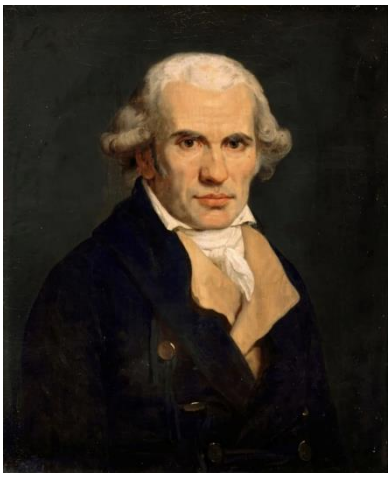


Рисунок 2 - Гаспар Монж (1746-1818), граф де Пелюз

Геометричне креслення

З сучасного її виду, геометричне креслення представлена двома напрямками накреслювальної геометрії:

- теоретичною наукою, що вивчає просторові фігури за допомогою їх проектування (прокладання) перпендикулярами на деякі три площини, які потім розглядаються поєднаними одна з одною;
- а також як інженерна дисципліна, що представляє двовимірний геометричний апарат і набір алгоритмів для дослідження властивостей геометричних об'єктів.

Практично нарисна геометрія обмежується дослідженням об'єктів тривимірного евклідового простору. Вихідні дані мають бути подані у вигляді двох незалежних проекцій. У більшості завдань та алгоритмів використовуються дві ортогональні проекції на взаємно перпендикулярні площині.

В даний час дисципліна не має практичної цінності через розвиток обчислювальної техніки та апарату лінійної алгебри (повсюдного застосування комп'ютерного моделювання), але, ймовірно, незамінна як складова загальної інженерної освіти на машинобудівних та будівельних спеціальностях.

Проекційне креслення

Існує два методи проектування.

1. ***Метод центрального проектування***, або конічної перспективи, що дає зображення предмета таким, яким ми його бачимо. У зображеннях, виконаних цим методом, лінії різного напрямку зменшуються не в однакове число разів, що не дозволяє судити про дійсні розміри тієї чи іншої частини предмета. Тому метод центральних проекцій не знайшов широкого застосування в машинобудуванні, але використовується в архітектурних проектах при виконанні перспектив будівель та живопису.
2. ***Метод паралельного проектування*** ґрунтується на припущенні нескінченної віддаленості центру проекцій. У цьому випадку проекції промені практично паралельні один одному, і розмірна невідповідність ліній, що малює центральним проекціям, виключається.

Продуктом проекційного креслення є креслення - графічний конструкторський документ, що містить зображення інженерного об'єкта (наприклад, деталі, складальної одиниці, виробу, будівлі, споруди тощо), а також дані, необхідні, залежно від конструктивного рівня, для його виготовлення, складання, монтажу, пакування, будівництва, контролю та ін. Зазвичай креслення містить двовимірні та тривимірні види, розміри, текстові написи та таблиці.

Класифікація креслень

Класифікацію креслень міждержавним стандартом проведено:

1. по галузях: технічні креслення, будівельні креслення;
2. за призначенням — у кожній із двох вище зазначених галузей.

Технічні креслення класифікуються за призначенням:

- креслення деталі - документ, що містить зображення деталі та інші дані, необхідні для її виготовлення та контролю;

- складальний креслення - документ, що містить зображення складальної одиниці та інші дані, необхідні для її складання (виготовлення) та контролю. До складальних креслень також відносять креслення, якими виконують гідромонтаж і пневмомонтаж;
- креслення загального виду - документ, що визначає конструкцію виробу, взаємодію його складових частин і пояснює принцип роботи виробу;
- теоретичний креслення - документ, що визначає геометричну форму (контур) виробу та координати розташування складових частин;
- габаритне креслення - документ, що містить контурне (спрощене) зображення виробу з габаритними, настановними та приєднувальними розмірами;
- електромонтажне креслення - документ, що містить дані, необхідні для виконання електричного монтажу виробу;
- монтажне креслення - документ, що містить контурне (спрощене) зображення виробу, а також дані, необхідні для його встановлення (монтажу) на місці застосування. До монтажних креслень також відносять креслення фундаментів, що спеціально розробляються для встановлення виробу;
- пакувальне креслення - документ, що містить дані, необхідні для виконання пакування виробу.

Будівельні креслення у складі проектної документації для будівництва класифікуються за призначенням:

1. креслення архітектурних рішень - креслення будівлі або споруди, що відображають авторський задум об'єкта з комплексним рішенням просторових, планувальних, функціональних та естетичних вимог до нього, зафіксований у вигляді контурного умовного зображення несучих та огороджувальних конструкцій;
2. креслення конструктивних рішень - креслення, що відображають у вигляді умовних зображень будівельних конструкцій (залізобетонні, кам'яні, металеві, дерев'яні, пластмасові тощо), застосовані в будівлях або спорудах, та їх взаємне розміщення та з'єднання;

За методом проектування: спочатку побудова 3D, потім креслення, а також у зворотному порядку.

По носію: цифрові, паперові.

Виконання креслень

Виконання креслень, коротко «креслення», здійснюється у рамках інженерної графіки, за правилами, що визначаються комплексом державних стандартів (ДСТУ).

З розвитком графічної статики за допомогою креслення стали легко і швидко вирішувати безліч чисельних завдань, що зустрічаються при проектуванні споруд і машин і потребують складних викладок алгебри.

Архітектурне креслення користується іншими умовними позначеннями та прийомами, але також вимагає точного дотримання розмірів, оскільки їх визначають при користуванні планом безпосереднім виміром за допомогою циркуля та масштабу. У заводських кресленнях, що даються в руки робітникам-виконавцям, переважно допускається більш грубе виконання, оскільки основні розміри зазвичай задаються, а самі креслення нерідко виконуються в натуральну величину.

За старих часів було прийнято ретельно обробляти всі інженерні, архітектурні та машинобудівні креслення: викреслювати тонкими лініями, ретельно розфарбовувати і навіть відтіняти округлі поверхні розмиванням туші.

Види паралельного проектування

Прямокутне проектування

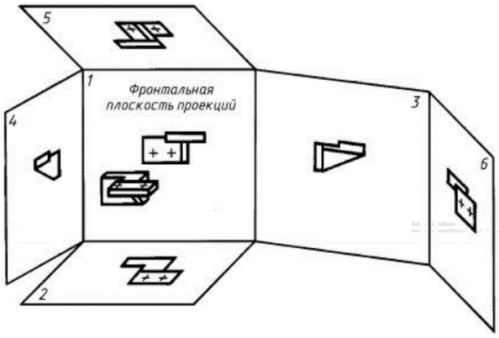


Рисунок 3 - Зображення предмета на кресленні методом прямокутного проектування

Зображення предметів повинні виконуватися на кресленнях (електронних моделях) всіх галузей промисловості та будівництва за методом прямокутного проектування. При цьому предмет передбачається розташованим між спостерігачем та відповідною площиною проекцій.

Встановлено такі назви видів, що одержуються на основних площинах проекцій (основні види):

- вид спереду (головний вид); на фронтальній площині проекцій П2;

- вид зверху; на горизонтальній площині проекцій П1;

- вигляд зліва; на профільній площині проекцій П3;

- вид справа;

- вид знизу;

- вид ззаду.

Аксонометрична проекція

Аксонометри́чна проекція (від грец. «вісь» + «вимірюю») спосіб зображення геометричних предметів на кресленні за допомогою паралельних проекцій.

Предмет із системою координат, до якої він віднесений, проектують довільну площину (картинна площина аксонометричної проекції) таким чином, щоб ця площина не збігалася з його координатною площиною. В цьому випадку виходять дві взаємозалежні проекції однієї фігури на одну площину, що дозволяє відновити положення в просторі, отримавши наочне зображення предмета. Так як картинна площина не паралельна жодній з координатних осей, є спотворення відрізків по довжині паралельних координатним осям. Це спотворення може бути рівним по всіх трьох осях - ізометрична проекція, однаковими по двох осях - диметрична проекція і з різними спотвореннями по всіх трьох осях - триметрична проекція.

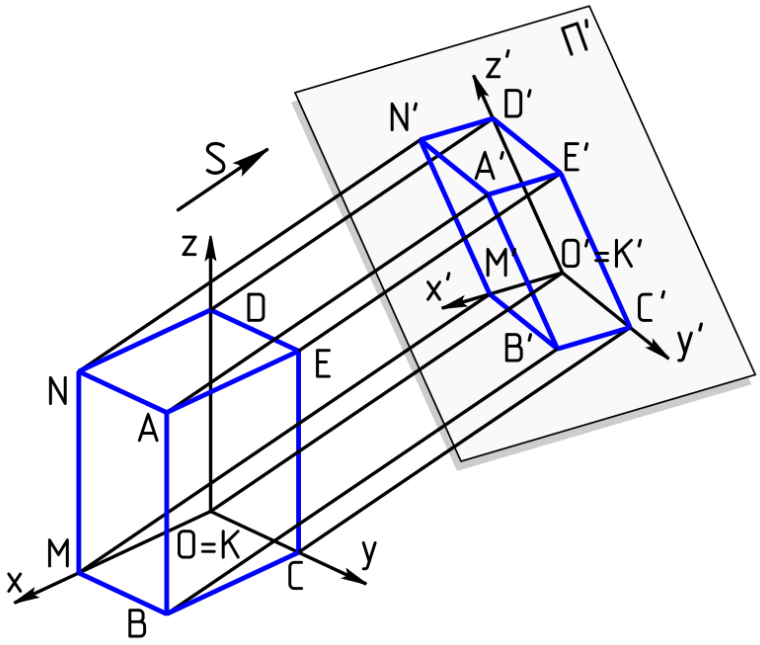


Рисунок 4 - Проеціювання паралелепіпеда на площину П' методом аксонометрії

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

AutoCAD

https://www-autodesk-com.translate.goog/products/autocad/overview?_x_tr_sl=auto&_x_tr_tl=uk&_x_tr_hl=uk&term=1-YEAR&tab=subscription&plc=ACDIST

AutoCAD — дво- та тривимірна система автоматизованого проектування та креслення, розроблена компанією Autodesk. Перша версія системи було випущено 1982 року. AutoCAD та спеціалізовані додатки на його основі знайшли широке застосування у машинобудуванні, будівництві, архітектурі та інших галузях промисловості. Програма випускається 18 мовами. Рівень локалізації варіює від повної адаптації до перекладу лише довідкової документації. Російськомовна версія локалізована повністю, включаючи інтерфейс командного рядка та всю документацію, крім посібника з програмування.

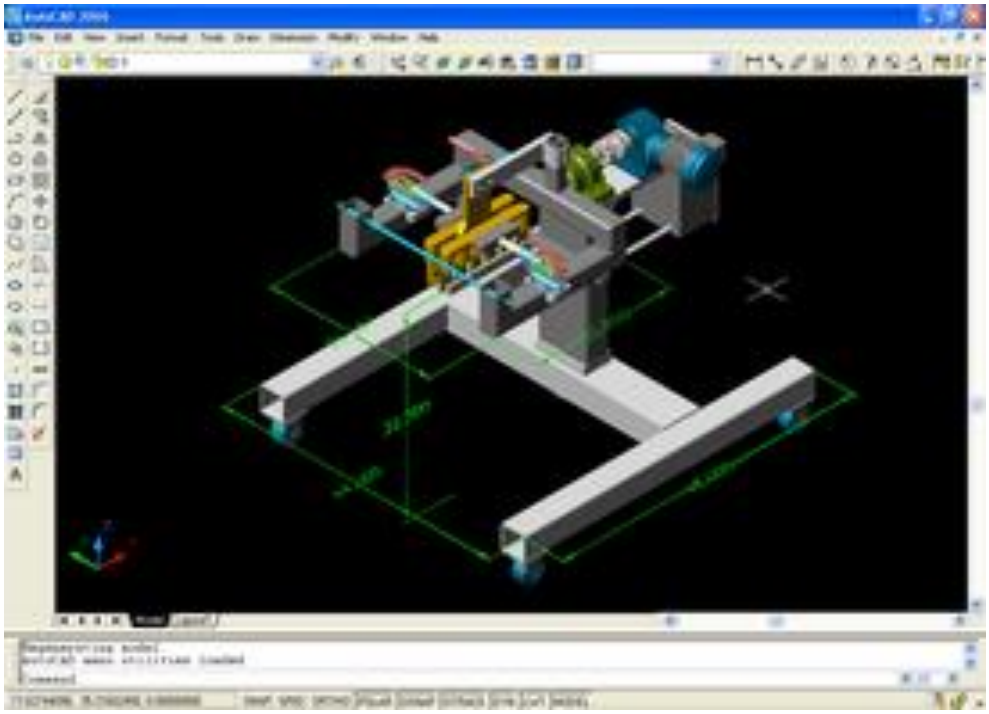


Рисунок 5 - AutoCAD

ФОРМАТИ ФАЙЛІВ

Основним форматом файлу AutoCAD є DWG – закритий формат, розроблений Autodesk. Для обміну даними з користувачами інших САПР пропонується використовувати відкритий формат DXF. Файли з розширеннями DWG і DXF може читати більшість сучасних САПР (Система автоматизованого проектування), оскільки ці формати є стандартом де-факто в області двовимірного проектування. Для публікації креслень та 3D-моделей (без можливості редагування) використовується формат DWF та DWFx, також створені компанією Autodesk.

Програма підтримує запис (за допомогою процедури експорту) файлів, формату DGN, SAT, STL, IGES, FBX та інших. А також читання (за допомогою процедури імпорту) файлів, формату 3DS, DGN, JT, SAT, PDF, STEP та деяких інших. Починаючи з версії 2012, AutoCAD дозволяє перетворювати файли, отримані з тривимірних САПР (таких як Inventor, SolidWorks , CATIA, NX тощо) у формат DWG.

SOLIDWORKS

(https://www.solidworks.com)

- SolidWorks (від англ. solid — тверде тіло та англ. works — працювати) — програмний комплекс САПР для автоматизації робіт промислового підприємства на етапах конструкторської та технологічної підготовки виробництва. В поліграфії – етикетно-пакувальна продукція.
- Забезпечує розробку виробів будь-якого ступеня складності та призначення. Працює під Microsoft Windows.
- Розроблено компанією SolidWorks Corporation, створеною з нуля Джоном Хірштіком, а з 1997 року незалежним підрозділом компанії Dassault Systemes (Франція).
- Програму почали розробляти в 1993 році, вона почала продаватися в 1995 і склала конкуренцію таким продуктам, як AutoCAD і Autodesk Mechanical Desktop, SDRC I-DEAS, Компас і Pro/ENGINEER.
- Система SolidWorks стала першою САПР, що підтримує тверде моделювання для платформи Windows. SolidWorks використовує ядро Parasolid.

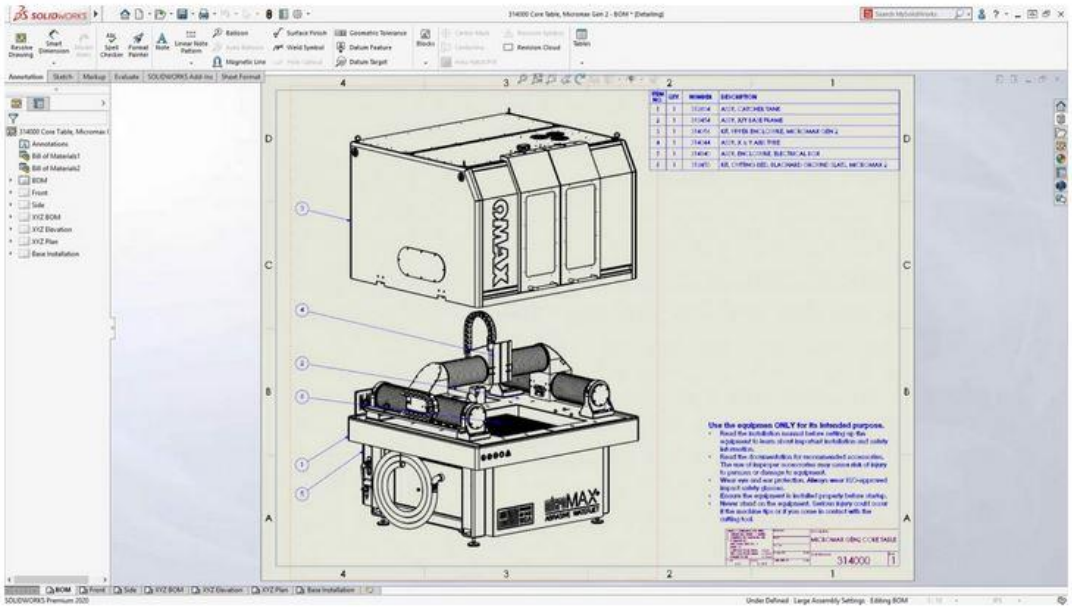


Рисунок 6 – Вікно SolidWorks

PRINECT PACKAGE DESIGNER

- Програма Prinect Package Designer значно спрощує процес проектування складних коробок і дисплеїв, а також розкладку деталей для подальшої висічки.
- Функціональність включає конструювання упаковки, виготовлення зразка, створення розкладки, введення параметрів висікання, генерування даних для налаштування пресів та різальних машин.
- За допомогою 3D-моделей та анімації фальцювання полегшується взаємодія з клієнтами та дизайнерами. Швидше і з більшою точністю можна налаштувати машину,

що фальцювально-склеює. В комплект поставки входить велика бібліотека типів складних коробок, елементів дисплеїв та упаковки, а також каталоги ECMA та FEFCO.

- Prinect Package Designer інтегрується у комплексний робочий потік Prinect. Налаштування післядрукарських машин за даними CAD суттєво заощаджує час та матеріал.

Можливості Prinect Package Designer

- Опція Synergy створює основу для ефективного проектування елементів упаковки зі змінними параметрами. Тут можна створювати бібліотеки складних коробок, дисплеїв, окремих елементів зі змінними розмірами.
- За допомогою опції Diemaker можна створювати креслення висікальних форм з перемичками, ножами та протиніжами, не кажучи вже про прості креслення контурів висікання.
- Винятково зручний інструментарій для проектування складних коробок і дисплеїв, конструювання упаковки зі змінними параметрами, створення форм для висікання 3D-моделі коробок для інтерактивної перевірки та показу замовнику
- Анімація послідовності фальцювання, налаштування фальцювально-склейувальних машин
- Великі бібліотеки, каталоги ECMA та FEFCO
- Опції для проектування упаковки зі змінними параметрами, форм для висікання.

Рисунок 7 – Вікно програми Prinect Package Designer

ADOBE SUBSTANCE 3D

Adobe Substance 3D – це екосистема програм і вмісту, яка надає безліч можливостей для 3D-дизайну. Додавайте текстури на ресурси та створюйте композиції за допомогою інтуїтивно зрозумілих інструментів. Керуйте робочими процесами у програмах Adobe Creative Cloud.

Рисунок 8 – Графіка в програмі Adobe Substance 3D

Інформаційні ресурси

1. https://www.autodesk.com
2. https://www.solidworks.com
3. https://www.heidelberg.com
4. https://www.adobe.com

ЛЕКЦІЯ 2.
ТЕМА: КОМП'ЮТЕРНА ГРАФІКА

План

1. Основні поняття комп'ютерної графіки.
2. Розвиток комп'ютерної графіки у ретроспективі.
3. Сфери застосування комп'ютерної графіки.
4. Моделі графічних об'єктів.

1. Основні поняття комп'ютерної графіки.

Формування комп'ютерної графіки як самостійного напрямку інформаційних технологій відноситься до 60-х років XX ст., коли А. Сазерлендом був створений спеціалізований пакет машинної графіки. Вперше подання інформаційних даних на екрані комп'ютера в графічному вигляді було продемонстровано на початку 50-х років спеціалістами Массачусетського технологічного інституту і згодом стало використовуватися в наукових і військових дослідженнях. За цей час комп'ютерна графіка пройшла шлях від окремих експериментів до одного з найважливіших інструментів сучасності. Розвиток комп'ютерної графіки проходить жваво: дещо швидко відживає, а водночас з'являється багато нового. Нині не можна собі уявити розвиток будь-якої галузі людської діяльності, пов'язаної з наукою чи технікою без використання комп'ютерної графіки. Вона є багатофункціональною складовою усіх інформаційних технологій і важливою компонентою для взаємодії людини з комп'ютером. На першому етапі свого розвитку комп'ютерна графіка була пасивною, однак це вже забезпечувало наочну форму сприйняття інформації, а, як відомо, більшу частину інформації (до 75%) про навколишній світ людина сприймає візуально, тобто за допомогою органів зору. З цього приводу можна згадати китайське прислів'я: «одна картинка коштує 1000 слів». Зображення не тільки місткий, але і доступний вид інформації, оскільки для сприйняття візуальної інформації від користувача вимагається менше зусиль. Інформація, що міститься у зображеннях, є найбільш концентрованою, найлегше сприймається та найшвидше обробляється. При пасивній графіці ЕОМ формує і виводить зображення на графічний пристрій під управлінням прикладних програм. Користувач не може безпосередньо керувати формуванням зображення, перш ніж воно не з'явиться на екрані. Пасивний контроль не дає можливості втрутитися в процес проектування конструкції (система працює в пакетному режимі без діалогу). Для модифікації вихідного зображення необхідний повторний запуск пакета прикладних програм. Тому виникла необхідність у розвитку діалогових систем комп'ютерної графіки, тобто систем інтерактивної графіки, щоб користувач оперативно міг вносити зміни в зображення безпосередньо в процесі його відтворення (в реальному масштабі часу). Нині всі сучасні графічні програми є системами інтерактивної комп'ютерної графіки і вони можуть відтворювати всі особливості реальних зображень. Дамо ряд визначень.

Графіка це вид мистецтва, що включає малюнок і художнє зображення (походить від грецьк. слова *графо*, що означає «пишу, креслю, малюю»).

Комп'ютерна графіка це галузь знань, що вивчає та розробляє методи і засоби синтезу, збереження й перетворення цифрових зображень за допомогою ЕОМ та інших технічних пристроїв.

Цифрове зображення це модель реального або синтезованого (створеного штучно) зображення, що зберігається в пам’яті ЕОМ у вигляді сукупності цифрових кодів. Режим, при якому користувач у реальному часі може впливати на весь процес формування зображення, тобто зміни в зображення можна вносити безпосередньо в процесі його відтворення (в режимі діалогу), називається інтерактивним. Інтерактивною КГ називають тоді, коли до складу графічної системи входять технічні і програмні засоби, що дозволяють динамічно формувати зображення.

Історично першими інтерактивними системами вважаються системи автоматизованого проектування (САПР), які з’явилися в 60-х роках XX ст. Також під **комп’ютерною графікою розуміють** область інформатики, у сферу інтересів якої входять всі аспекти формування зображення за допомогою комп’ютера. Цифрові зображення немислимі без комп’ютерної обробки графічної інформації, їх простіше зберігати, тиражувати, компонувати тощо. Цифрові зображення можна створити сканером, цифровим фотоапаратом, а потім відредагувати в програмі обробки зображень (наприклад, в Adobe Photoshop), а можна створити цифрові малюнки з нуля, застосувавши спеціальні програми (наприклад, у Corel Draw) або написавши відповідну програму на мові програмування (наприклад, Visual C++).

Задачі комп’ютерної графіки

Основними задачами КГ є **візуалізація** інформації, тобто створення зображень різних об’єктів і сцен (у загальному випадку тривимірних) на деякому двовимірному екрані (наприклад, на екрані монітора або на аркуші паперу), виконання різних дій із зображеннями, зберігання та передавання графічної інформації. Під синтезованим зображенням розуміють певне довільне візуальне подання інформації, яке отримується в комп’ютері на основі математичного опису (моделі) об’єкта згідно з директивами користувача. Таке зауваження вилучає з предмета КГ задачі обробки та розпізнавання зображень, отриманих шляхом фотографування, оскільки в цьому випадку ми не маємо справи з побудовою графічних об’єктів за певними правилами. При обробці зображень на основі початкового зображення одержують зображення, яке володіє іншими властивостями. Прикладом методів обробки зображень можуть бути підвищення контрасту, корекція кольорів, реставрація зображень.

Щодо тлумачення поняття **Комп’ютерна графіка існує багато думок. Так, комп’ютерна графіка** – це будь-які зображення, створені за допомогою комп'ютерів. Цей термін запропоновано у 1960 р. дослідниками Верном Хадсоном і Вільямом Феттер з корпорації Boing. Зауважимо, що Вільям Феттер був першим, хто зробив зображення людини за допомогою комп’ютера (Рис.1).

Комп'ютерна графіка - будь-які зображення, створені за допомогою комп'ютерів. Термін **«комп’ютерна графіка»** використовується в широкому сенсі для опису майже всього цифрового графічного матеріалу, технологій створення і обробки зображень, методів подання і маніпулювання графічними даними. Також це окрема галузь науки, що вивчає методи цифрового синтезу та управління візуальним контентом. У більш вузькому, або навіть повсякденному сенсі, термін застосовується до зображень, створених з використанням спеціалізованого графічного обладнання та програмного забезпечення.

Оскільки комп'ютерна графіка є великою областю, з неї можна виокремити безліч окремих тем, таких як дизайн користувальницького інтерфейсу, графіка спрайтів, векторна графіка, 3D-моделювання, шейдери, графічний дизайн, приховану візуалізацію поверхні за допомогою трасування променів і комп'ютерний зір. Загальна методологія багато в чому

залежить від фундаментальних наук про геометрію, оптику і фізику. Комп'ютерна графіка відповідає за те, щоб ефективно і осмислено відображати дані мистецтва і зображення. Також використовується для обробки даних зображення, отриманих з фізичного світу.

Рис. 1 – Перше зображення людини, створене за допомогою комп’ютера

Розвиток комп'ютерної графіки здійснив значний вплив на багато типів носіїв і радикально змінив анімацію, фільми, рекламу, відеоігри і графічний дизайн. Для візуалізації даних існує безліч інструментів.

Згенеровані комп'ютером зображення діляться на кілька типів:

- двомірну (2D);
- тривимірну (3D);
- анімовану графіку.

У міру поліпшення технологій, тривимірна комп'ютерна графіка стала більш поширеною, але двомірна як і раніше широко використовується. За останній час були розроблені інші спеціалізовані області, такі як інформаційна візуалізація, а наукова візуалізація використовується в метеорології, медицині, біології, картографія.

Види комп’ютерної графіки

Протягом всієї історії розвитку комп'ютерної графіки виділяються три її види: растрова, векторна і фрактальна графіки.

Дані види відрізняються між собою побудовою зображення, а також відображенням на екрані монітора. Розглянемо їх більш детально.

1. Характерним елементом растрової графіки вважається точка (піксель). На екрані монітора дана точка називається пікселем. Ілюстрації растрової графіки рідко створюються вручну за допомогою комп'ютерних програм, для цього використовують підготовлені зображення, виконані художниками на папері. Необхідно відзначити недоліки використання комп'ютерної растрової графіки. По-перше, растрові зображення містять великий обсяг даних. Другий недолік, погана якість зображення при його збільшенні.

2. Важливим елементом векторної графіки є лінія. Слід підкреслити, що обсяг пам'яті векторного зображення не залежить від розміру лінії, так як лінії представляються у вигляді формул. Векторна графіка призначена для створення зображень, але не для їх обробки. Векторна графіка використовуються в рекламних агентствах, редакціях і видавництвах. Оздоблювальною роботою вважається застосування різних шрифтів і елементарних геометричних елементів.

3. Основним елементом фрактальної графіки є рівносторонній трикутник. За допомогою фрактальної графіки створюються абстрактні зображення, де можна реалізувати такі прийоми як, горизонталі та вертикалі, діагональні напрямки, симетрію і асиметрію. Також при створенні і оформленні рекламних листівок і веб-сайтів використовують фрактальну графіку. На сьогоднішній день фрактальна графіка є одним з найшвидших і перспективних видів графіки. Зазначимо, що при збільшенні фрактального зображення його форма залишається незмінною.

2. Розвиток комп’ютерної графіки у ретроспективі.

Попередниками розвитку сучасної комп'ютерної графіки були досягнення в області електротехніки, електроніки та телебачення. Перша електронно-променева трубка (ЕПТ), **трубка Брауна, була винайдена в 1897 році. Вона дозволила створити осцилограф, що** представляє собою двомірний дисплей, реагував на програмний або призначений для користувача введення.

Електронно-променева трубка - електронний прилад, що має форму трубки, видовженої (часто з конічним розширенням) в напрямку осі електронного променю. ЕПТ складається з електронно-оптичної системи, відхиляючої системи і флуоресцентного екрана або мішені.

Відправною точкою розвитку комп'ютерної графіки можна вважати 1929 рік, коли в США Володимиром Зворикіним, який працював в компанії Westinghouse, була винайдена електронно-променева трубка (ЕПТ), що вперше дозволила отримати зображення на екрані без використання механічних рухомих частин. Основна сфера застосування ЕПТ - перетворення оптичної інформації в електричні сигнали і зворотне перетворення електричного сигналу в оптичний - наприклад, в видиме телевізійне зображення (рис. 1).

Рис. 2 – Електронно-променева трубка В. Зворикіна

Також початком ери власне комп'ютерної графіки можна вважати грудень 1951 року, коли в Массачусеттському технологічному інституті (MTI) для системи протиповітряної оборони військово-морського флоту США був розроблений перший дисплей для комп'ютера «Вихор». Винахідником цього дисплея був інженер з MTI Джей Форрестер.

Рис. 3 – Комп'ютер «Вихор»

Електронно-променева трубка слугувала у комп’ютері «Вихор» для виведення даних на екран, подібний телевізійному; пучок електронів, потрапляючи на вкриту фосфором внутрішню поверхню екрану, викликав його світіння. Щоб зберегти інформацію, що виводиться, її можна було віддрукувати на електричній друкарській машинці, подібній телетайпу.

Також слід згадати 1950 р., коли були створені перші радари з дисплеєм. Найбільш ранній проект - Whirlwind та SAGE, які впровадили ЕПТ в якості інтерфейсу.

Дуглас Т. Росс провів експеримент, в якому невелика програма захоплювала рух пальця і відображала вектор руху на екрані. Одна з перших інтерактивних відеоігор (Tennis for two; 1958 р.) з можливістю розпізнавання інтерактивної графіки створена для осцилографа Вільяма Хігінботама, щоб розважити відвідувачів Брукхейвенської національної лабораторії: був змодельований процес гри в теніс.

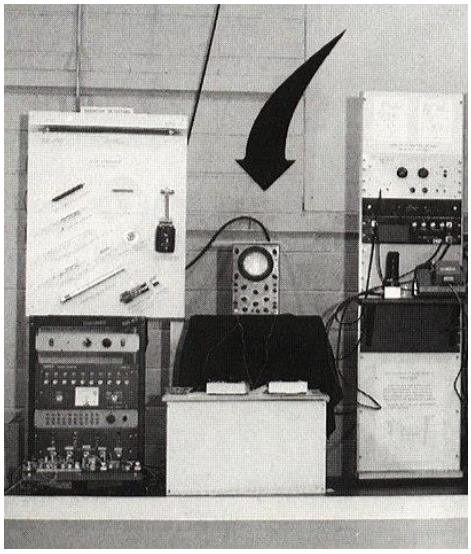


Рис. 4 – Обладнання для гри «Tennis for two»

Приблизно в той же час на території південної затоки Сан-Франциско створюється світовий комп'ютерний центр технологій (Кремнієва долина) при прямій участі Hewlett-Packard, що встановив міцні зв'язки зі Стенфордським університетом.

Того ж року розроблений комп'ютер TX-2, який інтегрував ряд нових людино-машинних інтерфейсів. TX-2 був комп'ютер побудований на транзисторах з величезним на ті часи ОЗУ на феритових сердечниках з розміром 64 тис. машинних слів розміром 36 біт. TX-2 запрацював в 1958 році. Його потужні характеристики дозволили Айвен Сазерленду створити революційну програму під назвою Sketchpad.

Світлове перо дозволяло малювати ескізи на комп'ютері з використання революційного програмного забезпечення Sketchpad. Стало можливим малювати прості фігури на екрані комп'ютера, зберігати їх в пам'яті, а потім відтворювати знову. Розробника даної програми вважають одним з засновників комп'ютерної графіки. Ця програма могла малювати досить прості фігури (точки, прямі, дуги кіл), могла обертати фігури на екрані.

Рис. 5 – Програма Sketchpad у роботі

У 1964 році General Motors представила систему автоматизованого проектування DAC-1, розроблену спільно з IBM [8]. У 1965 році фірма IBM випустила перший комерційний графічний термінал під назвою IBM-2250 (рис. 6).

Рис. 6 - IBM-2250

IBM 2250 Graphics Display Unit — векторна дисплейна система, розроблена компанією IBM для комп'ютерів серії IBM System/360. Анонсована у 1964 році.

У 1968 році групою під керівництвом Н. Н. Константинова була створена комп'ютерна математична модель руху кішки. Машина БЕСМ-4, виконуючи написану програму рішення диференціальних рівнянь, малювала мультфільм «Кішечка», який для свого часу був проривом. Для візуалізації використовувався алфавітно-цифровий принтер.

У 1960 р. Вільям Феттер, графічний дизайнер Боїнг, придумав і ввів у вживання фразу «комп'ютерна графіка». У 1961р. студент Массачусетського технологічного інституту Стів Рассел створив відеогру Spacewar!

Рис. 7 – Комп’ютерна гра «Spacewar!»

Однією з платформ для даної гри був PDP-1, що мав 18-бітове машинне слово і 4 кілослова основний пам'яті (еквівалентно 9 КБ) з можливістю розширення до 64 кілослов (144 КБ). Цикл реренамагнічювання пам'яті на феритових елементах займав 5 мікросекунд (приблизно відповідає тактовій частоті 200 КГц); відповідно більшість арифметичних операцій займали 10 мікросекунд (100 000 операцій в секунду), тому що вони мали два звернення до пам'яті: одне для інструкції, інше для операндів.

Машинне слово — машинозалежна і платформозалежна величина, що вимірюється в бітах або байтах, що дорівнює розрядності регістрів процесора і/або шини даних. Машинне слово визначає наступні характеристики апаратної комп'ютерної платформи:

- розрядність даних, що оброблюються процесором;
- розрядність адресованих даних (розрядність шини даних);

- максимальне значення беззнакового цілого типу, що безпосередньо підтримується процесором: якщо результат арифметичної операції перевершує це значення, то відбувається переповнення;
- максимальний об'єм оперативної пам'яті, безпосередньо адресується процесором.

У 1963 р. Е. Заєць, вчений з Bell Telephone Laboratory (BTL) разом з колегами створили першу комп'ютерну анімацію. Приблизно одночасно з цим інші вчені створювали комп'ютерну графіку для ілюстрації своїх досліджень. Також на початку 60-х було забезпечене бурхливе зростання автомобільної промисловості завдяки роботі П'єра Безьє з Renault, який використовував криві Пола де Кастельжау, тепер званими кривими Безьє. Криві Безьє вперше застосовані для 3D-моделювання кузовів автомобілів Рено. У 1966 р. Ральф Бер, головний інженер Sanders Associates, придумав домашню відеоприставку Agnavox Odyssey. Хоча пристрій було вкрай простим і використовував недорогі електронні компоненти, він дозволяв гравцеві переміщати точки світла на екрані. Це був перший споживчий комп'ютерний графічний продукт.

Рис. 8 - Agnavox Odyssey

У 1967 році Айвен Сазерленд (англ. Ivan Sutherland) описав і сконструював перший шолом, зображення на який генерувалося за допомогою комп'ютера. Шолом Сазерленда дозволяв змінювати зображення після руху голови (зоровий зворотний зв'язок). У 1968 р. Артур Аппель описав алгоритм, який назвали Ray casting, відправною точкою сучасної 3D-графіки і фотореалізму. В основі рейкастінга варто ідея випускати промені з «очей» спостерігача, один промінь на піксель, і знаходити найближчий об'єкт, який блокує шлях поширення цього променя. Використовуючи властивості матеріалу і ефект світла в сцені, алгоритм рейкастінга може визначити затінення даного об'єкта.

Рис. 9 – Шолом Айвена Сезерленда

Так, 1970-ті роки ознаменувалися серією проривів в області комп'ютерної графіки. Створюються складні алгоритми анімації і технології накладення текстур. Відбувається революція в видавничому середовищі завдяки створенню PostScript – мова опису сторінок, корпорація Adobe продовжує роботу над створенням програмного забезпечення для редагування фотографій (Adobe Photoshop і Adobe After Effects). Значний прогрес в 3D графіці стався після розробки *алгоритмів прихованої поверхневої детермінації*. За допомогою цих алгоритмів комп'ютер міг визначити, які поверхні знаходяться позаду інших об'єктів з точки

зору спостерігача, і, таким чином, повинні бути приховані при візуалізації. Анрі Гуро, Джим Блинн і Буй Туонг Фонг розробили модель затінення, що дозволила графіку вийти за рамки площині і точно відобразити глибину сцени. Джим Блінн став новатором в області впровадження карт рельєфу, техніки моделювання нерівних поверхонь. Створюються перші аркадні ігри, які використовували метод безперервного прокручування спрайту. Вперше в відеогрі з'являються персонажі в образі людини.

У 1975 році Бенуа Мандельброт опублікував роботу «Фрактальна геометрія природи», в якій стверджував, що випадкові на перший погляд форми є насправді складними геометричними фігурами, що складаються з менших фігур, які точно повторюють більшу.

У 1977 році Commodore випустила свій PET (Personal Electronic Transactor), а компанія Apple створила Apple-II. Слід зазначити, що поява цих пристроїв викликало змішані почуття: графіка була недостатньо якісною, а тогочасні процесори повільними. Однак ПК стимулювали процес розробки периферійного обладнання: недорогих пристроїв для побудови графів, а також графічних планшетів.

Рис. 10 – Commodore PET

Рис. 11 – Персональний комп’ютер Apple-II

До кінця 80-х програмне забезпечення стало швидко поширюватися: від комплексів управління до настільних видавничих комплексів. В кінці вісімдесятих років виникли нові напрями розвитку апаратних і програмних систем сканування, автоматичної оцифрування тощо. Оригінальний поштовх в таких системах повинна була створити машина Ozalid, яка б сканувала і автоматично векторизувала креслення на папері, перетворюючи його в стандартні формати CAD / CAM. Однак, акцент зрушився в сторону обробки, зберігання та передачі сканованих піксельних зображень.

Спостерігається модернізація і комерціалізація комп'ютерної графіки. У міру поширення домашніх комп'ютерів, тема, яка колись була академічною дисципліною, розширилася до користувальницької аудиторії, а число розробників значно зросло. З початку 80-х з появою 16-бітових процесорів з'явилася можливість створювати графіку з високою роздільною здатністю, а самі графічні станції стали більш інтелектуальними, автономними.

Типовими представниками таких графічних станцій були Orca 1000, 2000 і 3000, розроблені Orcatech of Ottawa. Художники і графічні дизайнери стали розглядати персональний комп'ютер, зокрема Commodore Amiga і Macintosh, як серйозний інструмент для дизайну, що економив час і дозволяв працювати точніше в порівнянні з ручними методами. Для студій графічного дизайну і бізнесу основним інструментом стає Macintosh. В області реалістичної анімації і 3D-рендеринга першою була Японія завдяки розробці графічної системи LINKS-1, найпотужнішим комп'ютером в світі на період 84-85 рр. Система обчислювала яскравість кожного пікселя, що становить поверхню, яка візуалізується, з конкретної точки зору з урахуванням просторового положення об'єкта. Кожен піксель оброблявся незалежно з використання трасування променів.

Рис. 12 – Commodore Amiga

Рис. 13 – Apple Macintosh

Серед кінематографічної продукції 80-х слід відзначити "Зоряні війни", легендарний відеокліп Dire Straits для пісні "Money For Nothing" (1985), мультфільми Pixar. Все це стало можливим з появою машини Silicon Graphics. 80-і роки – початок золотої ери відеоігор. Мільйонні продажі систем Atari, Nintendo і Sega відкрили комп'ютерну графіку для молодої аудиторії, а персональні комп'ютери на базі MS-DOS, Apple II, Mac і Amigas дозволили користувачам самим створювати відеоігри.

Рис. 14 – Кадр з відеокліпу групи Dire Straits на композицію *Money for nothing*

У 90-х роках фактично стираються відмінності між комп’ютерною графікою та обробкою зображень. Процесори робочих станцій досягли швидкодії, достатньої для того, щоб управляти як векторною, так і растровою інформацією. Крім того, з'являється можливість працювати з відео. Якщо поєднати згадані можливості, ми отримуємо комп'ютерне мультимедіа середовище.

Домашні комп'ютери стали виконувати завдання, які трохи раніше були долею суперкомп'ютерів і робочих станцій вартістю в тисячі доларів. 3D-моделлинг став доступний для домашніх комп'ютерів, популярність Silicon Graphics знизилася. Розробляється і популяризується полігональна 3D-графіка в реальному часі. Виходять у світ перші ігри: Wolfenstein 3D, Doom і Quake. З'являється відеопроцесор. У 1999 р. Nvidia випустила першу «домашню» змінну відеокарту GeForce 256. Також операційні системи приймають два загальних стандарти обробки графіки: DirectX (1995 р.) і OpenGL (1992 р.). Розвивається обробка відео, відео дизайн.

Рис. 15 – Логотипи бібліотек OpenG, Directx

У 2000-х роках комп'ютерна графіка проникає у майже всі сфери діяльності. Зростаюча складність графічного процесора стає вирішальною і визначальною. Навіть низькопродуктивні машини оснащуються 3D-сумісним графічним процесором. Фільм «Final Fantasy», випущений в 2001 році, був першим, повністю згенерували комп'ютером художнім фільмом, що використав також нову технологію захоплення руху. Індустрія відеоігор настільки розвинулася, що доходи галузі перевищили доходи від фільмів.

Після 2000 року комп'ютерна графіка, причому фотореалістична, впроваджена всюди і доступна для будь-якого, навіть для зовсім непідготовленого користувача домашнього комп'ютера. Разом з тим тривають розробки режимів надвисокої роздільної здатності в реальному часі, такі як Ultra HD. Розвиваються такі напрями, як віртуальна реальність, 3D зображення, технології переносу даних з тіла та інших об’єктів, доповнена реальність.

Зазначимо, що *доповнена реальність і віртуальна реальність* - протилежне відображення одного в іншому з тим, що кожна з технологій прагне надати користувачеві. Віртуальна реальність пропонує цифрове відтворення реальної обстановки життя, в той час як доповнена реальність забезпечує віртуальні елементи у вигляді накладення шарів на реальний світ.

3. Сфери застосування комп’ютерної графіки

Як було сказано вище, сучасна комп'ютерна графіка - це досить складна, ґрунтовно опрацьована і різноманітна науково технічна дисципліна. Деякі її розділи, такі як геометричні перетворення, способи опису кривих і поверхонь, до теперішнього часу вже досліджені досить глибоко. Ряд областей продовжує активно розвиватися: методи растрового сканування, видалення невидимих ліній і поверхонь, моделювання кольору і освітленості, текстурування, створення ефекту прозорості і напівпрозорості та інші.

Комп'ютерна графіка застосовується в чотирьох основних областях:

1. ***Відображення інформації.*** Проблема подання накопиченої інформації (наприклад, даних про кліматичні зміни за тривалий період, про динаміку популяцій тваринного світу, про екологічний стан різних регіонів і т.п.) найкраще може бути вирішена за допомогою графічного відображення.

Жодна з областей сучасної науки не обходиться без графічного представлення інформації. Крім візуалізації результатів експериментів і аналізу даних натурних спостережень існує велика область математичного моделювання процесів і явищ, яка просто немислима без графічного виведення. Наприклад, описати процеси, що протікають в атмосфері або океані, без відповідних наочних картин течій або полів температури практично неможливо. В геології в результаті обробки тривимірних натурних даних можна отримати геометрію пластів, що залягають на великій глибині.

У медицині в даний час широко використовуються методи діагностики, які використовують комп'ютерну візуалізацію внутрішніх органів людини. Томографія (зокрема, ультразвукове дослідження) дозволяє отримати тривимірну інформацію, яка потім піддається математичній обробці і виводиться на екран. Крім цього застосовується і двовимірна графіка: енцефалограми, міограми, що виводяться на екран комп'ютера або графічний пристрій.

2. ***Проектування***. У будівництві та техніці креслення давно являють собою основу проектування нових споруд або виробів. Процес проектування з необхідністю є ітеративним, тобто конструктор перебирає безліч варіантів з метою вибору оптимальних параметрів. Не останню роль в цьому відіграють вимоги замовника, який не завжди чітко уявляє собі кінцеву мету і технічні можливості. Побудова попередніх макетів - досить довгий і дорога справа. Сьогодні існують розвинені програмні засоби автоматизації проектно-конструкторських робіт (САПР), що дозволяють швидко створювати креслення об'єктів, виконувати розрахунки на міцність і т.п. Вони дають можливість не тільки зобразити проекції вироби, а й розглянути його в об'ємному вигляді з різних сторін. Такі засоби також надзвичайно корисні для дизайнерів інтер'єру, ландшафту.

3. ***Моделювання***. Під моделюванням в даному випадку розуміється імітація різного роду ситуацій, що виникають, наприклад, при польоті літака або космічного апарату, русі автомобіля і т.п. В англійській мові це найкраще передається терміном *simulation*. Але моделювання використовується не тільки при створенні різного роду тренажерів. У телевізійній рекламі, в науково-популярних та інших фільмах тепер синтезуються рухомі об'єкти, візуально мало поступаються тим, які можуть бути отримані за допомогою кінокамери. Крім того, комп'ютерна графіка надала кіноіндустрії можливості створення

спецефектів, які в попередні роки були просто неможливі. В останні роки широко поширилася ще одна сфера застосування комп'ютерної графіки - створення віртуальної реальності.

4. ***Графічний користувальницький інтерфейс***. На ранньому етапі використання дисплеїв як одного з пристроїв комп'ютерного виводу інформації діалог «людина-комп'ютер» в основному здійснювався в алфавітно-цифровому вигляді. Тепер же практично всі системи програмування застосовують графічний інтерфейс. Існує безліч різних програм-браузерів, що реалізують в тому чи іншому вигляді засоби спілкування в мережі, без яких доступ до неї важко собі уявити. Ці програми працюють в різних операційних середовищах, але реалізують, по суті, одні й ті ж функції, що включають вікна, банери, анімацію і т.д.

У сучасній комп'ютерній графіці можна виділити наступні основні напрямки: образотворча комп'ютерна графіка, обробка та аналіз зображень, аналіз сцен (перцептивна комп'ютерна графіка), комп'ютерна графіка для наукових абстракцій (когнітивна комп'ютерна графіка, тобто графіка, сприяє пізнанню).

Образотворча комп'ютерна графіка своїм предметом має синтезовані зображення. Основні види завдань, які вона вирішує, зводяться до наступних:

- побудова моделі об'єкта і формування зображення;
- перетворення моделі і зображення;
- ідентифікація об'єкта і отримання необхідної інформації.

Обробка та аналіз зображень стосуються в основному дискретного (цифрового) подання фотографій та інших зображень. Засоби комп'ютерної графіки тут використовуються для підвищення якості зображення; оцінки зображення - визначення форми, місця розташування, розмірів і інших параметрів необхідних об'єктів; розпізнавання образів - виділення і класифікації властивостей об'єктів (при обробці аерокосмічних знімків, введенні креслень, в системах навігації, виявлення і наведення).

Аналіз сцен пов'язаний з дослідженням абстрактних моделей графічних об'єктів і взаємозв'язків між ними. Об'єкти можуть бути як синтезованими, так і виділеними на фотознімках. До таких завдань відносяться, наприклад, моделювання "машинного зору" (роботи), аналіз рентгенівських знімків з виділенням і відстеженням об'єкта, що цікавить (внутрішнього органу), розробка систем відеоспостереження.

Когнітивна комп'ютерна графіка - новий напрям, поки ще недостатньо чітко окреслений. Це - комп'ютерна графіка для наукових абстракцій, що сприяє народженню нового наукового знання. Технічною основою для неї є потужні ЕОМ і високопродуктивні засоби візуалізації.

Одним з найбільш ранніх прикладів використання когнітивної комп'ютерної графіки є робота Ч. Страуса "Несподіване застосування ЕОМ в чистій математиці". У ній показано, як для аналізу складних алгебраїчних кривих використовується "n-мірна" дошка на основі графічного терміналу. Користуючись пристроями введення, математик може легко отримувати геометричні зображення результатів спрямованої зміни параметрів досліджуваної залежності. Він може також легко управляти поточними значеннями параметрів, "поглиблюючи тим самим своє розуміння ролі варіацій цих параметрів". В результаті отримано ряд нових знань.

Галузі застосування комп’ютерної графіки

За галузями застосування можна виділити такі напрями застосування комп’ютерної графіки:

1. ***Наукова графіка*** - перші комп'ютери використовувалися лише для вирішення наукових і виробничих завдань. Щоб краще уявити отримані результати, проводилася їх графічна обробка, будувалися графіки, діаграми, креслення розрахованих конструкцій. Перші графіки на машині отримували в режимі символьної друку. Потім з'явилися спеціальні пристрої для креслення і створення графіків чорнильним пером на папері. Сучасна наукова комп'ютерна графіка дає можливість проводити обчислювальні експерименти з наочним поданням їх результатів.

2. ***Ділова графіка*** - область комп'ютерної графіки, призначена для наочного представлення різних показників роботи установ. Планові показники, звітна документація, статистичні зведення - для них за допомогою комп'ютерної графіки створюються ілюстративні матеріали. Програмні засоби ділової графіки включаються до складу електронних таблиць.

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

4. ***Ілюстративна графіка*** - малювання, креслення, моделювання на екрані комп'ютера. Пакети ілюстративної графіки відносяться до прикладного програмного забезпечення загального призначення. Програмні засоби ілюстративної графіки називаються графічними редакторами.

5. ***Художня і рекламна графіка*** - популярна багато в чому завдяки розвитку фотографії, реклами і телебачення. За допомогою комп'ютера створюються друковані матеріали, різного роду рекламна продукція, мультфільми, комп'ютерні ігри, інтерактивні і відеоуроки, слайдові і відеопрезентації. Крім графічних редакторів, для цих цілей використовуються графічні пакети, що вимагають великих ресурсів комп'ютера за швидкодією і пам'яті. Відмінною особливістю цих графічних пакетів є можливість створення реалістичних зображень і рухомих картинок. Отримання малюнків тривимірних об'єктів, їх повороти, наближення, видалення, деформації пов'язані з великим обсягом обчислень. Передача освітленості об'єкта в залежності від положення джерела світла, розташування тіней, фактури поверхні вимагає розрахунків, які враховують закони оптики.

Комп'ютерна анімація - створення рухомих зображень. Художник створює на екрані малюнки початкового і кінцевого положення рухомих об'єктів, всі проміжні етапи розраховує і зображує комп'ютер, виконуючи розрахунки, що спираються на математичний опис даного виду руху. Отримані малюнки, що виводяться послідовно на екран з певною частотою, створюють ілюзію руху.

6. ***Мультимедіа*** - об'єднання високоякісного зображення на екрані комп'ютера зі звуковим супроводом. Найбільшого поширення системи мультимедіа отримали в області навчання, реклами, розваг.

7. ***Наукова робота.*** Комп'ютерна графіка є також однією з областей наукової діяльності. В області комп'ютерної графіки захищаються дисертації, а також проводяться різні конференції.

4. Моделі графічних об’єктів

Графічний об’єкт – це певний малюнок, одержаний за допомогою комп’ютера. Сукупність графічних об’єктів називають зображенням. Графічні зображення часто мають ієрархічну структуру, яка виникає в результаті процесу побудови знизу-вверх. Простіші базові графічні елементи, з яких будуються графічні об’єкти довільної складності, називаються графічними примітивами. Вони використовуються як будівельні блоки для об’єктів більш високого рівня. Графічні примітиви розглядаються як прості неподільні елементи зображення і генеруються однією окремою командою (підпрограмою). Вони можуть мати також апаратний та апаратно-програмний рівень формування. Як правило, множина примітивів містить такі елементи:

- точку;
- відрізок;
- ламану лінію (замкнену і незамкнену);
- трикутник;
- трикутний стріп (рис. 16, а);
- трикутний віяло (рис. 16, б);
- чотирикутник;
- чотирикутний стріп (рис. 16, в);
- опуклий многокутник (полігон).

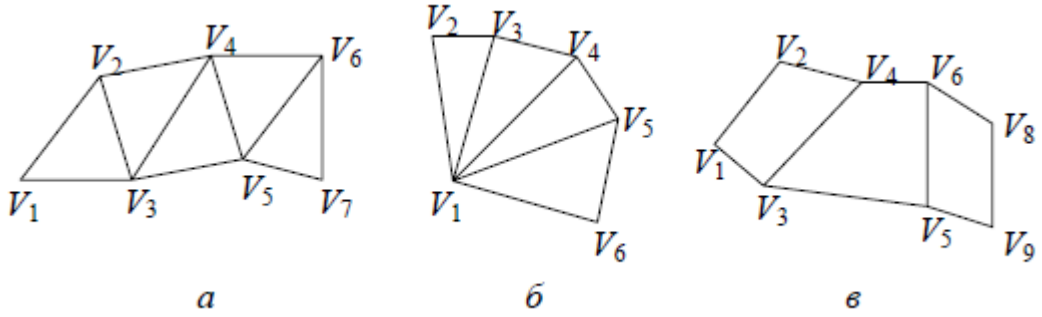


Рис. 16 - Елементи множини примітивів: а – трикутний стріп, б – віяло, в – чотирикутний стріп

Трикутний стріп – це множина трикутників T = {V_iV_i+1V_{i+2}}, що утворюється послідовністю вершин V_1,V_2,V_3,...,V_{n+2}. Перший трикутник стріпа $V_1V_2V_3$. Другий трикутник $V_2V_3V_4$ містить уже відоме ребро V_2V_3 і тому для його задання необхідно вказати тільки одну вершину і т.д. Тому стріп із 5 трикутників (рис. 16, а) задається послідовністю V_1,V_2,...,V_7 із семи вершин (замість п’ятнадцяти у випадку задання кожного трикутника окремо). Оскільки обробка кожної вершини в графічній системі займає певний час, то зменшення кількості обчислювальних вершин прискорює візуалізацію примітива.

Трикутний фен (віяло) – це множина трикутників T = {V1Vi+1Vi+2}, утворених послідовністю вершин V1,V2,...,Vn+2 (рис. 16, б), тобто трикутників, які задаються першою і кожною наступною парою вершин (пари не перетинаються). При заданні фену спочатку вказується загальна вершина, а потім решта вершин у порядку обходу спільних ребер.

Чотирикутний стріп – це множина чотирикутників Q = {V_{2i}-1V_{2i} V_{2i+2} $V2_{i+1}$}, утворених послідовністю вершин V_1,V_2,...,V_{2n+2}. Як видно з рис. 1.1, в, при заданні чотирикутного стріпа послідовність перерахування вершин інша, ніж при заданні незв'язаних чотирикутників.

Полігоном називається замкнена крива на площині, утворена відрізками прямих ліній. Відрізки прямих називаються ребрами або сторонами, кінцеві точки – вершинами. Дві вершини, що належать одному й тому ж ребру, називають суміжними. Відрізок прямої лінії, що лежить між двома несуміжними вершинами, називається діагоналлю.

Полігон називається простим, якщо він не має самоперетинів ребер. Простий полігон однозначно охоплює неперервну частину площини, яка є внутрішньою частиною полігона.
Для тривимірних додатків головною характеристикою полігона є не кількість вершин, а їх тип. Вершини полігона поділяються на опуклі і неопуклі (ввігнуті). Вершина називається опуклою, якщо внутрішній кут при вершині менше 180°, в протилежному випадку вершина неопукла. Нагадаємо, що область на площині називається опуклою, якщо для будь-яких двох точок області відрізок, що їх з'єднує, повністю лежить всередині цієї області. Будь-яка вершина в опуклому полігоні є опуклою (в неопуклому полігоні принаймні одна вершина неопукла). З цього випливає той факт, що при обході границі опуклого полігону в напрямку за годинниковою стрілкою в кожній вершині наступне ребро або продовжується по прямій лінії, або повертається вправо. В комп'ютерній графіці полігони особливо важливі як засоби задання поверхонь. В сучасних системах комп'ютерної графіки інформація про графічні об'єкти поділяється на дві категорії: параметри та атрибути. Примітиви теж характеризуються параметрами та атрибутами.

Параметри примітивів характеризують форму, розміри, місце розташування примітива.

Атрибути примітивів – це властивості, які визначають їх вигляд при візуалізації, тобто спосіб зображення заданого примітиву.

Для кожного типу об'єктів визначають свій набір атрибутів:

- для точки – розмір та колір;

- для лінії – колір, товщина і тип (лінія, яка використовується для малювання відрізків може бути суцільною або штриховою);

- для багатокутників – режим контуру чи заповнення. Цей параметр вказує спосіб задання багатокутників. Можна малювати тільки вершини або ребра, а можна зображати зафарбовані багатокутників, при цьому контур може малюватись одним кольором, а внутрішня область заливатись іншим.

У просторі простішими графічними примітивами є поліедри. Поліедри – це замкнені просторові об'єкти, що обмежені площинами. Серед об'ємних тіл поліедри займають таке ж важливе місце, як і полігони серед плоских фігур. Один полігон задає одну грань об'ємного об'єкта. Декілька граней у просторі складають об'ємний об'єкт у вигляді замкненої полігональної поверхні (багатогранника), або незамкненої поверхні (полігональної сітки).
Модель поліедра H={P, G} складається з двох масивів:

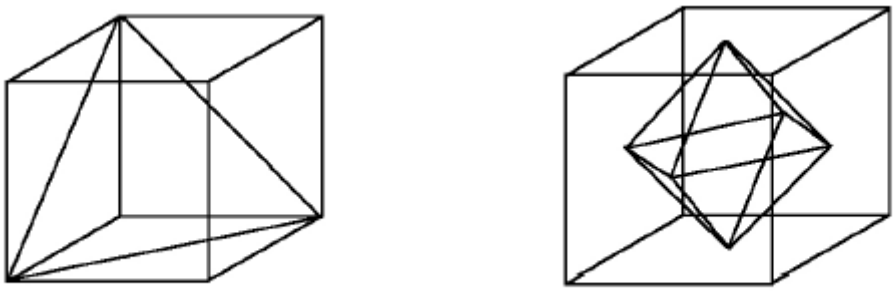
- координатного списку вершин P={P_1,…, P_n} пронумерованих у довільному порядку;

- масиву списків граней G={G_1,…, G_m}. Елемент G_i={g_{i1},..., g_{ini} , g_{i1}} масиву G є списком номерів вершин полігона і-тої грані, перерахованих у порядку обходу їх по замкненому контуру.

Полігональні моделі найбільш широко розповсюджені в сучасних системах тривимірної графіки, хоча вони мають і ряд недоліків, наприклад, приводять до складних алгоритмів візуалізації реалістичних зображень. Завдяки планарності всіх граничних поверхонь поліедра істотно спрощується розрахунок його перетинів із різними геометричними примітивами. Крім цього, часто криволінійні граничні поверхні об’єктів апроксимуються системою полігональних граней, що перетворює ці об'єкти на поліедри. У природі існує цілий ряд поліедральних об'єктів. Серед них правильні многогранники, призми, піраміди тощо.

Правильні многогранники (платонові тіла) – це опуклі многогранники, всі грані яких є правильними многокутниками і всі многогранні кути при вершинах рівні між собою. Існує 5 правильних многогранників (це довів ще Евклід): тетраедр, гексаедр (куб), октаедр, додекаедр, ікосаедр. Префікс перед грецьким коренем 'едр' означає кількість граней поліедра.

Для повного опису правильних многогранників унаслідок їхньої опуклості достатньо вказати спосіб знаходження всіх його вершин. Розглянемо деякі способи їх побудови. Вершинами тетраедра є чотири вершини куба, попарно несуміжні з жодним із його ребер (рис. 17, а). Вершини октаедра знаходяться в центрах граней куба (рис. 17, б).



а *б*

Рис. 16 - Приклади простіших поліедрів: а – тетраедр, б – октаедр

Опишемо спосіб конструювання ікосаедра. Побудуємо циліндр одиничного радіуса, вісь якого збігається з віссю аплікат z, а основи лежать у площинах z = 0,5 і z = – 0,5. Розбиваємо кожне з кіл, що лежать в основах циліндра, на 5 рівних частин, як це зображено на рис. 18.

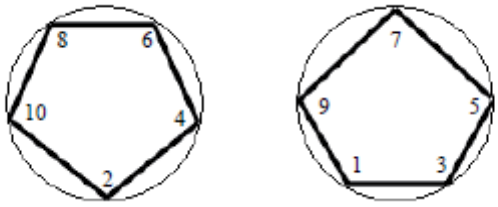


Рис. 18. - Послідовність з’єднання вершин

Далі нумеруємо ці точки проти годинникової стрілки парними та непарними числами і потім послідовно, відповідно до нумерації, з'єднуємо ці точки відповідними відрізками (рис. 19). Одержуємо десять середніх трикутних граней ікосаедра.

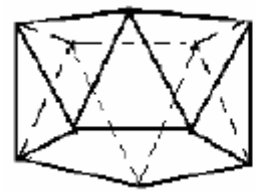
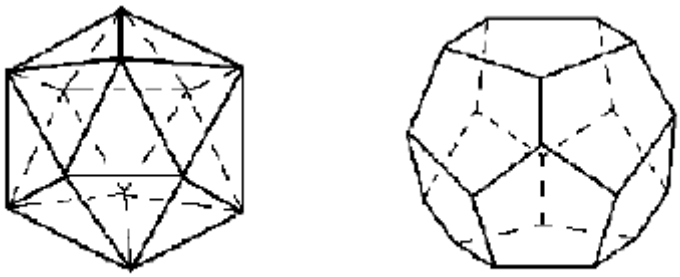


Рис. 19. - Середні грані ікосаедра

Для завершення побудови ікосаедра вибираємо на осі z дві точки так, щоб довжини бокових ребер у п'ятикутних пірамідах із вершинами в цих точках і основами, що співпадають з побудованими п'ятикутниками дорівнювали довжинам сторін побудованих правильних трикутників. Такими точками є точки з аплікатами $\pm \sqrt{5}/2$. У результаті описаних побудов одержуємо 12 точок (вершин), а опуклий многогранник із вершинами в цих точках буде мати 20 граней, кожна з яких є правильним трикутником (рис. 20, а). Декартові координати вершин побудованого ікосаедра легко обчислюються.



а *б*

Рис. 20 – Складніші поліедри: а – ікосаедр, б – додекаедр

Залишається побудувати додекаедр (кількість граней – 12) – найбільш складне платонове тіло. Вершини додекаедра є центрами трикутних граней ікосаедра. Отже, координати кожної вершини додекаедра можна знайти шляхом обчислення середнього арифметичного координат вершин ікосаедра. Далі необхідно з'єднати центри суміжних граней ребрами (рис. 21, б). Множина видимих на екрані об'єктів, упорядкованих згідно з просторовими відношеннями (наприклад, за відношенням дальше – ближче, лівіше – правіше), разом з атрибутами елементів поверхонь (колір, текстура, оптичні властивості матеріалів) і з атрибутами оточення (рівень освітлення, туман, димка тощо) утворюють сцену. Графічні об'єкти, з яких складається сцена, називаються елементами сцени. Для повного опису сцени необхідна, як мінімум, така інформація:

- морфологічна інформація – це зведення про форму, структуру кожного елемента незалежно від його розміщення та розмірів і місцезнаходження спостерігача. Методи моделювання змін форми об'єкта, тобто поступового перетворення одного образа в інший (анімація форми) називаються морфінгом (від грецьк. *morphe* – форма). Зазвичай перетворення форми об'єкта визначається за допомогою множини опорних точок і подальшої інтерполяції;

- геометрична інформація, яка включає характеристики розмірів елементів сцени, параметри взаємного розміщення елементів (віддаль між ними, кути), розміщення елементів відносно спостерігача і т.д.;

- інформація про джерела світла, про їх розміщення, про властивості матеріалу об'єктів.

Математичні моделі об'єктів графічних сцен

Для систем машинної графіки джерелом вхідної інформації є не самі фізичні процеси та об’єкти, а математичні моделі. Математична модель – це сукупність математичних співвідношень, за допомогою яких описуються основні характеристики реального об’єкта.

При побудові об’єктів і сцен різної природи у комп’ютерній графіці найчастіше використовують геометричну версію математичного моделювання, тобто кожний елемент сцени розглядається як геометричний об'єкт із певними властивостями, а сцена складається з множини геометричних об'єктів.

За способом представлення геометричних об'єктів можна виділити дві найчастіше використовувані моделі елементів графічних сцен:

- графічні примітиви;
- математичні моделі просторових об'єктів, які задаються рівняннями ліній і поверхонь.

У першій моделі вважається, що сцена формується з відомих геометричних фігур (примітивів): двовимірних (ліній, многокутників тощо) і тривимірних (кубів, пірамід, куль, циліндрів тощо). Примітиви розглядаються як прості неподільні елементи зображення. Тому програмна реалізація побудови сцени полягає в підборі відповідних фігур, які вибираються з бібліотеки примітивів. При використанні бібліотеки примітивів маємо справу з двома рівнями примітивів: двовимірними та тривимірними. Щоб побудувати елемент сцени, який відсутній в бібліотеці примітивів, потрібно сконструювати його з тих примітивів, які наявні в бібліотеці, використовуючи різні операції та композиції елементів. Це добре відомі операції, які застосовуються до суцільних тіл: повороти різного роду, зсув, перетворення подібності, симетричне відображення, розтяг, сегментація об'єкта (виділення тієї частини об'єкта, над якою буде проводитися перетворення), об'єднання (сума) об'єктів – $P_{r1} U P_{r2}$ (графічний об'єкт, кожна точка якого належить хоча б одному з примітивів), перетин (добуток) примітивів $P_{r1} \cap P_{r2}$ (тіло, кожна точка якого належить одночасно двом примітивам), різниця двох примітивів $P_{r1} \backslash P_{r2}$ (тіло, яке складається з точок P_{r1}, що не належать до P_{r2}).

У другому типі моделей геометричні елементи сцени розглядаються як функціональні залежності яскравості точок сцени від координат цих точок, тобто сцену можна побудувати за допомогою аналітичних рівнянь, які описують лінії та поверхні, що задають тіла сцени. Аналітичні моделі широко застосовуються при проектуванні геометричних об’єктів складної форми, при підготовці програм управління для станків з ЧПУ, при розкроюванні матеріалу тощо. Наведемо рівняння деяких ліній і поверхонь. Пряма лінія на площині задається неявним рівнянням $ax + by + c = 0$, або явним рівнянням $y = kx + b$. Відрізок, що визначається двома точками r0, r1, задається векторно-параметричним рівнянням вигляду $r = r_0(1 - t) + r_{1t}$, де $r=(x, y)$, $t \in [0, 1]$.

Загальна явна форма задання кривої на площині має вигляд $y = f(x)$, неявна форма – $f(x, y) = 0$. Явний вигляд кривої $y = f(x)$ має ряд недоліків. По-перше, не можна описати замкнені криві одним виразом. Такі криві необхідно розбивати на ряд сегментів і шукати аналітичне задання для кожного з них. По-друге, такий опис кривих не володіє інваріантністю відносно перетворення повороту. Щоб задати повернуту криву, необхідно виконати певну обчислювальну роботу. По-третє, при обчисленнях значень $f(x)$ виникають істотні обчислювальні складності, наприклад для кривих із великими кутами нахилу. Задання кривої

в неявній формі f(x, y) = 0 теж викликає ряд проблем. Параметрична форма задання кривої x = x(t), y = y(t), t∈[α, β] усуває недоліки функціонального і неявного способів задання кривої.

Параметрична форма рівнянь є найбільш універсальною і найбільш розповсюдженою для опису і побудови геометричних об'єктів завдяки різноманітності вибору функцій, що задають рух точки вздовж напрямів степенів вільності. Параметрична форма дозволяє задати замкнені й багатозначні криві. Поверхні можна задати явним рівнянням z = f(x, y), або неявним – F(x, y, z) = 0, чи в параметричній формі: x = x(τ, t), y = y(τ, t), z = z(τ, t), (τ, t)∈D. Поверхні першого порядку (площини), що задаються рівнянням ax+by+cz = d, та поверхні другого порядку, наприклад еліпсоїд x2/a2+ y2/b2 + z2/c2 = 1 – найбільш широко використовувані поверхні в КГ, оскільки площина є основою наближення поверхні довільної форми, а квадратичні поверхні в багатьох випадках добре передають візуальні образи криволінійних поверхонь.

Зазначимо, що серед поверхонь 2-го порядку тільки еліпсоїд (сфера як частковий випадок) локалізований у просторі, всі інші простягаються в нескінченність, тому вимагають для свого задання обмежуючих площин. Параметричне рівняння сфери має вигляд
x(τ, t) = Rsint, y(τ, t) = Rcost sinτ, z(τ, t) = Rcost cosτ, t∈ [0, π] , τ∈[0, 2π].

Для побудови моделі сфери використовують побудований раніше ікосаедр. Зауважимо, що ікосаедр уже є моделлю сфери (всі вершини ікосаедра лежать на сфері), єдиний недолік – мала кількість граней для зображення поверхні сфери. Для підвищення реальності зображення сфери діють так:

− кожна грань ікосаедра розбивається на чотири частини (трикутники). Додаткові вершини нових трикутників беруться на серединах сторін грані, як це показано на рис 21;

− нові вершини проектуються на поверхню сфери. Для цього з центрасфери через вершини проводяться промені і відшукуються точки перетину променів із поверхнею сфери. На цих точках будуються нові грані які краще апроксимують сферу;

− указані дії повторюються доти, поки не буде досягнуто задовільного зображення сфери. Аналогічним чином можна побудувати моделі циліндра, тора тощо.

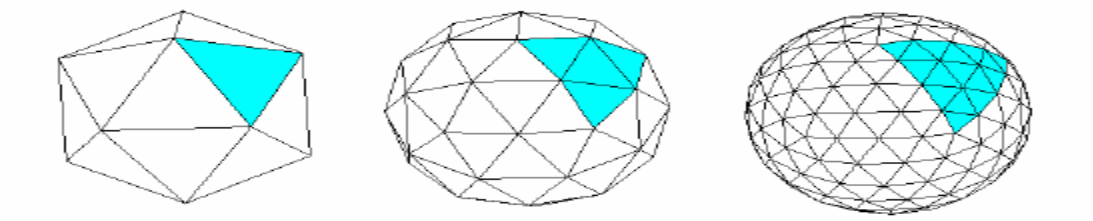


Рис 21. – Побудова зображення сфери

Фрагмент білінійної поверхні, розташованої між чотирма точками P_0,P_1,P_2,P_3, описується векторно-параметричним рівнянням $r = P_0(1 – u)(1 – v)+P_{1u}(1 – v)+P_2(1– u)v+P_{3uv}$, r=(x, y, z), u,v∈ [0,1]. За допомогою білінійних поверхонь описують складні 3D-поверхні. Лінійчаста поверхня загального вигляду, що утворюється внаслідок руху прямолінійної твірної вздовж двох напрямних ліній $r_1(v)$ та $r_2(v)$ задається рівнянням $r=r_1(v)(1 – u)+r_2(v)u$. Лінія в просторі є або перетином двох поверхонь, або слідом руху деякої точки, тому маємо дві форми моделі просторової кривої: неявна форма $\{F_1(p) = 0\}\cap\{F_2(p) = 0\}$, параметрична форма x = x(t), y = y(t), z = z(t)).

Прикладом параметричної просторової кривої є циліндрична гвинтова лінія (рис. 22), що задається рівнянням x(t) = rcost, y(t) = rsint, z(t) = ht, r,h > 0, t∈ [0, ∞). Ця крива лежить на поверхні циліндра радіусом r. При зміні параметра t на 2π змінні x та y повертаються до свого попереднього значення, а z збільшується на 2πh. Ця величина називається кроком спіралі.

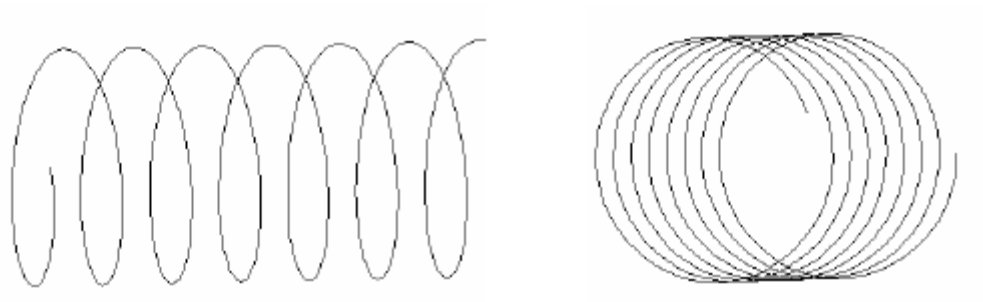


Рис. 22 – Циліндрична гвинтова лінія

Іншим прикладом параметричних кривих є циклічні криві. Простіші циклічні криві на площині утворюються додаванням відносного руху точки по колу з переносним рухом центра кола по кривій рц(t) і описуються рівнянням p(t)= $p_ц$(t) + r_c(wt), де p = (x, y), c(wt) = (sinwt, coswt). Випишемо рівняння деяких циклічних кривих на площині. Циклоїда – це траєкторія руху точки кола радіусом r, яке котиться по прямій y = 0 (рис. 23). Координати (рівняння) циклоїди мають вигляд x(t)= r(t – sint), y(t)= r(1 – cost), t ≥ 0. Епіциклоїда – це траєкторія руху точки кола радіусом r, якщо воно котиться зовні кола радіусом r_0 (рис. 24, а). Її параметричне рівняння: p(t) = (r0+r)(sint, cost) – r(sinwt, coswt), w = (r_0/r+1). Гіпоциклоїда – це траєкторія руху точки кола радіусом r, яке котиться всередині кола радіусом r_0 (рис. 24, б). Рівняння гіпоциклоїди має вигляд:
p(t) = (r_0 – r)(sint, cost)+r(–sinwt, coswt), w = (r_0/r – 1).

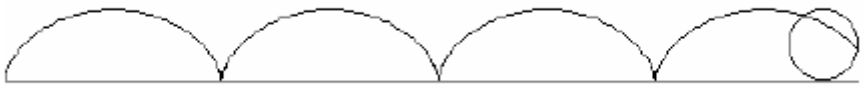


Рис. 23. – Циклоїда

Рис. 24 – Циклічні криві: а – епіциклоїда, б – гіпоциклоїда

Контрольні запитання

1. Дайте визначення понять «графіка» та «комп'ютерна графіка».
2. Співробітники якої компанії першими дали визначення поняття «Комп'ютерна графіка»?
3. Сутність поняття цифрове зображення.
4. Які завдання вирішує комп'ютерна графіка.
5. В чому полягає особливість інтерактивної комп'ютерної графіки?
6. Співробітниками якої компанії було винайдено електронно-променеву трубку?
7.Яка компанія розробила перший комерційний графічний термінал?
8. Яка програма вважається такою, що започаткувала принцип взаємодії людини з комп'ютером?
9. Які основні напрями застосування комп'ютерної графіки?
10. Які існують види комп'ютерної графіки?
11. Якою є сфера застосування когнітивної комп'ютерної графіки?
12. Що таке аналіз сцен?
13. Поняття графічного об'єкту.
14. В чому полягає відмінність трикутного стріпу від фену?
15. Яка інформація потрібна для опису сцени?
16. В чому полягає відмінність між графічними примітивами та математичними моделями просторових об'єктів?
17. В чому полягає перевага використання стріпів?
18. Що таке ікосаедр?
19. Яка з поверхонь другого порядку локалізована у просторі?
20. Що таке полігон?

Список використаних інформаційних джерел

1. https://www.youtube.com/playlist?list=PLOQZmjD6P2HlOoEVKOPaCFvLnjP865X1f - A Brief History of Graphics
2. Маценко В.Г. Комп'ютерна графіка: Навчальний посібник. – Чернівці: Рута, 2009 – 343 с.
3. https://blog.ted.com/the-math-behind-the-movies-an-interview-with-tony-derose-of-pixar
4. https://www.youtube.com/watch?v=ay8OMOsf6AQ&feature=youtu.be – виступ Бенуа Мальдеброт на конференції TED.
5. https://www.youtube.com/watch?v=l7W75UfAjRI – робота комп'ютера DAC-1.

ЛЕКЦІЯ 3.

ТЕМА: КОЛІРНІ МОДЕЛІ

План

1. Колірні моделі.
2. Адитивна колірна модель RGB.
3. Субтрактивна колірна модель CMYK.
4. Перцепційні колірні моделі.
5. Колірна модель та колірний простір.
6. Системи відповідності кольорів.
7. Кодування кольору. Палітра кольорів.

1. Колірні моделі.

Говорячи про колір, ми можемо сказати, що це в значній мірі суб'єктивний атрибут сприйняття. Проте при обробці зображень суб'єктивність є небажаною. Відповідно, постає потреба у розробці та використанні вимірювальних систем, які дозволять забезпечити однакове відтворення одного й того ж кольору моніторами, принтерами, камерами та іншими пристроями. Для цього існують спеціальні засоби, такі як:

- колірні моделі;
- системи відповідності кольорів;
- колірні режими.

Для опису кольору розроблені різноманітні колірні моделі. В їх основу покладено ідею використання універсальних мов, які дозволяють реалізувати способи точного опису кольору за допомогою стандартних математичних виразів. Без їх допомоги було би неможливо обробляти графічні зображення.

У сучасних комп'ютерних програмах маніпуляції з кольором відбуваються з використанням кольорових моделей та режимів.

Колірні моделі являють собою засоби для концептуального та кількісного представлення кольору. У свою чергу режим являє собою спосіб реалізації визначеної кольорової моделі у рамках конкретної графічної програми.

Колірні моделі (color model) використовуються для математичного опису визначених колірних областей спектра. Більшість комп'ютерних колірних моделей засновано на використанні трьох основних кольорів (tristimulus values), що відповідає сприйняттю кольору людським оком. Кожному основному кольору присвоюється певне значення цифрового коду, після чого всі інші кольори визначаються як комбінації основних кольорів. Такий підхід використовують художники при роботі над картиною на основі обмеженої палітри кольорів.

Не дивлячись на те, що колірні моделі представляють колір математично, таке представлення завжди буде здаватися неідеальним через відмінність від нашого сприйняття. Проте вони зручні при використанні у комп'ютерних програмах для однозначного визначення кольору.

Так, якщо надіслати на монітор сигнал R210 G154 B123, то на будь-якому правильно відкаліброваному пристрої має з'явитися один і той самий колір. Будь-яка колірна модель має відповідати трьом вимогам:

1. Реалізовувати визначення кольору деяким стандартним апаратно-незалежним способом;
2. Точно задавати діапазон відтворюваних кольорів, оскільки множина кольорів не є безкінечною;
3. Враховувати механізм сприйняття кольорів, тобто випромінене або відбите світло.

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

1. адитивні (RGB), які базуються на додаванні кольорів;
2. субтрактивні (CMYK), основу яких складає операція віднімання кольорів;
3. перцепційні (HSB, HLS), які базуються на сприйнятті;
4. універсальні (Lab, XYZ), тобто такі, що охоплюють весь спектр кольорів, які сприймає людина.

Перед тим, як перейти до розгляду конкретних колірних моделей, розглянемо способи виміру та опису кольору. В більшості колірних моделей використовується тривимірна система координат. Вона утворює кольоровий простір, в якому колір можна представити у вигляді точки з трьома координатами. Для оперування кольором у тривимірному просторі Г. Грасман вивів три закони, які становлять базу наукового аналізу синтезу кольору.

Тривимірність природи кольору (закон тривимірності). Око реагує на три різні кольорові складові.

Приклади:

- червоний, зелений і синій кольори;
- колірний тон (домінуюча довжина хвилі), насиченість (чистота) і яскравість (світлість).

Відповідно до цього закону будь-який колір (при дотриманні певних умов перегляду) можна представити у вигляді суміші (суми) певних кількостей трьох лінійно незалежних (базових) кольорів, тобто таких кольорів, кожен з яких не може бути представлений у вигляді суми будь-яких кількостей двох інших кольорів.

$$X \equiv \alpha A + \beta B + \gamma C,$$

Де *A*, *B* та *C* – лінійно незалежні базові кольори (наприклад, R,G та B). α, β та γ – вагові коефіцієнти (пропорції) відповідних базових кольорів.

У даному виразі замість знака рівності використовується знак тотожності, що відображає той факт, що ліва і права частини цього виразу еквівалентні тільки з точки зору сприйняття людини.

Закон адитивності. Колір суміші випромінювань залежить тільки від психофізичних (колометричних) характеристик складових кольорів, але не від їх спектрального складу.

Якщо

$$X_1 \equiv \alpha_1 A + \beta_1 B + \gamma_1 C;$$
$$X_2 \equiv \alpha_2 A + \beta_2 B + \gamma_2 C,$$

то

$$X_1 + X_2 \equiv (\alpha_1 + \alpha_2)A + (\beta_1 + \beta_2)B + (\gamma_1 + \gamma_2)C.$$

Слідство 1: якщо колір X_1 дорівнює кольору X та колір X_2 теж дорівнює кольору X, то слід, що колір X_1 дорівнює кольору X_2 незалежно від структури спектрів енергії кольорів X, X_1 і X_2.

Слідство 2: чотири кольори завжди лінійно залежні.

Прикладом вищесказаного слугує чотирьохкомпонентна модель CMYK. У ній одне і те ж відчуття кольору може бути досягнуто за допомогою різних значень вихідних компонентів: Cyan (Блакитний), Mangenta (Пурпурний), Yellow (Жовтий) і Black (Чорний). Так, друкований колір, визначений як 50 часткою блакитного, 50 жовтого, 75 пурпурового та 0 чорного, відповідає друкованому кольором з 50 часткою чорного, 25 пурпурового, 0 блакитного і 0 жовтого. Це можливо, оскільки чорну фарбу теоретично і практично можна розглядати як суміш трьох кольорових фарб.

Закон безперервності (колірний простір безперервно). Не існує такого кольору, до якого неможливо було б підібрати нескінченно близький. Іншими словами, якщо в суміші трьох кольорів один безперервно змінюється, а другий залишається постійним, то колір суміші буде змінюватися безперервно.

Розглянувши основні положення, що стосуються колірних моделей, перейдемо до розгляду колірної моделі RGB, що знайшла широке застосування у сфері оброблення та відображення графіки.

2. Адитивна колірна модель RGB.

Адитивний колір отримується на основі законів Г. Грассмана шляхом з'єднання променів світла, що мають різний тон. В основі цього явища покладено факт, відповідно до якого більшість кольорів видимого спектру можуть бути отримані шляхом змішування в різних пропорціях трьох основних колірних компонент. Цими компонентами (стимулами), які в теорії кольору іноді називаються первинними кольорами, є червоний (Red), зелений (Green) і синій (Blue) кольори. При попарному змішуванні первинних кольорів утворюються вторинні кольори: блакитний (Cyan), пурпурний (Magenta) і жовтий (Yellow). Слід зазначити, що первинні і вторинні кольори відносяться до базових кольорів.

Базовими кольорами називають кольори, за допомогою яких можна отримати практично весь спектр видимих кольорів. Для отримання нових кольорів за допомогою адитивного синтезу можна виконувати також різні комбінації з двох основних кольорів,

варіювання складу яких призводить до зміни результуючого кольору. На рис. 1 приведена схема отримання нових кольорів на базі двох первинних шляхом використання джерел зеленого і червоного кольорів, інтенсивністю кожного з яких можна керувати за допомогою фільтра. Можна побачити, що рівні пропорції первинних кольорів дають жовтий колір (рис. 2, вгорі); зниження в суміші інтенсивності зеленого кольору при тій же інтенсивності червоного дозволяє синтезувати помаранчевий колір (рис. 2, внизу). Подібні колометричні схеми дозволяють створити жовтий і оранжевий кольори у вигляді геометричного місця колірних точок - локусу. Проте у такий спосіб не можна отримати деякі кольори, наприклад блакитний, для створення якого потрібна наявність третього первинного кольору - синього (рис. 3).

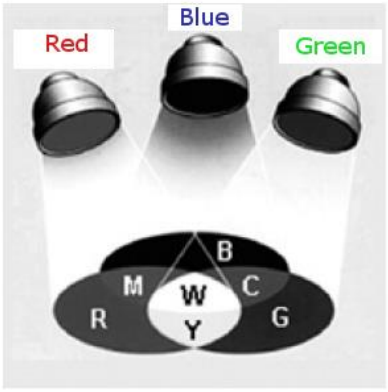


Рис. 1 – Принцип дії адитивної колірної моделі RGB. Шляхом проекції червоного, синього і зеленого кольорів на світлу поверхню можна отримати більшість кольорів видимої області спектра. При одночасному змішуванні трьох чистих кольорів виходить білий колір

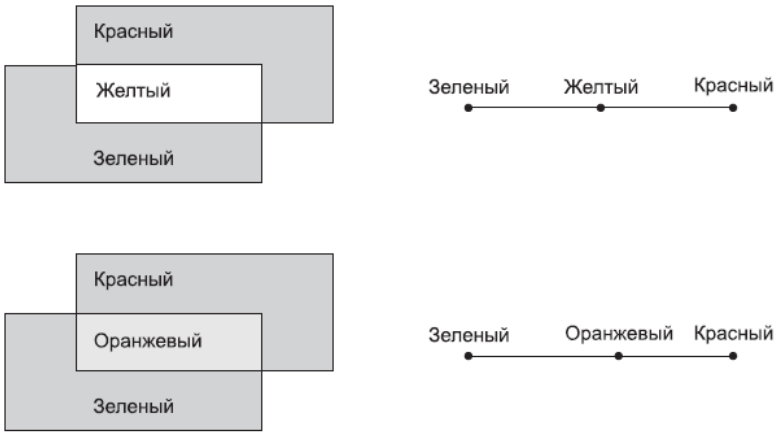


Рис. 2 – Адитивний синтез нових кольорів на базі різного процентного співвідношення двох первинних кольорів: червоного і зеленого

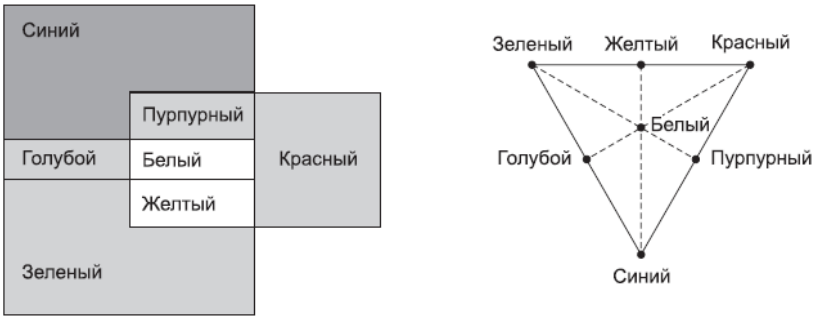


Рис. 3 –Колірна (зліва) і колориметрична (праворуч) схеми отримання колірного простору RGB-моделі за допомогою трьох первинних кольорів. Сторони трикутника утворюють безліч спектрально-чистих кольорів (локус)

Отримання нового кольору шляхом адитивного змішування декількох первинних кольорів визначає можливість отримання кольорового зображення у фотографії, кіно, телебаченні, поліграфії тощо. Використовувані для побудови RGB-моделі первинні, або адитивні, кольори мають ще одну назву: іноді, щоб підкреслити той факт, що при додаванні світла інтенсивність кольору збільшується, цю модель називають такою, що додає. Така значна кількість термінів, які використовуються для опису RGB-моделі, пов'язана з тим, що вона виникла задовго до появи комп'ютера і в кожній області її застосування виникали свої терміни.

Оскільки в моделі RGB використовуються три незалежних значення, її можна представити у вигляді тримірної системи координат (рис. 4).

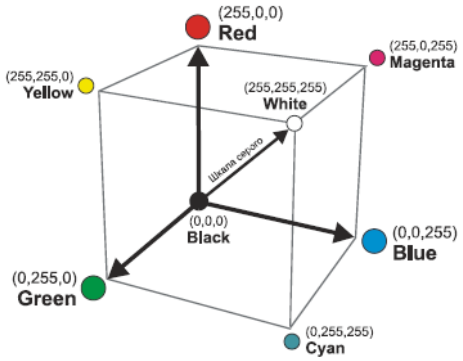


Рис. 4 – Графічне представлення моделі RGB

Кожна координата відображає внесок відповідної складової в конкретний колір в діапазоні від нуля до максимального значення. В результаті виходить куб, всередині якого і знаходяться всі кольори, утворюючи колірний простір моделі RGB.

Обсяг такого куба (кількість цифрових кольорів) легко порахувати: оскільки на кожній осі можна відкласти 256 значень, то 256 в кубі (або 2 в двадцять четвертій ступені) дає число 16 777 216. Важливо відзначити особливі точки і лінії цієї моделі.

Початок координат. У цій точці всі складові дорівнюють нулю, випромінювання відсутня, а це рівносильне темряві, отже, початок координат - це точка чорного кольору.

Точка, найближча до глядача. У цій точці всі складові мають максимальне значення, що означає білий колір.

Діагональ куба. На лінії, що з'єднує початок координат і точку, найближчу до глядача, розташовуються сірі відтінки: від чорного до білого. Це відбувається тому, що всі три складових однакові та розташовуються в діапазоні від нуля до максимального значення. Цей діапазон інакше називають сірою шкалою (Grayscale). У комп'ютерних технологіях зараз найчастіше використовуються 256 градацій (відтінків) сірого. Хоча деякі сканери мають можливість кодувати 1024 відтінків сірого і вище.

Як було зазначено вище, **математично колірну модель RGB найзручніше представляти** у вигляді куба. У цьому випадку кожна його просторова точка однозначно визначена значеннями координат X, Y і Z. Якщо по осі X відкладати червону складову, по осі Y - зелену, а по осі Z - синю, то кожному кольору можна поставити у відповідність точку всередині куба. При використанні цієї моделі будь-який колір може бути представлений в колірному просторі за допомогою вектора, описуваного рівнянням: C = rR + gG + bB

Дане рівняння ідентично рівнянню вільного вектора в просторі, яке розглядається в векторній алгебрі. При цьому напрямок вектора характеризує кольоровість, а його модуль висловлює яскравість.

На діагоналях (ахроматичної осі), що з'єднує точки з координатами (R, G, B) = (0, 0, 0) і (R, G, B) = (255, 255, 255), розташовані різні градації сірого, для яких значення червоного, зеленого і синього складових однакові.

Три вершини куба позначають чисті первинні кольори, інші три відображають подвійні змішування вихідних кольорів.

Насправді замість використання в якості первинних червоного, зеленого і синього кольорів можна взяти будь-які інші лінійно незалежні кольори. Просто шляхом змішування червоного, зеленого і синього можна отримати найбільшу **комбінацію кольорів**. Очевидним поясненням цього факту є особливість людського зору, пов'язана з наявністю в зоровому апараті людини трьох рецепторів, кожний з яких чуттєвим до червоних, зелених і синіх променів. Таким чином, утворення кольору за допомогою трьох випромінювачів синього, зеленого і червоного кольорів можна розглядати як спрямоване збудження трьох колірних рецепторів ока, в результаті чого виходить можливість викликати у глядача відчуття того або іншого кольору. Також обрані для використання у кольоровій моделі кольори залежать від технологічного процесу виготовлення елементів пристроїв виведення, що пов’язане зі складністю випромінення кольорів певної довжини.

Не дивлячись на переваги та широке розповсюдження кольорової моделі RGB, при її використанні на практиці виникають деякі складнощі:

- апаратна залежність, яка призводить до того, що на різних пристроях один колір може бути незначно відмінним;
- використання моделей, заснованих на різних системах передачі кольору, вимагає постійного перетворення кольору з однієї системи в іншу так як в RGB, наприклад,

працює монітор, а в CMYK - принтер. Таке перетворення загрожує втратою частини відтінків, а це в свою чергу погіршує якість ілюстрацій.

Пригадаємо кілька особливостей щодо відтворення зображення на екрані монітора. Так, зображення формується з сукупності пікселів. **Піксель** — найдрібніша одиниця цифрового зображення в растровій графіці. Він являє собою неподільний об'єкт прямокутної або круглої форми, що має певний колір. Будь-яке растрове комп'ютерне зображення складається з пікселів, розташованих по рядках і стовпцях. Якщо зображення збільшити, ви побачите ряди пікселів.

Для задання кольору та яскравості точок, які формують зображення монітора, необхідно задати значення інтенсивностей для кожної зі складових RGB-елемента (пікселя). Технологія реалізації цього з точки зору технологічної бази полягає в управлінні ступенем проникності елемента монітора перед світловим фільтром відповідного кольору.

У той же час число градацій інтенсивності визначає кольорову роздільність, або інакше кажучи, глибину кольору, яка характеризує максимальну кількість відтворюваних кольорів.

Також слід сказати, що кольорові зображення в моделі RGB будуються за допомогою трьох окремих зображень-каналів. На рис. 5 показано розкладання вихідного зображення на колірні канали.

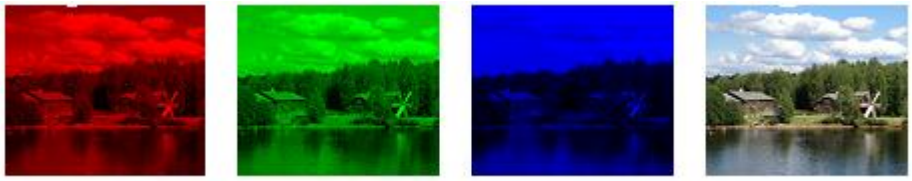


Рис. 5 – Розкладання зображення на кольорові канали

У моделі RGB для кожної складової кольору відводиться певна кількість біт, наприклад, якщо для кодування кожної складової відводити 1 байт, то за допомогою цієї моделі можна закодувати 2 ^ (3 * 8) ≈ 16 млн. кольорів. На практиці таке кодування надлишково, тому що більшість людей не здатне розрізнити таку кількість кольорів. Часто обмежуються так званим режимом «High Color» в якому на кодування кожної компоненти відводиться 5 біт. У деяких додатках використовують 16-розрядний режим в якому на кодування R і B складових відводиться по 5 біт, а на кодування G складової 6 біт. Цей режим, по-перше, враховує більш високу чутливість людини до зеленого кольору, а по-друге, дозволяє більш ефективно використовувати особливості архітектури ЕОМ. Кількість біт, що відводяться на кодування одного пікселя називається глибиною кольору. На рис. 6 наведені приклади кодування одного і того ж зображення з різною глибиною кольору.

Рис. 6 – Зображення з різною глибиною кольору, біт

З огляду на вищесказане, головний недолік RGB-моделі полягає в її розмитості. Це обумовлено тим, що на практиці RGB-модель характеризує колірний простір конкретного пристрою, наприклад монітора або сканера. Необхідний певний спільний знаменник. Проте будь-який RGB-простір можна зробити стандартним. Для цього необхідно всього лише однозначно визначити його. Наприклад, в графічних пакетах пропонується більш десяти заздалегідь визначених варіантів, важливе місце серед яких займає стандартне колірний простір для Інтернету - **sRGB** (standard RGB). З ініціативи двох фірм – Microsoft і HP - воно стандартизовано і широко використовується. Adobe Systems включила його до складу своїх продукції.

Фактично, модель sRGB – це абстрактний ідеальний стандартний монітор. Сьогодні цей простір є альтернативою системам управління кольором.

Говорячи про теорію кольору, можна почути такі терміни, як трикутник колірності, діаграма колірності, локус. Нижче розглянемо дані поняття.

Почнемо розгляд цих понять з принципу утворення площини одиничних кольорів. Площина одиничних кольорів (Q) (рис. 7) проходить через відкладені на осях координат яскравості поодинокі значення обраних основних кольорів. Одиничним кольором в колориметрії називають колір, сума координат якого дорівнює 1.

Тому можна вважати, що площину Q, яка перетинає осі координат в точках Br (R = 1, G = 0, B = 0), Bg (R = 0, G = 1, B = 0) і Bb (R = 0, G = 0, B = 1), є одиничним місцем точок в просторі RGB (рис. 7).

Кожній точці площини одиничних кольорів (Q) відповідає слід колірного вектора, що пронизує площину у відповідній точці на відстані від центру координат:

$$m = r + g + b = 1.$$

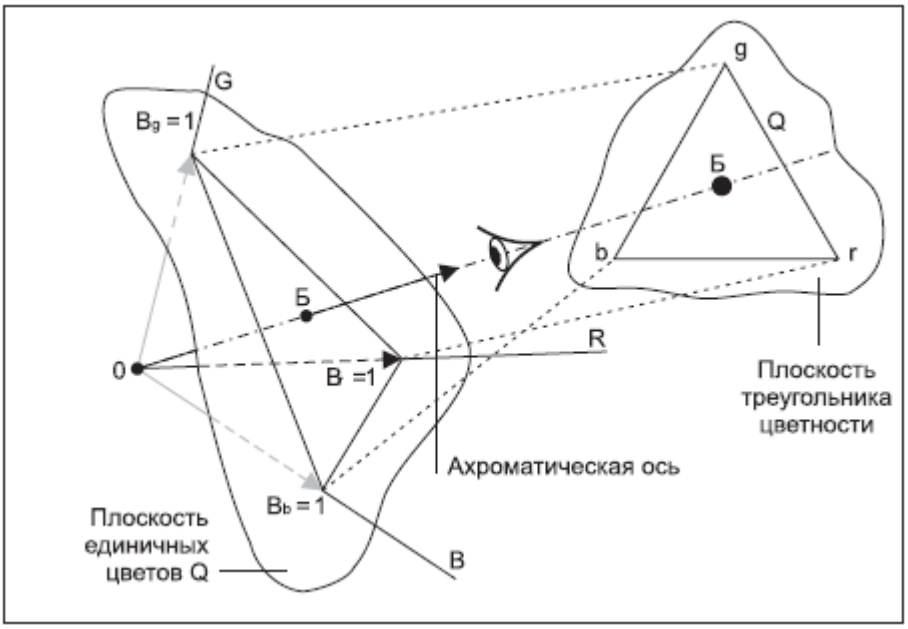


Рис. 7 – Площина одиничних кольорів та утворення діаграми колірності

Отже, кольоровість будь-якого випромінювання може бути представлена на площині єдиною точкою. Можна собі уявити і точку, відповідну білому кольору (Б). Вона утворюється шляхом перетину ахроматичної осі з площини Q (рис. 7).

У вершинах трикутника знаходяться точки основних кольорів. Визначення точок кольорів, одержуваних змішанням будь-яких трьох основних, проводиться за правилом графічного складання, а тому цей трикутник називається трикутником колірності, або діаграмою колірності. Часто в літературі зустрічається інша назва - локус, яку можна інтерпретувати як геометричну область всіх кольорів, відтворених даним пристроєм.

В колориметрії для опису кольору немає необхідності вдаватися до просторових представлень. Досить використовувати площину трикутника колірності (рис. 7). У ньому положення точки будь-якого кольору може бути поставлено лише двома координатами (наприклад, r і g). Третю (в даному випадку b) легко знайти по двом іншим, оскільки сума координат кольоровості завжди дорівнює 1. Тому будь-яка пара координат кольоровості може слугувати координатами точки в прямокутній системі координат на площині (більш докладно це буде розглянуто в розділі 3). Отже, ми з'ясували, що колір графічно можна виразити у вигляді вектора в просторі або у вигляді точки, що лежить всередині трикутника кольоровості.

Вище ми розглянули адитивну колірну модель RGB, що є популярною у сфері обробки растрових зображень. Проте для того, щоб створити комплексне уявлення про обробку графічної інформації, нам слід розглянути також інші колірні моделі. Нижче увага буде приділена колірній моделі CMYK.

3. Субтрактивна колірна модель CMYK

Субтрактивна модель використовується для підготовки не екранних, а друкованих зображень, тобто для пристроїв, які реалізують принцип поглинання (віднімання) кольорів. Друковані зображення відрізняються від екранних зображень тим, що їх бачать не у світлі, що випромінюється, а у відбитому світлі, оскільки аркуш паперу не випромінює світло. Папір поглинає деякі електромагнітні хвилі з оптичного діапазону, а решту відбиває, а наше око сприймає лише відбиті хвилі. Тому для підготовки друкованих зображень використовується не адитивна модель RGB, а субтрактивна модель CMY. На відміну від моделі RGB, біла точка в CMY – це відсутність фарб на папері. Назва цієї моделі складається з назв субтрактивних кольорів (протилежних до R, G, B) – блакитного (Cyan), пурпурного (Magenta) і жовтого (Yellow) (рис. 8), тобто, щоб отримати потрібний колір, базові кольори віднімаються від білого. Ці три кольори називаються доповнюючими, оскільки вони доповнюють основні кольори до білого, тобто змішування даного кольору і доповнюючого до нього дає білий колір.

Ці співвідношення можна подати у вигляді:

$$\begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} R \\ G \\ B \end{pmatrix},$$

тобто *доповнювальний колір = білий колір – даний колір*.

Обернене перетворення здійснюється за формулою:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} C \\ M \\ Y \end{pmatrix}.$$

Наприклад, коли на поверхню паперу нанести блакитний (cyan) колір, тоді червоне світло, що падає на папір, повністю поглинатиметься. Отже, блакитна фарба, так би мовити, віднімає червоний колір від білого, який є сумою червоного, зеленого і синього кольорів, тобто відбивається лише зелена та синя складові світла, що і дає блакитний колір (рис. 8).

Аналогічно жовта фарба (Yellow) поглинає синій колір, а пурпурна (Magenta) – зелений. Білий папір виглядає білим тому, що він відбиває всі кольори і жоден не поглинає (рис. 9).

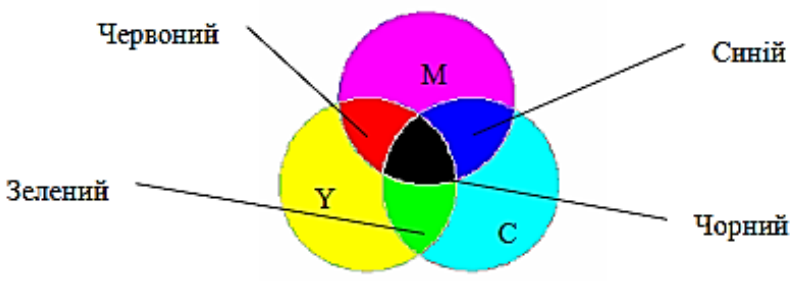


Рис. 8 – Кольори системи CMYK

При освітленні білим світлом, наприклад, синьої поверхні в шарі синьої фарби зі спектру білого кольору поглинаються червона та зелена частини спектру і в результаті ми бачимо синій колір (рис. 10) Чорний колір відповідає поглинанню всіх кольорів при відображенні (рис. 10). Якщо освітити червоний папір синім або зеленим світлом, то папір буде виглядати чорним, оскільки червоний папір не відбиває синій та зелений кольори, а поглинає їх (рис. 9).

Істотною проблемою в поліграфії є чорний колір. Теоретично його можна отримати змішуванням трьох доповнюючих фарб, але на практиці змішування цих трьох кольорів дає невизначений темно-коричневий колір. Отримати на папері чорний колір шляхом змішування трьох фарб складно і незручно через те, що реальні фарби не є абсолютно чистими, через великі витрати дорогого чорнила та високу вологість паперу на струменевих принтерах, через небажані візуальні ефекти, тому в принтерах до базових фарб CMY доводиться додавати ще й фарбу чорного кольору (blacK). Така модель кольору називається CMYK.

При друці малюнка на кольоровому принтері з чотирма кольорами драйвер принтера перетворює RGB-малюнок у модель CMYK. Однак багато відтінків, створених в кольоровій системі RGB, не вдається передати при друці на принтері. А це означає, що колірне охоплення системи CMYK менше, ніж колірне охоплення системи RGB. Водночас варто зазначити, що лише частину кольорів, які зустрічаються в природі і сприймаються людським зором, можна відтворити на екрані монітора, тобто колірне охоплення моделі RGB вужче, ніж колірне охоплення людського ока. Як видно, жодна з моделей не є повною за колірним охопленням. Під *колірним охопленням* розуміють діапазон кольорів, який може бути відтворений моделлю кольорів.

У типографіях кольорові зображення друкують у кілька етапів, накладаючи почергово на папір блакитний, пурпурний, жовтий і чорний відбитки. Так отримують повноколірну ілюстрацію. Тому готове зображення, отримане на комп'ютері перед друком, ділять на чотири складових однокольорових зображення. Цей процес називається діленням кольору (separations). Сучасні графічні редактори мають спеціальні засоби для виконання цієї операції. Ці програми самі визначають суміш CMYK.

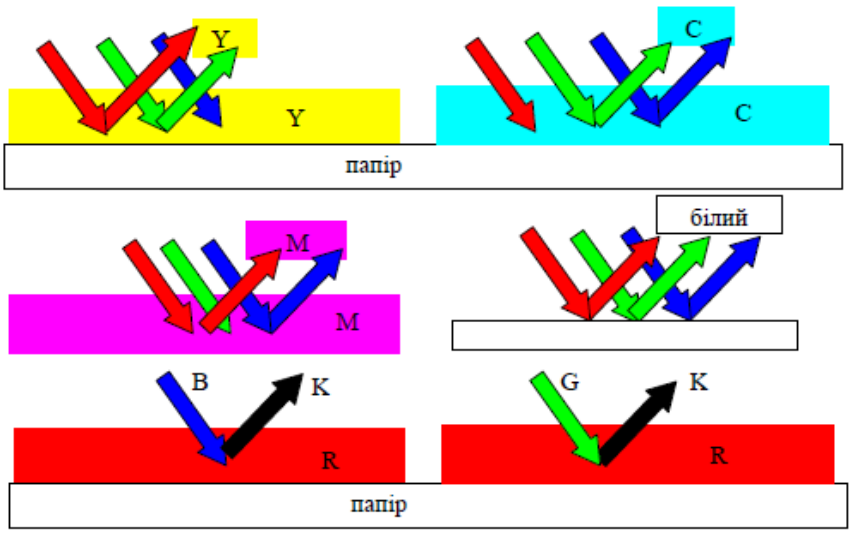


Рис. 9 – Поглинання (віднімання)кольорів

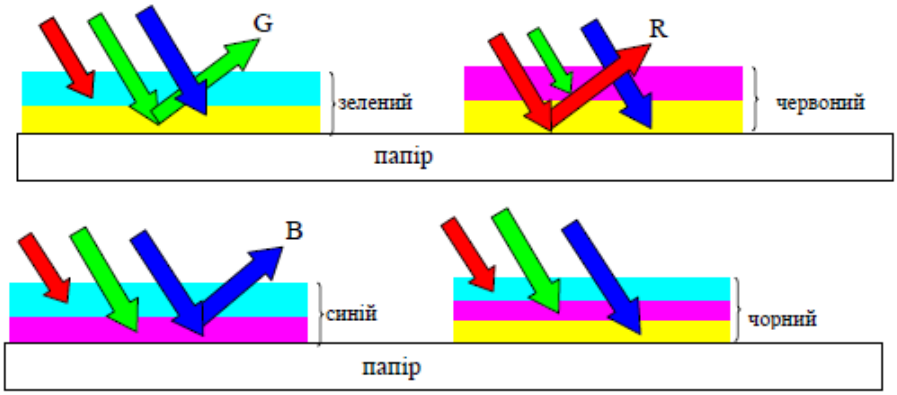


Рис. 10 – Субтрактивність для двох і трьох кольорів

У лазерному принтері є свій «комп'ютер» із необхідними програмами, що виконують перетворення графічних даних у зображення на папері. Кольорові принтери розділяють кольорові малюнки на 4 складові, а потім окремо друкують кожну частину на одному й тому ж аркуші паперу, тобто один аркуш проходить через принтер 4 рази. При цьому окремі кольорові точки, з яких складається малюнок, повинні бути трохи зсуненими одна від одної, щоб не було накладання кольорів. Для переходу від моделі CMY до моделі CMYK використовують такі співвідношення:

$K = min(C, M, Y), C = C - K, M = M - K, Y = Y - K.$

Тріадні і плашкові кольори

Раніше для роздруківки результатів виконаної в графічній програмі роботи на поліграфічному обладнанні можна було використовувати одну з двох схем друку: плашкова або багатошарова. Зараз ряд графічних програм і основні програми верстки підтримують комбінований спосіб друку шляхом додавання до багатошарового друку плашкових кольорів (Spot colors). Нагадаємо, що плашковими (або простими) кольорами називаються кольори, які відтворюються на папері готовими змішаними фарбами. Кожен плашковий колір репродукується за допомогою окремої друкованої форми (плашки). Багатошаровий друк засновано на використанні тріадних (інакше складених) кольорів і включає в себе як мінімум чотири процеси відповідно до кількості використовуваних барвників.

Тріадні кольори відтворюються шляхом оптичного змішування в різних пропорціях тріадних фарб (блакитної, пурпурної, жовтої), що застосовуються в стандартному чотири фарбовому друці. У графічних програмах всі колірні моделі працюють саме з тріадними кольорами. Тому відтворення плашкового кольору на екрані монітора за допомогою, наприклад, колірної моделі RGB призводить до апроксимації плашкового кольору тріадним кольором. Плашкова схема друку застосовується в тих випадках, коли кількість кольорів в малюнку менше чотирьох або коли окремі кольори можна отримати шляхом змішування фарб (наприклад, неонові або ті, що імітують металізовану поверхню). У разі необхідності відтворення кольору або створення спеціальних колірних ефектів можлива реалізація плашкового друку з великим кількістю кольорів або поєднання плашкового і багатошарового друку.

Деякі плашкові кольори можна точно передати за допомогою тріадних фарб, інші знаходяться за межами колірного охоплення CMYK. Наприклад, пастельні, неонові або металізовані фарби не мають аналогів в колірній системі CMYK, а відтінок зеленого кольору легко замінити його складовим аналогом. У графічних редакторах для підтримки плашкових кольорів введені особливі плашкові канали. За реалізацію вони схожі на альфа-канали, але являються колірними. Такі канали можуть бути створені безпосередньо або шляхом перетворення зі звичайних альфа-каналів. Зображення з плашковими каналами широко поширені в поліграфії, особливо в газетній продукції, і призначені для друку плашковими кольорами. Це дозволяє скоротити витрати на видання за рахунок зменшення кількості фарб. Плашкові кольори використовуються і в високоякісній поліграфії для розширення діапазону переданих кольорів, друку точних кольорів (наприклад, фірмових), друку спеціальними фарбами (металізованими, люмінесцентними і т. п.), виготовлення форм для лакування та фольгування.

Різниця між плашковими та тріадними кольорами прямо пов'язана з процесами взаємодії світла з чорнилом, що використовуються для створення цих фарб. Чорнило для плашкового друку непрозорі, тому вони відбивають світло поверхневим шаром. В результаті для отримання на папері, наприклад, пурпурного кольору необхідно використовувати пурпурні чорнила. Це дозволяє, в свою чергу, отримувати дуже яскраві тони і спеціальні ефекти типу металізації і переливу відтінків. Чорнило для багатошарового друку прозорі, тому світло відбивається не їх поверхневим шаром, а поверхнею матеріалу, на який вони нанесені. Це призводить до того, що відтворення кольору відбувається за рахунок видалення зі спектру

зайвих компонентів шляхом поглинання їх шаром фарби. В результаті для відтворення пурпурного кольору на поверхню паперу необхідно накласти два типи чорнил - бірюзового та синього кольорів. Вони поглинуть синю і зелену частини спектра, залишивши (відбивши) для нашого ока тільки пурпурову частина спектра.

4. Перцепційна колірна модель HSB

Моделі RGB, CMY, CMYK орієнтовані на роботу з технічними засобами. Якщо модель RGB найприйнятніша для комп'ютера, модель CMYK – для типографій, то модель HSB найзручніша для людини. Вона проста та інтуїтивно зрозуміла. Ця модель дозволяє задавати кольори, опираючись на інтуїтивні поняття тону кольору H (Hue), насиченості кольору S (Saturation) та яскравості кольору B/V (Brightness/ Value). Тому цю модель називають HSB або HSV. Регулюючи три згадані компоненти, можна отримати стільки ж кольорів, як і при роботі з іншими моделями.

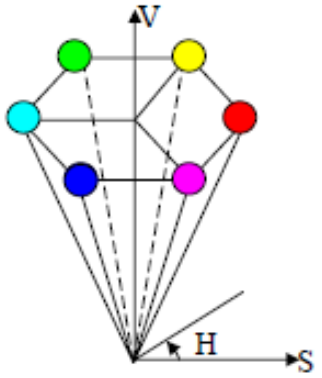


Рис. 11 – Модель HSB

Деякі графічні редактори дозволяють працювати з моделлю кольорів HSB. Модель HSB зручна для застосування в тих графічних редакторах, які зорієнтовані не на обробку готових зображень, а для створення власних художніх творів. У цій моделі використовується циліндрична система координат, а множина всіх допустимих кольорів є конусом, покладеним на вершину (рис. 3.11). Основа конуса – яскраві кольори, що відповідають значенню V=1 (конус має одиничну висоту).

Значення H змінюється в градусах від 0^0 до 360^0, оскільки кольори веселки розміщуються на колі в такому порядку: червоний, оранжевий, жовтий, зелений, блакитний, синій, фіолетовий. Червоному кольору відповідає кут 0^0, зеленому – 120^0 і т.д. Кольори, що доповнюють один одного до білого, на колі знаходяться навпроти. Величина S змінюється від 0 до 1. Значення V = 0 відповідає чорному кольору. При S = 0 (тобто на осі V) маємо сірі відтінки. Білий колір кодується як S = 0, V = 1. При S = 0 значення H не має змісту.

Модель HSB зручна для вибору кольорів на екрані. Спочатку на екрані можна зобразити спрощену палітру, а потім збільшити або зменшити яскравість. Наприклад, так діють при моделюванні затінення об'єктів або сутінків.

Графічні редактори дозволяють працювати з кольоровими зображеннями в різних моделях. Наприклад, у Paint Brush for Windows для установки кольору використовуються дві

моделі – RGB та HSV (існують відповідні формули зв’язку між параметрами R, G, B та H, S, V).

Звернемо увагу на відмінності між моделями HSL і HSV - обидві циліндричні конфігурації, з відтінком, їх кутовим вимірюванням, що починається з червоного на 0 °, проходячи через зелений колір на 120 ° і синьому на 240 °, і потім розгортаємо назад до червоного кольору в 360 °. У кожній геометрії, центральна вертикальна вісь включає нейтральні, безбарвні, у відтінках сірого кольору, в межах від чорного в легкості 0 або цінності 0, підставі, до білого в легкості 1 або цінності 1, вершині. (наприклад, на рис.12 це від 0 до 10). В обох конфігураціях, сукупні первинні і вторинні кольори - червоні, жовті, зелені, блакитні, сині, і фуксин і лінійні суміші між суміжними парами їх, тобто чистими кольорами, які влаштовані навколо зовнішнього краю циліндра з насиченістю 1; в HSV вони мають цінність 1, в той час як в HSL вони мають світлість S. У моделі HSV (рис. 13), змішуючи ці чисті кольору з білим - створюються так звані відтінки, що зменшує насиченість, а змішуючи їх з чорним кольором отримуємо відтінки, наприклад, незмінну насиченість листя. У HSL і відтінки мають повну насиченість, і тільки суміші з обома чорно-білими квітами - названі тонами - мають насиченість менше ніж 1.

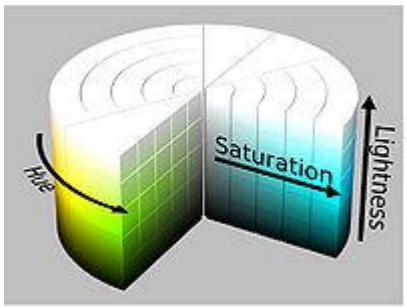


Рис. 12 – Модель HSL

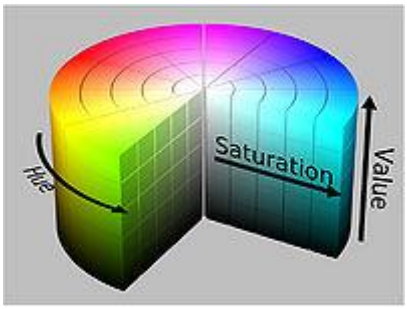


Рис. 13 – Модель HSV

Таким чином, шкала Brightness моделі HSB змінює яскравість від чорного до найбільш яскравого. Шкала Lightness моделі HSL дозволяє змінювати яскравість (або світлість) кольору від чорного до білого, не торкаючись шкали Saturation. Точка найбільшої колірної насиченості знаходиться посередині шкали. В цьому полягає основна відмінність. Той чи інший вид моделі зустрічається в різних редакторах, тому координати одного і того ж кольору можуть подекуди відрізнятися.

5. Колірна модель та колірний простір.

Після розгляду механізмів функціонування основних колірних моделей слід зупинитися на взаємозв'язку колірної моделі і колірного простору. Колірна модель визначає відносини між величинами, а колірний простір - абсолютні значення цих величин в якості кольорів. В деяких колірних моделях (наприклад, CIE Lab) колірний простір фіксований, оскільки ці моделі створені на базі експериментів по сприйняттю кольору оком людини. Такі моделі прийнято називати універсальними або апаратно незалежними. Для них поняття колірної моделі і колірного простору еквівалентні. Для інших колірних моделей (RGB, CMY,

CMYK і т. Д.) може бути створено множину різних колірних просторів. Так як ці моделі різні для різних колірних просторів і пристроїв, їх називають апаратно-залежними.

Колірний простір (color space) - тривимірний простір для зображення кольорів. Кожному кольору відповідає точка простору, а кожній точці – єдиний колір, тобто встановлена однозначно відповідність. Таким чином, це геометрична інтерпретація можливих координат кольору. Можна сказати що колірний простір - опис кольору деякого гіпотетичного пристрою. Колірний RGB-простір представимо як деякий ідеальний монітор, що не міняє характеристик з плином часу: для будь-якого кольору, заданого складовими червоного, зеленого і синього, в цьому просторі однозначно визначимо абсолютний колір. Саме колірний простір дає інтерпретацію RGB-даними документа. Те ж саме з CMYK-файлами, але, строго кажучи, при роботі в моделі CMYK ми маємо не за забарвленням простір, а опис кольору, тобто не виконується важлива вимога однозначності завдання кольорів.

Колірна модель - математична модель опису представлення кольорів у вигляді наборів чисел (зазвичай з трьох, рідше - чотирьох) значень, званих колірними компонентами або колірними координатами. Якщо використовується система з більшим, ніж три, числом компонентів, кодується спектр відбиття або випромінення джерела, що дозволяє більш точно описати фізичні властивості кольору. Всі можливі значення кольорів, що задаються моделлю, визначають колірний простір. У кожній моделі представлений певний діапазон кольорів у вигляді дво- або тривимірного простору.

6. Системи відповідності кольорів.

Для спрощення процедури ідентифікації кольору провідні фірми, що спеціалізуються в області поліграфії та на виробництві барвників, створили системи відповідності кольорів.

Розглянемо процеси ідентифікації та підбору кольорів на прикладі поліграфічної галузі. Виготовлення зразка ретельно контролюється з метою мінімізації варіацій кольорів. Кожному кольору присвоюється своє унікальне ім'я і вказується тип пігменту або склад суміші з різних пігментів, необхідних для його реалізації. Вказується також ідентифікований з даними пігментом тип паперу. На додаток до цієї таблиці, яка використовується як довідник, користувач отримує зразки кольорів, які можна вирізати і прикріпити до зображення. Завдяки цими зразками система забезпечує точний візуальний контроль відповідності того, що ми бачимо на екрані, з тим, що ми отримаємо на друку. Типовими прикладами атласів кольорів (або, як їх ще називають, колірних зразків) є каталоги фірм TRUMATCH і Pantone, відомі під назвами Colorfinder і Process Color Guide.

Різниця між плашковими та тріадними кольорами прямо пов'язана з процесами взаємодії світла з чорнилом, що використовуються для створення цих фарб.

Чорнило для плашкового друку непрозорі, тому вони відбивають світло поверхневим шаром. В результаті для отримання на папері, наприклад, пурпурного кольору необхідно використовувати пурпурні чорнило. Це дозволяє, в свою чергу, отримуємо дуже яскраві тони і спеціальні ефекти типу металізації і іризації (переливу відтінків).

Чорнило для багатошарового друку прозорі, тому світло відбивається не їх поверхневим шаром, а поверхнею матеріалу, на який вони нанесені. Це призводить до того,

що відтворення кольору відбувається за рахунок видалення зі спектру зайвих компонентів шляхом поглинання їх шаром фарби. В результаті для відтворення пурпурного кольору на поверхню паперу необхідно накласти два типу чорнила - бірюзового та синього кольорів. Вони поглинуть синю і зелену частини спектра, залишивши (відбивши) для нашого ока тільки пурпурову частина спектра.

7. Кодування кольору. Палітра кольорів.

Щоб комп'ютер мав можливість працювати з кольоровими зображеннями, необхідно вміти подавати кольори у вигляді чисел, тобто кодувати колір за допомогою комбінації бітів. Кількість бітів для подання кольору кожного пікселя, називається *бітовою глибиною*. Спосіб кодування кольору залежить від моделі кольорів та формату числових даних у комп'ютері. Для моделі RGB кожну з компонент можна зобразити з допомогою чисел, обмежених певним діапазоном (наприклад, дробовими числами від 0 до 1 або цілими від 0 до деякого максимального значення). Зараз досить розповсюджений формат True Color, в якому під кожний піксель відводиться 24 біти, тобто кожна компонента подається у вигляді байта, що дає 256 градацій для кожної компоненти: R = 0 –255, G = 0 –255, B = 0 –255. Це дає можливість закодувати 256 ×256 ×256 = 224=16777216 градацій кольорів, що значно перевищує кількість кольорів, які розрізняє людське око. Такий спосіб кодування кольорів називається *компонентним*. У комп'ютері коди зображень True Color подаються трьома байтами (рис. 3.13) або упаковуються в довге ціле – 32 біти (так, наприклад, зроблено в API Windows).

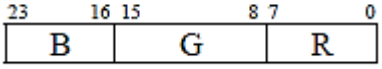


Рис. 14 – Трибайтовий код

У 32-бітній моделі RGBA старші 8 бітів використовуються для задання компоненти маски (Alpha-канали). Маска створюється деякими графічними програмами для спеціальних ефектів, наприклад для задання прозорості, туману. Хоча немає видимих причин використовувати більше кольорів, реально використовуються 48-бітні і навіть 64-бітні моделі. Ці моделі використовуються в кольорових системах вищого рівня, зокрема в професійних графічних системах. При роботі із зображеннями в системах комп'ютерної графіки часто доводиться шукати компроміс між якістю зображення (якомога більше кольорів) і ресурсами, необхідними для збереження зображення. Якщо для кодування кольору виділити 1 біт, то можна закодувати 2 різних кольори (чорно-білий режим). Виділення одного байта дозволяє закодувати 256 різних відтінків кольорів. Два байти дозволяють визначити 65536 кольорів (рис. 13). Цей режим називається Hight Color.

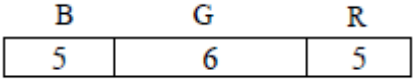


Рис. 15 – Двобайтовий код

У випадку True Color/Hight Color кольорова палітра не потрібна, оскільки в трьох/двох байтах достатньо інформації про колір пікселя. При обмеженні кількості кольорів, тобто для зменшення обсягу пам'яті використовують палітру. Кольорова палітра – це набір кольорів.

Палітру можна сприймати як таблицю кольорів, у якій указано індекс (номер) кольору та сам код того чи іншого кольору. Кожен кольоровий відтінок задають одним числом, причому це число визначає не код кольору, а номер кольору в таблиці. Сам колір визначається за цим номером у таблиці-палітрі. Ця таблиця зберігається разом із графічним файлом. Програма, що здійснює візуалізацію даних, читає з файла індекси і використовує відповідні їм кольори для зображення пікселів на екрані. Різні зображення можуть мати різні кольорові палітри. Якщо відтворити зображення з іншою палітрою, то зелена ялинка може стати рожевою (в різних палітрах під однаковим номером можуть зберігатися різні кольори). У зв'язку з цим виникає проблема відповідності кольорів при перегляді Web-графіки. Для оформлення Web-сторінок, особливо раніше, не застосовувалася графіка, що має кодування кольору вище 8 бітів через невисоку швидкість передачі даних в Internet. Зважаючи на це, було прийнято рішення – всі браузери (програми перегляду Web-сторінок) завчасно налаштовувати на певну фіксовану Web-палітру. У цій палітрі не 256 кольорів, які дозволяє кодувати 8 бітів, а лише 216. Це пов'язано з тим, що в Інтернеті працюють з різними комп'ютерами і не всі комп'ютери можуть відтворити 256 кольорів. Якщо розробник Web-сторінки використовуватиме лише цю палітру при створенні ілюстрації, то користувачі побачать малюнок правильно. Найбільш часто використовуються палітри з 16 та 256 кольорів. Як приклад наведемо стандартну палітру 16-кольорових відеорежимів. При 8-бітній глибині кольору можна задати 256 кольорів. Очевидно, що тут можна виділити під кожну компоненту певну кількість бітів (наприклад, для R – 3, для G – 3, для B – 2), але така технологія надто обмежує можливості передачі кольорів. Тому для такої малої глибини кольору краще використовувати інший підхід – із множини 256 × 256×256 кольорів вибираються довільні 256 кольорів, які нумеруються індексами від 0 до 255. Вибрані кольори записуються в таблицю кольорів (палітру), а у відеопам'ять замість коду кольорів записується 8- бітний індекс (номер кольору), який при візуалізації автоматично замінюється кольором, який у палітрі відповідає цьому індексу. У графічних системах, наприклад OpenGL, для роботи з палітрами кольорів є спеціальні оператори. З їх допомогою можна створити палітру, вказавши для кожного індексу набір R, G, B компонент кольору і в будь-який момент можна вибрати довільний індексний колір із палітри.

Таблиця 1 – Підіндекси компонент RGB

Номер кольору	R	G	B	Назва кольору	Колір
0	0	0	0	Чорний	
1	128	0	0	Темно-червоний	
2	0	128	0	Зелений	
3	128	128	0	Коричн.-зелений	
4	0	0	128	Темно-синій	
5	128	0	128	Темно-пурпурний	
6	0	128	128	Синьо-зелений	
7	128	128	128	Сірий 50%	
8	192	192	192	Сірий 25%	
9	255	0	0	Червоний	
10	0	255	0	Яскраво-зелений	
11	255	255	0	Жовтий	
12	0	0	255	Синій	
13	255	0	255	Фіолетовий	
14	0	255	255	Блакитний	
15	255	255	255	Білий	

У реальних додатках, що працюють із палітрами кольорів, часто виникає задача створення такої палітри, яка б рівномірно охоплювала весь спектр моделі кольорів RGB.
Задача. З усіх $(256)^3$ кольорів рівномірно вибрати 256 кольорів і призначити їм відповідні індекси.

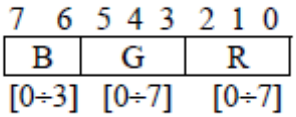


Рис. 16 – Підіндекси компонент RGB

Рис. 17 – Відображення інтервалів

Розв'язування. Розіб'ємо поле індексу справа наліво на 3 підполя довжиною 3, 3 і 2 біти, відповідно (рис. 1.15). Ці поля визначатимуть підіндекси компонент R, G, B. Для компоненти B відведемо 2 біти, оскільки зміна синього кольору менше сприймається людським оком, ніж зміна кольорів R та G (колір B має найменший діапазон електромагнітних хвиль). Таким чином, індекс кольору в палітрі *Ic* можна подати у вигляді $Ic = I_B \cdot 2^6 + I_G \cdot 2^3 + I_R$, де I_R, I_G, I_B – значення підіндексів.

Для вибори кольору слід рівномірно відобразити інтервали підіндексів [0, 3], [0, 7], [0, 7] на Інтервал [0, 255] (рис. 15). для компонент R и G одержуємо відповідно значення кольору $[255 \times I_R / 7]$ и $[255 \times I_G / 7]$, а для компоненти B - значення $[255 \times I_B / 3]$. Замість цілої частини можна брати найближче ціле. Обчислимо, який колір буде відповідати в палітрі індексу 214. Число 214 розкладемо за наведеною вище формулою. Маємо $214 = 3 \cdot 2^6 + 2 \cdot 2^3 + 6$, тому $I_R = 6$, $I_G = 2$, $I_B = 3$. Тоді для компонентів кольору матимемо: $R = [255 \times 6/7] = [218{,}57] = 218$, $G = [255 \times 2/7] = [72{,}86] = 72$, $B = [255 \times 3/3] = 255$. Отже, індексу 214 в рівномірній палітрі з 256 кольорів відповідає колір (218, 72, 255).

В даній лекції ми розглянули адитивні, субтрактивні та перцепційні колірні моделі. Показано відмінності та сфери застосування різних колірних моделей. Також була звернена увага на ряд питань, що пов’язані з поданням кольорів.

Контрольні запитання

1. Що таке колірна модель?

2. Яким вимогам має відповідати колірна модель?

3.. Чим колірна модель відрізняється від колірного режиму?

4. Назвіть та поясніть закони Г. Грасмана.

5. Який зв’язок законів Грасмана та колірної моделі RGB?

6. Охарактеризуйте адитивні колірні моделі.

7. Охарактеризуйте субстрактивні колірні моделі.

8. Охарактеризуйте перцепційні колірні моделі.

9. Охарактеризуйте графічне представлення моделі RGB.

10. Яким чином подаються кольори для їх подальшого сприйняття комп’ютером?

Список використаних інформаційних джерел

1. https://www.youtube.com/watch?v=15aqFQQVBWU – відео про зображення, пікселі та кольорову модель RGB.
2. https://www.youtube.com/watch?v=9hirYMZ7PQc – відео про відмінність між колоровими моделями RGB та CMYK.
3. https://habr.com/ru/post/181580/
4. Маценко В.Г. Комп’ютерна графіка: Навчальний посібник. – Чернівці: Рута, 2009 – 343 с.
5. https://ed.ted.com/lessons/how-do-we-see-color-colm-kelleher – відео про колір.
6. https://www.youtube.com/watch?v=9hirYMZ7PQc&t=4s – відео про колірну модель CMYK.
7. http://popel-studio.com/blog/article/cvetovaya-model-hsb.html - стаття про відмінності між колірними моделями HSB та HSL.

ЛЕКЦІЯ 4.

ТЕМА: ВЕКТОРНА ГРАФІКА

План

1. Основні поняття векторної графіки.

2. Переваги та недоліки векторної графіки.

3. Об’єкти векторної графіки .

4. Структура векторної ілюстрації.

5. Шрифти у векторній графіці.

1. Основні поняття векторної графіки.

Векторна графіка – спосіб представлення об’єктів та зображень в комп’ютерній графіці, заснований на математичному описі елементарних геометричних об’єктів, що називають примітивами. До примітивів, як правило, відносять точки, лінії, криві Безьє, кола та окружності, багатокутники. Об’єкти векторної графіки є графічними зображеннями математичних об’єктів.

Термін векторна графіка використовується для того, щоб пояснити відмінність від растрової графіки, де зображення представлено у вигляді графіної матриці. При виведені на монітори векторна графіка попередньо перетворюється у растрову.

Зображення, створене в векторних програмах, ґрунтується на математичних формулах, а не на координатах пікселів. Тому векторні файли містять набори інструкцій для побудови геометричних об'єктів - ліній, еліпсів, прямокутників, багатокутників і дуг (рис. 1). Відповідно до цього основу векторних зображень складають різноманітні лінії або криві, звані *векторами,* або, по-іншому, *контурами.* Кожен *контур* являє собою незалежний об'єкт, який можна редагувати: переміщати, масштабувати, змінювати. Відповідно до цього векторну графіку часто називають також *об'єктно-орієнтованою* графікою.

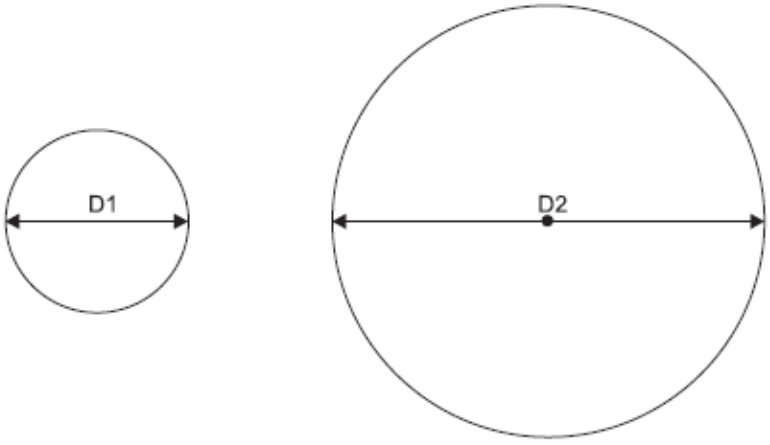


Рис. 1 – З векторною графікою ви фактично вже познайомилися на уроках шкільної математики, коли писали формулу площі круга $S = \pi R^2$. Для зміни розміру фігури досить

вказати нове значення параметра (в даному випадку діаметра). Згадайте, що виконання аналогічного перетворення в растрової зображенні призведе до появи так званих «ступенів»

Проілюструємо різницю в механізмах роботи растрових і векторних редакторрів на прикладі опису одного і того ж відрізка прямої. У векторному форматі задаються координати початку і кінця прямої, колір і товщина лінії. Для збереження такої інформації на диску буде потрібно всього кілька байтів пам'яті. У растровому форматі задаються координати і колір кожної точки (пікселя), що входить в цей відрізок прямої. А оскільки кількість вхідних в неї пікселів залежить від дозволу, то обсяг інформації, необхідної для опису відрізка прямої (а значить, необхідний для її запам'ятовування обсяг пам'яті), буде визначатися встановленим дозволом. З наведеного прикладу видно, що векторний формат, як правило, більш компактний (хоча складні малюнки, що містять сотні і тисячі об'єктів, можуть перевищувати розміри растрових зображень). Разом з тим він зовсім не придатний для зберігання сканованих зображень, наприклад фотографій. А ось малюнки і креслення набагато зручніше і практичніше робити саме в векторному вигляді.

При традиційному 8-бітовому кодуванні комп'ютери можуть використовувати для уявлення букв, цифр і символів тільки 128 знаків - набір ASCII. Однак при роботі з мовами, в яких існують множинні знаки наголосу або діакритичні знаки (наприклад, в польському або румунською мовами) або навіть абсолютно інший алфавіт, потрібні додаткові програмні рішення. Набори символів можна розширювати до 256 знаків і навіть більше, але такий підхід не завжди виявляється коректним вирішенням проблеми. Unicode - це інший стандарт кодування символів, свого роду оболонка «СуперASCII», яка дає можливість використовувати в одному шрифті близько 65 тисяч символів з механізмами для зручного переходу з одного підмножини до іншого. Слід зазначити, тільки з впровадженням нового шрифтового формату OpenType отримує все більшу підтримку у виробників програмних продуктів. Растрові и векторні зображення суттєво відрізняються за способом представлення графічної информации. *Векторні зображення* - це зображення, які побудовані з графічних прімітівів, что задаються своими характерними параметрами (для кожного примітіву свої параметри).

У растровій графіці основним елементом зображення є точка, а у векторній графіці - лінія (контур), яку ще називають вектором. Контур может бути прямою або кривою лінією, замкненим або відкрітім. Кожний контур має дві або ьільше опорних точки (вузлів). Між двома вузлами містіться сегмент контуру. Форму контуру задають через опорні точки. Над контуром можна виконувати операции комбінування та об'єднання. У векторній графіці все складається з ліній. У растровій графіці теж є лінії, проте вони розглядаються як сукупність точок, й чим довша растрова лінія, тим більше пам'яті вона займає. У векторній графіці обсяги пам'яті, які займає лінія, не залежить від розмірів лінії, оскількі лінія подається у вигляді формули (кількох параметрів). Оскількі лінія є основним об'єктом векторної графіки, а складніші об'єкти у векторній графіці складаються з простішіх (ліній), то векторну графіку називають ще об'єкто-орієнтованою графікою. У ній зображення створюється шляхом комбінування різніх прімітівів. Файли векторної графіки для створення зображення містять тисячі різніх команд типу "нарисувати лінію від *A* до *B* ", набори параметрів, інформацію про колір, дані про шрифт, что можуть бути включені в малюнок. Ключовим моментом векторної графіки є використання для опису об'єктів команд та математичних формул. Для кожного об'єкта (або класу обєктів) візначається набір параметрів, який визначає його зовнішній вигляд. Але хоча об'єкти векторної графіки зберігаються в пам'яті як набір параметрів,

зображення об'єктів на екран віводяться у вигляді точок-пікселів, оскількі такою є будова екранів.

Перед виведенням кожного об'єкта на екран відбувається процес розкладання векторного зображення в растр, тобто програма обчислює координати екранних точок зображення об'єкта. Тому векторну графіку ще називають обчислювальною графікою.

Крім геометричних параметрів векторні об'єкти мають параметри кольору контура, кольору заповнення, текстури тощо. Для векторних об'єктів (коло, квадрат, лінія) колір стосується об'єкта в цілому. Колір об'єкта зберігається в його векторному описі. У растрових об'єктах потрібно зберігаті інформацію про колір для кожного пікселя. Векторна графіка не має зазначених недоліків растрової графіки, тому векторна графіка "економна" в плані обсягів файлів, оскількі зберігає не саме зображення, а тільки параметри зображення. Так, для збереження лінії другого порядку у векторній графіці потрібно приблизно 10 параметрів (20 байтів). Складні композіції, які складаються з 1000 об'єктів, займають всього десятки або сотні $к_б$.

Найважлівішій аспект векторної графіки полягає в тому, что можна змінювати розміри векторного малюнка без втрати його якості. У векторній графіці легко розв'язується питання масштабування. При збільшенні зображення можна розглядаті деякі деталі зображення, тобто маємо можливість змінюваті розміри векторного малюнка без втрати якості. Це найбільша перевага векторної графіки. Недоліком векторної графіки є її програмна залежність. Кожна програма зберігає дані в своєму форматі, тому зображення, створене в одному редакторі, як правило, не конвертується в формат іншої програми без похибок. Виключення складає лишь формат файлу AI програми Adobe Illustrator. Формат AI підтрімується практично усіма програмами векторної графіки. Крім цього, оскількі основним об'єктом векторної графіки є лінія, то засоби векторної графіки складно створюваті художні ілюстрації, смороду не здатні показати оригінал так реалістічно, як це дозволяє зробити растрова ілюстрація. Тому векторна графіка використовується не для створення художніх композіцій, а для оформлювальніх та проектно-конструкторських робіт. Векторні засоби широко застосовуються в рекламних агентствах, редакціях, видавництво, студіях дизайну. Разом із цім векторна графіка знайшла широке застосування в комп'ютерній анімації. Для збереження або передачі мультфільмів по лініях зв'язку не потрібна обробка всех кадрів фільму, достатньо відправити лишь перший та останній кадр фільму, форму траєкторії, швидкість показу кадрів тощо, а браузер користувача згенерує всі кадри для проглядання фільму.

2. Переваги та недоліки векторної графіки.

Одним з головних достоїнств цього виду графіки є можливість необмежено збільшувати або зменшувати зображення без втрати якості і практично без збільшення розмірів вихідного файлу.Як було зазначено вище, це пов'язано з тим, що векторна графіка містить тільки опису об'єктів, які формують зображення, а комп'ютер або пристрій друку інтерпретує їх необхідним чином. Векторну графіку значно легше редагувати, оскільки готові зображення не є «плоскою» картинкою з пікселів, а складено з об'єктів, які можуть накладатися один на одного, перекриватися, залишаючись в той же час абсолютно незалежними один від одного. Векторним програмам властива висока точність малювання (до сотої частки мікрона). Векторна графіка економить дисковий простір, необхідне для

зберігання зображень. Це пов'язано з тим, що зберігається не саме зображення, а тільки деякі основні дані (математична формула об'єкта), використовуючи які програма щоразу відтворює зображення заново. Опис колірних характеристик майже не збільшує розмір векторного файлу. Векторні зображення, як правило, займають менший об'єм пам'яті компьютера в порівнянні з растровими. Набагато простіше описати коло радіусом 10 і центром в точці $x = 20$, $y = 30$, ніж пам'ятати всі пікселі масиву, відповідних даній окружності. Для векторних редакторів характерно прекрасна якість друку малюнків і відсутствие проблем з експортом векторного зображення в растрове.

Практично неможливо здійснити експорт зображення з растрового формату в векторний. Спробуйте, наприклад, відсканувати складне зображення, а потім вирізати його на плоттері. І навпаки, зворотне перетворення (тобто перетворення векторного зображення в растрове) виконується практично автоматично не тільки за допомогою графічних редакторів, а й буфера обміну Windows. Векторна графіка обмежена в живописних засобах і не дозволяє отримувати фотореалістичні зображення з тією ж якістю, що і растрова. Причина в тому, що, на відміну від растрової графіки, мінімальної областю, яка зафарбовується однорідним кольором, є не піксель, а об'єкт. Спробуйте описати, скажімо, туманний ранок математичними формулами. Векторний принцип опису зображення не дозволяє автоматизувати введення графічної інформації, як це робить сканер для растрової графіки. В реальному житті не існує векторних моніторів або векторних сканерів. У векторній графіці неможливо застосувати велику бібліотеку ефектів (фільтрів), використовуваних при роботі з растровими зображеннями. Інакше кажучи, жоден сучасний професійний графічний пакет не є чисто векторним або чисто растровим, а поєднує в собі елементи як того, так і іншого виду графіки. Наприклад, векторний редактор CorelDRAW має як власні, так і підключаються (plug-ins) інструменти для оброблення растрових зображень, а версії графічного редактора Photoshop включають розширені інструментальні можливості для роботи з векторними об'єктами.

3. Об’єкти векторної графіки.

Основним логічним елементом векторної графіки є геометричний об'єкт. Як об'єкт приймаються прості геометричні фігури (так звані примітиви - прямокутник, окружність, еліпс, лінія), складені фігури або фігури, побудовані з примітивів, колірні заливання, в тому числі градієнти. Важливим об'єктом векторної графіки є **сплайн. Сплайн** - це крива, за допомогою якої описується та чи інша геометрична фігура. На сплайнах побудовані сучасні шрифти TryeType і PostScript. Об'єкти векторної графіки легко трансформуються і модифікуються, що не має практично ніякого впливу на якість зображення. Масштабування, поворот, викривлення можуть бути зведені до елементарних перетворень над векторами. Якщо в растровій графіці базовим елементом зображення є точка, то у векторній графіці - лінія. Лінія описується математично як єдиний об'єкт, і тому об'єм даних для відображення об'єкта засобами векторної графіки істотно менше, ніж в растровій графіці.

Лінія - елементарний об'єкт векторної графіки. Як і будь-який об'єкт, лінія має властивості: формою (пряма, крива), товщиною, кольором, накресленням (суцільна, пунктирна). Замкнені лінії набувають властивість заповнення. Охоплюваний ними простір може бути заповнений іншими об'єктами (текстури, карти) або вибраним кольором. Найпростіша незамкнута лінія обмежена двома точками, які називаються

вузлами. Вузли також мають властивості, параметри яких впливають на форму кінця лінії і характер сполучення з іншими об'єктами. Всі інші об'єкти векторної графіки складаються з ліній. Наприклад, куб можна скласти з шести зв’язанних прямокутників, кожен з яких, в свою чергу, утворений чотирма зв'язаними лінями.

Криві Безьє

На початку 70-х рр. професор П'єр Безьє, проектуючи на комп'ютері корпусу автомобілів «Рено», вперше застосував для цієї мети особливий вид кривих, описуваних рівнянням третього порядку, які згодом стали відомі під назвою **криві Безьє**. Оскільки ці лінії мають особливе значення як для векторної, так і для растрової графіки, має сенс розглянути їх більш детально. В даний час криві Безьє присутні в будь-якому сучасному графічному пакеті. Досить сказати, що всі комп'ютерні шрифти складаються з кривих Безьє. Вони також знаходять широке застосування і в растровій графіці. Так, в програмі Photoshop використовується термін контур (path), який базується на кривих Безьє. Саме за допомогою цього інструменту можна виділити на сканованій фотографії потрібний об'єкт (наприклад, для його вирізання), який буде використано при створенні фотомонтажу. Навчившись працювати з ними в одній програмі, Ви легко зможете працювати з безліччю інших - повторимося, принцип тут однаковий. Відрізками такої кривої можна апроксимувати контур будь-якої складності. У цьому випадку він буде складатися з набору кривих Безьє. У місцях згину сформована з відрізків крива Безьє може мати злами. Однак за допомогою функцій згладжування керуючі точки сусідніх відрізків легко вибудовуються в одну лінію, після чого злам зникає. Гнучкість в побудові і редагуванні кривих Безьє багато в чому визначається характеристиками вузлових і керуючих точок.

Поява кривих Безьє викликало справжній переворот в відео та тривимірній графіці. Це пов'язано з тим, що до появи формул Безьє контури комп'ютерних персонажів були ламаними, поверхні - гранованими, а рух – переривчастим, простим, стрибкоподібним, неприродним. Використання кривих Безьє дозволило реалізувати найбільш загальний і інтуїтивно зрозумілий спосіб управління рухом. Параметрами кривої можна поставити у відповідність параметри руху комп’ютерного персонажа. В результаті рух буде відбуватися за тими ж правилам. Таким чином, криві Безьє використовується не тільки в двомірній комп'ютерній графіці, але і в тривимірній графіці, відео та анімації.

Як уже зазначалося, в основі векторної графіки лежить використання матетичних уявлень про властивості контурів, основу яких становить елементарний об'єкт векторної графіки - лінія. З її допомогою можна легко побудувати будь-який більш складний об'єкт. Наприклад, об'єкт чотирикутник можна створити за допомогою чотирьох ліній, а куб - за допомогою 12 ліній або чотрикутників. Таким чином, ілюстрація складається з простих об'єктів, як з кубиків. Завдяки цьому процес малювання в векторних редакторах фактично сводиться до створення контурів (об'єктів) потрібної форми і наданню їм визначених заливок і обводок. Цей принцип лежить в основі всіх програм векторної графіки. Розрізняються лише прийоми роботи і деякі спеціальні ефекти.

У той же час побудова лінії поряд з використанням для її опису математичного апарату передбачає завдання ряду додаткових атрибутів, що визначають її основні властивості: форму, товщину, колір, стиль (суцільна, пунктирна і т. п.). Кількість перерахованих атрибутів

залежить від виду лінії. Відкриті лінії, наприклад, на відміну від замкнутих, не мають атрибута заливки. Замкнені контури крім обведення можуть мати певну користувачем заливку.

Поряд з лінією (line) іншим основним елементом векторної графіки є вузол (опорна точка). Як уже зазначалося, лінії і вузли служать для побудови контурів, які можна представити у вигляді прямої, кривої або форми. Кожен контур має кілька вузлів. У векторних редакторах (як, втім, і в растрових) форму контуру змінюють шляхом маніпуляції вузлами. Це можна зробити одним із таких способів:

- переміщенням вузлів;
- зміною властивостей вузлів;
- додаванням або видаленням вузлів.

При виділенні вузлової точки криволінійного сегмента у неї з'являються одна або дві крапки, що управляють, з'єднані з вузловою точкою дотичними лініями. Керуючі точки зображуються чорними зафарбованими точками. Місце розташування дотичних ліній і керуючих точок визначає довжину і форму (кривизну) криволінійного сегмента, а їх переміщення призводить до зміни форми. У локалізованих версіях векторних програм поряд з терміном дотичні використовуються й інші терміни: важелі управління, направляючі тощо. Форма і колір керуючих точок також залежать від використовуваного редактора.

Колір в векторній графіці

Різні векторні формати мають різними колірними можливостями. Найпростіші формати, які можуть не містити взагалі ніякої інформації про колір, використовують колір за замовчуванням тих пристроїв, на які вони виводяться, інші формати здатні зберігати дані про колір в інших форматах. Не в залежності від того, яку кольорову модель застосовує векторний формат, на розмір файлу він не впливає, крім тих випадків, коли файл містить растрові образи. У звичайних векторних об'єктах значення кольору відноситься до всього проекту в цілому. Колір об'єкту зберігається у вигляді частини його векторного опису. Деякі векторні файли можуть створити растровий ескіз зображень, які зберігаються в них. Ці растрові картинки, іноді звані короткими описами зображень, зазвичай представляють собою ескізи векторних малюнків в цілому. Опис зображення, особливо корисний в ситуаціях, коли ви не хочете відкривати весь файл, щоб подивитися, що в ньому зберігається або коли Ви не можете бачити векторний малюнок під час його використання.

Перша ситуація виникає, коли вам необхідно знайти файл за допомогою однієї з багатьох спеціально розроблених для цього програм. Для полегшення пошуку потрібного векторного файлу такі програми можуть зчитувати растровий ескіз зображення та інші хара до теристики, наприклад, векторний формат, час створення, бітову глибину з зображенія і так далі.

Друга ситуація виникає, коли в будь-якому видавничому пакеті міститься на сторінках векторний рисунок. Зображення, яке Ви побачите, буде растровим ескізом справжнього векторного малюнка, у якого не можна змінити розмір, обрізати або як - то інакше обробити зображення.

4. Структура векторної ілюстрації.

Структуру будь-якої векторній ілюстрації можна представити у вигляді ієрархічного дерева. У такій ієрархії сама ілюстрація займає верхній рівень, а її складові частини - більш низькі рівні ієрархії.

1. Верхній ієрархічний рівень займає сама картинка, яка об'єднує в своєму складні об'єкти + вузли + лінії + заливкі.

2. Наступний рівень ієрархії - об'єкти, які представляють собою різноманітні векторні форми.

3. Об'єкти ілюстрації складаються з одного або декількох контурів: замкнутих і відкритих. Контуром називається будь-яка геометрична фігура, створена за допомогою інструментів векторної програми і представляюча собою обриси того чи іншого графічного об'єкту (окружність, прямокутник і т.п.). Замкнутий контур - це замкнута крива, у якій початкова і кінцева точки збігаються (окружність). Відкритий контур має чітко позначені кінцеві точки (синусоїдальна лінія).

4. Наступний рівень ієрархії складають сегменти, які виконують функції цеглинок, які використовуються для побудови контурів. Кожен контур може складатися з одного або декількох сегментів. Початок і кінець кожного сегмента називають вузлами, або опорними точками, оскільки вони фіксують положення сегмента, «прив'язуючи» його до певної позиції в контурі. Переміщення вузлових точок призводить до модифікації сегментів контуру і до зміни його форми. Замкнуті контури (форми) мають властивість заповнення кольором, текстурою або растровим зображенням. Заливка - це колір або візерунок, виведений в замкнутій області, обмеженій кривою.

5. На самому нижньому рівні ієрархії розташовані вузли та відрізки ліній, що з'єднують між собою сусідні вузли. Лінії поряд з вузлами виконують функції основних елементів векторного зображення.

Математичні основи векторної графіки

Як було зазначено вище, якщо основним елементом растрової графіки є піксель (точка), то в векторній графіці в ролі базового елементу виступає лінія. Це пов'язано з тим що будь-який об'єкт в ній складається з набору ліній, з'єднаних між собою вузлами. Як уже зазначалося раніше, окрема лінія, що з'єднує сусідні вузли, називається *сегментом* (в геометрії їй відповідає відрізок). Сегмент може бути заданий за допомогою рівняння прямої або рівняння кривої лінії, вимагаючи для свого опису різної кількості параметрів. Для більш повного розуміння механізму формування векторних об'єктів розглянемо способи представлення основних елементів векторної графіки: *точки, прямої лінії, відрізка прямий, кривої другого порядку, кривої третього порядку, кривих Безьє.* У векторній графіці *точці* відповідає *вузол.* На площині цей об'єкт перед представляється двома числами **(X, Y),** які задають його положення щодо початку координат. Для опису *прямої лінії* використовується рівняння $\mathbf{Y = aX + b.}$ Тому для побудови даного об'єкта потрібно завдання всього двох параметрів: **a** і **b.** результатом буде побудова нескінченної прямої в декартових координатах. На відміну від прямої *відрізок прямої* вимагає для свого опису двох додаткових параметрів, що відповідають початку і кінцю відрізка (наприклад, $\mathbf{X_1}$ $\mathbf{X_2}$**).** До класу *кривих другого порядку* відносяться параболи, гіперболи, еліпси і окружності, тобто всі

лінії, рівняння яких містять змінні в ступені не вище другого. У векторній графіці ці криві використовуються для побудови базових форм (примітивів) у вигляді еліпсів і кіл. Криві другого порядка не мають точок перегину. Классичне рівняння, що використовується для опису цих кривих вимагає для свого задання п'ять параметрів:

$$x^2 + a_1 y^2 + a_2 xy + a_3 x + a_4 y + a_5 = 0.$$

Для побудови відрізка кривої потрібно задати два додаткові параметри. На відміну від кривих другого порядку, *криві третього порядку* можуть мати точку перегину. Наприклад, графік функції **Y = X³** (рис. 2) включає точку перегину на початку координат (0, 0). Саме ця особливість даного класу функцій позволяє використовувати їх в якості основних кривих для моделювання різних природних об'єктів у векторній графіці. Слід зазначити, що згадані раніше прямі і криві другого порядку є окремим випадком кривих третього порядку.

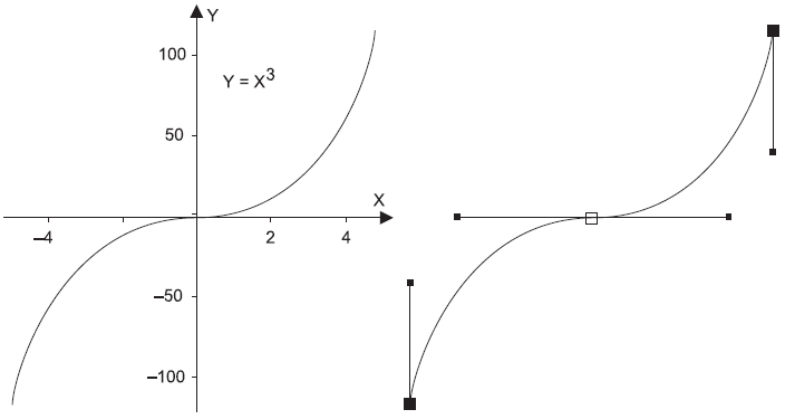


Рис. 2 – Подання кривої лінії за допомогою кривих третього порядку: *зліва* - класичний варіант; *праворуч* - крива Безьє

Класичне рівняння, яке використовується для опису рівняння третього порядку, вимагає для свого завдання дев'яти параметрів:

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

Для опису відрізка кривої третього порядку потрібно на два параметри більше. *Криві Безьє* - це приватний вид кривих третього порядку, що вимагає для свого опису меншої кількості параметрів - восьми замість одинадцяти. В основі побудови кривих Безьє лежить використання двох дотичних, проведених до крайніх точок відрізка лінії. На кривизну (форму) лінії впливає кут нахилу і довжина відрізка дотичної, значеннями яких можна управляти в інтерактивному режимі шляхом перетягування їх *кінцевих точок.* Таким чином, дотичні виконують функції віртуальних важелів, позволяющих управляти формою кривої.

Текстові об'єкти

Сучасні графічні векторні програми надають повноцінні можливості для роботи як з графічними, так і з текстовими об'єктами. За допомогою них до тексту, як різновиду об'єктів, може бути застосований весь спектр потужних засобів графики, недоступний для більшості текстових процесорів. Причому ці засоби доступні як для окремих символів і рядків тексту, так і для великих абзаців. Основу текстових об'єктів складають символи, організовані в *шрифти.*

5. Шрифти у векторній графіці.

Шрифт - це загальний термін, яким називають набір друкованих або відобравантажують текстових символів певного стилю (наприклад, жирний або курсив) і певного розміру (наприклад, 10 пунктів), що мають конкретне накреслення (наприклад, Times New Roman). Перш термін «шрифт» був зрозумілий тільки графічним дизайнерам, видавцям і тим, хто мав справу з великими друкованими пресами, але не з периферією накнязівських комп'ютерів. Ті, хто використовував тоді комп'ютери, звикли бачити на комп'ютерному екрані досить грубі, моноширинні букви і цифри, які в кращому випадку виглядали як видрукувані на друкарській машинці. Але в 1984 р відбулися дві події, які стали визначальними для шрифтів. По-перше, компанія Apple Computer представила комп'ютер Macintosh, по- друге, Hewlett-Packard випустила перший принтер LaserJet. Macintosh запропонував комп'ютерному світу концепцію множинних шрифтів, які дійсно виглядали як шрифти в книгах і журналах. Більше не було необхідності використовувати символи, в яких заголовна W мала таку ж ширину, що і рядкова.

За способом організації шрифти поділяються на дві великі групи: *растрові* та *векторні*. *Растрові шрифти* (bitmap font) представляють собою точкові (растрові) зображенія, які були розглянуті нами раніше. Вони добре пристосовані для швидкого виведення на екран. Іншими словами, це спеціальні, службові шрифти, які використовує для своїх потреб комп'ютер. У складі Windows поставляється кілька базових растрових шрифтів, включаючи MS Serif, MS SansSerif, Courier, Small Fonts і Symbol. При бажанні їх число можна збільшити шляхом установки додаткових fon-шрифтів. Оскільки растровий шрифт являє собою набір точок (пікселів), він погано піддається масштабування, тобто не існує ефективного способу зміни розмірів шрифту і доводиться для кожного кегля зберігати окремі набори символів. Спроба масштабування такого шрифту при відчутному коеффіцієнті збільшення призводить до появи так званого сходового ефекту, коли символи здаються створеними з великих блоків без згладжування стиків. Для мінімізації цього ефекту застосовують операцію згладжування, призначення якої полягає в спеціальному розмиття кордонів (кромок) растрових об'єктів шляхом поміщення додаткових пікселів проміжного кольору (тону) між граничними пікселями і пікселями фону. Однак при невеликому дозволі застосування згладжування до символів маленького розміру (менше 10 пунктів) призводить до їх змазування (рис. 3).

Рис . 3 – Збільшені букви реєстрового шрифту з застосуванням згладжування для дозволу : *1* - 300 dpi; *2* - 72 dpi

Растрові шрифти для принтера і екрану були популярні в видавничих системах раніше через те, що процесори у комп'ютерів і принтерів володіли недостатнім швидкодією, тому формування зображень букв заново при кожному перемальовуванні сторінки на екрані або на

принтері займало дуже багато часу. Зараз растрові шрифти використовуються досить рідко, їх практично витеснили векторні шрифти.

Векторні шрифти (outline font) є математичною моделлю, де кожен символ складається з набору точок (вузлів), з'єднаних лініями таким чином, що вони утворюють контур символу. Тому такі шрифти на- викликають також *контурними* (масштабованими). Так само як і розглянуті раніше об'єкти, їх описують за допомогою будь-яких математичних засобів (Векторів, дуг, кривих Безьє, сплайнів і т. П.). На рис. 4 наведено типовий вид символів векторного шрифту при відображенні їх в CorelDRAW в режимі Wireframe (Каркас). Векторні шрифти можуть легко масштабироваться шляхом зміни пропорцій між точками, які, в свою чергу, змінюють довжину ліній, що з'єднують ці точки. Однак якісне відтворення векторних шрифтів при малих розмірах також пов'язано з серйозними проблемами, т. к. вони утворені невеликим кількістю пікселів, що сильно огрубляет контур знака. До недавнього часу на персональних комп'ютерах переважали два формати цифрових векторних шрифтів: Type 1 фірми Adobe (часто ці шрифти називаючиються «PostScript-шрифтами») і TrueType фірми Microsoft. Найчастіше в одній і тій же програмі є сусідами не тільки шрифти в обох форматах, але навіть одні й ті ж гарнітури і в TrueType, і в Type 1 поданні. У Windows 2000 введений новий тип векторного шрифту, званий OpenType, який увібрав в себе властивості як шрифтів типу Type 1, так і TrueType.

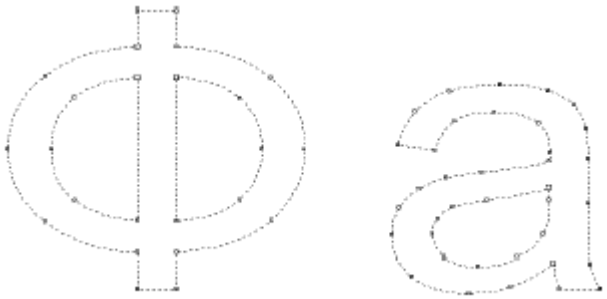


Рис . 4 – Як і будь-які інші векторні об'єкти , символи векторного шрифту складаються з вузлів і з'єднують їх сегментів (контурів)

Розглянемо властивості і можливості кожного з перерахованих векторних форматів шрифтів більш докладно.

Формат Type 1

Цей формат був створений фірмою Adobe Systems Inc. в 1985 р, а в 1990 р публічно розкритий і документований. Даний формат повністю сумісний з мовою описания сторінок PostScript, випущеним в тому ж 1985 року і підтримується всіма PostScript-пристроями. Фірма Adobe створила безліч шрифтових форматів, заснованих на мові опису сторінок PostScript. Найпоширеніший з них - Type 1. Символи в шрифтах Type 1 описуються за допомогою кривих Безьє. Завдяки мові PostScript формат Type 1 більш, ніж TrueType, сумісний з про- грамами Adobe, він не вимагає конвертації при друку на PostScript-пристроях і, як наслідок, породжує менше помилок. Шрифт в форматі Type 1 (або PostScript) відрізняється від TrueType головним чином тим, що контур будується з кривих третього, а не другого порядку. использова- ня кривих вищого порядку і обумовлює основні переваги

PostScript-шрифтів перед шрифтами TrueType. За рахунок більшого числа ступенів волі PostScript-лінія не має зламів в точках сполучення фрагментів, що дозволяє точніше, ніж TrueType, передавати їх форму на друку. Інакше кажучи, символи шрифту Type 1 є більш гладкими, ніж TrueType. Шрифт Type 1 (PostScript) для Windows складається з двох компонентів: растрового (екранного) PFM (PostScript Font Metrics) і векторного (принтерного) (PostScript Font Binary) файлів або з трьох файлів: файлу з розширенням PFB, який містить інформацію про контури символів; текстового файлу з розширенням AFM (Adobe Font Metrics), що містить ін-формацію про ширину символів і кернінг; текстового файлу INF , що містить додаткову інформацію, яка потрібна для інсталяції. В процесі інсталяції Windows генерує PFM -файл, в основі якого лежить інформація з AFM- і INF -файлів. Далі використовуються тільки PFB- і PFM -файли. Деякі виробники генерують PFM -файли самостійно і поставляють своїм клієнтам тільки два цих файли. Цього достатньо для нормального ви-користування. Деякі також додають AFM -файли, а деякі постачають всі 4 файли.

Формат TrueType

Шрифтовий формат TrueType був розроблений в середині 80-х рр. компанією Apple для операційної системи комп'ютерів Macintosh. Сьогодні ж під такими шрифтами, як правило, мають на увазі TTF -шрифти фірми Microsoft. TrueType-шрифти створені на мові опису сторінок TrueImage і використовують для формування контуру символу криві другого порядку. Кожна ділянка контуру символу характеризується або задається двома точками (кордонами ділянки) і напрямом лінії на кожній з меж. Часто для завдання напряму використується третя точка, що лежить на перетині дотичних до кривої на її кінцях. Отже, для побудови шрифтових знаків в TrueType застосовуються квадратичні сплайни. Утворені ними фрагменти символів малюються на екрані швидше, ніж криві Безьє, що застосовуються для побудови символів в PostScript-шрифти. Їх недолік - менша точність відображення кривих при друці в порівнянні з кривими Безьє. Шрифти TrueType спроектовані таким чином, щоб однаково чітко виводити на екран і друк при будь-якому кеглі. При друку шрифтів TrueType на струменевому принтері, лазерних принтерах або TrueImage-сумісних пристроях інформація, яка використовується комп'ютером, передається безпосередньо на принтер. При друку TrueType-шрифтів на принтері PostScript драйвер принтера зазвичай перетворює їх в PostScript-сумісні шрифти (наприклад, Type 1), що призводить до втрати чи спотворення частини інформації про форму шрифту, товщині ліній. При друку TrueType-шрифтів на лазерному принтері драйвер посилає їх непосредственно на принтер або перетворює в растрові зображення. Шрифт TrueType складається з одного файлу TTF . У Windows шрифти TrueType розміщуються в папці Windows \ Fonts . Відзначимо, що з точки зору користувача використання шрифтів TrueType дещо простіше (наприклад, не вимагає установки спеціальних програм типу ATM), а самі шрифти більш поширені і коштують дешевше Type 1. Крім того, користувачеві, кінцевим продуктом якого є роздрукований на принтері лист, не потрібно мати високу якість растеризации за допомогою. Але якщо мова йде про підготововку матеріалів для поліграфії (і отже – фотоскладального апарату, який працює на PostScript), то різницю стає відчутною. Шрифти TrueType (TTF) поставляються в систему Windows різними додатками. У тих випадках, коли шрифти приходять з додатком, таким як CorelDRAW,

управління шрифтами бере на себе операційна система. Ці шрифти стають доступними всім програмам, які підтримують TrueType Fonts. OpenType - нова шрифтова технологія, розроблена спільно компаніями Adobe і Microsoft. Специфікації були опубліковані в 1997 році, перші шрифти були випущені в 2000 році в ОС Windows, де цей тип шрифтів широко представлений поряд з двома групами традиційних шрифтів.

Шрифти OpenType нагадують шрифти TrueType, але можуть містити шрифтові дані будь-якого з двох форматів: як PostScript, так і TrueType. відповідно, вони об'єднують можливості двох конкуруючих технологій, що забезпечує їм додаткову перевагу - міжплатформову сумісність. Розглянемо деякі переваги шрифтів OpenType. OpenType підтримує кодування Unicode. Завдяки цьому шрифти можуть містити понад 65 000 символів, включаючи всі західні символи і повні набори символів з багатьох нелатинських алфавітів (наприклад, японських або китайських), лігатури, а також широкий спектр надрядкових, підрядкових, математичних символів, буквиц, особливі форми букв, інші спеціальні знаки. Те, що раніше не містилося в 256-символьний файлі і виносилося в розширні шрифтові набори (expert set), технологія OpenType вміщує в один файл. Шрифти OpenType можуть ефективно стискуватися. Менший розмір файлу шрифту полегшує його впровадження в інші файли. Це важливо для PDF-файлів і веб-сторінок. Техніка стиснення залежить від типу шрифту OpenType. Для стистення шрифтів PostScript OpenType застосовується Adobe Compact Font Format (CFF). Для стиснення шрифтів TrueType OpenType застосовується Agfa MicroType Express. Покращена також типографіка, тобто шрифти OpenType можуть включати широкий набір гліфів, включаючи лігатури, дроби, історичні символи (числа старого стилю, капітельні символи) і ін. Вони допускають автоматичну заміну визна ленних символів на них. Шрифти OpenType можуть містити кілька варіантів оптичних размірів шрифту всередині одного сімейства шрифтів, при цьому шрифти різного розміру грунтуються на різних наборах векторних контурів, для покращеного відображення на екрані і підвищеної читабельності символів малого розміру. Покращений кернинг, де символи з однотипною формою (наприклад, лівий край символів c, e, d) можуть кернінговатися ідентично. Це зменшує розмір таблиць кернінгу і розширює число кернінгових пар.

Контрольні запитання

1. Що таке векторна графіка?
2. Назвіть переваги та недоліки векторної графіки.
3. Що таке криві Без’є?
3. Які об’єкти відносяться до векторної графіки?
4. Опишіть структуру векторної ілюстрації.
6. Чим растрові шрифти відрізняються від векторних? Наведіть приклади.
7. Назвіть формати шрифтових файлів.

Список використаних інформаційних джерел

8. Маценко В.Г. Комп’ютерна графіка: Навчальний посібник. – Чернівці: Рута, 2009 – 343 с.
9. https://www.youtube.com/watch?v=TswPqn5bdeI&t=573s – векторна та растрова види графіки.

10. https://www.youtube.com/channel/UCF6F8LdCSWlRwQm_hfA2bcQ – курс відео про комп'ютерну графіку.
11. https://www.youtube.com/watch?v=ZJSCl6XEdP8 – серія уроків про формат векторної графіки SVG.
12. https://medium.com/coding-artist/a-beginners-guide-to-vector-graphic-design-815cb1cb4d70
13. https://www.youtube.com/watch?v=Xu28TZnH1kU – відео щодо роботи з векторною графікою.
14. https://svg-art.ru/?page_id=897#comm_m – стаття про роботу з елементом Path.
15. http://tutorials.jenkov.com/svg/svg-viewport-view-box.html – стаття про атрибути viewBox та ViewPort.

ЛЕКЦІЯ 5.

ТЕМА: РАСТРОВА ГРАФІКА

План

1. Растрові зображення та їх основні характеристики.
2. Виведення зображення на растрові прилади.
3. Методи покращення растрових зображень.
4. Базові растрові алгоритми.
5. Особливості використання растрової графіки.

1. Растрові зображення та їх основні характеристики.

Растрова графіка – це вид комп'ютерної графіки, який має справу зі створеням, обробкою та зберіганням растрових зображень. Растрове зображення є масивом кольорових точок або пікселів. Растрові зображення зберігаються у різних графічних форматах.

Растрове зображення — зображення, яке являє собою сітку (растр), зазвичай прямокутну, пікселів відображених на моніторі, папері та інших відображальних пристроях і матеріалах. Створюються растрові зображення фотоапаратами, сканерами, безпосередньо в растровому редакторі, також шляхом експорту (растеризацією) з векторного редактора або у вигляді знімків екрану. Характеристиками растрового зображення є:

- Кількість пікселів (number of pixels) – зазвичай вказують кількість пікселів по ширині і висоті (наприклад, 1024 × 768, 1920 × 1080)
- Кількість використовуваних кольорів або глибина кольору (color depth), тобто обсяг пам'яті в бітах, що використовуються для одного пікселя (наприклад, 24 біта).
- Колірний простір (color space) – RGB, CMYK, XYZ, YCbCr та інші.
- Роздільна здатність (resolution) – довідкова величина, яка вказує на рекомендований розмір зображення (наприклад, ppi).
- Лінійний розмір надрукованого зораження (наприклад, 10 см *15 см).

Як було сказано вище, растр – це матриця комірок (пікселів). Піксель – це найдрібніша одиниця цифрового зображення в растровій графіці.Він являє собою елемент неподільної форми, зазвичай квадратної та має свій колір. Сукупність пікселів різного кольору утворює зображення. Залежно від розташування пікселів в просторі розрізняють квадратний, прямокутний, гексагональний чи інші типи растра. Можна виділити такі види пікселів:

1. Фізичний піксель (physical pixel) комірка на матриці, одиниця роздільної здатності дисплею пристрою.

2. Віртуальнйи піксель (device independent pixel) незалежний ві пристрою піксель, може бути менше або більше фізичного.

3. Змінений піксель, який з'являється в результаті масштабування зображення на екрані.

Для опису розташування пікселів використовують різноманітні системи координат. Загальним для всіх таких систем є те, що координати пікселів утворюють дискретний ряд значень (необов'язково цілі числа). Часто використовується система цілих

координат - номерів пікселів з (0, 0) в лівому верхньому кутку. Таку систему ми будемо використовувати і надалі, бо вона зручна для розгляду алгоритмів графічного виведення.

Роздільна здатність растру характеризує відстань між сусідніми пікселями - крок дискретної сітки растра. Роздільну здатність вимірюють кількістю пікселів на одиницю довжини. Найбільш популярна одиниця виміру - **ppi** *(pixels per inch)* - кількість пікселів в одному дюймі довжини (2.54 см). Не слід ототожнювати крок з розмірами пікселів - розмір пікселів може дорівнювати кроку, а може бути як менше, так і більше кроку.

Можна сказати, що для КГ найбільш зручний растр з однаковим кроком для обох осей, тобто dpi *X* = dpi *Y*. Це зручно для багатьох алгоритмів виведення графічних об'єктів. Інакше можуть виникати проблеми, наприклад, при рисуванні кола на екрані дисплея.

Форма пікселів растра визначається особливостями будови графічного виведення (рис. 1). Наприклад, пікселі можуть мати форму прямокутника або квадрата, які за розмірами дорівнюють кроку растра (дисплеї моніторів); пікселі можуть мати круглу форму і за розмірами можуть не дорівнювати кроку растра (принтери).

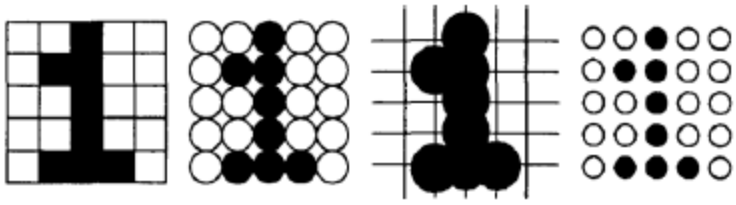


Рис. 1 Одне зображення на різних растрах

Кількість кольорів (глибина кольору) - важлива характеристика будь-якого зображення, не тільки растрового. Відповідно до психофізіологічних дослідженнь, очі людини здатні розрізняти 350 000 кольорів.

З точки зору кольору можна класифікувати зображення наступним чином:

• двобарвні (бінарні) - 1 біт на піксель. Серед двоколірних найбільш часто зустрічаються чорно-білі зображення.
• напівтонові - градації сірого або іншого кольору. Наприклад, 256 градацій (1 байт на піксель).
• кольорові зображення (2 біта на піксель і більше). Глибина кольору 16 бітів на піксель (65536 кольорів) отримала назву **High Color,** 24 біта на піксель (16. 7 млн. кольорів) - **True Color.** У комп'ютерних графічних системах використовують і більшу глибину кольору - 32, 48 і більше бітів на піксель. Як приклад розглянемо растровий рисунок (рис. 2).

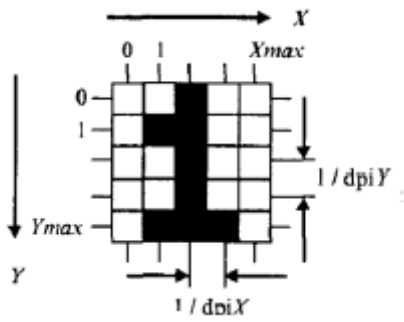


Рис. 2 Растр

Недостатня кількість кольорів призводить до появи зайвих контурів на гладких поверхнях циліндра і кулі.

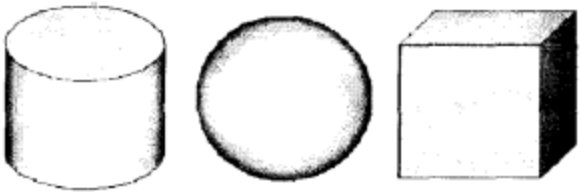


Рис. 3 Фігури 256 градацій сірого, 100 ppi

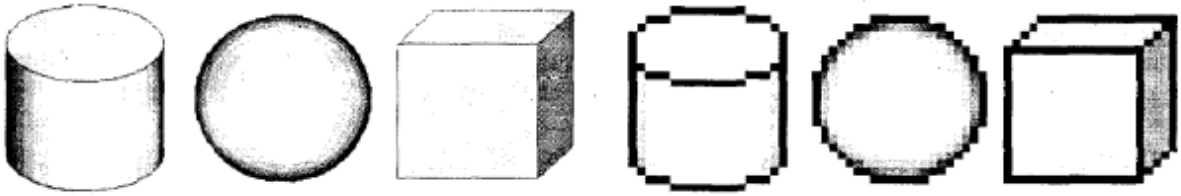


Рис. 4 Фігури 8 градацій сірого, роздільна здатність зменшена в 8 разів

Око людини з нормальним зором здатне розрізняти об'єкти з кутовим розміром близько однієї хвилини. Якщо відстань до об'єкта дорівнює *R*, то можна приблизно оцінити цей розмір *(dP)*, як довжину дуги, що дорівнює *R*a* (рис. 5). Можна припустити, що людина розрізняє дискретність растру (крок) також відповідно до цього мінімально помітного розміру. Інакше кажучи, якщо відстань між окремими точками (пікселями) менше ніж *dP*, то ці точки вже не сприймаються як окремі точки. Тоді можна оцінити мінімальну роздільну здатність растрового зображення, яке людиною вже не сприймається як растрове, наступної величиною; ppi = 25, 4 / *dP* [мм].

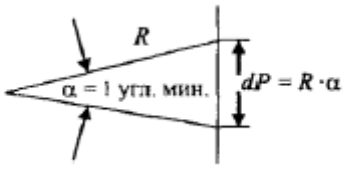


Рис. 5 Оцінка роздільної здатності растра

Наведемо кілька значень dpi для різних *R* (табл.1). Якщо вважати відстань, з якого людина зазвичай розглядає друковані документи, рівним 300 мм, то можна оцінити мінімальну роздільну здатність, при якій вже не помітні окремі пікселі, як приблизно 300 dpi (приблизно 0.085 мм). Лазерні чорно-білі принтери повністю задовольняють такій вимозі.

Таблиця 1

Відстань *R*, мм	**Розмір *Dp*, мм**	**Роздільна здатність, dpi**
500	0,14	181
300	0,09	282

Кількість кольорів - 256 градацій сірого, що дозволяє отримати здатність - приблизно 100 dpi. Відповідно розглядаючи дисплей з відстані 0,5 м мінімальна роздільна здатність відповідає приблизно 200 dpi, якщо в дисплеї роздільна здатність становить 100-120 dpi - це

погано; наприклад, дисплей розміром 15" по діагоналі повинен забезпечувати не 1024x768 пікселів, а вдвічі більше.

3.2 Виведення зображень на растрові пристрої

Відеосистема персонального комп'ютера орієнтована на растровий метод виведення зображення. Монітор є основним пристроєм оперативного виведення інформації. Сьогодні випускаються монітори стандарту SVGA. Якість зображення на екрані монітора визначається як можливостями самого монітора, так і можливостями контролера SVGA (відеоконтролера). Екран дисплейного монітора подається як набір дискретних точок, що утворюють регулярну прямокутну решітку – растр (від лат. rastrum – граблі). Точки растру формують зображення.

Важливими характеристиками моніторів є роздільна здатність, кількість пікселів на дюйм, довжина діагоналі, розмір пікселя та формат. Так, роздільна здатність подається у пікселях, наприклад 1024x768 пікселів.

Слід зазначити, що не в залежності від показника ppi зображення, при виводі на екран монітора буде використовуватися показник ppi монітора. Проте це може впливати на розмір зображення, що переглядається. Таким чином, ми можемо прийти до висносвку, що в тих випадках, коли зображення не планується буди надрукованим, показник ppi не відіграє ключої ролі. Натомість, важливим є розмір зображення.

Для розрахунку показника ppi екрану монітора слід спочатку розрахувати роздільну здатність діагоналі d_p за наступною формулою:

$$d_p = \sqrt{Wp^2 + Hp^2},$$

де *Wp* та *Hp* висота та ширина екрана у пікселях.

Дізнавшись d_p, її необхідно поділити на діагональ пристрою:

$$PPI = \frac{dp}{di},$$

Де *PPI* – кількість пікселів на дюйм; *di* – діагональ пристрою.

Щоб порахувати кількість пікселів на 1 см, отримане значення слід поділити на 2,54.

Для ілюстрації впливу значення ppi на виведення зображення наведемо наступний приклад. Так, якщо вивести зображення квадрату розміром 109x109 пікселів на монітори з ppi 109 та 72 пікселі відповідно, у другому випадку зображення буде здаватися більшим (рис.).

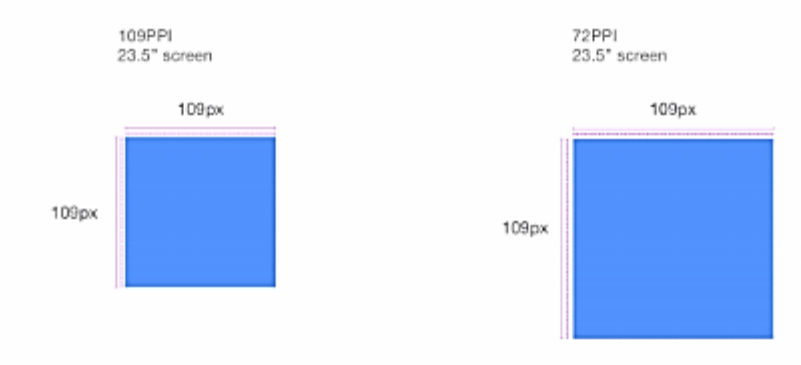


Рис. 6 Виведення зображення на монітори з різним показником ppi

Альтернатива растровим пристроям - векторні пристрої виведення зображень. У цих пристроях інструмент промальовує тільки зображувані фігури, і його траєкторія руху визначається виведеним зображенням. Зображення складається з графічних примітивів, якими можуть бути відрізки прямих - вектори (звідки і назва методу виведення), дуги, кола. До векторних пристроїв виведення статичних зображень відносяться плоттери.

Принтер це пристрій для одержання твердих копій цифрових зображень, для друку на папері тексту, графіків, зображень, креслень тощо. Нині найрозповсюджені лазерні, струменеві, матричні принтери. Лазерні принтери володіють широким діапазоном роздільної здатності: 300dpi, 600 dpi, 1200 dpi (для видавничих систем, що забезпечує досить високу якість зображення). Роздільна здатність принтера вимірюється в точках на дюйм. Наприклад, 300 dpi означає здатність принтера надрукувати на відрізку довжиною один дюйм 300 окремих точок.

Для того, щоб визначити, який розмір зображення може бути надруковано в високій якості, слід поділити ширину та висоту зображення на бажаний показник dpi та перевести отримані значення в сантиметри. Припустимо, зображення має розмір 1200x1500 пікселів. Бажаною роздільною здатністю є показник 300 dpi. Далі розрахуємо:

1200/300*2,54=10,16 см

Виконаємо розрахунок для другого розміру:

1500/300*2,54=12,7 см

Це означає, що дане зображення може бути надруковано з показником 300 dpi в розмірах 10,16x12,7 см.

3. Методи покращення растрових зображень

Розглянемо деякі з існуючих методів покращення якості зображень, які грунтуються на суб'єктивному сприйнятті роздільної здатності і кількості кольорів. При одних і тих же значеннях технічних параметрів пристрою графічного виводу можна створити ілюзію збільшення роздільної здатності або кількості кольорів. Причому, суб'єктивне поліпшення однієї характеристики відбувається за рахунок погіршення іншого.

Спершу розглянемо методи усунення ступеневого ефекту. З формальної точки зори ступеневий ефект, або aliasing, являє собою накладення сигналів рпи їх дискретизації. Тобто частина інформації про зображенні, яке підлягає рендерингу, втрачається, так як кількість пікселів на екрані є обмеженою.Часто під аліасінгом розуміють ефект сходинок на прямих лініях.

Термін алиасінг був вперше введений в області обробки сигналів, в якій він спочатку описував ефект, що виникає, коли різні безперервні сигнали стають невиразними (або починають спотворювати один одного) при дискретизації. У 3D-рендеринга цей термін зазвичай має більш конкретне значення: він відноситься до безлічі небажаних артефактів, які можуть виникати, коли 3D-сцена рендерится для відображення на екрані, що складається з фіксованої сітки пікселів. Приклад аліасінгу можна побачити на рисунку 7.

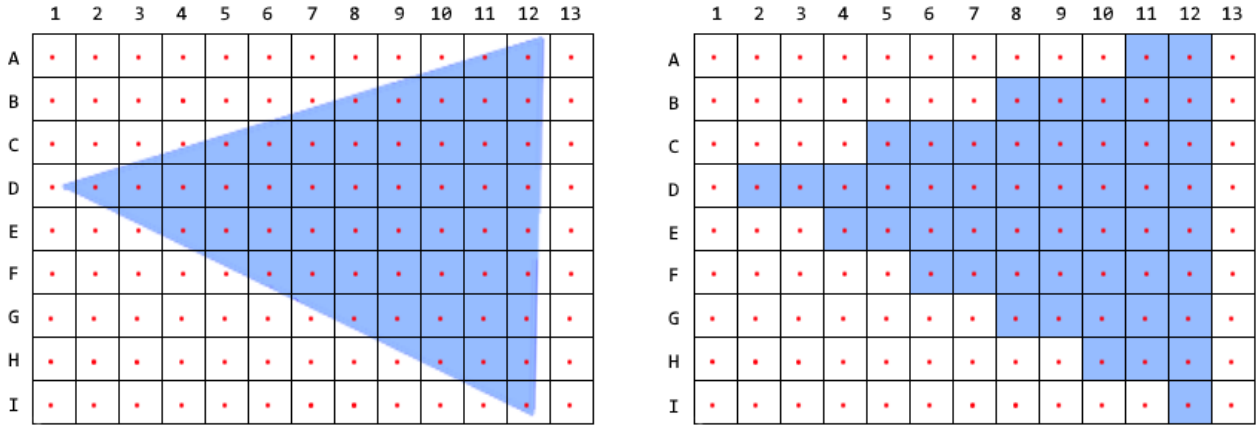


Рис.8 Приклад аліасінгу

Аліасінг прийнято поділяти на такі дві групи, просторовий (spatial) та часовий (temporal).

На рисунку 8 (а,б,в) показаний алиасінг в простій сцені, що складається з єдиного білого трикутника на чорному тлі. На етапі растеризації стандартного рендеринга семплюється центральна позиція кожного пікселя: якщо він знаходиться в трикутнику, то піксель буде зафарбований білим, в іншому випадку він закрашивается чорним. В результаті виходить добре помітний ефект «драбинки», один з найвідоміших артефактів аліасинга.

При ідеальному згладжування для кожного пікселя визначається, яка частина його площі закрита трикутником. Якщо піксель закритий на 50%, то він повинен бути заповнений кольором на 50% між білим і чорним (середнім сірим). Якщо він закритий менше, то повинен бути пропорційно темніше, якщо більше - то пропорційно світліше. Повністю закритий піксель є білим, повністю незакритий - чорним. Результат цього процесу показаний на четвертому малюнку. Однак виконання цього обчислення в реальному часі в загальному випадку є складним завданням.

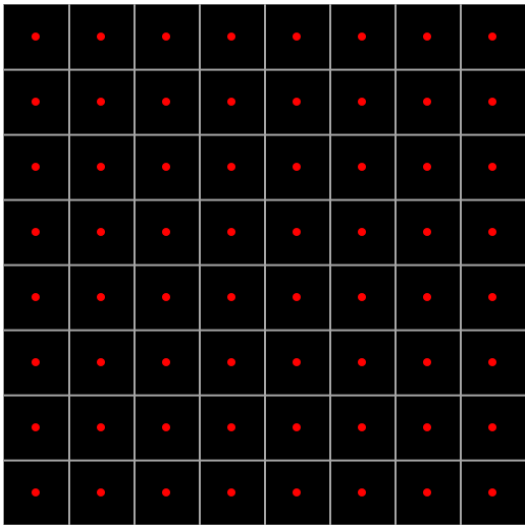


Рис.8а Сітка 8x8 зі згладженими краями

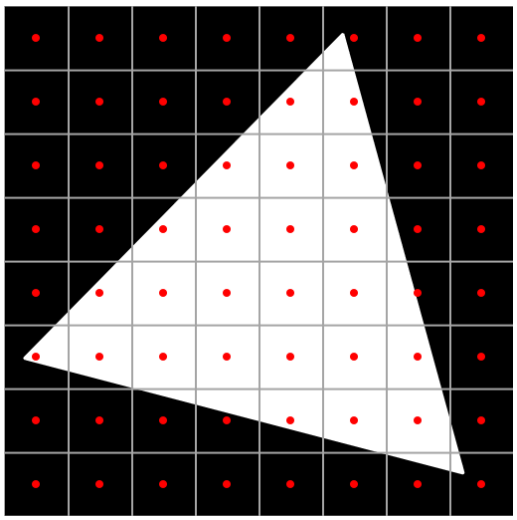


Рис.8б Сітка 8x8 з трикутником

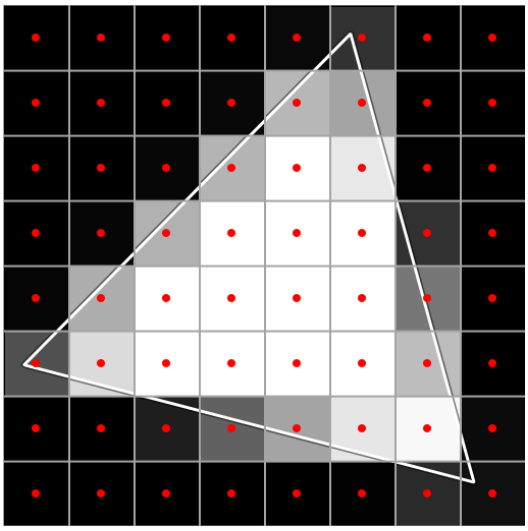


Рис.8в Сітка 8x8 зі згладженими краями

Хоча всі артефакти аліасінга можна звести до проблеми дискретизації уявлення безперервного сигналу на фіксованій сітці, що складається з обмеженої кількості пікселів, конкретні причини їх виникнення дуже важливі для вибору способу їх ефективного способу згладжування. Як буде видно надалі, деякі методи антиаліасинга можуть справлятися з простим геометричним аліасингом, показаним на рисунку 8, але терпіти невдачу при виправленні аліасинга, створюваного іншими процесами рендеринга.

Тому щоб в повній мірі обговорити відносні сильні і слабкі сторони технік згладжування, ми згрупували артефакти аліасинга, що виникають при 3D-рендеринга, в п'ять окремих категорій. Це групування залежить від точних умов генерування артефактів. На рисунку 2 показані ці типи аліасинга на реальному прикладі, відрендерене за допомогою OpenGL.

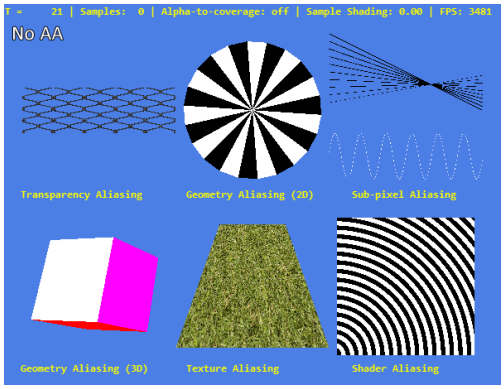


Рис.9 Різні типи аліасінгу

Найпоширенішим типом аліасингу, про який ми вже говорили, є **геометричний алиасінг.** Цей артефакт виникає, коли якийсь примітив сцени (зазвичай трикутник) частково перетинається з пікселем, але це часткове перекриття не враховується в процесі рендеринга.

Алиасінг **прозорості** виникає в текстурованих примітивах з частковою прозорістю. Верхнє ліве зображення на рисунку 9 створене з використанням одного прямокутника, заповненого частково прозорою текстурою сітчастого паркану. Оскільки сама текстура - це просто фіксована сітка пікселів, її потрібно семпліювання в точках, на які накладається кожен піксель зображення, і для кожної такої точки має прийматися рішення, чи потрібна в ньому прозорість. В результаті виникає та ж проблема семпліювання, яку ми вже зустрічали на геометрії.

Незважаючи на те, що фактично він є типом геометричного аліасінгу, **підпіксельний алиасінг** вимагає особливого розгляду, так як він ставить унікальні завдання для аналітичних методів згладжування, які нещодавно отримали велику популярність в рендерингу компю’терних ігор. Ми детально розглянемо їх нижче. Підпіксельний алиасінг виникає тоді, коли структура, що раструється, накладається менш ніж на один піксель в сітці буфера кадрів. Таке найчастіше відбувається в разі вузьких об'єктів - шпилів, телефонних або електричних ліній, або навіть мечів, коли вони знаходяться досить далеко від камери.

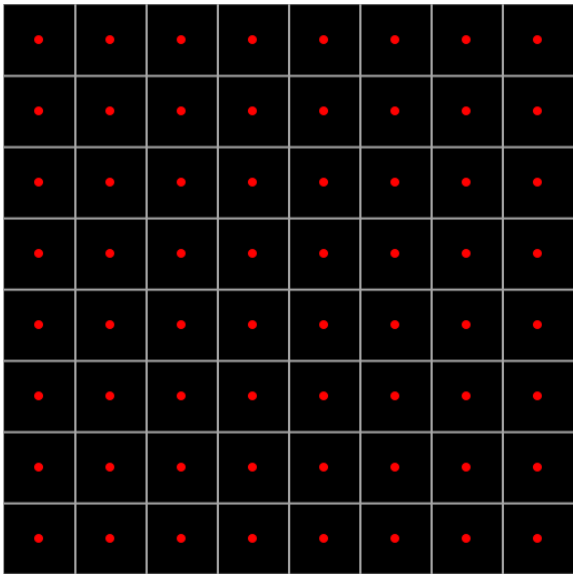


Рис.10а Підпіксельний аліасінг

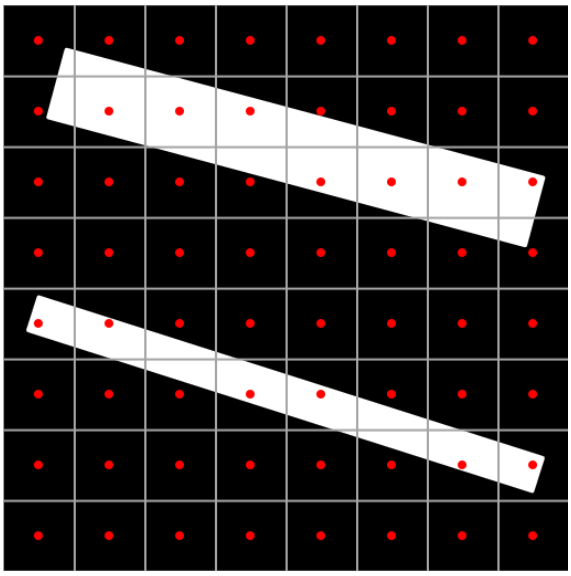


Рис.10б Підпіксельний аліасінг

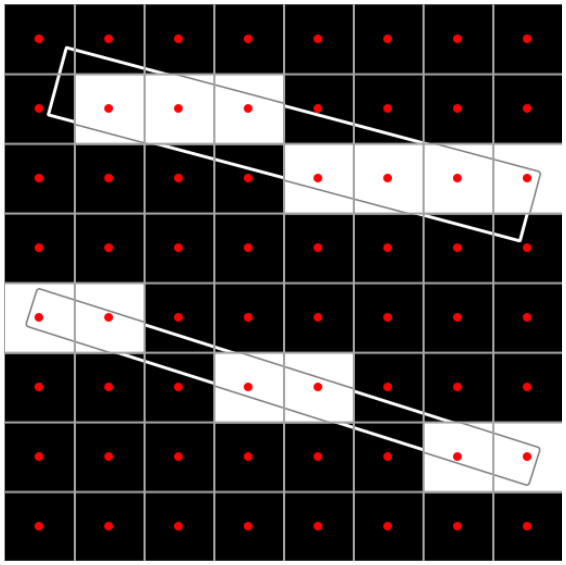


Рис.10в Підпіксельний аліасінг

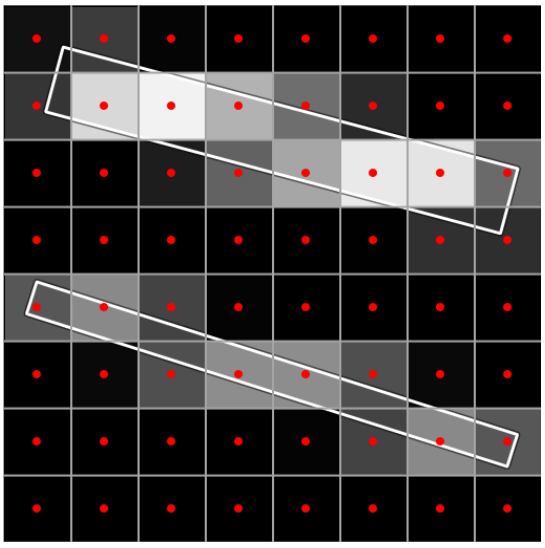


Рис.10г Підпіксельний аліасінг

На рисунку 10 показаний подпіксельний алиасінг в простій сцені, що складається з двох відрізків прямих. Верхній має ширину в один піксель, і хоча при растеризації він демонструє знайомий артефакт- «драбинку» геометричного аліасинга, результат все одно в цілому відповідає за формою вхідних даних. Нижній відрізок має ширину півпікселя. При растеризации частина перетинаються їм стовпців пікселів не має одного центру пікселя в межах відрізка. В результаті він розділяється на кілька незв'язаних фрагментів. Те ж саме можна помітити на прямих лініях і кривої синусоїди на рисунку 9.

Текстурний алиасінг виникає при недостатньому семплірuванні текстури, особливо у випадках анізотропного семпліювання (це випадки, коли поверхня сильно нахилена відносно екрану). Зазвичай артефакти, створювані таким типом аліасинга, не очевидні на нерухомих скріншотах, але проявляються в русі як мерехтіння і нестійкість пікселів. На рисунку 11 показано на кількох кадрах програми-прикладу в режимі анімації.

Рис. 11 Анімована високочастотна текстура з артефактами мерехтіння

Текстурний алиасінг зазвичай можна запобігти використанням mip-текстурування і фільтрацією високоякісних текстур, але він все одно іноді залишається проблемою, особливо з деякими версіями драйверів популярних відеопроцесорів, субдискретизующих високоанізотропних текстур. На нього також впливають різні методи антиалиасінгу, тому він теж включений в приклад.

І, нарешті, **шейдерний алиасінг** виникає, коли програма пиксельного шейдера, виконується для кожного пікселя і визначає його колір, генерує результат з алиасингом. Таке часто трапляється в іграх з шейдерами, що створюють контрастне освітлення, наприклад, дзеркальні засвітки на підставі карти нормалей, або з техніками контрастного освітлення типу cel shading або заднім підсвічуванням. У прикладі це апроксимується одним шейдером, обчислює функцію синуса для координат текстури і зафарбовувати все негативні результати чорним, а всі позитивні - білим.

Розглянемо техніки згладжування на основі семплювання. Вивчивши артефакти аліасингу та всіх типів аліасингу, які можуть виникнути при рендерингу 3D-сцени, ми можемо почати дослідження технік антиалиасінгу. Ці техніки можна розбити на дві категорії: техніки, які намагаються знизити алиасінг збільшенням кількості семплів та техніки, які намагаються

пом'якшити артефакти аліасингу аналізом і постобробкою згенерованих зображень. Категорія технік згладжування на основі сем плювання більш проста, тому варто почати з неї.

Давайте знову розглянемо наш перший приклад з трикутником в сітці 8 × 8 пікселів. Проблема зі стандартним рендерингом полягає в тому, що ми семпліруємо тільки центр кожного пікселя, що призводить до генерування потворною «драбинки» на ребрах, які не є повністю горизонтальними або вертикальними. З іншого боку, обчислення покриття кожного пікселя неможливо в реальному часі.

Інтуїтивним рішенням буде просте збільшення кількості семплів, узятих на піксель. Ця концепція показана на рисунку 5.

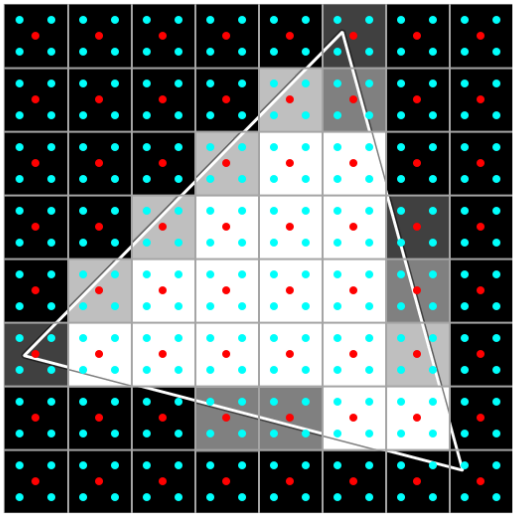


Рис. 12 Трикутник, растеризований з чотирма впорядкованими семплами на піксель

Центри пікселів знову помічені червоними крапками. Однак в кожному пікселі семплюється насправді чотири окремих місця (вони позначені бірюзовими точками). Якщо трикутник не закриває жоден з цих семплів, то піксель вважається чорним, а якщо закриває їх усіх, то білим. Тут цікава ситуація, коли закрита тільки частина пікселів: якщо закритий один з чотирьох, то піксель буде на 25% білим і на 75% чорним. У разі двох з чотирьох співвідношення 50/50, а при трьох закритих семплах результатом буде світліший відтінок в 75% білого.

Ця проста ідея є фундаментом всіх методів антиалиасінга на основі семплювання. У цьому контексті також варто зауважити, що коли кількість семплів на піксель прямує до нескінченності, то результат цього процесу буде прагнути до «ідеального» згладженому наприклад, показаному раніше. Очевидно, що якість результату сильно залежить від кількості використаних семплів - але і продуктивність теж. Зазвичай в іграх використовується 2 або 4 семплів на піксель, а 8 і більше зазвичай застосовуються тільки в потужних PC.

Існують і інші важливі параметри, зміна яких може впливати на якість отриманих результатів методів антиалиасинга на основі семплювання. В основному це розташування семплів, тип семплів і групування семплів.

Розташування семплів всередині пікселя сильно впливає на кінцевий результат, особливо в разі невеликої кількості семплів (2 або 4), яке найчастіше використовується в графіці реального часу. У попередньому прикладі семпли розташовуються так, як ніби вони є центрами відрендереного зображення в чотири рази більше вихідного (16 × 16 пікселів). Це інтуїтивно зрозуміло і легко досягається простим рендерингом зображення більших розмірів. Цей метод відомий як антиаліасинг на впорядкованій сітці (ordered grid anti-aliasing, OGAA), також його іноді називають субдіскретізацією (downsampling). Зокрема, його реалізують примусовим збільшенням дозволу рендеринга в порівнянні з роздільною здатністю монітора.

Однак упорядкована сітка часто неоптимальна, особливо для майже вертикальних і майже горизонтальних ліній, в яких якраз найбільш очевидні артефакти аліасинга. На рисунку 6 показано, чому так відбувається, і як повернена або розріджена сітка семпліювання забезпечує набагато кращі результати:

У 1999 році Ішик і Куніеда представили перший варіант цієї техніки, призначеної для використання в реальному часі, яка виконувалася скануванням пар рядків і стовпців зображення, і могла бути реалізована апаратно.

У загальному випадку, всі чисто аналітичні методи антиалиасінга виконуються в три етапи:

1. Розпізнавання розривів в зображенні.
2. Відтворення геометричних ребер з паттерна розривів.
3. Згладжування пікселів, які перетинають ці ребра, змішуванням кольорів кожної їх боку.

Окремі реалізації аналітичного антиалиасінга розрізняються тим, як реалізовані ці кроки.

Зауважте, що на відміну від MSAA, ці аналітичні методи не цікавить, чи були причинами артефактів аліасинга геометрія, прозорість або навіть обчислення шейдеров. Всі ребра обробляються однаково. На жаль, те ж саме відноситься і до ребер екранного тексту, хоча спотворення при SMAA1x і менше, ніж при FXAA.

Обидва методи не справляються з антиалиасінгом в разі підпіксельного аліасингу, проте вони обробляють цю невдачу по-різному: SMAA1x просто вирішує взагалі не впливати на окремі білі пікселі синусоїди, а FXAA змішує їх з їхнім оточенням. Більш бажана обробка залежить від контексту і особистих переваг.

Більш об'єктивно можна сказати, що SMAA1x обробляє деякі кути ліній в тесті 2D-геометрії і криві в прикладі з шейдерним алиасінгом точно краще, ніж FXAA, забезпечуючи більш гладкий результат, який приємніше естетично і ближче до «ідеального» зображення. Так вийшло завдяки складнішого етапу відтворення ребер і змішування, подробиці якого пояснені в статті про SMAA 2012 року Хіменеса et alia.

Ми отримали достатнє розуміння безлічі методів антиалиасинга (аналітичних і на основі семплінгу), які активно використовуються зараз в іграх. Настав час трохи поміркувати. Які техніки антиалиасінга можна буде побачити на новому поколінні консолей та яким чином

можна буде пом'якшити недоліки наявних методів, і як нове обладнання дозволити використовувати нові алгоритми.

У цьому майже вертикальному випадку ідеальний результат з чотирма семплами повинен мати п'ять різних відтінків сірого: чорний при повністю незакритих семплах, 25% білого при одному закритому семплФ, 50% при двох і так далі. Однак растеризація з упорядкованою сіткою дає нам всього три відтінки: чорний, білий і 50/50. Так відбувається, не тому що впорядковані семпли розташовані в два стовпці, а тому, коли один з них закривається майже вертикальним примітивом, інший теж швидше за все буде закритий.

Як показано на зображенні з розрідженим семплінгом, цю проблему можна обійти, змінивши положення семплів всередині кожного пікселя. Ідеальним розташуванням семплів для згладжування є розріджений. Це означає, що при N семплів ніякі два семпли не мають одного загального стовпчика, рядка або діагоналі в сітці NxN. Методи антиалиасінга, в яких використовуються такі сітки, називають виконуючими антиалиасінг на розріджених сітках (sparse grid anti-aliasing, SGAA).

Найпростіший підхід до антиаліасінг зображення на основі семплювання полягає в тому, що всі обчислення виконуються для «реального» пікселя кожного семплу. Хоча цей підхід високоефективний для видалення всіх типів артефактів аліасинга, він також є дуже обчислювально витратним, тому що при N семплах збільшує в N раз витрати на затінення, растеризування, займану смугу пропускання і пам'ять. Техніки, при яких всі обчислення виконуються для кожного окремого семпли, називаються згладжуванням суперсемплінг (super-sampling anti-aliasing, SSAA).

Приблизно на початку цього століття в графічне обладнання була вбудована підтримка антиалиасінгу мультісемплінг (multi-sample anti-aliasing, MSAA), що є оптимізацією суперсемплінг. На відміну від випадку SSAA, в MSAA кожен піксель затінюється тільки один раз. Однак для кожного семпли обчислюються значення глибини і СТЕНС, що забезпечує ту саму якість згладжування на ребрах геометрії, що і в SSAA, при значно меншій зниженні продуктивності. Крім того, можливі подальші поліпшення продуктивності, особливо зайнятої смуги пропускання, якщо підтримується стиснення Z-буфера і буфера кольору. Вони підтримуються у всіх сучасних архітектурах відеопроцесорів. Через способу оптимізації семпліювання MSAA, з алиасінгом прозорості, текстур і шейдерів таким чином безпосередньо справлятися неможливо.

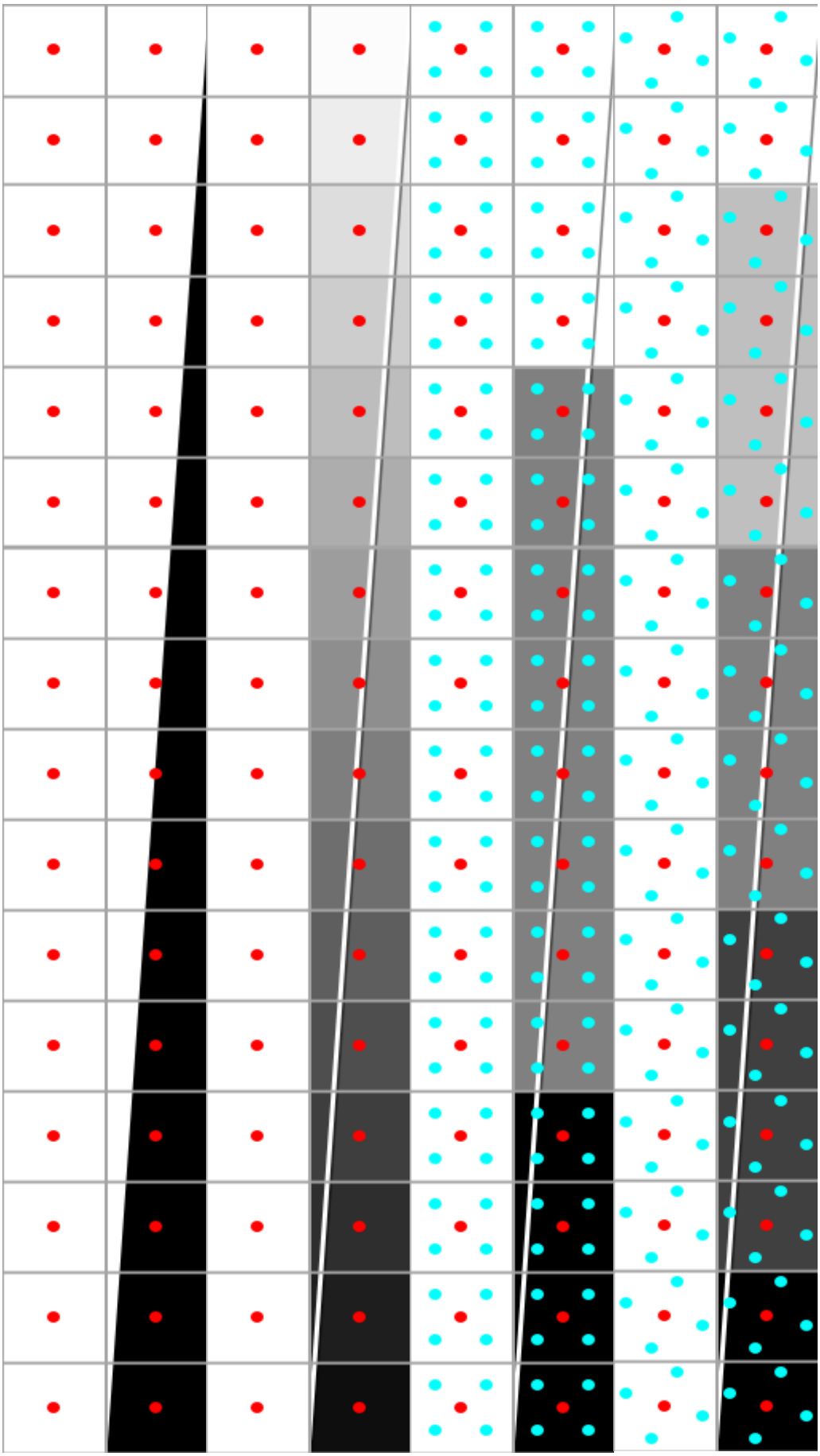


Рис. 13 а) Сцена з майже вертикальною лінією; б) ідеально згладжена пастеризація; в) пастеризація з чотирма впорядкованими семплами; г) згладжування з чотирма розрідженими семплами.

Третій тип семпліювання був представлений компанією NVIDIA в 2006 році в технології антиалиасінгц покриття семпліювання (coverage sampling anti-aliasing, CSAA). MSAA відокремлює затінення від попиксельного обчислення глибини і СТЕНС, а CSAA додає семпли покриття, які не містять значень кольору, глибини або СТЕНС - в них зберігається тільки двійкове значення покриття. Такі виконавчі семпли використовуються для допомоги в змішуванні готових семплів MSAA. Тобто режими CSAA додають семпли покриття до режимам MSAA, але не має сенсу виконувати семпліюання покриття без створення безлічі семплів MSAA. У сучасному обладнанні NVIDIA використовується три режими CSAA: 8xCSAA (4xMSAA / 8 семплів покриття), 16xCSAA (4x / 16), 16xQCSAA (8x / 16) і 32xCSAA (8x / 32). У AMD є схожа реалізація з 4x EQAA (2x / 4), 8xEQAA (4x / 8) і 16xEQAA (8x / 16). Додаткові семпли покриття зазвичай тільки незначно впливають на продуктивність.

Останнім інгредієнтом методів АА на основі семплювання є спосіб угруповання семплів, тобто те, як окремі семпли, згенеровані при рендеринге, збираються в кінцевий колір кожного пікселя. Як показано на рисунку 7, для цієї мети використовуються різні фільтри угруповання. На рисунку показані пікселі 3 × 3 - бірюзові точки позначають позиції семплів, а жовтий відтінок позначає фільтр угруповання семплів.

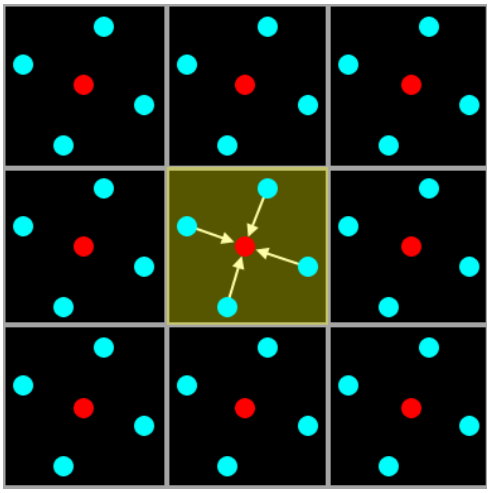


Рис. 14 а Сцена з майже вертикальною лінією

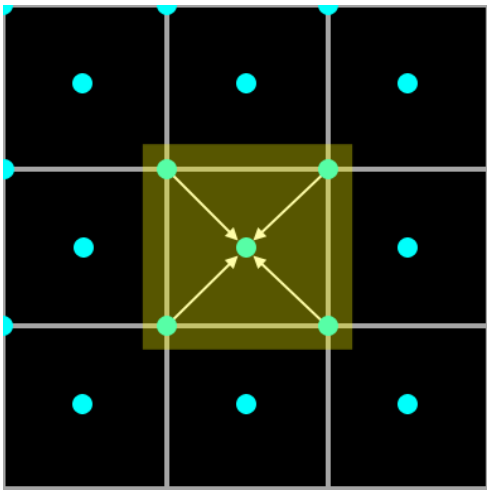


Рис. 14 а Сцена з майже вертикальною лінією

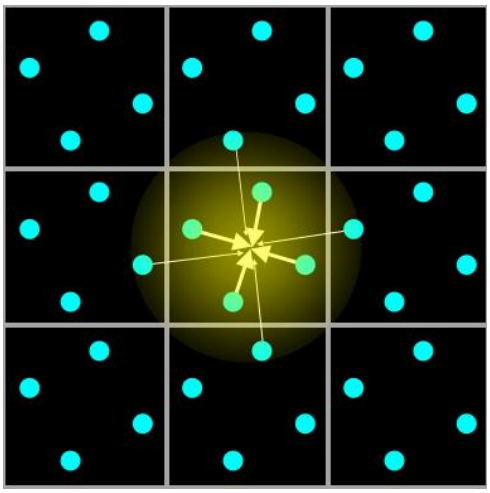


Рис. 14 а Сцена з майже вертикальною лінією

Потрібно зауважити ще один важливий пункт - особливо в світлі подальшого обговорення аналітичних методів AA - всі ці методи на основі семплювання однаково добре застосовуються і до підпіксельного алиасінгу, і в звичайному геометричному алиасінгу.

Техніки на основі семплінгу інтуїтивно зрозумілі і досить добре працюють з досить великою кількістю семплів, але в той же час вони затратні з точки зору обчислень. Ця проблема посилюється, коли застосовуються методи рендеринга (наприклад, відкладене затінення), які можуть ускладнити застосування ефективних типів семпліювання з апаратним прискоренням. Тому досліджуються інші способи зменшення візуальних артефактів, створюваних алиасінгом при 3D-рендерингу. Такі методи рендерять звичайне зображення з одним семплом на піксель, а потім намагаються ідентифікувати і усунути аліасінг, аналізуючи зображення.

Хоча ідея згладжування згенерованих комп'ютером зображень популяризована завдяки статті Решетова 2009 року щодо морфологічного антиалиасінгу (який часто називають MLAA), вона ні в якому разі не є новою. Жуль Блументаль привів стислий опис цієї техніки в своїй статті 1983 роки для SIGGRAPH «Edge Inference with Applications to Antialiasing», яка активно застосовується в сучасних методах:

«Ребро, що семплюється по точках для відображення на растровому пристрої, і непаралельно осі дисплея, виглядає як драбинка. Цей артефакт аліасингу часто виникає в комп'ютерних зображеннях, згенерованих дво- та тривимірними алгоритмами. Точна інформація про ребра часто більше недоступна, але з безлічі вертикальних і горизонтальних сегментів, які формують цю драбинку, можна витягти апроксимацію вихідного ребра з точністю, яка перевершує точність растра. Таким чином можна забезпечити згладжування ребра драбинки.

Такі витягнуті ребра можна використовувати для повторного затінення пікселів, які вони перетинають, таким чином згладжуючи витягнуті ребра. Згладжені витягнуті ребра виявляються більш привабливою аппроксимацією реальних ребер, ніж аналоги з аліасінгом.

Проте комбіновані методи на основі семплювання і аналітичного антиалиасинга вже широко використовуються. Ці алгоритми створюють кілька семплів сцени - за допомогою традиційного мульти- або суперсемплірованія або за допомогою тимчасового накопичення між кадрами - і поєднують їх з аналізом, генеруючи кінцеве зображення з антиалиасинг. Це дозволяє їм знизити проблеми подпіксельного аліасинга і тимчасової нестабільності односемплових чисто аналітичних методів, проте все одно дає набагато кращі результати на геометричних ребрах, ніж чисті методи семплювання зі схожими характеристиками продуктивності. Дуже просту комбінацію додаткового семпліювання і аналітичного АА можна отримати поєднанням односемпловой аналітичної техніки на зразок FXAA з Субдіскретізація з буфера з більш високою роздільною здатністю. Більш складними прикладами таких методів є SMAA T2x, SMAA S2x і SMAA 4x, а також TXAA. Методи SMAA докладно пояснюються і порівнюються в цій статті, в той час як NVIDIA реалізувала власний підхід до TXAA тут. Є висока ймовірність, що такі комбіновані методи будуть більш широко використовуватися в майбутньому.

Ще одним варіантом, який поки ще не одержав широкого поширення, але має великий потенціал на майбутнє, є кодування в процесі рендеринга додаткової геометричній інформації, яка пізніше буде використовуватися на етапі аналітичного антиалиасинга. Приклади такого підходу - це антиалиасінг з геометричною постобробкою (Geometric Post-process Anti-Aliasing, GPAA) і антиалиасінг з використанням буфера геометрії (Geometry Buffer Anti-Aliasing, GBAA).

Нарешті, загальний пул пам'яті ЦП і відеопроцесора нових консольних платформ і майбутніх архітектур PC можуть дозволити використовувати техніки, призначені для експлуатації таких загальних ресурсів. В недавній статті «Asynchronous Adaptive Anti-aliasing Using Shared Memory» Берринджер і Меллер описують техніку, що виконує традиційний односемпловий рендеринг, в той же час розпізнаючи важливі пікселі (наприклад, що знаходяться на ребрі) і растерізіруя для них в ЦП додаткові розріджені семпли [5] . Хоча це вимагає серйозної реструктуризації процесу виконання рендеринга, результати виглядають багатообіцяюче.

Розглянемо далі таку важливу тему для комп’ютерної графіки, як дизеринг, або псевдотонування. Псевдотонування як і раніше залишається унікальним методом не тільки з практичної точки зору (наприклад, підготовка повнокольорового зображення для друку на чорно-білому принтері), а й за художніх міркувань. Дизеринг також знаходить застосування в веб-дизайні, де цей метод використовується для скорочення числа кольорів зображення, що зменшує розмір файлу (і також трафік) без шкоди для якості. Він також використовується при

зменшенні цифрових фотографій у форматі RAW в 48 або 64 біта на піксель до RGB в 24 біта на піксель для редагування.

Зосередимо увагу на основних моментах **дизеринга** зображень. На рис. 15 знаходиться повнокольорове зображення.

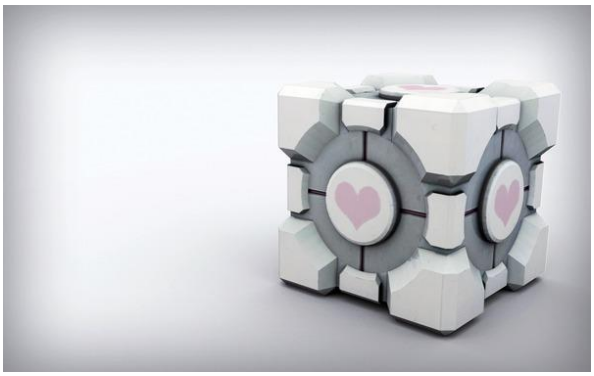


Рис. 15 Зображення кубу

На сучасному ЖК-або світлодіодному дисплеї - будь то монітор комп'ютера, смартфон або телевізор - це повнокольорове зображення можна виводити без проблем. Але уявімо ПК старіший з підтримкою обмеженої палітри кольорів. Якщо ми спробуємо відобразити наше зображення на такому ПК, це може виглядати приблизно так (рис.16).

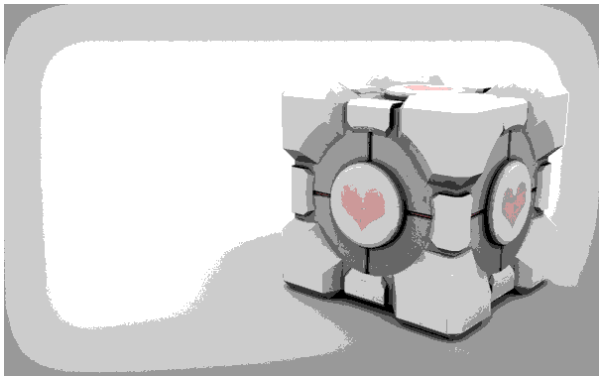


Рис. 16 Зображення з обмеженою веб-палітрою кольорів

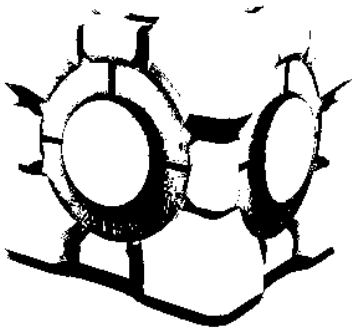


Рис. 17 Зображення кубу, надруковане на чорно-білому принтері

Проблеми виникають кожного разу, коли зображення відображається на пристрої, який підтримує менше кольорів, ніж містить зображення. Тонкі градієнти в оригінальному документі можуть бути замінені плямами однорідного кольору, і в залежності від обмежень пристрою, вихідне зображення може стати невпізнанним.

Псевдотонування (або дизеринг) - це спроба вирішити цю проблему. Псевдотонування працює через наближене вираження недоступних кольорів доступними, для чого доступні кольори змішуються так, щоб імітувати недоступні. Як приклад тут наведено зображення куба, знову погіршене до кольорів уявного старого ПК, тільки на цей раз застосовано псевдотонування (рис.18).

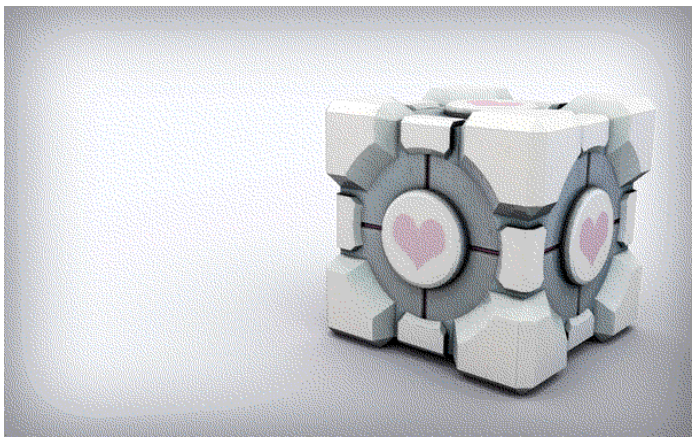


Рис. 18 Зображення кубу після застосування псевдо тонування

Якщо уважно подивится, ви побачите, що це зображення використовує ті ж кольори, що і його копія без псевдотонування. Але ці кілька кольорів розташовані так, що здається, ніби присутньо багато інших кольорів. Як інший приклад: тут чорно-біла версія нашого зображення зі схожим псевдо тонуванням (рис. 19).

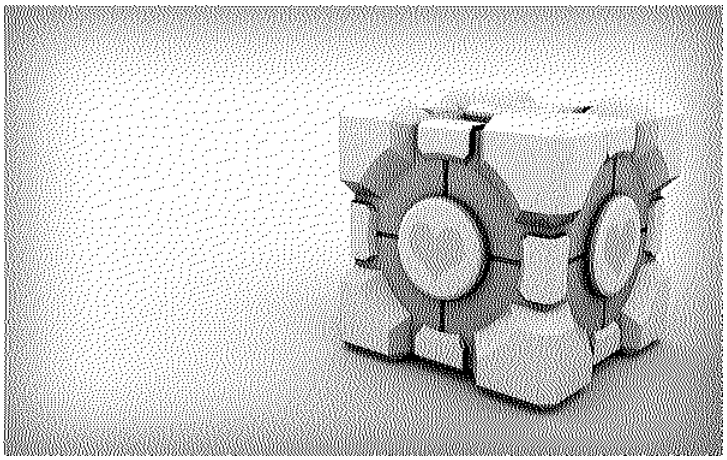


Рис. 19 Зображення кубу після застосування псевдо тонування

Незважаючи на наявність тільки чорного і білого кольорів, ми все ще можемо розрізнити форму куба аж до сердець по обидва боки. Дизеринг - надзвичайно потужний метод, і його можна використовувати в будь-якій ситуації, коли дані повинні бути

представлені в більш низькій роздільній здатності, ніж те, для якого їх створювали. Основну увагу буде приділено зображенням, але ті ж методи можуть бути застосовані до будь-яких двомірних даних (або до одновимірних даними, що ще простіше). Основна ідея псевдотонування пов'язана з розсіюванням помилок.

Розсіювання помилок працює наступним чином: припустимо, що нам потрібно звести фотографію в градаціях сірого до чорних і білих кольорів, щоб ми могли надрукувати її на принтері, який підтримує тільки чистий чорний (чорнило) або чистий білий (папір без чорнила) колір. Перший піксель на зображенні - темно-сірий, зі значенням 96 за шкалою від 0 до 255, де 0 - чистий чорний колір, 255 - чистий білий (рис. 20).

Рис. 20 Візуалізація значень для прикладу

При перетворенні такого пікселя в чорний або білий ми використовуємо просту формулу: значення кольору ближче до 0 (чорному) або 255 (білому)? Так як 96 ближче до 0, ніж до 255, тому ми робимо піксель чорним.

На цьому етапі стандартний підхід - просто переміщатися до наступного пікселя і виконувати те ж порівняння. Але проблема виникає, якщо у нас є купа подібних сірих пікселів зі значенням 96 - всі вони перетворюються в чорні. У нас виходить з величезний фрагмент порожніх чорних пікселів, які погано уявляють оригінальний сірий колір.

Розсіювання помилок слід більш розумному підходу до проблеми. Як ви могли припустити, розсіювання помилок працює шляхом «розсіювання» - або поширення - помилки кожного обчислення в сусідні пікселі. Якщо алгоритм знаходить сірий піксель зі значенням 96, він також визначає, що 96 ближче до 0, ніж до 255 - і тому робить піксель чорним. Але тоді алгоритм враховує «помилку» у його перетворенні. Зокрема, помилку в тому, що сірий піксель, який ми змусили бути чорним, насправді був на відстані в 96 кроків від чорного.

Коли він переміщається до наступного пікселя, алгоритм розсіювання помилок додає помилку попереднього пікселя до поточного пікселя. Якщо наступний піксель також має сірий колір 96, замість того, щоб зробити його чорним, алгоритм додає помилку 96 з попереднього пікселя. Це призводить до значення 192, яке насправді ближче до 255 - і, отже, ближче до білого. Таким чином, алгоритм робить цей піксель білим і знову враховує помилку. У цьому випадку помилка становить -63, тому що 192 на 63 менше, ніж 255 - то значення, на яке цей піксель поміняли.

У міру того, як алгоритм продовжує роботу, розсіювання помилок призводить до чергування чорних і білих пікселів, що досить добре імітує сірий колір значення 96 для цього сегмента. Це куди краще, ніж фарбувати все пікселі поспіль чорним. Як правило, коли ми закінчуємо обробку рядка зображення, ми відкидаємо значення помилки, яке ми відстежували, і починаємо заново з помилкою «0» з наступного рядка зображення.

Нижче наведено приклад зображення нашого куба із застосуванням описаного алгоритму. Зокрема, кожен піксель перетворюється в чорний або білий, помилка перетворення відзначається і передається наступному пікселю праворуч (рис. 21).

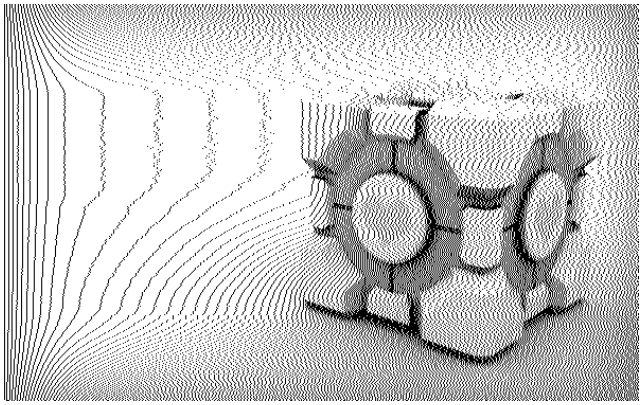


Рис. 21 Застосування псевдотонування з розсіюванням помилок

На жаль, псевдотонування з розсіюванням помилок має свої власні проблеми. Як би там не було, псевдотонування завжди призводить до помітних точок або до пунктирною вигляду. Це неминучий побічний ефект роботи з невеликою кількістю доступних кольорів - через їх кількості ці кольори будуть повторюватися знову і знову.

У наведеному вище простому прикладі алгоритму розсіювання помилок очевидна інша проблема - якщо у вас є блок дуже схожих кольорів, і ви штовхаєте помилку тільки вправо, все «точки» виявляються в одному і тому ж місці. Це призводить до ліній точок, які майже так само відволікають, як і оригінальна версія без псевдотонування.

Проблема в тому, що ми використовуємо тільки одномірне розсіювання помилок. Поширивши помилку тільки в одному напрямку (вправо), ми не дуже добре її розподіляємо. Оскільки у зображення вимірювання два (горизонтальне і вертикальне), чому б не звернути помилку в декількох напрямках. Це розподілить помилку більш рівномірно, що, в свою чергу, дозволить уникнути дивних ліній точок, розглянутих у прикладі розсіювання помилок вище.

Існує багато способів розсіювання помилки в двох вимірах. Наприклад, ми можемо поширити помилку на один або кілька пікселів вправо, вліво, вгору і вниз.

Для простоти розрахунків всі стандартні формули дізерінга просувають помилку тільки вперед. Якщо обійти зображення попіксельно, починаючи з верхнього лівого кута і рухаючись вправо, враховувати помилки назад (наприклад, вліво і / або вгору) необхідності не буде. Причина цього очевидна - якщо закидати помилку назад, доведеться повернутися до пікселів, які вже оброблені, що призводить до більшої кількості помилок при русі назад. У підсумку вийде нескінченний цикл поширення помилок.

Таким чином, для стандартного поведінки циклу (починаючи з верхнього лівого кута зображення і рухаючись вправо) ми хочемо, щоб рух пікселів йшов тільки вправо і вниз.

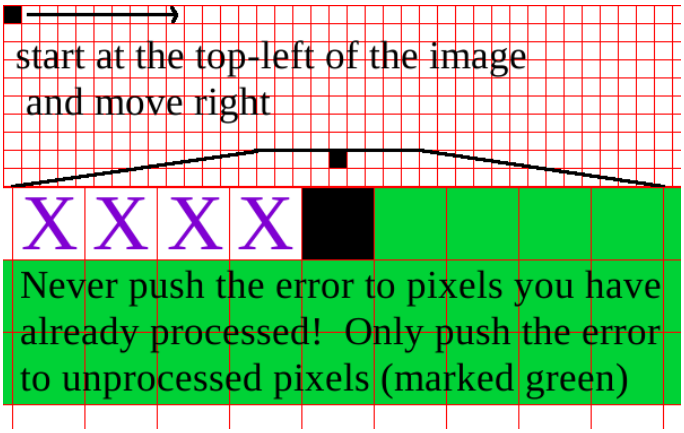


Рис. 22 Поширення помилок під час дизерингу

Щодо алгоритмів розсіювання помилок можна сказати наступне. Перша і можливо найвідоміша формула розсіювання помилок була опублікована Робертом Флойдом і Луїсом Стейнбергом в 1976 році. Розсіювання помилок відбувається за такою схемою:

```
      X   7
  3   5   1

    (1/16)
```

У наведених вище позначеннях X означає поточний піксель. Дріб в нижній частині є дільник для помилки. Інакше кажучи, формула Флойда-Стейнберга може бути записана у вигляді:

```
          X    7/16
3/16  5/16   1/16
```

Але таке позначення довге і неакуратне, тому будемо дотримуватися оригіналу. Повернемося до нашого оригінального наприклад перетворення пиксельного значення від 96 до 0 (чорний) або до 255 (білий). При фарбуванні пікселя в чорний ми отримуємо помилку 96. Ми поширюємо цю помилку оточуючим пікселям, поділивши 96 на 16 (= 6), потім множимо її на відповідні значення, наприклад:

```
          X    +42
+18    +30    +6
```

Шляхом поширення помилки декільком пікселям, кожному з різним значенням, ми зводимо до мінімуму всі відволікаючі смуги з точками, помітні в вихідному прикладі алгоритму розсіювання помилок. На рис. 23 зображення куба із застосуванням алгоритму Флойда-Стейнберга.

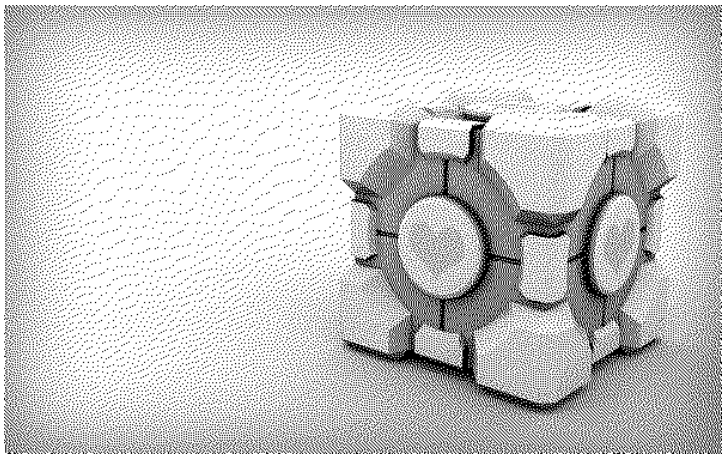


Рис. 23 Алгоритм розсіювання помилок Флойда-Стейнберга.

Алгоритм розсіювання помилок Флойда-Стейнберга - ймовірно, найбільш відомий алгоритм розсіювання помилок. Він дає достатньо гарну якість, а також вимагає тільки один передній масив (одновимірний масив шириною в зображення, де зберігаються значення помилок, поширювані до наступного рядка). Крім того, оскільки його дільник 16, замість поділу можна використовувати бітові зрушення. Так алгоритм досягає високої швидкості роботи навіть на старому обладнанні.

Що стосується значень 1/3/5/7, використовуваних для поширення помилки - вони були обрані спеціально, тому що вони створюють рівномірний картатий візерунок для сірого зображення.

Якщо ви порівняєте зображення вище з результатами псевдотонування іншої програми, ви можете виявити невеликі відмінності. Подібне очікувано. Є багато змінних, які можуть вплинути на точність виведення алгоритму псевдотонування. Серед них займають важливу роль настпні:

1. Відстеження помилок числами з плаваючою комою або цілими числами. Цілочисельні методи втрачають деяку дозвіл через помилки квантування.
2. Зменшення знебарвлення. Деяке програмне забезпечення зменшує похибка по заданому значенню (можливо, 50% або 75%), щоб зменшити обсяг знебарвлення сусідніх пікселів.
3. Поріг відсікання для чорного або білого. Значення 127 або 128 є загальноприйнятими, але на деяких зображеннях може бути корисно використовувати інші значення.
4. Для кольорових зображень грає важливу роль то, як розраховується яскравість. Я використовую формулу яскравості HSL ([max (R, G, B) + min (R, G, B)] / 2). Інші використовують ([r + g + b] / 3) або одну з формул ITU. YUV або CIELAB дозволять досягти результатів ще краще.
5. Гамма-корекція або інші попередні модифікації. Часто корисно нормалізувати зображення перед його перетворенням в чорно-біле. Вибір методу для цього очевидним чином вплине на результат.
6. Напрямок циклу. Я обговорював стандартний підхід «зліва направо, зверху вниз», але деякі розумні алгоритми дізерінга будуть слідувати змієподібною шляху, де спрямованість зліва направо змінюється на протилежну. Так можна зменшити плями однорідних точок і дати більш різноманітний зовнішній вигляд, але їх складніше реалізувати.

Для демонстраційних зображень ми не виконували попередню обробку вихідного зображення. Всі колірні відповідності виконуються в просторі RGB з відсіченням 127 (значення <= 127 встановлені на 0). Напрямок циклу - стандартні зліва направо зверху вниз.

Які конкретні методи псевдотонування Ви можете використовувати, залежить від вашої мови програмування, обмежень обробки і бажаного результату.

4. Базові растрові алгоритми.

Графічні пристрої, що використовуються в КГ, є в основному растровими, тому при синтезі графічних зображень, що складаються з окремих простих елементів, виникає необхідність у растрових алгоритмах.

Растровий алгоритм – це алгоритм, який для роботи програми, що реалізує графічні примітиви, враховує властивість растра. Хоча більшість графічних бібліотек містить у собі достатню кількість простіших растрових алгоритмів, часто виникає необхідність явної побудови растрових алгоритмів. При побудові графічних зображень вдаються до певного набору графічних примітивів, однак найчастіше в графічних системах використовуються лінії. Зобразити криву на дискретній поверхні (екрані) означає знайти таку її цілочислову апроксимацію, яка дасть можливість відтворити неперервну криву. Наприклад, для відрізка прямої потрібно отримати послідовність дискретних точок (пікселів), що апроксимує неперервну лінію (рис. 24).

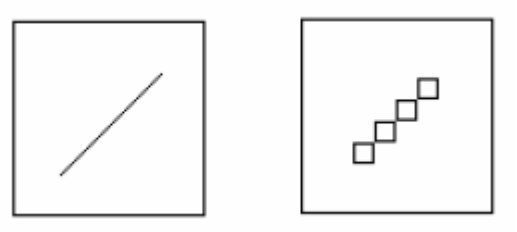


Рис. 24 Наближення пікселями неперервної лінії

Методи побудови ліній на дискретній поверхні поділяються на 2 класи:

• числові методи (методи прямих обчислень координат);
• інкрементні методи.

Числові методи базуються на числовому аналізі рівнянь кривих, тобто на обчисленні значень функцій. Розглянемо пряму лінію загального вигляду. Для її зображення теж необхідно обчислювати координати кожного пікселя. Відомо декілька методів розрахунку координат кожного пікселя лінії. Нехай задано координати кінців відрізка $A(x1, y1)$, $B(x2, y2)$, тоді рівняння прямої, що проходить через дві точки A і B, має вигляд $y = ax+b$, де

$$a = \frac{y_2 - y_1}{x_2 - x_1}, \qquad b = \frac{x_2 y_1 - x_1 y_2}{x_2 - x_1}.$$

Для простоти вважатимемо, що $0 \le y2 - y1 < x2 - x1$, тобто цикл у програмі буде здійснюватись по x.

Недоліком такої програми є те, що в циклі виконується множення та використовуються операції з плаваючою крапкою. А це зменшує швидкість виведення графічного зображення. Обчислення значень функції $y = ax + b$ можна уникнути, якщо використати рекурентне співвідношення $y = y + a$, оскільки при зміні x на одиницю значення функції y змінюється на a. Оскільки виділення цілої частини значення y не завжди може привести до коректного графічного результату, то покращити зовнішній вигляд відрізка можна за рахунок округлення значень функції до ближчого цілого, тобто це означає, що з двох пікселів, які лежать на одній

вертикалі (так, що відрізок прямої проходить між ними), завжди вибирається той піксель, який лежить ближче до прямої. Для цього досить порівняти дробову частину значення *y* з 0,5.

У цьому алгоритмі для обчислення координати y в тілі циклу є тільки одна операція '+', тому порівнюючи наведені алгоритми, останній кращий за швидкодією. Але незважаючи на те, що вхідні дані є цілочисловими величинами і всі результати теж мають бути цілочисловими, ця модифікація алгоритму все одно використовує дійсні числа. Окрім цього, оскільки для обчислення y в останньому варіанті використовується операція y:= y + a, то з кожним кроком циклу будуть накопичуватись похибки за рахунок неточного представлення дробових чисел у комп'ютері та за рахунок похибки арифметичних операцій із плаваючою крапкою. Тому в результаті виконання циклу може статися так, що на останньому кроці y ≠ y2 (хоча з математичної точки зору тут все коректно, при x = x2 маємо y = y2). Це теж необхідно враховувати при використанні даного алгоритму. Отже, числові методи мають перевагу завдяки простоті та ясності побудови алгоритму і можливості працювати з дробовими значеннями координат точок відрізка. Однак у числових методів є й свої недоліки:

1) використання операцій із плаваючою крапкою та операцій множення або ділення в циклі зумовлює малу швидкість обчислень, хоча це суттєво залежить від процесора (у сучасних комп'ютерах, де процесори використовують ефективні засоби прискорення, наприклад, конвеєр арифметичних операцій із плаваючою крапкою, час виконання цілочислових операцій не набагато менший);

2) при обчисленні координат шляхом додавання приросту може накопичуватись похибка обчислення координат.

Інкрементні алгоритми

У 1965 році Брезенхем (Bresenham J.E.) запропонував підхід для створення інкрементних методів (тоді ці методи використовувались для графопобудівників). Основною ідеєю інкрементних методів є теж рекурентні співвідношення для обчислення y, але в них уже не застосовуються дійсні обчислення й операції множення та ділення, а застосовуються лише цілочислові операції додавання і віднімання. Інкрементні алгоритми виконуються як послідовність обчислень координат сусідніх точок шляхом додавання відповідних приростів так, щоб сусідня цілочислова точка була найближчою до неперервної кривої. Прирости розраховуються на основі аналізу функції похибок. У циклі виконуються лише цілочислові операції порівняння, додавання та віднімання. У такий спосіб досягається більш висока швидкодія обчислень координат кожного пікселя порівняно з числовими методами. На сьогодні існує багато спеціалізованих інкрементних алгоритмів для генерування кривих того чи іншого типу (наприклад відрізків, кіл, еліпсів тощо). В основному вони реалізовані апаратно.

Алгоритм Брезенхема для відрізка

Серед графічних примітивів найбільшу питому вагу мають відрізки прямих. У зв'язку з цим при розробці графічних засобів алгоритмам лінійної інтерполяції приділяють особливу увагу. Розглянемо спочатку простіший випадок формування відрізка прямої, коли пряма проходить через початок координат (0,0) і точку з координатами (*n*, *m*). Рівняння цієї прямої має вигляд $mx - ny = 0$.

При цьому виникає задача за відомою точкою (x, y), що найкраще наближає пряму, визначити наступну точку (x+1, y) чи (x, y+1). Введемо в розгляд функцію $F(x, y) = mx - ny$. Нехай значення $F(x, y) = f \neq 0$. Знайдемо

$$F(x + 1, y) = m(x + 1) - ny = mx - ny + m = f + m,$$
$$F(x, y + 1) = mx - n(y + 1) = mx - ny - n = f - n.$$

Щоб зробити вибір між двома точками, потрібно знати, яке з двох чисел $f + m$ чи $f - n$ знаходиться ближче до нуля. Якщо значення $f + m$ ближче до нуля, то рух із точки (x, y) відбудеться в точку (x +1, y), а якщо $f - n$ ближче до нуля, то в точку (x, y +1).

Для знаходження приросту координат (x, y) побудуємо функцію похибок $S(x, y)=F(x + 1, y) + F(x, y + 1) = 2f + m - n$ і виробимо ознаку вибо-
ру наступної точки. Для цього потрібно розглянути такі випадки:

1) якщо $f + m < 0$, $f - n < 0$, тобто $f + m$ є ближчим до нуля (рис. 25, *а*), а $S = 2f + m - n < 0$, то наступною є точка (x + 1, y);

2) якщо $f + m > 0$, $f - n > 0$, тобто $f - n$ є ближчим до нуля (рис. 25, *б*), а $S = 2f + m - n > 0$, то рух відбувається в точку (x, y + 1);

3) якщо $f + m > 0$, $f - n < 0$, тобто $f + m$ є ближчим до нуля (рис. 25, *в*), а $S = 2f + m - n < 0$, то надалі вибираємо точку (x + 1, y);

4) якщо $f + m > 0$, $f - n < 0$, тобто $f - n$ є ближчим до нуля (рис. 5.2, *г*), а $S = 2f + m - n > 0$, то рух відбувається в точку (x, y + 1);

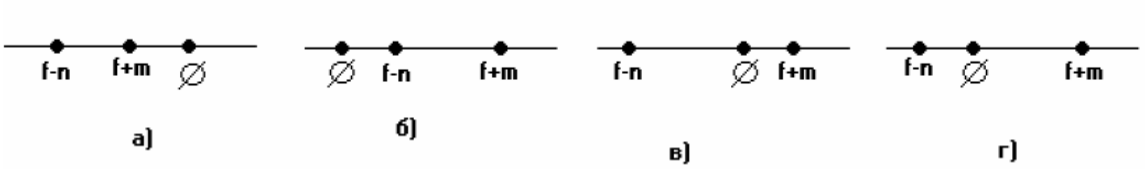


Рис. 25 Розміщення точок

Отже, вибір наступної точки визначається знаком функції похибок S: якщо $S > 0$, то наступною є точка (x, y +1), а якщо $S \leq 0$, то (x+1, y). При переході до наступної точки змінюється значення S, тому для простого обчислення S потрібно мати рекурентну формулу. Оскільки $S(x, y) = 2f(x, y) + m - n$, то початкове значення $S0 = S(0, 0) = m - n$. Розглянемо

$S(x+1, y) = 2f(x+1, y)+m - n = 2(mx - ny + m) + m - n = 2f + m - n + 2m = S(x, y)+2m$;

$S(x, y+1) = 2f(x, y+1)+m - n = 2(mx - ny - n) + m - n = 2f + m - n - 2n = S(x, y) - 2n$;

Таким чином, маємо такі рекурентні формули для S:

$S = S + 2m$, якщо $S \leq 0$;

$S = S - 2n$, якщо $S > 0$.

Для побудови кола $x2 + y2 = R2$ можна реалізувати алгоритм шляхом прямого обчислення координат. Або, маючи програму побудови відрізка, коло можна апроксимувати хордами, але коло буде недостатньо гладким, поки кількість хорд не стане більшою 36. При високих роздільних здатностях потрібна ще більша кількість хорд. Ці методи вимагають багато обчислень, що зменшує швидкість побудови кола. Тому розглянемо інкрементний метод Брезенхема для кола.

Для спрощення алгоритму растрової розгортки кола можна використати його симетрію відносно координатних осей і прямих $y = \pm x$. Тому достатньо побудувати растрове зображення

для 1/8 частини кола, а решту частин побудувати за рахунок симетрії кола (якщо точка (*x*,*y*) належить колу, то і точки (–*x*, *y*), (*x*, –*y*), (–*x*, –*y*), (*y*, *x*), (–*y*, *x*), (*y*, –*x*), (–*y*, –*x*) належать колу).

Цей алгоритм більш складний, ніж алгоритм побудови кола. Внаслідок симетрії еліпса відносно осей координат координати точок еліпса обчислюватимуться лише в першій чверті.

Канонічне рівняння еліпса має вигляд $\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$, де *a* та *b* – велика і мала півосі еліпса. Зведемо рівняння еліпса до такого вигляду:

$$\frac{x^2}{a^2}+\frac{y^2}{(b+0.5)^2}=1$$

Звідси маємо

$$\frac{x^2}{a^2}+\frac{(2y)^2}{(2b+1)^2}=1, \quad \text{або} \quad (2b+1)^2x^2+4a^2y^2=(2b+1)^2a^2.$$

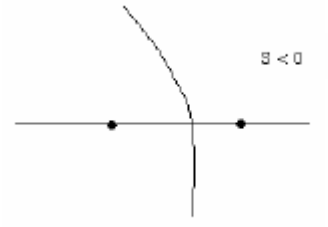
або (2*b* + 1)2 *x*2 + 4*a*2 *y*2 = (2*b* + 1)2 *a*2.

Позначимо (2*b* + 1)2 = *m*, *a*2 = *n*, *F*(*x*,*y*) = *m**x*2 + 4*n**y*2 – *mn*, тоді рівняння еліпса матиме вигляд *F*(*x*, *y*) = 0.

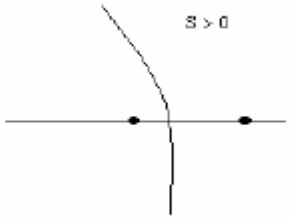
Ідея алгоритму полягає ось у чому: для кожного *y*, що змінюється від *b* до 0, необхідно підібрати ціле значення *x* так, щоб *F*(*x*, *y*) було найближчим до нуля (точка (*x*, *y*) найкраще апроксимує еліпс). Якщо *F*(*x*, *y*) < 0, то збільшуємо *x* (*x*:= *x* +1), якщо *F*(*x*, *y*) > 0, то зменшуємо *y* (*y*:= *y* – 1). Але знак функції *F*(*x*, *y*) не може бути ознакою вибору точки (*x*, *y*). Для вибору точки (*x*, *y*), що найкраще апроксимує еліпс, утворюємо функцію похибок *S*(*x*, *y*).

S(*x*, *y*) = *F*(*x*, *y*) + *F*(*x*+1, *y*) = 2*m*(*x*2 + *x*) + 8*n**y*2 – 2*mn* + *m*.

Тоді, якщо *S*(*x*, *y*) < 0, то збільшуємо *x*:= *x* + 1, (рис. 5.7, *а*); якщо *S*(*x*, *y*) стає більшим нуля, то ми підібрали потрібну точку *x* (цей піксель зображаємо на екрані) і здійснюємо перехід до наступного *y* (*y*:= *y* – 1). Отже, точка *x* – це перша точка, для якої *S*(*x*, *y*) > 0.



а



б

Для початкової точки (0, *b*) *S*0 = 8*nb*2 – 2*mn* + *m*. Для наступних точок (*x*, *y*) потрібно мати значення S(*x*, *y*), але його не можна знайти безпосередньо, тому що в цей вираз входять операції множення (втрачається швидкодія алгоритму). Для обчислення значень S(*x*, *y*) побудуємо рекурентні співвідношення без операцій множення та ділення.

Нехай *S*(*x*+1, *y*) = *S*(*x*, *y*) + *P*(*x*, *y*), тоді *P*(*x*, *y*) = (*S*(*x*+1, *y*) – *S*(*x*, *y*)) = 2*m*((*x*+1)2 +(*x*+1)) + 8*n**y*2 – 2*mn* + *m* –2*m*(*x*2 + *x*) – 8*n**y*2 +2*mn* – *m* = 4*mx* + 4*m*.

З останнього співвідношення видно, що при збільшенні *x* на одиницю, *P* збільшується на 4*m*, тому для *P* маємо рекурентне співвідношення

P(*x* +1, *y*) = *P*(*x*, *y*) + *dp*, де *dp* = 4*m*, *P*(0, *y*) = 4*m*.

Аналогічно при зміні y на $y - 1$ маємо
$Q(x, y) = S(x, y-1) - S(x, y) = 2m\,(x2 + x) + 8n(y - 1)2 - 2mn + m - 2m \times$
$(x2 + x) - 8ny2 + 2mn - m = -16ny + 8n$, тобто $Q(x, y-1) = Q(x, y) + dq$,
$dq = 16n$, $Q(0, b) = -16nb + 8n$.

Растрові алгоритми зафарбовування і заповнення областей Область можна задавати двома способами: шляхом виведення ліній контуру (границі області) (рис. 6.1, *а*) або зафарбувавши всі пікселі області (рис. 26, *б*).

а *б*

Рис. 26 Зафарбовування області

Якщо область задається пікселями, що лежать усередині області, то всі пікселі з області мають одне й теж значення old і жоден піксель із границі такої області не має значення old. Такі області називаються внутрішньо заданими. Алгоритми, які працюють з такими областями так, щоб пікселі з атрибутом *old* перевести в пікселі зі значенням *new,* називаються *внутрішньо заповнюючими алгоритмами*.

Області, що задаються своїми замкнутими контурами (границями), називаються *гранично визначеними*. Пікселі, що належать границі, мають значення *bound*, а коди пікселів внутрішньої частини області відмінні від *bound.* Алгоритми заповнення таких областей називаються *гранично заповнюючими алгоритмами*. При цьому пікселі області потрібно зафарбувати в колір *new*, а колір контуру не повинен змінитися. Користувач може інтерактивно визначати область, задавши границю із пікселів, що прилягають один до одного. Ці області потрібно зафарбувати кольором *new*. Після вибору області для рекурсивного алгоритму зафарбовування користувач повинен указати ще довільну затравочну точку, що належить області. Далі в цьому алгоритмі зафарбовування області потрібно прочитати сусідні пікселі, визначити, чи вони належать області і не зафарбовані, якщо так, то зафарбувати їх. При цьому виникає питання, які пікселі ($x1$, $y1$), ($x2$, $y2$) можна вважати сусідніми. Важливим поняттям растрових областей є їх зв'язність – можливість з'єднати два пікселі області растровою лінією, тобто послідовним набором пікселів, що належать цій області. Використовуючи зв'язність можна, рухаючись від точки затравки, досягти і зафарбувати всі пікселі області. За способом доступу до сусідніх пікселів області поділяються на 4-зв'язні та 8-зв'язні.

4-зв'язними називають ті області, в яких кожен піксель області може бути досягнутий з іншого пікселя області за допомогою комбінацій переміщень на один піксель у чотирьох напрямках: горизонтальних (вліво, вправо), або вертикальних (вверх, вниз), тобто якщо координати сусідніх пікселів задовольняють умову

$$|x1 - x2| + |y1 - y2| \le 1|$$

Такі пікселі називають ще околом фон Неймана. У 8-зв’язних областях до цих напрямків додаються ще і діагональні, тобто *8-зв’язними* називаються області, в яких кожний піксель може бути досягнутий з іншого пікселя цієї ж області за допомогою послідовності переміщень на один піксель у восьми напрямках (рис. 6.2). У цьому випадку сусідніми вважаються пікселі, якщо їх координати відрізня ються відповідно не більше ніж на одиницю, тобто

$$|x1-x2| \le 1 \text{ і } |y1-y2| \le 1.$$

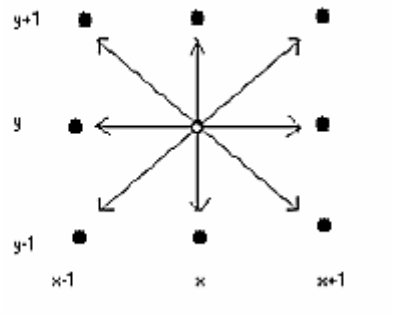


Рис. 27

Із наведених означень випливає, що довільних два 4-зв’язних пікселі є 8-зв’язними, але не навпаки. Тому, в загальному випадку 4-зв’язну область можна заповнити як 4-зв’язними так і 8-зв’язними алгоритмами, але не навпаки, Тобто 8-зв’язну область не завжди можна заповнити 4-зв’язним алгоритмом. Розглянемо два квадрати зображені на рис. 6.3. За алгоритмом заповнення для 4-зв’язних областей буде заповнено лише один квадрат, а алгоритм для 8-зв’язних областей заповнить два квадрати.

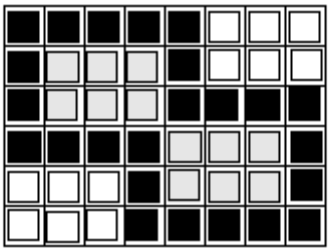


Рис. 28

Методи заповнення областей можна поділити на дві групи: методи заповнення з затравочною точкою та методи растрової розгортки. У методах заповнення з точкою затравки заповнення починається з деякої точки, що знаходиться всередині замкнутого контуру (затравочна точка). Далі відшуковуються сусідні з нею точки, які розміщені всере- дині контуру. Якщо така точка знайдена, то вона стає новою затравоч- ною точкою і для неї рекурсивно ведеться пошук нових точок. Подібні алгоритми використовуються тільки для растрових пристроїв. У методах растрової розгортки в порядку сканування рядків визначають чи точка лежить всередині контуру і якщо так, то її зафарбовують.

Обробка растрових зображень

Обробка растрових зображень передбачає перетворення растрових даних. Існує багато різних варіантів обробки растрових зображень. Можна виділити три основних класи алгоритмів обробки растрових даних:

− точкові алгоритми, в яких значення пікселів змінюються на основі значень самих пікселів та їх координат;

− просторові алгоритми, в яких значення пікселів змінюється на основі вихідних значень самих пікселів і пікселів навколо них;

− алгоритми геометричних перетворень, коли розміщення пікселів в зображенні змінюється на основі геометричних перетворень. Для кожного з цих класів розроблено цілий ряд алгоритмів, причому застосування цих алгоритмів некомутативне, тому важливим є порядок їх застосування.

Розглянемо деякі поняття, що лежать в основі цих алгоритмів. Нехай a_{ij} елемент зображення, тобто піксель з координатами (i, j), тоді A = { aij }, i = 0, 1,…, m, j = 0, 1, …, n растрове зображення, тобто матриця пікселів. Якщо aij ∈{0, 1}, то растрове зображення називається бінарним і складається тільки з чорних та білих пікселів.

Якщо aij ∈{0, 1, …, N – 1}, то растрове зображення називається на півтоновим і кожний піксель може приймати N відтінків сірого (градацій яскравості).

$$\text{Якщо } a_{ij} \in \left\{ \begin{pmatrix} x_{ij}^1 \\ x_{ij}^2 \\ x_{ij}^3 \end{pmatrix} \right\}, \text{ де } x_{ij}^k \in \{0,\ 1,\ \ldots,\ N-1\},$$

то зображення називається кольоровим. Колір кожного пікселя визначається координатами ($x1$, $x2$, $x3$) в просторі кольорів. Розглянемо алгоритмічні основи обробки бінарних і на півтонових зображень.

Точкові алгоритми. Точкові алгоритми прості і найбільш часто застосовуються в алгоритмах обробки зображень. Вони застосовуються для бінарних і напівтонових зображень. В цих алгоритмах для зміни яскравості пікселя в зображенні використовується тільки вихідна яскравість цього пікселя, інколи координати цього пікселя; ніякі інші значення не використовуються.

Нові значення яскравості обчислюються на основі деякого алгоритму. Точкові алгоритми сканують зображення піксель за пікселем, здійснюючи перетворення точок зображення. Якщо перетворення залежить тільки від яскравості пікселів, то процес обробки зображень краще реалі зувати на основі таблиць перетворення. Якщо перетворення в точкових алгоритмах враховує і розміщення пікселів, то воно задається формулою. Точкові алгоритми не можуть модифікувати деталі зображення. Найпростіші алгоритми цього класу це алгоритми побудови негативних зображень, або алгоритми інверсії яскравості пікселів. Негативне зображення одержується шляхом віднімання вихідного значення яскравості f(x, y) кожного пікселя зображення від максимально можливого значення яскравості fmax:

$g(x, y) = f_{max} - f(x, y)$, де $g(x, y)$ – нове значення яскравості пікселя (x, y). Негативні зображення необхідні для виділення яскравих частин зображення, оскільки людське око краще сприймає деталі в темній області зображення, ніж в світлій. Друга важлива група алгоритмів цього класу це алгоритми бінаризації – напівтонових зображень, тобто перетворення їх в чорно-біле (бінарне) зображення. Ці алгоритми базуються на пороговій обробці напівтонових зображень шляхом розділення всіх пікселів зображення на два класи за ознакою яскравості: пікселі об'єкта і фону. Формула перетворення яскравості в алгоритмах бінаризації має вигляд

$$g(x, y) = \begin{cases} f_{\max}, якщо \ f(x, y) \geq f_0; \\ f_b, якщо \ f(x, y) < f_0, \end{cases}$$

де $f0$ – деяке порогове значення яскравості (рис. 6.10).

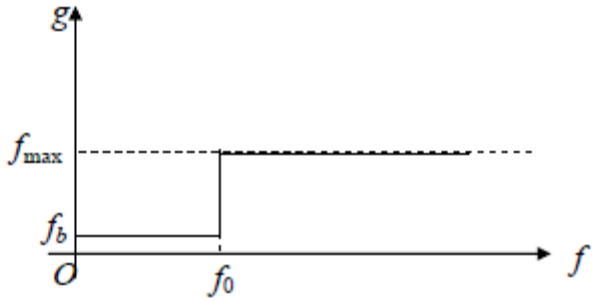


Рис. 29

При цьому важливо правильно вибрати порогове значення $f0$, щоб не втратити корисну інформацію. Особливо це суттєво, коли зображення містить деякий шум. Такі перетворення виконують для того щоб в зображенні залишити тільки ту інформацію, яка необхідна для розв'язування задач, наприклад для знаходження контурів об'єкта.

За допомогою точкових алгоритмів можна розв'язати задачу збіль-
шення контрастності зображення. Зображення з низьким контрастом можуть бути як надто світлими так і надто темними. Зображення з високим контрастом мають як темні так і світлі області, тобто використовують весь діапазон яскравості. Зображення з низьким контрастом складається з тонів обмеженого діапазону яскравості. Задача збільшення контрасту полягає в розтягуванні діапазону яскравості вихідного зображення на
всю шкалу $[fb, fmax]$. Цю задачу можна розв'язати за допомогою точкового перетворення яскравості за формулою $g(x, y) = af(x, y) + b$ де a, b підібрані коефіцієнти.

Просторові алгоритми. В просторових алгоритмах використовується інформація про групи пікселів, на відміну від точкових алгоритмів у яких використовується інформація тільки про один піксель. Група пікселів зображення, які приймають участь в просторових алгоритмах називається *областю примикання*. Область примикання визначається матрицею значень яскравості пікселів з непарною кількістю рядків і стовпців. Пікселі, в яких старе значення яскравості замінюється на нове, розміщені в центрі області притягання. Більшість просторових алгоритмів використовують фільтри. Фільтр зручно задавати матрицею чисел малого розміру (3 × 3, або 5 × 5), яку називають я*дром розгортки*. Розміри, структура ядра згортки і значення коефіцієнтів, що містяться в ядрі визначають тип просторового алгоритму, дозволяють вплинути на значення пікселя зображення і одержати різні спецефекти.

Більший розмір матриці підвищує гнучкість процесів згортки, однак виникають труднощі з примежовими пікселями, оскільки маска згортки для них виходить за межі зображення. Тому на практиці обмежуються невеликими розмірами матриці, та додатково опрацьовують пікселі на краях зображення. Щоб перетворити один піксель в зображенні, значення його яскравості множиться на число в центрі ядра, а яскравості пікселів, що оточують центральний піксель множаться на відповідні коефіцієнти ядра. Потім ці всі добутки сумуються і в результаті одержується нове значення яскравості для центрального пікселя. Цей процес повторюється для кожного пікселя зображення (так фільтрується зображення). Коефіцієнти ядра визначають результат процесу фільтрування. Якщо сума коефіцієнтів більша за одиницю, то яскравість збільшується; якщо сума менша за одиницю, то яскравість зменшується. Найбільш широко розповсюджені просторові алгоритми обробки растрових зображень – це алгоритми розмивання, збільшення чіткості, тиснення і акварельний ефект.

В *алгоритмі розмивання* в зображенні пом'ягшуються різкі границі за рахунок перерозподілу яскравості, тобто шляхом усереднення швидких змін яскравості (рис. 6.11, *б*). Ядро розмивання складають коефіцієнти менші за одиницю, а їх сума дорівнює одиниці. Це означає, що в результаті фільтрації кожний піксель вбирає інформацію від сусідніх пікселів, але повна яскравість зображення залишається незмінною. Результуюче зображення таким чином буде більш розмитим порівняно з оригіналом. Ядро розмивання, наприклад, має вигляд

$$\begin{pmatrix} 0{,}05 & 0{,}05 & 0{,}05 \\ 0{,}05 & 0{,}6 & 0{,}05 \\ 0{,}05 & 0{,}05 & 0{,}05 \end{pmatrix} \quad \text{або} \quad \begin{pmatrix} 0{,}1 & 0{,}1 & 0{,}1 \\ 0{,}1 & 0{,}2 & 0{,}1 \\ 0{,}1 & 0{,}1 & 0{,}1 \end{pmatrix}.$$

Ступінь розмивання можна змінити

− збільшенням розміру ядра;

− підбором коефіцієнтів ядра зі зменшенням впливу центрального коефіцієнта;

− проведенням багатокрокової фільтрації з ядром розмивання. Зауважимо, що у випадку кольорових зображень ядро розмивання застосовується до червоної, зеленої і синьої компонент кольору кожного пікселя зображення. В алгоритмах збільшення чіткості застосовується ____101 .нше ядро,

оскільки необхідно збільшити чіткість зображення, тобто підкреслити різницю між яскравостями сусідніх пікселів і виділити непомітні деталі (рис.6.11, *в*).

В ядрі чіткості центральний коефіцієнт більший за одиницю, а решту елементів ядра від'ємні числа, сума яких на одиницю менша, ніж центральний коефіцієнт. Це означає, що при обробці зображення з великими змінами яскравості нове значення яскравості центрального пікселя різко збільшується, тобто великі зміни яскравості збільшуються, а області постійної яскравості залишаються незмінними. Результуюче зображення одержується більш чітким, ніж оригінал. При повторній обробці чіткість знову може бути збільшена, але при цьому ніякі нові деталі з нічого не з'являться. При обробці пікселів в зображенні застосовуються такі ядра чіткості (високочастотні маски):

$$\begin{pmatrix} -0{,}1 & -0{,}1 & -0{,}1 \\ -0{,}1 & 1{,}8 & -0{,}1 \\ -0{,}1 & -0{,}1 & -0{,}1 \end{pmatrix}, \quad \begin{pmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{pmatrix}, \quad \begin{pmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{pmatrix}.$$

Знову, як і раніше у випадку кольорового зображення червона, зелена і синя складові обробляються маскою окремо, а потім об’єднуються, щоб сформувати 24-бітне значення кольору.

Алгоритми тиснення перетворюють зображення так, що об’єкти сцени виглядають видавленими на деякій поверхні (рис. 6.11, *г*). Тиснення виконується за допомогою ядра згортки так, як і в попередніх алгоритмах. Кожен піксель зображення обробляється ядром тиснення розміру 3 × 3.

Наведемо приклади ядра тиснення

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}, \quad \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}.$$

Зауважимо, що сума коефіцієнтів в ядрі тиснення дорівнює нулю. Це означає, що тим пікселям, які не знаходяться на границях переходу від від одного кольору до іншого присвоюються нульові значення, а тим що знаходяться на таких границях ненульові значення. Тобто, якщо всі дев'ять пікселів, що знаходяться в області матриці тиснення мають однакові яскравості, то значення центрального пікселя після перетворення стане рівним нулю (чорний колір). Після обробки пікселя ядром тиснення до одержаної яскравості (у випадку кольорового зображення до кожної складової R, G, B) добавляють число 128. Суми, що перевищують 255 заокруглюються до 255. Аналогічно конструюються маски для акварельних зображень.

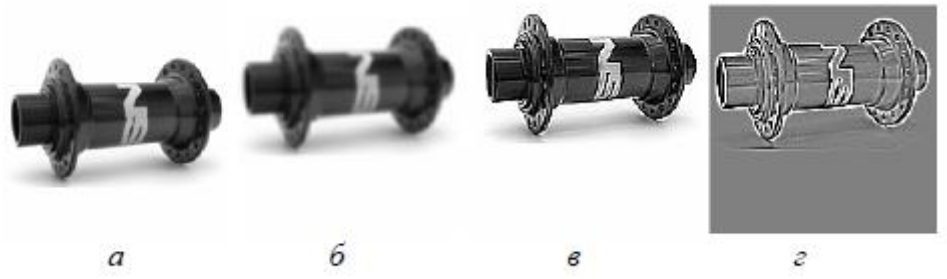


Рис. 30

До просторових алгоритмів відносяться ще *медіанні фільтри* за допомогою яких можна відфільтрувати шумові ефекти. Ці алгоритми не працюють за принципом згортки, тобто нове значення яскравості пікселя не обчислюється шляхом матричного множення, а замість цього всі пікселі в області примикання сортуються в зростаючому порядку за яскравістю пікселів і вибирається середнє значення яскравості для нового значення розглядуваного пікселя. Наприклад, потрібно змінити значення яскравості центрального пікселя в такому вікні

$$\begin{pmatrix} 43 & 75 & 50 \\ 57 & 126 & 50 \\ 24 & 83 & 50 \end{pmatrix}.$$

Відсортувавши значення пікселів, одержимо послідовність 24, 43, 50, 50, 50, 57, 75, 83, 126 в якій 50 є середнім значенням. В результаті середнє значення яскравості 50 замінить 126 у вихідному зображенні, тобто випадковий шум, що міститься в зображенні буде усунено.

Алгоритми геометричних перетворень.

Алгоритми геометричних перетворень растрових зображень змінюють місце розміщення і/або структуру пікселів зображення на основі геометричних перетворень. Геометричні перетворення не завжди змінюють яскравість елементів зображення, але вони

завжди змінюють розміщення пікселів зображення, тобто значення яскравості пікселів займають нову позицію. Геометричні перетворення растрових даних широко застосовуються, наприклад при розпізнаванні образів, накладанні текстури, забезпеченні різних ефектів у зображеннях. Геометричні перетворення растрових даних – це переміщення, масштабування, зсув і поворот зображення. Ці перетворення будуть вивчатимуться далі.

Криві Без’є

Крива Безьє розроблені математиком *П'єром Без'є*. Криві і поверхні Безьє були використані в 60-х роках компанією "Рено" для комп'ютерного проектування форми кузовів автомобілів. В даний час вони широко використовуються в комп'ютерній графіці.

Криві Безьє описуються в параметричній формі:

Значення *t* виступає як параметр, якому відповідають координати окремої точки лінії. Параметрична форма опису може бути зручніше для деяких кривих, ніж завдання у вигляді функції y = *f* (x), оскільки функція *f* (x) може бути набагато складніше, ніж *P x (t)* і *P y (t),* крім того, *f* (x) може бути неоднозначною.

Багаточлени Безьє для P x і P y мають такий вигляд:

$$P_x(t) = \sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} x_i, \quad P_y(t) = \sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} y_i,$$

де *x i* і *y i* - координати точок-орієнтирів *P i* , а величини - це відомі з *комбінаторики,* так звані *поєднання* (вони також відомі як коефіцієнти *бінома Ньютона):*

$$C_m^i = \frac{m!}{i!(m-i)!}.$$

Значення та можна розглядати і як ступінь полінома, і як значення, яке на одиницю менше кількості точок-орієнтирів.

Розглянемо криві Безьє, класифікуючи їх за значеннями *m.*

m = 1 (по двох точках)

$$P(t) = (1-t)P_0 + t_1.$$

m = 2 (по трьом точкам)

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

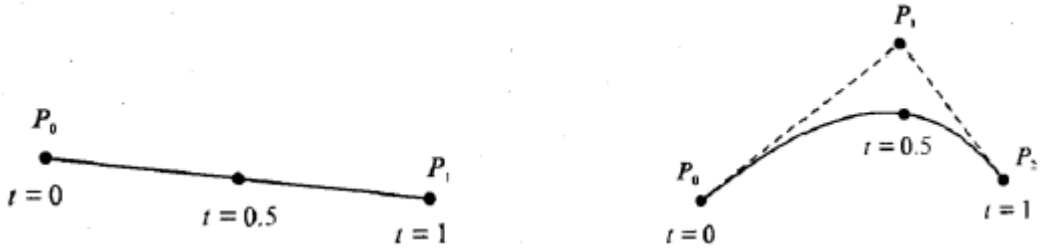


Рис. 31 Крива Безьє M=1; M=2

$$P(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3.$$

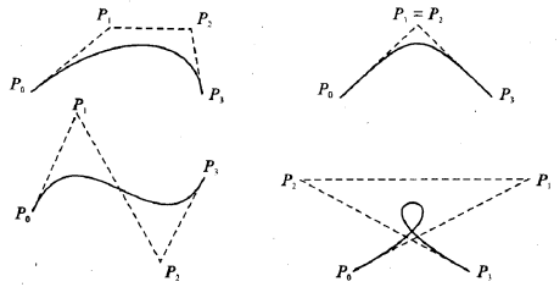


Рис. 32. Кубічні криві Безьє (m = 3)

Геометричний алгоритм для кривої Безьє

Цей алгоритм дозволяє обчислити координати (x, *y)* точки кривої Безьє за значенням параметра *t.*

1. Кожна сторона контуру багатокутника, який проходить по точках-орієнтирах, ділиться пропорційно значенню *t.*

2. Точки ділення з'єднуються відрізками прямих і утворюють новий багатокутник. Кількість вузлів нового контуру на одиницю менше, ніж кількість вузлів попереднього контуру.

3. Сторони нового контуру знову діляться пропорційно значенню t. І так далі. Це продовжується до тих пір, поки не буде отримана єдина точка поділу. Ця точка і буде точкою кривої Безьє (рис. 31).

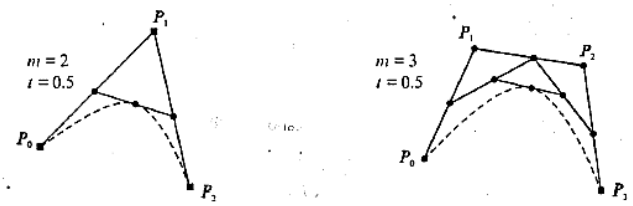


Рис. 33. Геометричний алгоритм для кривих Безьє

5. Особливості використання растрової графіки.

Особливості растрової графіки зумовлені її перевагами та недоліками. Так, однією з переваг растрової графіки є простота і, як наслідок, технічна реалізація (автоматизація) введення (оцифровки) образотворчої інформації. Існує розвинена система зовнішніх пристроїв введення зображень (до них відносяться сканери, відеокамери, цифрові фотокамери, графічні планшети).

Растрове зображення має переваги при роботі з фотореалістичними об'єктами, наприклад сценами природи або фотографіями людей. Справа в тому, що наш світ створений як растровий. І його об'єкти важко уявити в векторному, тобто математичному, поданні. Фотореалістичність має на увазі, що в растровій програмі можна отримувати мальовничі ефекти, наприклад туман або серпанок, домагатися найтоншого нюансування кольору, створювати перспективну глибину і нерізкість, розмитість і т. д. Формати файлів,

призначені для збереження точкових зображень, є стандартними, тому не має вирішального значення, в якому графічному редакторі створено те чи інше зображення.

Що стосується недоліків, можна сказати наступне. При першій же спробі що-небудь намалювати в програмі растрової графіки - наприклад, в Photoshop, - вона буде потребувати інформації про дозвіл (кількості точок на одиницю довжини) і про глибину кольору (кількості колірних бітів на піксель). Нічого цього знати в векторній програмі не потрібно. Обсяг файлу растрової графіки однозначно визначається добутком площі зображення на дозвіл і на глибину кольору (якщо вони приведені до єдиної розмірності).

Неможливо збільшити зображення для розгляду деталей. Оскільки зображення складається з точок, то збільшення зображення приводить тільки до того, що ці точки стають більшими. Ніяких додаткових деталей при збільшенні растрового зображення розглянути не вдається. Більш того, збільшення точок растру візуально спотворює ілюстрацію і робить її грубою (пикселізація). Текст в растровій графіці виявляється проблемою. У більшості растрових програм Ви, як правило, редагуєте текст під час його створення, але коли ви натискаєте мишею в будь-якому місці на екрані, друкований символ закріплюється там, де він був би завданий на полотно. Якщо ви хочете відредагувати вже набраний раніше текст, то не можете просто помістити курсор між двома буквами, видалити одну і почати знову набирати. З цією проблемою можна зіткнутися, наприклад, працюючи в растровому графічному редакторі MS Paint. Крім того, при великому дозволі файл растрового тексту буде надто великим.

Контролні запитання

1. Що таке растрова графіка?
2.Назвіть базові алгоритми обробки растрових зображень.
3.Яким чином відбувається вивидення растрових зображень.
4. Що таке шейдер?
5. Якими є особливості використання растрової графіки.

Список використаних інформаційних джерел

1. Петров М. Н.Компьютерная графика: Учебник для вузов. 3-е изд. (+CD). — СПб.: Питер, 2011. — 544 с.: ил.
2. Маценко В.Г. Комп'ютерна графіка: Навчальний посібник. – Чернівці: Рута, 2009 – 343 с.
3.https://www.youtube.com/watch?v=H2E3yO0J7TM&index=3&list=PLlrATfBNZ98foTJPJ_Ev03o2oq3-GGOS2 – серія відео про OpenGL.
4. https://www.youtube.com/watch?v=GvLEAHRmPl0 – відео про антіелайзинг.
5. https://www.gamingscan.com/what-is-anti-aliasing/ – стаття про антіелайзинг.
6. https://ux.pub/rukovodstvo-dizajnera-po-dpi-i-ppi/
7. http://www.detaillook.com/49-ppi-i-ego-vliyanie.html
8. https://habr.com/ru/post/343876/ – стаття про антіаліасінг.
9. https://habr.com/ru/post/326936/ – стаття про псевдо тонування.

ЛЕКЦІЯ 6.
ТЕМА: ФРАКТАЛЬНА ГРАФІКА

План

1. Основні поняття фрактальної графіки.
2. Історія фрактальної графіки.
3. Основні види фрактальних зображень
4. Застосування фрактальної графіки.

1. Основні поняття фрактальної графіки

Поняття фрактала, фрактальної геометрії з'явилося в кінці 70-х років минулого століття і стало широко застосовуватися математиками і комп'ютерними художниками. Фрактали знайшли широке застосування в комп'ютерній графіці завдяки компактності математичного апарату, необхідного для їх відтворення на екрані комп'ютера. За останній час з'явилося багато праць (особливо на Заході), присвячених фракталам (наприклад, [18, 20]). В цих працях вивчаються математичні основи фракталів, а також наводяться оригінальні графічні зображення фракталів, одержаних за допомогою комп'ютерів.

Слово *фрактал* походить від латинського fractus і в перекладі означає "складений із фрагментів''. Цей термін запропонований математиком Б. Мандельбротом у 1975 р. для позначення самоподібних структур.

Поняття фрактала як математичного об'єкта досі залишається невизначеним – зроблено лише окремі спроби визначити фрактал. Перша спроба базується на топологічній розмірності множин (точка має розмірність 0, лінія – 1, плоска фігура – розмірність 2). Фігури з дробовою розмірністю описують властивості фракталів. Але це визначення не точне, оскільки трапляються фрактали з розмірністю 2. У комп'ютерній графіці використовують таке означення фрактала.

Фракталом, за Мандельбротом, називається геометрична фігура, яка складається з частин, які в певному розумінні подібні за формою на всю фігуру, тобто це геометрична фігура, в якій один і той самий фрагмент повторюється при кожному зменшенні масштабу (частина об'єкта схожа на сам об'єкт).

Характерною рисою фрактала є інваріантність (самоподібність) відносно масштабу, наприклад, гірський камінь має такі ж обриси, як і гірський хребет.

Існують *конструктивні (геометричні) та динамічні (алгебраїчні) фрактали.* Найбільш наочними є *геометричні фрактали*, оскільки їхня форма може бути описана як послідовність простих геометричних операцій. На площині їх отримують за допомогою деякої ламаної, яка називається *генератором*. За один крок алгоритму кожен із відрізків ламаної замінюється на ламану-генератор у відповідному масштабі. У результаті нескінченного повторення кроків цього алгоритму одержуємо геометричний фрактал. Тому алгоритми побудови фракталів в більшості випадків носять рекурсивний характер. Рекурсивність обумовлює властивість самоподібності фрактала. Прикладом геометричного фрактала є крива Коха. Послідовні наближення до фрактала Коха зображено на рисунку.

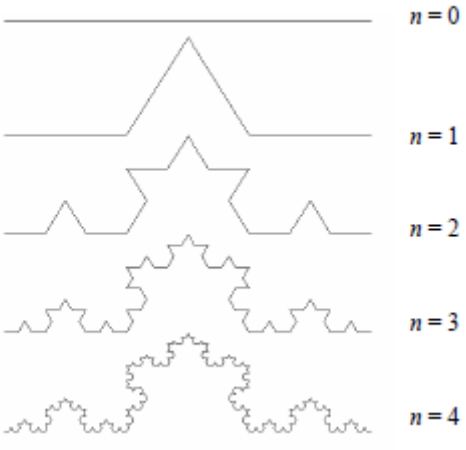


Рис. 1 - Фрактал Коха

Прикладом конструктивного фрактала може бути дерево, стовбур якого розділений на менші гілки. У свою чергу кожна з цих гілок ділиться на дві менші і т.д. Практично ми зупинимося на певному кроці (уявно цей процес можна продовжити до нескінченності). Фігура, що з'явиться в результаті цих операцій і є фракталом, в якому кожна його частина у свою чергу подібна на весь фрактал (рис. 10.2).

Поряд із конструктивними фракталами були відкриті множини, схожі на фрактали (вони можуть володіти масштабною інваріантністю лише наближено). Такі множини називаються *динамічними* (алгебраїчними) фракталами. Як правило, вони виникають у нелінійних дискретних динамічних системах. Якщо нелінійна система досить складна, то вона у своєму розвитку проходить етапи стійкого і хаотичного розвитку. Нелінійні системи володіють кількома стійкими станами (атракторами). Існують області початкових даних, з яких система потрапляє в область притягання атрактора. Замальовуючи області притягання різними кольорами, можна отримати кольоровий фазовий портрет цієї системи.

Динамічні фрактали одним із перших описав у 1918 р. французький математик Г. Жуліа. На той час були ще відсутні зображення фракталів. Комп'ютери зробили видимим те, чого не уявляли в часи Жуліа.

Створення фрактальної художньої композиції полягає не в малюванні, а в програмуванні. У програмних засобах зображення автоматично генеруються шляхом математичних розрахунків. Цей вид графіки використовується при створенні заставки на ТВ.

Базовим елементом фрактальної графіки є сама математична формула, ніяких об'єктів в пам'яті комп'ютера не зберігається і зображення будується по рівняннях.

Згідно означення *фрактал* – це структура, що складається з частин, які подібні до цілого.

Поняття фрактал вперше ввів Мандельброт. Ще до нього видатними вченими були відкриті класичні фрактали: множини Кантора, криві Пеано, функції Вейєрштрасса, сніжинки Коха, і коврик Серпинського. Але тільки Мандельброт та його учні зуміли звести розрізнені фрактали в єдину стрункy науку, відкривши при цьому нові фрактали, які моделювали різні природні об'єкти та явища. Завдяки виходу фундаментальних праць по фрактальній геометрії розпочалося її широке застосування для опису різноманітних явищ та процесів – від фрактального броунівського руху до кіноіндустрії [1,2]. В основі концепції фрактальності лежить властивість виділяти об'єкти різного масштабу згідно ієрархічного принципу

організації Всесвіту. Основна гіпотеза, що лежить в основі фракталів – це само подібність. тобто вигляд фрактальної структури не змінюється при обмежених масштабних перетвореннях. Фрактальним підходом можна описувати структури неживої природи: лінії берегів, рельєф місцевості [3], обриси хмар, структури корисних копалин, так і живої: системи кровообігу людини, будови нирок і легенів, які нагадують по структурі дерева з кроною, процесів: економічних, водоспадів, турбулентних процесів, які використовуються при прогнозі погоди. Алгоритми фрактальної геометрії використовують для стиснення зображень, дистанційному зондуванні і радіолокації, моделюванні фракталоподібних розсіювальних систем, еволюційних обчисленнях, тощо.

Самоподібність фракталів використовується для синтезу зображень об'єктів природи, яким теж властива самоподібність, наприклад, листу папороті.

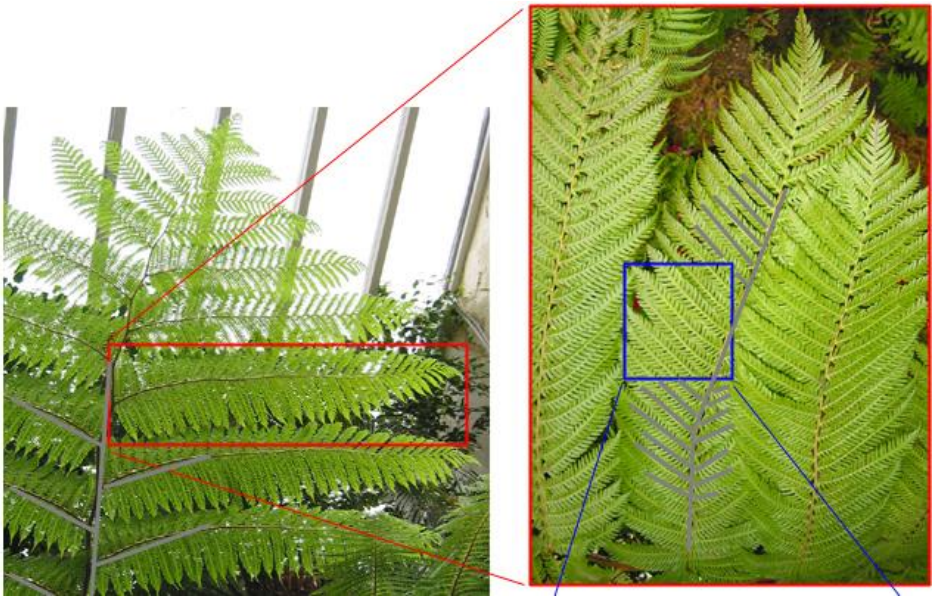


Рис. 2 – Самоподібність листа папороті

Для порівняння, на рисунку показано зображення листа папороті, згенерованого за допомогою фрактальної графіки. Помітно, що і в даному випадку яскраво простежується самоподібність.

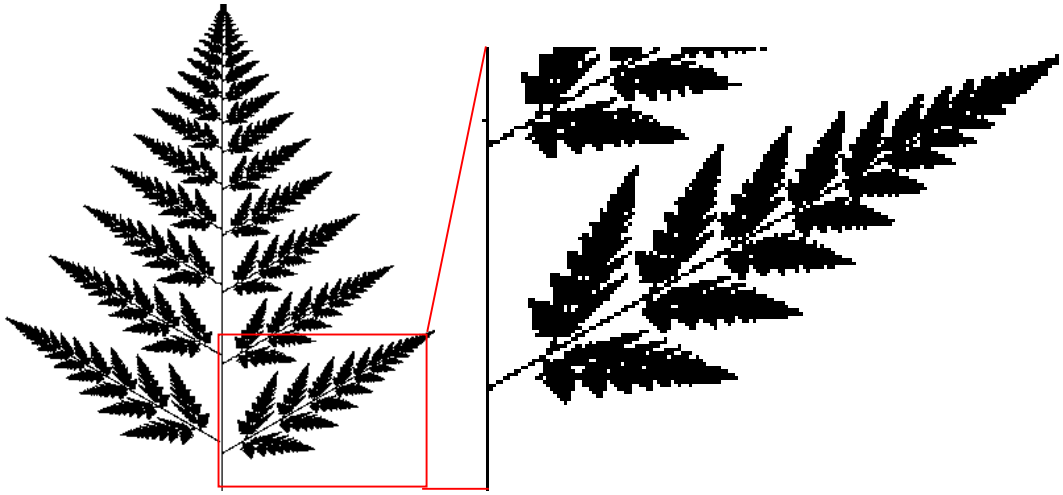


Рис. 3 – Самоподібність фракталу (лист папороті)

В самому загальному випадку невелика частина фрактального зображення містить інформацію про весь фрактал.

Звідси випливає, що елементарним примітивом фрактальної графіки є фрактал. В зв’язку з великою різноманітністю фракталів, використання засобів фрактальної графіки для синтезу реалістичних зображень стає дуже складною справою.

Роль фракталів в машинній графіці на цей час є досить великою. За їх допомогою можна задати криві або поверхні дуже складної форми. Фрактальна графіка активно використовується при генерації зображень, яким властива самоподібність – хмар, гір, поверхні моря.

2. Історія фрактальної графіки

Перші ідеї фрактальної геометрії виникли в 19 столітті. Кантор за допомогою простої рекурсивної (повторюваної) процедури перетворив лінію в набір незв'язаних крапок (так звана Пилом Кантора). Він брав лінію і видаляв центральну третину і після цього повторював те ж саме з відрізками. Пеано намалював особливий вид лінії (див.мал.1.1). Для її малювання Пеано використовував наступний алгоритм:

Рис. 4 - Алгоритм Пеано

На першому кроці він брав пряму лінію і замінював її на 9 відрізків довгою в 3 рази меншою, ніж довга вихідної лінії (Частина 1 і 2 рис. 4). Далі він робив те ж саме з кожним відрізком вийшла лінії. І так до нескінченності. Її унікальність у тому, що вона заповнює всю площину. Доведено, що для кожної точки на площині можна знайти точку, що належить лінії Пеано. Крива Пеано і пил Кантора виходили за рамки звичайних геометричних об'єктів. Вони не мали чіткої розмірності. Пил Кантора будувалася на підставі одновимірної прямої, але складалася з точок, а крива Пеано будувалася на підставі одновимірної лінії, а в результаті виходила площина

У багатьох інших областях науки з'являлися завдання, вирішення яких призводило до дивних результатів. Аж до 20 століття йшло накопичення даних про такі дивні об'єкти, без якої або спроби їх систематизувати. Так було, поки за них не взявся Бенуа Мандельброт – батько сучасної фрактальної геометрії і слова фрактал. Народження фрактальної геометрії прийнято пов'язувати з виходом у 1977 р. книги Б. Мандельброта "Фрактальна геометрія природи". Визначення фрактала, дане Мандельбротом, звучить так: "фракталом називається структура, що складається з частин, які в якомусь сенсі подібні цілому". Працюючи в IBM математичним аналітиком, він вивчав шуми в електронних схемах, які неможливо було описати за допомогою статистики. Поступово зіставивши факти, він прийшов до відкриття нового напрямку в математиці – фрактальної геометрії. Сам Мандельброт вивів слово *fractal* від латинського слова *fractus*, що означає розбитий (поділений на частини). І одне з визначень фрактала – це геометрична фігура, що складається з частин і яка може бути поділена на частини, кожна з яких представлятиме зменшену копію цілого.

Мандельброт досліджував перетворення комплексної площини. Втім, перетворення, досліджене Мандельбротом, можна представити просто як перетворення площини. Мандельброт розглядав траєкторії крапок, які виходять при цьому перетворенні, і вивчав

залежність картини, що виходить, від параметра С. Здавалося б, нічого цікавого чекати не доводилося: настільки простим здавалося перетворення. Фіксуючи параметр С, Мандельброт спробував встановити ті області на площині, виходячи з яких, крапки не "тікають" на нескінченність, а утворена при ітераційному процесі послідовність залишається в обмеженій околиці. Виявилось, що значення таких параметрів С утворюють зв'язну множину з дивно химерною межею, і форма основної частини множини повторюється і повторюється в різних масштабах. Це множина і було названо множиною Мандельброта яка зобажена на рис. 5.

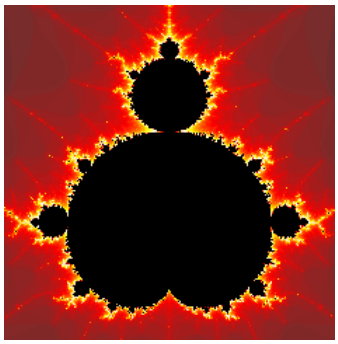


Рис. 5 - Множина Мандельброта

Щоб уявити собі фрактал, розглянемо приклад, наведений у книзі Б.Мандельброта «Фрактальна геометрія природи» став класичним – «Яка довжина берега Британії?». Все залежить від довжини інструменту, яким ми будемо користуватися. Помірявши берег за допомогою кілометрової лінійки ми отримаємо довжину. Однак ми пропускаємо невеликі заливи та пів-острови, які за розміром набагато менше лінійки. Зменшивши розмір лінійки до, скажімо, 1 метра – ми врахуємо ці деталі ландшафту, і, відповідно довжина берега стане більшою. Тепер виміряємо довжину берега за допомогою міліметрової лінійки, ми тут врахуємо деталі, які більше міліметра, довжина буде ще більшою. Відповіддю на таке, на перший погляд, просте питання може поставити в глухий кут – довжина берега Британії нескінченна.

Основна властивість фракталів – самоподібність. Будь-який мікроскопічний фрагмент фрактала відтворює його глобальну структуру. У найпростішому випадку частина фрактала являє собою просто зменшений фрактал.

Тому, щоб побудови фрактал візьміть звичайний мотив і повторюйте його, постійно зменшуючи розміри. І вийде структура, яка відтворює цей мотив у всіх масштабах. Беремо відрізок і середню його третину переламуємо під кутом на n градусів. Потім повторюємо це з кожною з частин вийшла ламаної – і так до нескінченності.

Якщо на кожному кроці не тільки зменшувати основний мотив, але також зміщувати і повертати його, можна отримати більш цікаві та реалістично виглядають освіти, наприклад, лист папороті або навіть цілі їх зарості. А можна побудувати досить правдоподібний фрактальний рельєф місцевості і покрити її дуже симпатичним лісом. У 3D Studio Max, наприклад, для генерації дерев використовується фрактальний алгоритм. І це не виняток – більшість текстур місцевості в сучасних комп'ютерних іграх представляють фрактали. Гори, ліс і хмари на картинці – фрактали.

Файли фрактальних зображень мають розширення fif. Зазвичай файли у форматі fif виходять трохи менше файлів у форматі jpg, але буває і навпаки. Найцікавіше починається, якщо розглядати картинки з усе більшим збільшенням. Файли у форматі jpg майже відразу демонструють свою дискретну природу – з'являється горезвісна драбинка. А ось fif файли, як

і належить фракталам, зі зростанням збільшення показують все нову ступінь деталізації структури, зберігаючи естетику зображення.

Фрактал (лат. fractus– дроблений) – термін, що означає геометричну фігуру, яка володіє властивістю самоподібності, тобто складену з декількох частин, кожна з яких подібна всій фігурі загалом. Слід зазначити, що фрактал – це не кінцева форма (її ніхто не бачив), а лише закон побудови цієї форми. Деякі вчені справедливо вважають термін "фрактал" геном формоутворення.

Проте об'єкти, які тепер називаються фрактальними зображеннями, досліджувались задовго до того, як їм було дано таку назву.

В етноматематиці, наприклад в роботах Рона Еглаша "Африканські Фрактали", згадується про наявність фрактальних геометричних фігур в мистецтві тубільців. Так, роботи американського експресіоніста, Джексона Поллока багато чим дуже схожі на фрактали. У 1872 р. Карл Веєрштрасс знайшов приклад функції з не інтуітивною особливістю, скрізь неперервної, але ніде недиференційованої – графік цієї функції тепер називався б фракталом. У 1904 р. Хельга Фон Кох розробив геометричне означення схожої функції, яка має назву Сніжинки Коха (див. рис. 6). Ідею самоподібних кривих, що складаються із частин, схожих на ціле, було далі розвинено Полем П'єром Леві у своїй роботі він описав нову фрактальну криву, відому тепер як Крива Леві (див. рис. 7).

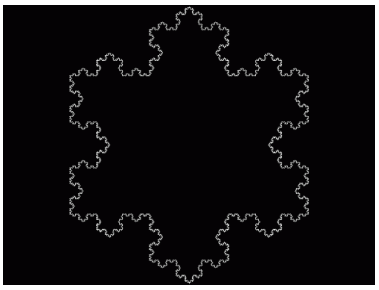


Рис. 6 - Сніжинка Коха

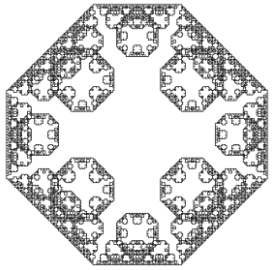


Рис. 7 - Крива Леві

Ґеорг Кантор навів приклади підмножин дійсних чисел із незвичними властивостями – ці множини Кантора тепер також визнаються як фрактали. Ітераційні функції на комплексній площині досліджувались в кінці XIX та на початку XX століття Анрі Пуанкаре, Феліксом Кляйном, П'єром Фату та Ґастоном Жюліа. Проте, на жаль, у них забракло засобів, щоб відобразити красу відкритих ними об'єктів.

Заслуговує на увагу і той факт, що поява фракталів (ще не отримавших цієї назви) в математичній літературі близько ста років тому, була зустрінута з неприязню, як це бувало і в історії розвитку багатьох інших математичних ідей. Один відомий математик, Шарль Ерміта, навіть охрестив їх монстрами. Принаймні, загальна думка визнала їх патологією, що представляє інтерес тільки для дослідників, які зловживають математичними примхами, а не для справжніх учених. У результаті зусиль Бенуа Мандельброта таке ставлення змінилося, і фрактальна геометрія стала шанованою прикладною наукою, а з надзвичайно стрімким розвитком комп'ютерної техніки – і основою для появи нового типу графіки, що на сьогодні є одним з найперспективнішим видом комп'ютерної графіки.

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

Її можливості важко переоцінити. Фрактальна комп'ютерна графіка дозволяє створювати абстрактні композиції, де можна реалізувати такі композиційні прийоми як, горизонталі і вертикалі, діагональні напрями, симетрію і асиметрію тощо.

З чим же ж можна порівняти фрактальні зображення? Наприклад, зі складною структурою кристала, зі сніжинкою, елементи якої вибудовується в одну складну структуру. Цю властивість фрактального об'єкта може бути вдало використано при складанні декоративної композиції або для створення орнаменту. На сьогоднішній день розроблені алгоритми синтезу коефіцієнтів фрактала, які дозволяють відтворити копію будь-якої картини як завгодно близько до вихідного оригіналу.

З точки зору машинної графіки фрактальна геометрія незамінна при генерації штучних хмар, гір, поверхні моря. Фактично завдяки фрактальній графіці знайдений спосіб ефективної реалізації складних неевклідових об'єктів, образи яких дуже нагадують природні (див. рис. 8). Геометричні фрактали на екрані комп'ютера – це візерунки, побудовані самим комп'ютером за заданою програмою. Крім фрактальної графіки все більше набувають популярності такі напрямки вивчення фракталів як анімація, живопис, музика, література.

Рис. 8 - Приклад картини, створеної за допомогою засобів фрактальної графіки

3. Основні види фрактальних зображень

Незважаючи на те, що більшість людей асоціює поняття фракталу та фрактальної графіки з чимось хаотичним, з тим, що не можна структурувати, фрактали можна поділити на три основні групи (відповідно до того, що лежить в їх основі, будь-то геометричні операції, формули чи певні дії з кольором): геометричні, алгебраїчні та стохастичні.

Геометричні фрактали

Фрактали цього класу найбільш наочні. Саме з них починалася історія фракталів. Зображення цього типу отримують за допомогою геометричних маніпуляцій над початковою фігурою, яку називають ініціюючим елементом, наприклад, у двовимірному просторі такою фігурою є деяка ламана, у тривимірному ж – площина. Шляхом застосування деякого набору правил до початкового елементу, він перетворюється на більш складну геометричну фігуру, фрактал з якої можна отримати, якщо провести нескінченну кількість ітерацій над кожною частиною новоутвореної фігури.

Фрактали цього типу будуються поетапно. Спочатку зображується основа, потім деякі частини основи заміняються на фрагмент. На кожному наступному етапі частини вже побудованої фігури, аналогічні заміненими частинам основи, знову заміняються на фрагмент,

взятий у потрібному масштабі. Кожного разу масштаб зменшується. Коли зміни стають візуально непомітними, вважається, що побудована фігура наближена до фракталу і дає уявлення про його форму. Проте для отримання самого фракталу, потрібна нескінченна кількість етапів. Змінюючи основу, можна отримати багато різних геометричних фракталів.

Геометричні фрактали з одного боку являються об’єктом серйозних наукових досліджень, а з іншого – їх можна легко розпізнати і "побачити", навіть не будучи спеціалістом у даній галузі. Таке поєднання рідко зустрічається в сучасній математиці, де всі об’єкти задаються за допомогою слів і символів, зрозумілих лише людям із відповідною освітою.

Виявляється, що більшість геометричних фракталів можна намалювати на листочку у клітинку. Слід звісно ж пам’ятати, що отримані таким шляхом зображення будуть лише кінцевим наближенням нескінченних по своїй суті фракталів. Наприклад, маючи достатньо великий аркуш паперу, олівець та запас вільного часу можна намалювати такий трикутник Серпінського, що з відстані в кілька метрів неозброєне око буде сприймати це зображення як справжній фрактал. Очевидно, що комп’ютер виконає це набагато краще, збільшивши точність намальованого.

Прикладами геометричних фракталів слугують: сніжинка Коха, лист, трикутник Серпінського, криві Гільберта, криві Серпінського, килим Серпінського тощо.

Одним із перших досліджень вчених у галузі фрактальної графіки була сніжинка Коха. Її отримують із трьох копій кривої Коха, інформація про яку вперше з’явилась у статті шведського математика Хельге фон Коха в 1904 р. Ця крива була придумана як приклад неперервної лінії, до якої неможливо провести дотичну в жодній точці. Лінії з такою властивістю були відомі й раніше (наприклад, Карл Вейерштрас побудував таку лінію ще у 1872 р.), але особливістю кривої Коха була простота її конструкції.

Процес її побудови виглядає наступним чином: береться одиничний відрізок, ділиться на три рівні частини і замінюється середнім інтервалом рівностороннього трикутника без цього сегмента (див. мал. 1.6). У результаті утворюється ламана, що складається з чотирьох ланок довжиною 1/3. На наступному кроці потрібно повторити операцію для кожної з чотирьох одержаних частин. Таким чином, для отримання кожного з наступних поколінь, всі ланки попереднього покоління необхідно замінити за даним правилом. Крива n-го покоління при будь-якому кінцевому n називається предфракталом. На рис. 9 представлені 7 поколінь кривої. Коли n буде прямувати до нескінченності, крива Коха стає фрактальним зображенням.

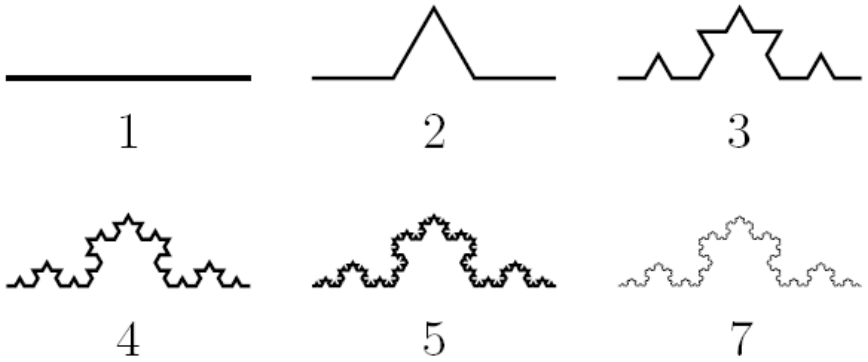


Рис. 9 - Етапи побудови кривої Коха

Для того, щоб отримати сніжинку Коха, за початковий елемент, замість одиничного відрізка, береться рівносторонній трикутник, кожна сторона якого і дорівнюватиме даному відрізку. Провівши відповідні перетворення з кожною стороною, отримаємо видозмінену замкнуту фігуру, яку прийнято називати сніжинкою Коха.

Для одержання іншого фрактального об'єкта потрібно змінити правила побудови. Для цього потрібно взяти за нульове покоління фракталу трикутник, знайти середину кожної сторони цього трикутника та з'єднати їх. Ця дія повторюється для кожного отриманого трикутника безліч кількість разів.

Отриманий таким чином фрактал був запропонований ще в 1915 р. польським математиком і названий трикутник Серпінського. Проте фрактал цього типу можна отримати ще й іншими способами (див. рис. 10).

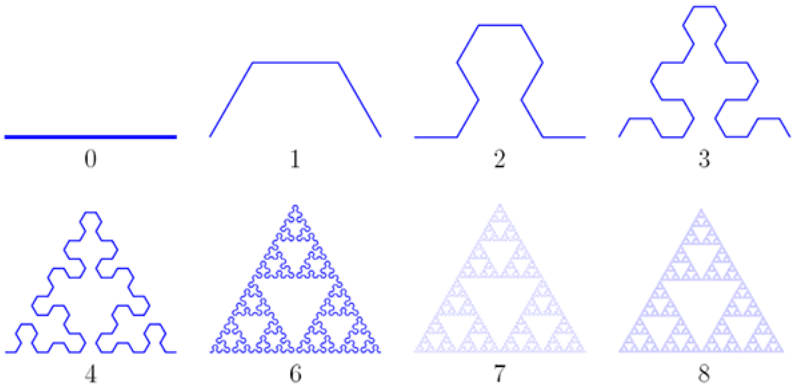


Рис. 10 - Один з способів побудови трикутника Серпінського

Як і інші фрактали, він має свої властивості:

1) Трикутник Серпінського має нульову площу. Після кожної ітерації площа того, що залишилось множиться на ¾, тобто стає все меншою і поступово наближається до нуля.
2) Цікавим є несподіваний зв’язок фракталу з комбінаторикою. Якщо в трикутнику Паскаля, всі парні числа виділити білим, а непарні – чорним, то видимі числа в деякому наближені сформують трикутник Серпінського.

На сьогодні існує безліч різновидів даного фракталу: квадрат Серпінського, піраміда Серпінського, Губка Менгера, що різняться зовнішньо, але будуються за одним і тим же правилом, змінюється лише початковий елемент або розмірність простору, як наприклад в піраміді Серпінського.

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

Алгебраїчні фрактали

Алгебраїчні фрактали – це найбільша група фракталів, яка отримала назву через те, що будують їх на основі алгебраїчних формул, іноді досить простих. Вони виникають при дослідженні нелінійних динамічних систем (звідки й інша назва – динамічні фрактали). Поведінку такої системи можна описати комплексною нелінійною функцією (многочленом)

Як приклад візьмемо будь-яку початкову точку z_0 на комплексній площині. Розглянемо нескінчену послідовність чисел на комплексній площині, кожне наступне з яких отримується з попереднього $z_0, z_1 = f(z_0), \quad z_2 = f(z_1), \dots z_{n+1} = f(z_n)$. В залежності від початкової точки z_0, така послідовність може вести себе по різному: прямувати до

нескінченності при $n \to \infty$; прямувати до 0; приймати декілька фіксованих значень і не виходити за їх межі; можливі і більш складні варіанти.

Щоб проілюструвати алгебраїчні фрактали звернемося до множини Manlelbrota (див. рис. 11).

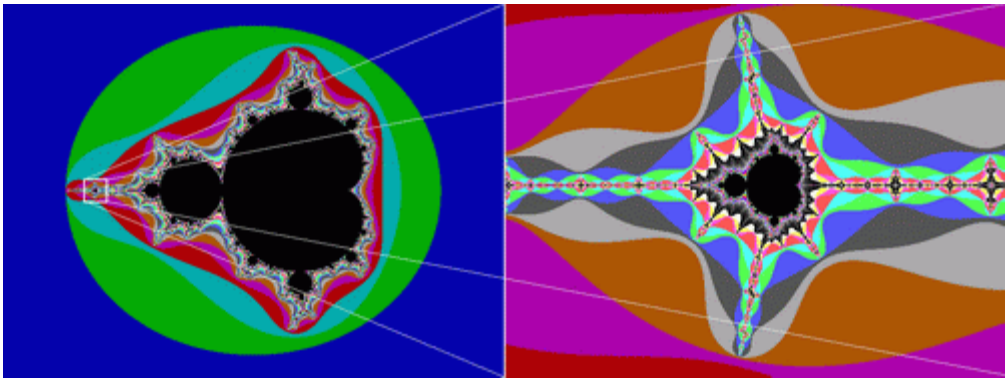


Рис. 11 - Множина Мандельброта

Для побудови цього фракталу необхідні комплексні числа. Комплексне число – це число, що складається з двох частин: дійсної та уявної, і позначається воно $a + bi$. Дійсна частина a – це звичайне число в нашому уявленні, а bi – уявна частина (i – називають уявною одиницею, і прийнято вважати, що при піднесенні до квадрату вона дає -1)

Комплексне число можна зобразити як точку на площині, у якої координата X це дійсна частина a, а Y це коефіцієнт при уявній частині *b*. Функціональна множина Мандельброта визначається за правилом:

$z_{i+1} = z_i * z_i + C$, де і z_i – іCмплексні змінні.

В залежності від параметра C функція $f(z)$ може прямувати до нескінченності або залишатись обмеженою. При цьому всі значення C при яких послідовність обмежена і утворюють таким чином множину Мандельброта. Дана множина була детально вивчена самим Мандельбротом та іншими математиками, які відкрили немало цікавих властивостей цієї множини.

Ще один тип фракталів, що включає в себе клас динамічних фракталів – так звані басейни. Одним із яскравих прикладів є басейни Ньютона. Формула для їх побудови засновані на методі розв'язання нелінійних рівнянь, який був відкритий великим математиком ще в XVII столітті. Застосовуючи загальну формулу метода Ньютона (де *n =0, 1, 2…)* для розв'язання рівняння $f(z) = 0$ до многочлена $z^k - a$ отримаємо послідовність $z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)}$

$z_{n+1} = ((k-1)z_n^k - a)/kz_n^{k-1}$.

Якщо в останній формулі підставити *k* =2, а в якості початкового наближення взяти $z_0 = a$, то отриманий фрактал будуватиметься за формулою $f(z) = z^3 - 1$ (див. рис.12)

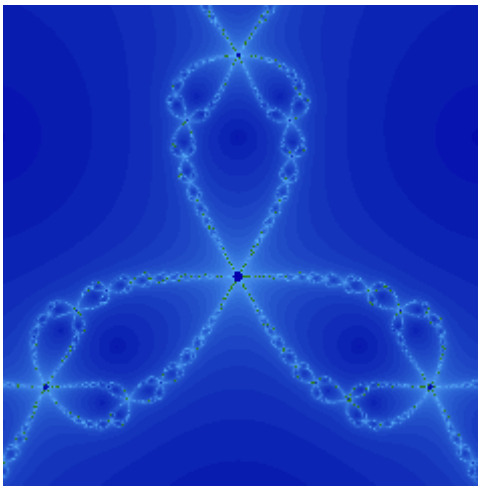


Рис.12 - Басейн Ньютона 3 порядку

Ще одним із різновидів алгебраїчних фракталів є пузирі Галлея. (див. рис. 13) Такі фрактали виходять, якщо в якості правила для побудови використовувати формулу Галлея для пошуку наближених значень коренів функцій. На відміну від метода Ньютона, даний спосіб ефективніший: послідовність сходиться до нуля функції швидше.

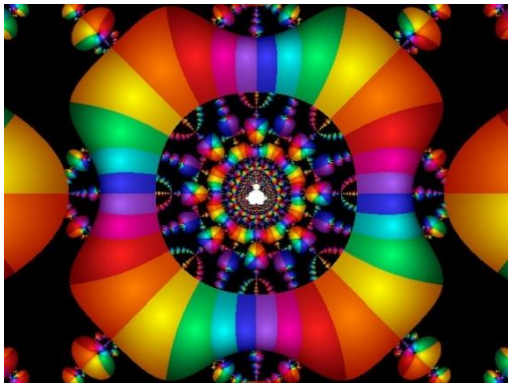


Рис. 13 - Метод Галлея

Крім вищезгаданих до динамічних фракталів також належить множина Жюліа, яку можна зобразити як множину точок, що мають конкретний тип поведінки. Фрактальні зображення даного типу часто використовують в моделюванні, кінематографі та для створення різного роду спецефектів.

Стохастичні фрактали

Фрактали, що отримуються у випадку, якщо в ітераційному процесі випадковим чином замінити якісь з параметрів називаються стохастичними. Термін "стохастичність" походить від грецького і означає "припущення".

Стохастичним природним процесом є броунівський рух. За допомогою комп'ютера такі процеси задаються досить просто: потрібно задати послідовність випадкових чисел і налаштувати відповідний алгоритм.

При цьому отримуються об'єкти досить схожі на природні – несиметричні дерева, порізані берегові лінії тощо. Двохвимірні стохастичні фрактали використовуються для моделювання рельєфу, ландшафтів, поверхні морів.

Спроможність фрактальної графіки моделювання образів живої природи обчислювальним шляхом часто застосовують для автоматичної генерації незвичайних ілюстрацій.

Типовим представником даного класу фракталів є "Плазма" (див. рис. 14).

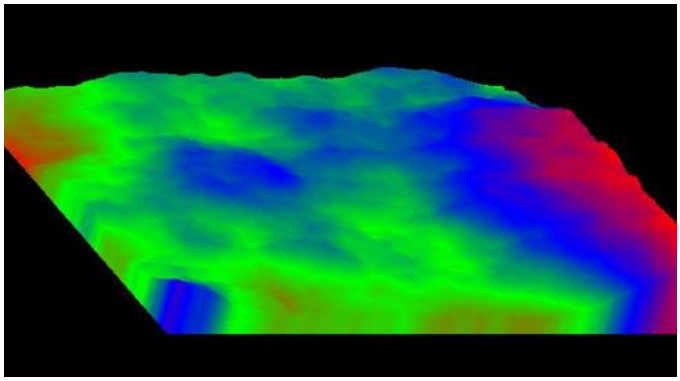


Рис. 14 - Плазма

Для її побудови візьмемо прямокутник і для кожного його кута визначимо колір. Далі знаходимо центральну точку прямокутника і розфарбовуємо її у колір рівний середньому арифметичному кольорів по кутах прямокутника плюс деяке випадкове число. Чим більше випадкове число – тим більш "рваним" буде малюнок. Якщо, наприклад, взятий колір крапки за висоту над рівнем моря, то отримаємо замість плазми – гірський масив.

Саме на цьому принципі моделюються гори в більшості програмах. За допомогою алгоритму, схожого на плазму будується карта висот, до неї застосовуються різні фільтри, накладаються текстури.

Системи ітерованих функцій (IFS)

Група фракталів стала відомою завдяки роботам Майкла Барнслі. Він намагався кодувати зображення за допомогою фракталів. Запатентувавши ідеї з кодування за допомогою фракталів, він створив фірму «Iterated Systems». І через деякий час вона випустила програмний продукт «Images Incorporated», в якому зображення переводлись з растрової форми у фрактальну. Це дозволило домогтися високих ступенів стиснення. При низьких ступенях стиснення якість малюнків поступалося якістю формату JPEG, але при високих картинки виходили більш якісними (див. рис. 15).

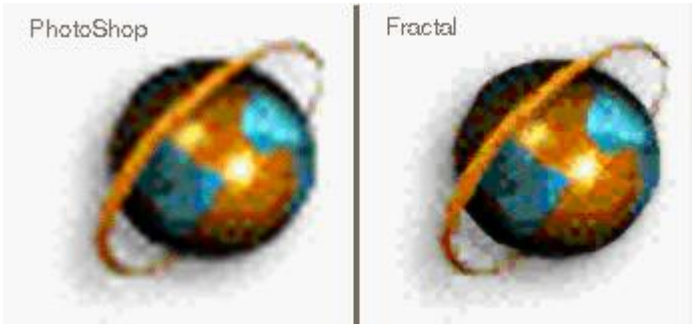


Рис. 15 - Приклад зображення виконаного в PhotoShop та фрактального

Якщо в L-systems (алгебраїчних фракталах) йшлося про заміну прямої лінії якимсь полігоном, то в IFS ми в ході кожної ітерації замінюємо якийсь полігон (квадрат, трикутник, коло) на набір полігонів, кожен їх яких підданий афінних перетворень. При афінних перетвореннях вихідне зображення змінює масштаб, паралельно переноситься уздовж кожної з осей і обертається на деякий кут.

4. Застосування фрактальної графіки

Сьогодні фрактальна геометрія пронизує багато наукових галузей, таких як: астрофізика, біологія, криптографія, навіть музика та література. Розглянемо деякі з прикладів використання фракталів та фрактальних зображень в науці.

Економіка. *Фрактальний аналіз ринків* – новий напрямок аналізу валютного та фондового ринку. Родоначальником фрактального аналізу ринків є Бенуа Мандельброт, який окреслив теорію у своїй книзі у співавторстві з Річардом Л. Хадсоном "(Не)слухняні ринки: фрактальна революція у фінансах." Фрактальний аналіз ринків, на відміну від теорії ефективних ринків, постулює залежність майбутніх цін від їх минулих змін. Таким чином, процес ціноутворення на ринках глобально детермінований, залежний від "початкових умов ", тобто минулих значень. Локально ж процес ціноутворення випадковий, тобто в кожному конкретному випадку ціна має два варіанти розвитку. Фрактальний аналіз ринків безпосередньо виходить з фрактальної теорії і запозичує властивості фракталів для отримання прогнозів.

Природничі науки. У фізиці фрактали природним чином виникають при моделюванні нелінійних процесів, таких як: турбулентний перебіг рідини, складні процеси дифузії-адсорбції, полум'я, хмари тощо. Фрактали використовуються при моделюванні пористих матеріалів, наприклад, в нафтохімії. У біології вони застосовуються для моделювання популяцій і для опису систем внутрішніх органів (система кровоносних судин).

Фрактали в астрофізиці. Ніхто не знає скільки точно зірок на небі, але, дивлячись на них, ви коли-небудь задумувались над тим як вони формувались? Астрофізики вважають, що ключем рішення цієї проблеми є фрактальна природа міжзоряного газу. Фрактальні розподіли є ієрархічними так як сліди диму, чи наприклад купчасті хмари в небі. Турбулентність формує як хмари в небі, так і своєрідні хмари у космосі, надаючи їм нелінійного, але повторюваного патерну, який неможливо було б описати без допомоги фрактальної геометрії та людини, що на належному рівні змогла б пояснити особливості цих перетворень і спрогнозувати можливі подальші перспективи, використовуючи засоби обробки фрактальних зображень [1].

Фрактали в біології. Біологи традиційно моделюють природу та природні процеси з використанням евклідової геометрії. Вони представляють серцебиття, як синусоїду, хвойні дерева, як конуси, клітинні мембрани у вигляді графіків або простої поверхні. Тим не менш, вчені прийшли до висновку, що багато природних об'єктів краще характеризується з використанням фрактальної геометрії. Біологічні системи і процеси, як правило, характеризуються багаторівневими підструктурами і водночас з повторювальною загальною картиною. Вчені виявили, що базова структура хромосома є деревоподібною, кожна хромосома складається з безлічі "міні-хромосом", і тому може розглядатися як фрактал. Самоподібність була виявлена також в наборі ДНК. На думку деяких учених-біологів фрактальні властивості ДНК можуть бути використані для вирішення еволюційних відносин серед тварин. Можливо, у майбутньому біологи, співпрацюючи з кваліфікованими спеціалістами з комп'ютерних наук, використають фрактальну геометрію для створення всеосяжної моделі структур і процесів, що спостерігаються в природі.

Радіотехніка. Використання фрактальної геометрії при проектуванні антенних пристроїв було вперше застосоване американським інженером Натаном Коеном, який жив в центрі Бостона, де була заборонена установка зовнішніх антен на будівлях. Натан вирізав із алюмінієвої фольги фігуру у формі кривої Коха і наклеїв її на лист паперу, потім приєднав до

приймача. Коен заснував власну компанію і налагодив серійний випуск такого типу радіоантен.

Інформатика. *Стиснення зображень.* Фрактальне стиснення зображень – це алгоритм стиснення зображень із втратами, заснований на застосуванні систем ітеруючих функцій (*IFS,* як правило є афінними перетвореннями) до зображень. Даний алгоритм відомий тим, що в деяких випадках дозволяє отримати дуже високі коефіцієнти стиснення для реальних фотографій природних об'єктів, що недоступно для інших алгоритмів стиснення зображень у принципі. Ці алгоритми були засновані на ідеї про те, що замість самого зображення можна зберігати стискаюче відображення, для якого це зображення (або деяке близьке до нього) є нерухомою крапкою. Один з варіантів даного алгоритму був використаний фірмою Microsoft при виданні своєї енциклопедії.

Через складну ситуацію з патентуванням широкого розповсюдження алгоритм не отримав. Майклом Барнслі та іншими було отримано кілька патентів на фрактальний стиск у США. Ці патенти покривають широкий спектр можливих змін фрактального стиснення і серйозно стримують його розвиток. Дані патенти не обмежують досліджень у цій області, тобто можна придумувати свої алгоритми на основі запатентованих і публікувати їх. Також можна продавати алгоритми в країни, на які не поширюються отримані патенти. Крім того, термін дії більшості патентів – 17 років з моменту прийняття і він закінчується для більшості патентів найближчим часом, відповідно використання методів, що покривалися цими патентами, стане гарантовано вільним.

Децентралізовані мережі. Система призначення IP-адрес у мережі Netsukuku використовує принцип фрактального стиснення інформації для компактного збереження інформації про вузли мережі. Кожен вузол мережі Netsukuku зберігає всього 4 Кб інформації про стан сусідніх вузлів, при цьому будь-який новий вузол підключається до загальної мережі без необхідності в центральному регулюванні роздачі IP-адрес, що, наприклад, характерний для мережі Інтернет.

Криптографія. У криптографії використовують фрактальні хаотичні системи, які мають таку властивість: їх поведінка суттєво залежить від заданих початкових умов, при цьому стан системи в певний момент часу неможливо виразити простою формулою. Хаотична поведінка фракталів використовується в криптографії для шифрування даних і генерації випадкових послідовностей чисел. У фрактальній криптографії немає циклів, замість них використовуються ітерації. Так, для обчислення n-ої ітерації функції, необхідно спочатку обчислити (n-1)-шу ітерацію і т. д. до найпершої. Як і в звичайній криптографії, чим більше виконано ітерацій – тим краще, тому що проітерована багато разів точка фрактала далеко віддаляється від початку і неможливо обчислити траєкторію точки не виконавши кожен крок рекурсії. Отже, якість шифрування виходить значно вищою в порівняні зі звичайними методами. Ще одна з характеристик фракталів, що використовується в криптографії – це чутливість до початкових умов, тобто два близьких початкових значення будуть розходитись по мірі роботи системи. Таку поведінку важко передбачити аналітичними методами без знання секретного ключа. Це усуває один тип атак, а саме атаки перебору ключів, які перебирають усі можливі ключі для розшифровки даних. Атаки є успішними дуже рідко, оскільки будь-яке намагання зламати ключ залежить від довжини ключа [2].

Фрактальна графіка. Фрактали широко застосовуються в комп'ютерній графіці для побудови зображень природних об'єктів, таких як: дерева, кущі, гірські ландшафти, поверхні

морів тощо. Фрактали широко використовують і в кінематографі, де, наприклад, режисерська задумка вимагає реалістичного пейзажу неіснуючої планети. Так, приміром, в "Поверненні Джедая" фрактали були використані для створення географії Місяця. З використанням фракталів можуть будуватися не тільки ірреальні зображення, але і цілком реалістичні (наприклад, фрактали нерідко використовуються при створенні хмар, снігу, берегових ліній, дерев і кущів тощо). Застосовувати фрактальні зображення можна в різних сферах, починаючи від створення звичайних текстур і фонових зображень, і закінчуючи фантастичними ландшафтами для комп'ютерних ігор або книжкових ілюстрацій. А створюються подібні фрактальні шедеври (так само як і векторні) шляхом математичних розрахунків. Будь-хто може створити своє власне оригінальне зображення. Так як це зробила італійка Сільвія Кордедда (див. рис. 16) Створивши серію фрактальних квітів, що тепер зачаровують не лише людей з інформаційною освітою, а й художників та мистецтвознавців.

Рис. 16 - Фрактальнне зображення Сільвії Кордедди

Народження фрактальної геометрії прийнято пов'язувати з виходом у 1977 р. книги "Фрактальна геометрія природи" Бенуа Мандельбротта. Його і вважають батьком сучасної фрактальної геометрії і слова фрактал. Визначення фрактала, дане Мандельбротом, звучить так: "фракталом називається структура, що складається з частин, які в якомусь сенсі подібні цілому". Мандельброт вивів слово *fractal* від латинського слова *fractus*, що означає розбитий (поділений на частини). І одне з визначень фрактала – це геометрична фігура, що складається з частин і яка може бути поділена на частини, кожна з яких представлятиме зменшену копію цілого.

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

Фрактали можна поділити на три основні групи (відповідно до того, що лежить в їх основі, будь-то геометричні операції, формули чи певні дії з кольором): геометричні, алгебраїчні, стохастичні та системи ітерованих функцій IFS.

Основними галузями застосування є: економіка (фрактальний аналіз ринків), фізика (моделювання нелінійних процесів), біологія (моделювання природних процесів), радіотехніка (використання фрактальної геометрії при проектуванні антенних пристроїв), інформатика **(**фрактальне стиснення зображень при застосуванні систем ітеруючих функцій IFS), децентралізовані мережі (призначення IP-адрес у мережі Netsukuku).

Контрольні запитання

1. Що називають фрактальною графікою?
2. Історія виникнення фрактальної графіки.
3. Що таке L-системи в контексті фрактальної графіки?
4. Яким чином можна синтезувати фрактальне зображення?
5. Назвіть сфери застосування фрактальної графіки.

Список використаних інформаційних джерел

1. Маценко В.Г. Комп’ютерна графіка: Навчальний посібник. – Чернівці: Рута, 2009 – 343 с.
2. Морозенко О.П. Інженерна графіка. / Навчальний посібник. Д.: НМетАУ. – 2013р. 105 стор.
3. Ванін В.В. Інженерна графіка. / К.: Видавнича група BHV., 2009 - 400 стор., іллюстр.

ЛЕКЦІЯ 7.
ТЕМА: ФОРМАТИ ГРАФЧНИХ ФАЙЛІВ

Графічний формат – *це метод запису графічної інформації в графічних файлах. Мета розробки графічних форматів – ефективно і логічно організовувати, зберігати і відновлювати графічні дані.*

Формат файлу – це структура даних, що зберігаються в цьому файлі, тобто порядок розташування даних у файлі.

У графічних файлах міститься інформація про спосіб подання зображення. Наприклад, растрове зображення представляється у вигляді двовимірної матриці, елементами якої є числа. У найпростішому випадку ці числа відповідають значенням яскравості пікселів (в термінах RGB і CMYK).

Векторні зображення представляються у файлі інакше: там немає поняття пікселів, але є поняття об'єктів (лінії, області заливки кольором та ін.), які мають свій спосіб опису.

Щоб файли різних форматів відрізнялися між собою, в різних системах, таких як, наприклад Window, використовують **розширення** – це додаткові символи до імені файлу, що пишуться через крапку. Наприклад, в файлі *Picture.jpg*, *Picture* – ім'я файлу, а *.jpg* – його розширення. Це простий спосіб пояснити всім програмам, якого формату цей файл. Так, програма PhotoShop не зможе відкрити файл .exe, навіть якщо він містить графічну інформацію.

Інакше, **формат** – це деякий опис (специфікація) того, що саме, де і в якому вигляді має бути представлено у файлі.

Для існуючого формату програмістами створюються як відповідні файли, так і програми їх перегляду і редагування. Проте буває так, що деякі графічні редактори зберігають зображення у файлах, що не реалізують всіх можливостей відповідного формату. Тому важливо правильно вибрати формат файлу у разі його збереження і подальшого його використання.

Таким чином, створивши зображення (за допомогою сканера, цифрового фотоапарата, графічного редактора), необхідно зберегти його у вигляді файлу. Для цього потрібно оптимізувати параметри, як самого зображення, так і власне файлу. Головне завдання при цьому – знайти компроміс між якістю зображення і обсягом файлу, що містить його.

В даний час існує більше двох десятків форматів графічних файлів, наприклад, BMP, GIF, TIFF, JPEG, PCX, WMF, CUR та ін. Є файли, які, крім статичних зображень, можуть містити анімаційні кліпи та / або звук, наприклад, GIF, PNG, AVI, SWF, MPEG, MOV та інше.

Важливою характеристикою цих файлів є здатність представляти дані, що містяться в них, в стислому вигляді.

Принципи стиснення графічних даних

Стиснення графічних даних (компресія) має дуже велике значення під час створення файлів з мультимедійною інформацією. Без нього файли мали б неприйнятно великий об'єм.

Введемо деякі поняття.

Алгоритми стиснення – набір інструкцій, що в скінченної кількості кроків призводить до перетворення вхідного коду цифрового зображення в код меншого об’єму, усуваючи надмірність.

Метод стиснення – більш загальне поняття, ніж алгоритм, метод стиснення може бути реалізований різними алгоритмами , тобто реалізується одна і та ж ідея, але по-різному, залежно від мети розробки.

Код – це скінченна послідовність бітів, а довжина коду – це число бітів в ньому.

Ті ж самі дані можуть бути записані по-різному. Можна записати у вигляді як вони є, а подати в упакованому вигляді, використовуючи алгоритм стиснення. Наприклад, інформацію у форматі PDF можна записати за допомогою різних алгоритмів стиснення для різних видів даних.

Всі існуючі в даний час алгоритми стиснення інформації можна розділити на два класи:

1 Алгоритми стиснення без втрати інформації (lossless) і алгоритми, що допускають втрату інформації. Стиснення без втрат засноване на видаленні надмірності вихідного представлення інформації, тобто на застосуванні більш економного кодування. Таке стиснення ще називають зворотним.

Під надмірною інформацією розуміють таку, що її видалення не помітне людині в силу обмеженості її органів чуття.

2 Стиснення з втратами (lossy) базується на видаленні деякої частини інформації. У ряді випадків ці втрати виявляються практично непомітними для зору або цілком допустимими. Це стосується головним чином зображень типу фотографії. Досвід показує, що досить часто за рахунок незначної втрати якості такого зображення можна істотно скоротити обсяг файлу. Стиснення з втратами називають також незворотним.

Розглянемо основну ідею алгоритмів стиснення даних без втрат на простому прикладі. Припустимо, вихідна інформація представлена у вигляді такої послідовності букв: АААААББВВВАААААААА. Можна цю інформацію подати більш економно: А4Б2В3А7. Тут число вказує кількість повторень літери, зазначеної ліворуч від числа.

Алгоритм декодування цієї послідовності очевидний: кожну букву необхідно записати стільки разів, скільки вказано числом праворуч від неї. В даному випадку ми скоротили обсяг даних в 2 рази, причому без втрат. Дану задачу вдалося виконати, оскільки вихідне подання інформації було надмірним.

Багато зображень (наприклад, темні лінії на білому фоні) містять великі області однакових пікселів. Кожному пікселю відповідає одне число (яскравість у відтінках сірого або індекс) або кілька чисел (три, за кількістю базових кольорів). Для таких зображень описаний вище алгоритм стиснення дає хороші 6результати.

Метод стиснення RLE легко реалізується, але не є ефективним для більшості растрових файлів. Найкраще цей метод працює із зображеннями, які містять обмежену кількість кольорів і великі області однотонного зафарбовування, і гірше – з відсканованими або іншими «фотореалістичними» зображеннями, так як в них немає довгих рядків однакових пікселів, які можна ефективно стиснути. Коли кожен піксель зображення відрізняється від сусіднього, то метод RLE призводить до збільшення розміру отриманого файлу в порівнянні з вихідним файлом.

Переваги методу: швидко працює; не вимагає додаткової пам'яті при компресії/декомпресії. Ступінь стиснення для деяких зображень може бути істотно підвищена за рахунок зміни порядку кольорів у палітрі зображення.

Метод LZW (названий за першими літерами прізвищ його розробників: Lempel, Ziv і Welch) полягає в пошуку однакових послідовностей (фраз) пікселів у всьому файлі. Виявлені послідовності зберігаються в таблиці, їм присвоюються більш короткі коди (ключі). Так, якщо в зображенні є набори з червоного, жовтого і зеленого пікселів, що повторюються в зображенні, наприклад, 50 разів, то алгоритм LZW виявляє це, привласнює даному набору пікселів окремий код (число), а потім зберігає цей код 50 разів. Метод LZW так само, як і RLE, краще працює на ділянках однорідних, вільних від шуму кольорів. Він дає набагато кращі результати, ніж RLE, при стисненні довільних графічних даних, але процеси кодування і розпакування відбуваються повільніше. Метод LZW використовується, наприклад, при створенні файлів формату GIF.

Приклад дії алгоритму LZW.

Розглянемо послідовність ABABAAACAAAAD і для простоти припустимо, що кожна буква являє собою 2-бітову величину даних. Шифратор і дешифратор LZW починають з однієї і тієї ж таблиці. Вони обидва відстежують ситуації, коли таблиця розширюється. Для нашого прикладу початкова кодова таблиця виглядає наступним чином:

A: 00 B: 01 C: 10 D: 11.

Алгоритм LZW шукає найдовшу послідовність, яку він здатний розпізнати як найдовшу з тих, що можна вставити в таблицю. Він знаходить першу величину A і розпізнає її, потім перевіряє послідовність AB і не розпізнає її. Тоді він розширює коди у вихідній таблиці і вносить новий елемент в таблицю для тієї величини, якої не розпізнає. Таблиця шифрування стає наступною:

A: 000 B: 001 C: 010 D: 011 AB: 100.

Дешифратор при цьому отримує ті ж самі дані і додає в таблицю код для AB. Тепер і шифратор, і дешифратор можуть розпізнати наступний екземпляр послідовності AB. Таким чином, один 3-бітовий код замінює два 2-бітових. У наведеному простому прикладі це невеликий виграш, але він набагато істотніше в реальних ситуаціях. Навіть у цьому випадку всі багаторазові появи величини будуть вкладені в таблицю і, отже, переведені в 3-бітовий код.

Методом LZW стискаються файли до 1/3 або 1/4 від їх початкового розміру. Насичені зображення, що містять великі блоки однотонного забарвлення або повторювані кольорові ділянки, можуть стискатися ще більше – до 1/10 їх початкового розміру. Разом з тим, відскановані фотографії або аналогічні зображення, що не містять повторення кольорових ділянок, стисненню методом LZW піддаються погано. Відсутність повторюваних значень пікселів робить процес для таких файлів важким і дуже рідко успішним (як і у випадку стиснення, способом RLE).

Застосовується при 8-бітових зображеннях, побудованих на комп'ютері. Стиснення може бути досягнуто в 1000 разів тільки на одноколірних зображеннях розміром кратним приблизно 7 Мб.

LZW є універсальним, і саме його варіанти використовуються в звичайних архіваторах.

Алгоритм Хаффмана, запропонований в 1952 р, став одним з класичних алгоритмів. Він використовує тільки частоту появи однакових байтів в зображенні і зіставляє символи вхідного потоку, що зустрічаються частіше, з ланцюжком біт меншої довжини, а що зустрічаються рідко – з ланцюжком більшої довжини. Для збору статистики виконується два проходи по зображенню.

Алгоритм заснований на запису в файл таблиці відповідності кодованих символів і ланцюжків, що кодують. У деяких випадках використовується або постійна таблиця, або таблиця, яка будується в процесі архівації/розархівації. Ці прийоми позбавляють від двох проходів по зображенню і необхідності зберігання таблиці разом з файлом. Кодування з фіксованою таблицею застосовується в якості останнього етапу архівації в JPEG і в алгоритмі CCITT Group 3.

Алгоритм практично не застосовується до зображень в чистому вигляді, а використовується як один з етапів компресії в більш складних схемах.

Коефіцієнти компресії: 8, 1.5, 1.

Переваги: єдиний алгоритм, який не збільшує розмір вихідних даних.

Алгоритм Хаффмана з фіксованою таблицею CCITT Group 3 був запропонований третьою групою по стандартизації Міжнародного Консультаційного Комітету з Телеграфу і Телефону (Consultative Committee International Telegraph and Telephone)(США) для обробки дворівневих чорнобілих зображень в телекомунікаціях. При роботі алгоритму послідовності поспіль чорних і білих точок у зображенні замінюються числом, рівним їх кількості. А цей ряд вже, у свою чергу, стискається по Хаффману з фіксованою таблицею. Такі схеми кодування і відповідно декодування є досить швидкими і не вимагають спеціального обладнання з надання інформації декодеру Group 3 для виявлення та коригування помилок.

Всі сучасні факс-машини і факс-модеми підтримують це стиснення, крім того, воно реалізовано у форматі TIFF, як одна з можливостей вибору.

Алгоритм застосовується для двоколірних чорно-білих зображень, в яких переважають великі простори, заповнені білим кольором.

Коефіцієнти компресії: кращий коефіцієнт в межі 213, середній 2, в гіршому випадку файл збільшується в 5 разів.

Переваги: надзвичайно простий у реалізації, швидкий і легко реалізується апаратно.

Наведені вище алгоритми стиснення без втрат досить універсальні і покривають всі типи зображень, але, за сьогоднішніми мірками, у них занадто маленький коефіцієнт стиснення.

Алгоритми стиснення з втратами

Останнім часом з'явилися системи, у яких дуже високі вимоги до ступеня компресії. Наприклад,це стосується гіпертекстової системи WWW, де оформлення інформаційних або рекламних сторінок стає все більш яскравим і кольоровим, що позначається на розмірі зображень.

Це призвело до створення нового типу алгоритмів – алгоритмів стиснення із втратою інформації. У них можна задавати коефіцієнт стиснення, з яким буде пов'язана певна ступінь втрати якості. При такому підході важливий компроміс між розміром та якістю зображень.

Одна з серйозних проблем комп'ютерної графіки полягає в тому, що на сьогодні не знайдений адекватний критерій оцінки втрат якості зображення.

Найкраще втрати якості зображень оцінюються візуально. Відмінним вважається стиснення, після якого неможливо розрізнити на око початкове і розархівоване зображення. Якщо тільки при порівнянні двох зображень, що знаходяться поряд, можна сказати, яке з них піддавалося архівації, то стиснення вважається хорошим. При подальшому збільшенні ступеня стиснення, як правило, стають помітні побічні ефекти, характерні для даного алгоритму. Тому алгоритми архівації з втратами не рекомендується використовувати при стисненні зображень, які в подальшому необхідно або друкувати з високою якістю, або обробляти програмами розпізнавання образів.

Розглянемо деякі алгоритми стиснення з втратами.

Алгоритм JPEG – один з найновіших і досить потужних алгоритмів. Практично він є стандартом де-факто для повнокольорових зображень. Стиснення в JPEG здійснюється за рахунок плавності зміни кольорів у зображенні.

Алгоритм розроблений в 1991 році групою експертів в області фотографії спеціально для стиснення 24-бітних зображень. JPEG – Joint Photographic Expert Group – підрозділ в рамках ISO. В цілому, алгоритм заснований на дискретному косинусоідальному перетворенні (ДКП), що застосовується до матриці зображення для отримання деякої нової матриці коефіцієнтів. Для отримання вихідного зображення застосовується зворотне перетворення.

ДКП розкладає зображення по амплітудах деяких частот. Таким чином, при перетворенні ми отримуємо матрицю, в якій значна кількість коефіцієнтів або близькі, або дорівнюють нулю. Крім того, завдяки недосконалості людського зору, можна апроксимувати коефіцієнти більш грубо без помітної втрати якості зображення.

Для цього використовується квантування коефіцієнтів (quantization). У самому простому випадку – це арифметичний побітовий здвиг вправо. При цьому перетворенні втрачається частина інформації, але можуть досягатися великі коефіцієнти стиснення.

Наведемо опис конвеєра операцій, використовуваних в алгоритмі JPEG. Нехай ми стискаємо 24-бітове зображення.

Крок 1. Переклад зображення з колірного простору RGB в колірний простір YCrCb (або YUV), У ньому Y – складова яскравості, а складові Cr, Cb – компоненти, що відповідають за колір (хроматичний червоний і хроматичний синій).

За рахунок того, що людське око до компоненти яскравості чутливе більше, ніж до кольору, з'являється можливість стискати масиви для Cr- і Cbкомпонент з великими втратами (і з великими коефіцієнтами стиснення). Саме тому, при суміщенні кольорового і чорно-білого телебачення на сигнали, що несуть інформацію про колірні компоненти, виділяється більш вузька смуга частот.

Перехід з колірного простору RGB в колірний простір YCrCb і навпаки задається за допомогою матриці переходу:

$$\begin{vmatrix} Y \\ Cb \\ Cr \end{vmatrix} = \begin{vmatrix} 0.299 & 0.587 & 0.114 \\ 0.5 & -0.4187 & -0.0813 \\ 0.1687 & -0.3313 & 0.5 \end{vmatrix} * \begin{vmatrix} R \\ G \\ B \end{vmatrix} + \begin{vmatrix} 0 \\ 128 \\ 128 \end{vmatrix}.$$

Зворотне перетворення здійснюється множенням вектора YUV на зворотну матрицю

$$\begin{vmatrix} R \\ G \\ B \end{vmatrix} = \begin{vmatrix} 1 & 0 & 1.402 \\ 1 & -0.34414 & -0.71414 \\ 1 & 1.772 & 0 \end{vmatrix} * \left(\begin{vmatrix} Y \\ Cb \\ Cr \end{vmatrix} - \begin{vmatrix} 0 \\ 128 \\ 128 \end{vmatrix} \right).$$

Крок 2. Розбиття у вихідному зображенні компонент Y, Cb, Cr на робочі матриці ДКП (дискретного косинус-перетворення) розміром (8 x 8). Для компоненти яскравості Y робочі матриці вибираються з зображення без дискретизації. Для колірних компонент Cr і Cb матриці набираються через рядок і через стовпець, тобто з вихідної матриці фрагмента зображення розміром (16 x 16) виходить тільки одна робоча матриця ДКП. При цьому втрачається 3/4 інформації про колірні складові зображення, що відповідно збільшує ступінь компресії в два рази. Практика показує, що на результуючому RGB-зображенні це позначається не сильно.

Крок 3. Застосування ДКП до кожної робочої матриці.

Для фрактального алгоритму компресії розроблений набір методів, за допомогою яких можна регулювати ступінь стиснення і ступінь втрат.

По-перше, можна обмежити кількість афінних перетворень, забезпечивши ступінь стиснення не нижче фіксованої величини.

По-друге, можна вимагати, щоб у ситуації, коли різниця між фрагментом, що обробляється, і найкращим його наближенням буде вище певного порогового значення, або в випадку, коли алгоритм не може підібрати для фрагмента подібного йому зображення, цей фрагмент обов'язково необхідно подробити на більш дрібні фрагменти. Для кожного з цих фрагментів виконати алгоритм підбору подібного йому зображення. Але цю процедуру неможливо повторювати до нескінченності, інакше кількість перетворень стане занадто великою і алгоритм перестане бути алгоритмом компресії.

По-третє, можна заборонити дробити фрагменти розміром менше, припустимо, чотирьох точок. Зміною порогових значень і пріоритетів цих умов досягається гнучке управління коефіцієнтом компресії зображення в діапазоні від побітової відповідності до будь-якого ступеня стиснення. Причому, ефективність цього управління набагато вища, ніж у найближчого «конкурента» – алгоритму JPEG.

Коефіцієнти стиснення: 2-2000 (Задається користувачем).

Рекурсивний алгоритм. Англійська назва цього стиснення – **wavelet**. Перекладається також як хвильове стиснення, і як стиснення з використанням сплесків. Цей вид компресії використовує ідею когерентності (від лат. *cohaerens* — «що знаходиться в зв'язку») областей зображення. Алгоритм орієнтований на кольорові і чорно-білі зображення з плавними переходами і ідеально підходить для зображень типу рентгенівських знімків.

Зображення спрощено можна представити як таблицю, в комірках якої зберігаються кольори кожного пікселя. Замість кольору в комірках записуються значення яскравості. Значення яскравості знаходяться в діапазоні від 0 до 255.

Ідея алгоритму полягає в тому, що в чорно-білих зображеннях яскравість сусідніх пікселів відрізняється на невелике значення, зазвичай близька до 0.

Всі значення яскравості розбиваються на пари і знаходять полусуму значення яскравості пари пікселів b_{1i} і їх полу різницю b_{2i}

Так два числа a_i, і a_{i+1} завжди можна представити у вигляді суми і різниці чисел $b_{1i} = (a_i + a_{i+1})/2$ і $b_{2i} = (a_i - a_{i+1})/2$.

У результаті вихідна послідовність a_i може бути попарно переведена в послідовність b_{1i}, b_{2i}.

Приклад. Стискається рядок з 8 значень яскравості пікселів

(a_i): ((220, 211), (212, 218), (217, 214), (210, 202)).

На першому кроці отримані наступні послідовності:

b_{1i}: (215.5, 215, 215.5, 206) і b_{2i}: (4.5, □ 3, 1.5, 4),

причому, значення b_{2i} досить близькі до 0.

Операція повторюється для b_1, як для a_i.

Дана дія виконується як би рекурсивно, звідси і назва алгоритму. В результаті аналогічних дій з (215.5, 215, 215.5, 206) отримують послідовності (215.25, 210.75); (0.25, 4.75). (Знання значень величин полусуми a і полурізниці d дасть можливість знайти і самі числа: перше значення числа в парі =**a** – **d**, друге значення числа в парі = **a** + **d**.)

Ці коефіцієнти округлюються до цілих і стискаються, наприклад, за допомогою алгоритму Хаффмана з фіксованими таблицями, а потім поміщаються в файл.

У наведеному прикладі перетворення застосовувалося тільки два рази. Реально застосування wavelet-перетворення виконується 4-6 разів. Більш того, додаткове стиснення можна отримати, використовуючи таблиці алгоритму Хаффмана з нерівномірним кроком. В результаті досягаються помітні коефіцієнти стиснення.

Алгоритм для двовимірних даних реалізується аналогічно. Для зображення розміром (512 x 512) пікселів після першого перетворення будуть отримані чотири матриці розміром (256 x 256) елементів.

У першій матриці – B_1 –зберігатиметься зменшена копія зображення, у другій – B_2 – усереднені різниці пар значень пікселів по горизонталі, в третій – B_3 – усереднені різниці пар значень пікселів по вертикалі, в четвертій – B_4 – усереднені різниці значень пікселів по діагоналі: Вихідне зображення

B_1	B_2
B_3	B_4

Ці перетворення повторюються вдруге і отримуємо замість першої матриці чотири матриці розміром (128 x 128).

Якщо провести перетворення в третій раз, то в підсумку будуть отримані: чотири матриці розміром (64 х 64), три матриці розміром (128 х 128) і три матриці розміром (256 х 256). На практиці, при записі у файл, значеннями, одержаними в останньому рядку, нехтують, відразу отримуючи виграш приблизно на третину розміру файлу, показники зменшення такі: 1–1/4 –1/16 –
1/64 ....

Алгоритм дуже легко дозволяє реалізувати можливість поступової «прояви» зображення при його передачі по мережі. Крім того, оскільки на початку зображення фактично зберігається його зменшена копія, то спрощується показ «огрубіння» зображення по заголовку.

На відміну від JPEG даний метод оперує не блоками фіксованого розміру для одного зображення, наприклад, 8 х 8 пікселів, а блоками змінних розмірів:

2 х 2, 4 х 4, 8 х 8 тощо.

За рахунок цього вдається уникнути розщеплення зображення на «мозаїчні» квадрати.

Коефіцієнт стиснення задається і варіюється в межах 5-100 разів (задається користувачем). При спробі задати більший коефіцієнт на різких межах, особливо що проходять по діагоналі, проявляється «сходовий ефект» – сходинки різної яскравості розміром в декілька пікселів.

На закінчення представлені таблиці 1, 2, в яких порівнюються параметри різних алгоритмів стиснення зображень, розглянутих вище.

Таблиця 1 – Параметри алгоритмів

Алгоритм	Особливості зображення, за рахунок яких відбувається стиснення
RLE	Однакові кольори, що йдуть поряд: 2 2 2 2 2 2 15 15 15
LZW	Однакові підланцюжки: 2 3 15 40 2 3 15 40
Хаффмана	Різна частота появи кольору: 2 2 3 2 2 4 3 2 2 2 4
CCITT-3	Переважання білого кольору в зображенні, великі області, заповнені одним кольором
Рекурсивний	Плавні переходи кольорів і відсутність різких меж
JPEG	Відсутність різких меж
Фрактальний	Подібність між елементами зображення

Таблиця 2 – Порівняння параметрів алгоритмів

Алгоритм	К-ти стиску	Симетричність по часу	Призначення (зображення)	Втрати	Розмірність
RLE	32, 2, 0.5	1	3,4-х бітні	Ні	1D

LZW	1000, 4, 5/7	1.2-3	1-8 бітні	Ні	1D
Хаффмана	8, 1.5, 1	1-1.5	8 бітні	Ні	1D
CCITT-3	213(3), 5, 0.25	~1	1-бітні	Ні	1D
JBIG	2-30	~1	1-бітні	Ні	2D
Lossless JPEG	2 рази	~1	24-бітні, сірі	Ні	2D
JPEG	2-20	~1	24-бітні, сірі	Так	2D
Рекурсивне стиснення	2-200	1.5	24-бітні, сірі	Так	2D
Фрактальний	2-2000	1000-10000	24-бітні, сірі	Так	2.5D

Тенденції розвитку алгоритмів стиснення останніх років:

- орієнтація на фотореалістичні зображення з 16 мільйонами кольорів (24 біта);
- використання стиснення з втратами, можливість за рахунок втрат регулювати якість зображень;
- використання надмірності зображень у двох вимірах;
- поява сильно асиметричних алгоритмів; підвищення ступеня стиснення зображень.

Стиснення відео зображень

Відео зображення мають такі поняття: *кодеки, контейнери, потоки*, які також містять алгоритми, формати та розширення.

Кодек – є з'єднання з перших букв слів **ко**дера і **дек**одера або компресора і декомпресора, тобто ця програма реалізує алгоритм кодування.

Головне завдання кодеків формування потрібних аудіо та відео потоків. Усі потоки повинні бути поміщені в контейнер, який є необхідним, щоб записати файл для відтворення або передачі по мережі.

Контейнер визначає структуру файла, можливі кодеки, їх параметри і додаткові данні, таких як, наприклад, субтитри. Кодеки і контейнери визначають формат відео.

Можна сказати, що кодеки – це формат стиснення, контейнери — це формат упаковки в файлу або потік. Роздільна здатність відео — кількість точок, не щільності крапок.

Головною проблемою передачі відеоряду є стиснення великих масивів інформації і стрімке відновлення зображення для відтворення фільму зі швидкістю 30 кадрів/с.

Стиснення, яке використовується в кодеках, може бути внутрікадровим (interframe) і міжкадровим (intraframe). Внутрікадровий стиск виконується на кожному фреймі з використанням різних методів стиснення: RLE, JPEG, WIC (Wavelet Image Compression).

У міжкадровому стисненні використовується система ключових і проміжних кадрів. У ключові кадри записується повна інформація, в проміжні – відмінності або «дельти» в інформації ключових кадрів. Такий тип стиснення називається кодуванням з прогнозом.

Крім цього кодування можливе застосування двонаправленого передбачення, при якому поточний кадр кодується на основі відмінностей між ним, попереднім і наступним кадрами.

Цей підхід реалізовано в стандарті **MPEG**, заявленому в 1992 році групою експертів у галузі рухомих зображень (Motion Picture Experts Groups), для кодування і синхронізації цифрового відео та звуку в єдиному потоці даних.

Відеодані MPEG містять три типи закодованих кадрів:

- I-фрейми (Inter Frames) – внутрікадрове кодування; – P-фрейми (Predicted Frames) – кодування з пророкуванням;
- B-фрейми (Bidiretional Frames) – двоспрямоване пророкування.

Тобто в P-фрейми записані відмінності між поточним кадром і попереднім I- або P-фреймом, а у B-фрейм – відмінності між поточним кадром і двома (попереднім і наступним) I- або P-кадрами.

Після міжкадрового кодування, в якому буде знята тимчасова надмірність, I- P- або B-фрейм стискається методом: JPEG, що знижує вже просторову надмірність. Результати тимчасового і просторового кодування будуть записані в потік даних MPEG.

За рахунок об'єднання просторового і тимчасового стискування може бути отримана результативна ступінь стиснення - до 200: 1 (прийнятною вважається ступінь стиснення між 16: 1 і 40: 1). Цей метод є асиметричним. Ступінь стиснення залежить від типу програми, що кодує, та пов'язана з якістю результуючого відео (чим вище необхідна якість, тим нижче ступінь стиснення).

В MPEG-потоці I-кадри, як правило, йдуть через одинадцять інших. Типовий приклад такої послідовності фреймів:

I B B P B B P B B P B B I B B P B B P B B P B ...

В декодер поступає спочатку кадр I, потім P, і після цього формуються два фрейми B. Перші десять кадрів цього відео потоку будуть декодовані в такому порядку: I P B B P B B P B B (0 3 1 2 4 5 6 9 7 8).

Звичайний розмір I-фрейма – 150 Кбіт, P-фрейма – 50 Кбіт, B-фрейма – 20 Кбіт, а частота відеоданих – 1,15 Мбіт/с. У підсумку, методи компресії стандарту MPEG дозволяють зводити швидкості надходження відео та аудіо даних до швидкостей обміну звичайних CD-ROM.

Групою JPEG розроблений кодек **MJPEG** (Motion JPEG) -– альтернатива MPEG. Цей кодек на апаратному рівні в реальному масштабі часу стискає за схемою JPEG «перехоплені» відео кадри з коефіцієнтом від 10: 1 до 20: 1. Компресія аудіо інформації цим кодеком не реалізується, стислі стандартним методом аудіодані включаються в потік MJPEG.

Переваги MJPEG:

- стиснення в реальному масштабі часу;
- можливість доступу до довільного кадру, так як проводиться тільки внутрикадровий стиск.

Недоліки:

- великий обсяг файлів MJPEG в порівнянні з файлами MPEG;
- менша швидкість відтворення;
- «погіршення якості відео з підвищенням коефіцієнта JPEG- стиснення.

Огляд графічних форматів

Розроблені наступні формати: растрові, векторні і метафайлові.

Растрові формати характеризуються тим, що кожна їхня точка (піксель) описується окремо. Задається піксельна карта зображення, по якій програма візуалізації реконструює зображення на відображаючій поверхні пристрою виводу, Це дає великі можливості для редагування, тому що всі пікселі можна обробляти незалежно один від одного. Такий формат використовується для зберігання реальних зображень – фотографій і відео зображень.

Найбільш поширені такі формати: Microsoft BMP, GIF, ICO, JPEG, PSD, PNG, TGA, TIFF.

Векторні формати характерні тим, що в них використовуються математичні описи елементів зображення – ліній або фігур (в векторній графіці ці об'єкти прийнято називати контурами). За математичними описами графічних форм (лінії, криві, сплайни) програма візуалізації виводить зображення, закодоване у векторному файлі, на поверхню пристрою виводу.

Векторні файли на відміну від растрових не підтримують стиск даних, але допускають двійкове кодування як альтернативу символьного кодування, що дозволяє зменшити обсяг файлу, але погіршує його читабельність і міжплатформову переносимість.

Можна назвати такі формати: SVG (Scalable Vector Graphics), EPS (Encapsulated PostScript), CDR (CorelDraw), AutoCAD DXF, Microsoft SYLK.

Метафайлові формати зберігають растрові і векторні дані, використовуються для їх транспортування і переміщення між апаратними платформами і для переміщення між програмними платформами.

Найбільш поширені формати: PDF, WPG, Macintosh PICT і CGM.

З розвитком комп'ютерних технологій до трьох вище перелічених форматів додалися наступні: сцени, анімації, об'ємні, мультимедіа, гіпертекстові і гіпермедіа, а також аудіо формати, VRML, формати опису шрифтів, мова опису сторінок, гібридні.

Розглянемо їх призначення.

Формати сцен. Містять інструкції, що дозволяють відновити зображення сцени цілком (просто векторні формати містять двовимірні описи частин зображення).

Файли формату сцени (іноді їх називають файлами опису сцени) були розроблені для зберігання стислого представлення зображень (або сцени). Векторні файли містять описи частин зображення, а файли сцени містять інструкції, які дозволяють програмі візуалізації відновити все зображення повністю. На практиці це іноді важко визначити, чи ми маємо справу з векторним форматом або форматом сцени.

Формати анімації. Найбільш прості – зберігають зображення цілком для їх подальшого відображення в циклі. Більш ускладнені – зберігають одне зображення і колірні таблиці. Ще більш складні – зберігають відмінності між двома послідовно відображеними фреймами.

Для мультиплікаційної анімації швидкість відображення 10 - 15 фреймів/с, для відео анімації – 20 фреймів/с і більше.

Об'ємні формати. Зберігають описи форми і кольору об'ємних моделей (уявних і реальних). Об'ємні об'єкти конструюються з багатокутних і гладких поверхонь, об'єднаних з описом таких атрибутів, як колір, текстура, відображення, за допомогою яких програма візуалізації реконструює об'єкт.

В результаті, моделі поміщаються в сцени з джерелами світла і камерами. 3D-дані використовуються програмами моделювання та анімації – 3D Studio фірми Autodesk, Lightwave фірми NewTek. Ці програми можуть відкоригувати зовнішній вигляд зображення після його візуалізації, змінюючи систему освітлення, відносне розташування елементів сцени і їх текстуру, і приписати елементам сцени рух.

Після цього програма створює ряд растрових файлів або кадрів, які збираються у фільм.

Мультимедіа формати. Об'єднують для зберігання в одному файлі графіку, аудіо- і відеодані. Це формати RIFF (Microsoft), Quick Time (Apple), MPEG і FLI (Autodesk).

Гіпертекст і гіпермедіа. Система гіпертексту забезпечує нелінійний доступ до даних. Потік гіпертексту можна декодувати по мірі надходження. Гіпертекстові мови швидше мови програмування, а не растрові або векторні формати, призначені для послідовної передачі потоків даних.

Гіпертекст забезпечує створення структури, яка дозволяє користувачеві обмежувати мультимедіа дані, відображати їх і інтерактивно переміщатися в них. Система WWW – це гіпертекст і гіпермедіа, в якій зберігаються великі інформаційні ресурси у вигляді файлів GIF, JPEG, PostScript, PDF і AVI.

Гіпермедіа – сплав гіпертексту і мультимедіа. Сучасні гіпертекстові мови і мережеві протоколи підтримують текст і шрифти, нерухому і динамічну графіку, аудіо-, відео- та об'ємні дані.

Формати аудіо файлів. Спочатку аудіо інформація зберігалася в аналоговій формі на магнітній стрічці. Для запису на CD-ROM або жорсткий диск вона піддається дискретизації з подальшим кодуванням (цей же процес проходять і відеодані). Після кодування вона зберігається або як необроблений потік цифрових даних або у форматі аудіо файлу.

Концептуально формати аудіо інформації ідентичні графічним форматам. У графічних файлах задаються кількість відліків на піксель, кількість колірних площин і кількість бітів на відлік. У заголовках аудіо файлів – аналогічна інформація про кількість відліків в секунду, про кількість каналів і про кількість бітів на відлік.

Формати мови моделювання віртуальної реальності. VRML створений на базі формату SGI Invertor. Це формат опису тривимірних сцен, розроблений в 1992 році для обміну інформацією тривимірного моделювання фірмою Silicon Grafics. У нього додані засоби з'єднання з URL в системі WWW.

В результаті 3D-дані засобами VRML кодуються у формат, придатний для обміну в Internet по протоколу HTPP.

Формати шрифтів (растрових, штрихових, сплайнових контурних). Файли шрифтів містять описи наборів буквено-цифрових знаків і символів в компактному і зручному для доступу форматі.

Оскільки з них зручно вибирати дані, пов'язані з окремим знаком, то вони іноді використовуються для зберігання графічної інформації.

Формати мов опису сторінки. Page Description Language (PDL) є мовами, що інтерпретуються і використовуються для опису компонування шрифтів і графіки сторінок і для передачі інформації на пристрої друку (принтери) і на пристрої відображення – екрани GUI (Graphic User Interface).

Гібридні формати. На даний момент досліджується можливість об'єднання неструктурованого тексту і растрових даних (змішаний текст), а також інтеграції інформації, об'єднаної в запису, і растрових даних (змішана база даних). Такі змішані формати можуть ефективно використовуватися для зберігання графічних даних.

Приклади растрових форматів

Формат BMP (BitMap)

Растровий формат, створений Microsoft, орієнтований на застосування в операційній системі Windows. Він використовується для представлення растрових зображень в ресурсах програм. Підтримуються тільки зображення в моделі RGB з глибиною кольору до 48 біт на піксель.

Формат передбачає використання найпростішого алгоритму стиснення RLE.

Формат TIFF (Tagged Image File Format)

Формат **TIFF** створений фірмою *Aldus* спеціально для зберігання відсканованих зображень. Виняткова гнучкість формату зробила його дійсно універсальним. Хоча з моменту його створення пройшло вже багато часу, досі є основним форматом, використовуваним для зберігання відсканованих зображень і розміщення їх у видавничих системах і програмах ілюстрування.

Версії формату існують на всіх комп'ютерних платформах, що робить його виключно зручним для перенесення растрових зображень між ними.

TIFF підтримує монохромні, індексовані, напівтонові і повнокольорові зображення в моделях RGB і CMYK з 8- і 16-бітними каналами. Великою перевагою формату залишається підтримка практично будь-якого алгоритму стиснення.

Найбільш поширеним є стиснення за алгоритмом **LZW**, що забезпечує дуже високу ступінь компресії. Цей же алгоритм використовується численними програмами стиснення загального призначення, що підтримують формат ZIP.

Формат PSD (PhotoShop Document)

Власний формат програми **Adobe PhotoShop**. Єдиний формат, який підтримує всі можливості програми. Найбільш використовуваний для зберігання проміжних результатів редагування зображень, оскільки зберігає їх пошарову структуру.

Всі останні версії продуктів фірми Adobe Systems підтримують цей формат і дозволяють імпортувати файли PhotoShop безпосередньо.

До недоліків формату можна віднести недостатню сумісність з іншими поширеними додатками і відсутність можливості стиснення.

Формат JPEG (Joint Photographic Experts Group)

Формат **JPEG** вперше застосував новий принцип стиснення зображень з втратами інформації. Він заснований на видалення із зображень тієї інформації, яка не сприймається (або слабо сприймається) людським оком. Позбавлене надлишкової інформації зображення займає набагато менше місця, ніж ісходне. Ступінь стиснення, а, отже, і кількість інформації, що видаляється плавно регулюється. Низькі ступені стиснення дають кращу якість зображення, а високі можуть істотно його погіршити.

Формат **JPEG** являється лідером серед графічних форматів.

Найбільш широко JPEG використовується при створенні зображень для електронного розповсюдження або в Інтернеті. Компактність файлів JPEG робить цей формат незамінним у тих випадках, коли розмір файлів критичний, наприклад, при передачі по каналах зв'язку. У поліграфії використовувати його не рекомендується.

JPEG підтримує напівтонові і повнокольорові зображення в моделях RGB і CMYK. Не підтримується прозорість. Використовується формат JPEG тільки для фотографічних зображень. На рисунках з чіткими межами і великими областями, що залиті, сильно проявляються дефекти стиснення. Особливо характерна – поява 'бруду' навколо темних ліній на світлому фоні і видимих квадратних областей. Останній дефект пов'язаний з тим, що алгоритм стиснення обробляє зображення квадратними блоками зі стороною 8 пікселів. При збереженні документів у форматі JPEG можна вказати необхідну ступінь стиснення, а, отже, і якості зображення.

Формат GIF (Graphics Interchange Format)

Формат створений найбільшою онлайновою службою CompuServe (нині підрозділ AOL, America Online) спеціально для передачі растрових зображень в глобальних мережах. Орієнтований на компактність і використовує алгоритм стиснення **LZW**, що не приводить до втрати якості.

Використовується тільки за своїм первісним призначенням – в Internet, оскільки підтримує тільки індексовані зображення. Версія дозволяє зберігати в одному файлі кілька індексованих зображень (майже як шари в Photoshop).

Браузери здатні демонструвати всі ці зображення по черзі, отримуючи в результаті нескладну анімацію. У файлі анімації зберігаються не тільки кадри анімації, але й параметри її демонстрації. GIF-анімація в силу своєї простоти найбільш поширена в Internet.

Крім того, один з кольорів в палітрі індексованого зображення можна оголосити прозорим. У браузері ділянки цього кольору будуть прозорими, крізь них буде видно фон сторінки.

Формат PNG (Portable Network Graphics)

Як видно з назви, формат **PNG** призначений для передачі зображень по мережі Internet. Це досить 'молодий' формат для Web-графіки, що конкурує з GIF. Всі останні версії браузерів підтримують його без спеціальних модулів.

Формат підтримує напівтонові і повнокольорові RGB-зображення з єдиним альфа-каналом, а також індексовані і монохромні зображення без альфа-каналів. Альфа-канал служить маскою прозорості.

Таким чином, формат PNG – єдиний з поширених форматів, що дозволяє отримувати повнокольорові зображення з прозорим фоном і являється універсальним форматом.

У форматі **PNG** використаний потужний алгоритм стиснення **LZW**. Будучи орієнтований на Web-графіку, формат **PNG** не підтримує багатоканальних зображень, колірних профілів і контурів обтравки.

Формат TGA (Targa)

Досить давній формат, створений спеціально для роботи з графічним акселератором Truevision. Цей акселератор широко використовується додатками на платформі DOS.

Формат підтримує 24-бітові RGB-зображення з одним альфа-каналом, а також напівтонові, індексовані і 16-бітові RGB-зображення без альфа-каналів. Відсічні контури і колірні профілі не підтримуються.

Формат ICO

ICO (Windows icon) дозволяє зберігати значки файлів в Microsoft Office. Такий формат має пряму аналогію CUR (Windows cursors), яка зберігає курсори.

Стандартний розмір – це квадратний значок із стороною 16, 32, 48 пікселів. Можуть використовуватися значки і з іншими розмірами, наприклад: 24, 40, 60, 72, 92, 108, 128 пікселів.

Такі значки зберігаються в стислому вигляді. Колір значків може бути природним (True Color), High Color або з певною палітрою (може складатися з 256, 16 або 2 кольорів).

Формат **ICO** дуже близький до **BMP**. Головна їхня відмінність – маска, яка «кладеться» на задній план зображення.

Формат FLM (Filmstrip)

Формат **FLM** – власний формат Adobe Premier, програми редагування відеоінформації та створення презентацій. PhotoShop вміє відкривати кадри, створені в Adobe Premier, і редагувати їх. Але якщо змінити колірну модель отриманого документа або видалити альфа-канал, то неможливо знову зберегти зображення в цьому форматі.

Векторні формати

Найпростіші векторні формати використовуються електронними таблицями: .WKS і .WKI (Lotus 1 – 2 – 3), .XLS (EXCEL) – і були розроблені для Intel). В даний час для підтримки обміну між різними електронними таблицями і платформами використовуються Lotus DIF (Date Interchange Format) і MS SYLK (Symbol Link Format – формат символьного зв'язку).

Але, в основному, більшість векторних форматів призначено для зберігання креслень і рисунків, розроблених САПР.

Переваги векторних файлів:

- зручні для зберігання зображень, складених з елементів, заданих лініями (кола, багатокутники), або з таких елементів, які можуть бути розкладені на найпростіші геометричні об'єкти (наприклад: шрифти True Type були розроблені на базі онлайнових кривих);
- легко масштабуються;
- файли, що містять дані у форматі ASCII, можуть бути модифіковані засобами текстового редактора.

Недоліки:

- важко застосовувати для зберігання реалістичних зображень, в яких колірна інформація може змінюватися на піксельному рівні;
- зовнішнє уявлення залежить від сумісності програми візуалізації з програмою, яка створила зображення, а також від складності набору геометричних примітивів та операцій рисування;
- векторні дані краще відображаються на векторних пристроях виводу, таких як плоттери і дисплеї з довільним скануванням, на растрових дисплеях векторна графіка ефективно відображається тільки при високому розділі;
- візуалізація векторних даних може зажадати значно більше часу, ніж візуалізація растрового файлу такої же складності, так як кожен елемент зображення має бути відтворений окремо і в певній послідовності.

Формат SVG

SVG (Scalable Vector Graphics) – це векторна графіка. Підтримує як статичну, так і анімовану графіку.

Переваги:

- формати файлів **SVG** можуть «прочитуватися» в текстових редакторах;
- векторний формат SVG – масштабованість;
- можливість використання растрових форматів в документах розширенням **SVG.** Формат зображення може бути PNG, GIF або JPG;
- передбачена анімація за допомогою мови SMI; – SVG є відкритим форматом.

Формат EPS

EPS (Encapsulated PostScript) – розширення для формату PostScript. Компанією Adobe був створений цей формат на базі мови PostScript. Такий формат використовується дизайнерами в поліграфії. Має можливість утримувати як растрові і векторні зображення, так і їх комбінації. Основні колірні моделі для **EPS**: RGB, CMYK.

Формат CDR

CDR (CorelDraw) – векторний графічний формат, який використовується програмою CorelDraw. Через постійне оновлення програми інформація, наприклад, створена в програмі CorelDraw2, може не «читатися» в пізніших версіях CorelDraw3, 4, 5.

Формат CDR гарантує високу якість зображень. Головний недолік:
документи такого формату не «відображаються» в інших графічних програмах.

Формат PDF

PDF(Portable Document Format)– це портативний формат документа. Комп'ютерні програми, які дозволяють працювати з таким форматом документа, – це Adobe Acrobat.

Він використовується як документ «оригінал макету» в додрукарській підготовці.

Для перегляду документів даного формату використовують програму Adobe Reader.

Перевага формату PDF в тому, що він дозволяє «стискати» інформацію. Тим самим можливе зберігання інформації з великим об'ємом. Всі документи з легкістю проглядаються. Це має велике значення при архівації та здійсненні додрукарської підготовки.

У 2007 році формат PDF став міжнародним стандартом ISO 32000.

Формат CGM

CGM (Computer Graphics Metafile) – графічний формат для зберігання та обміну даних. Тип: метафайл. Розширення – .cgm Призначений для професійних програм, що працюють з векторною графікою.

AutoCAD DXF (Data eXchange Format)

Формат обміну даними. Підтримує не тільки основні векторні елементи (коло, багатокутник), але й складні об'єкти, часто використовувані в САПР (тривимірні об'єкти, розмірні лінії і штрихування).

Метафайли

Можуть містити векторні і растрові дані одночасно. Метафайли широко застосовуються для перенесення даних на різні апаратні платформи і для обміну даними зображення між програмними платформами.

Метафайл може бути прочитаний будь-якою програмою, що підтримує растрові або векторні дані та підтримує даний метафайловий формат.

Наприклад, багато настільно-видавничих систем (НВС) можуть маніпулювати векторними даними і роздруковувати їх, але не в змозі відображати їх на екрані. Наприклад, Adobe PageMaker, Adobe InDesign, Microsoft Office Publisher, QuarkXPress, LaTeX та інші.

Щоб уникнути цього, в метафайл включають растрове подання того ж зображення. В цьому випадку програма може прочитати це растрове зображення, відтворити його на екрані і маніпулювати їм, а для виведення на друк використати векторні дані із того ж метафайла.

Відображення і друк файлів PostScript є прикладом такої організації даних.

Наряду з двійковими існують символьно-орієнтовані (ASCII) метафайлові формати, що значно полегшує перенос даних із однієї комп'ютерної системи в іншу. Метафайли в цілому містять заголовок, за яким йдуть один або кілька типів даних зображення (найпростіші метафайли дуже схожі на файли векторного типу). Вони поєднують переваги і недоліки растрових і векторних форматів. І в конкретному метафайлі потрібно орієнтуватися на переважаючий в ньому тип даних.

Питання для самоперевірки

1. Визначіть мету стиснення даних.
2. В чому різниця між алгоритмами без втрати інформації і з втратою інформації?
3. Назвати етапи конвейера стиснення JPEG.
4. На чому основана схема фрактального алгоритму стиснення?
5. Вкажіть основні характеристики алгоритму стиснення wavelet.
6. Приведіть схему стиснення і характеристики алгоритму RLE.
7. Приведіть схему стиснення і характеристики алгоритму LZW.
8. Приведіть схему стиснення і характеристики алгоритму Хаффмана и CCITT-3.
9. Приведіть схему стиснення і характеристики алгоритму JPEG.
10. В чому є принцип стиснення відео зображень?
11. Як видалити надмірність відео даних в MPEG-потоці?
12. Що таке кодеки?
13. Які типи графічних форматів ви знаєте?
14. Які елементи графічних файлів ви знаєте?
15. Які переваги та недоліки растрових форматів?
16. Описати особливості форматів BMP, TIFF, PSD.
17. Призначення і правила використання формату GIF, і PNG.
18. Описати особливості використання форматів TGA, ICO, FLM. 19 Які переваги та недоліки векторних форматів?
19. 20 Описати особливості використання форматів SVG, EPS, CDR. 21 Призначення і правила використання форматів PDF, CGM, DXF 22 Які переваги та недоліки метафайлів?