

## Лекція 1. Вступ до комп'ютерної графіки

Один рисунок вартий тисячі слів.

(китайська мудрість)

### 1.1. Предмет вивчення і галузі застосування комп'ютерної графіки.

Сучасна комп'ютерна графіка визначається багатьма напрямками і різноманітними застосуваннями. Для одних з них, наприклад, основою є автоматизація креслення технічної документації, для інших - проблеми оперативної взаємодії людини і комп'ютера. Сучасне розширення можливостей комп'ютерів, створених для виконання обчислень, дає можливість комп'ютеру сприймати і наочно зображувати результати обчислень, будувати необхідні комплексні креслення, схеми і т.п.. Зорове сприйняття графічної інформації для людини має важливе значення, обсяг і швидкість сприйняття зорових образів значні. Для представлення особливостей чи креслення будь-якого процесу досить декількох секунд, під час яких ми розглядаємо креслення, графік функції чи інше наочне зображення. Отже, важливість наочного представлення комп'ютером інформації важко переоцінити.

Комп'ютерна графіка (КГ) - це спеціальна галузь комп'ютерних дисциплін, що вивчає методи і засоби створення та обробки зображень за допомогою програмно-апаратних обчислювальних комплексів.

Комп'ютерна графіка – дуже обширна галузь комп'ютерних знань, яка пройшла розвиток як складна наукова дисципліна.

Сфера застосування КГ включає чотири основні області:

1. *Відображення інформації*
2. *Проектування*
3. *Моделювання (імітація)*
4. *Сфера розваг*
5. *Графічний інтерфейс користувача.*

Розглянемо основний зміст застосування КГ для зазначених областей.

*Відображення інформації.* Проблема подання накопиченої інформації найкраще може бути вирішена за допомогою графічного відображення.

Наприклад, дані про кліматичні зміни за тривалий період, динаміка популяцій тваринного світу, екологічний стан різних регіонів.

Сучасна наука використовує графічне представлення інформації

- візуалізація результатів експериментів;
- аналіз даних натурних спостережень;
- наочні картини результатів математичного моделювання процесів і явищ.

Наприклад, у геології в результаті обробки тривимірних натурних даних можна отримати геометрію пластів, що залягають на великій глибині (Рис.1); у медицині широко використовуються методи діагностики, що використовують комп'ютерну візуалізацію внутрішніх органів людини.

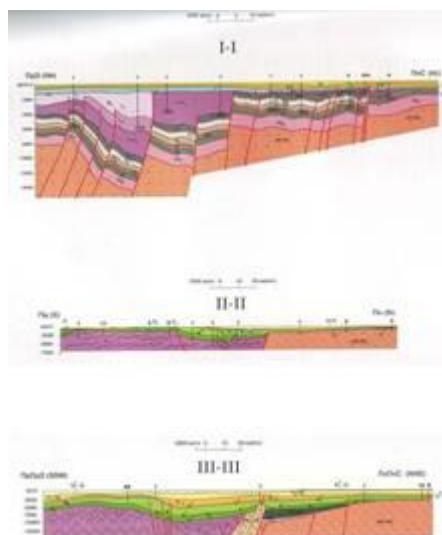


Рис. 1. Приклад візуалізації геологічних даних.

*Проектування (дизайн).* У будівництві та техніці креслення є основою проектування нових споруд або виробів.

Процес проектування є ітеративним, тобто конструктор перебирає безліч варіантів з метою вибору оптимального за певними параметрами.

Існують розвинені програмні засоби автоматизації проектно-конструкторських робіт (САПР), що дозволяють швидко створювати креслення об'єктів, виконувати розрахунки на міцність і т.п.

*Імітація.* Створення тренажерів у різних галузях використовує методи віртуальної реальності. Віртуальна реальність полягає у відображені тривимірного простору на двовимірних пристроях.



Рис.2. Візуалізація у тренажерах.

*Графічний інтерфейс користувача.* Розділ комп'ютерних дисциплін “Людино-машинна взаємодія” використовує методи та засоби комп'ютерної графіки для створення зручних інтерфейсів програмних систем.

Основні напрями КГ:

- 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, сприяюча пізнанню).

1) Образотворча комп'ютерна графіка своїм предметом має синтезовані зображення.

Основні види завдань, які вона вирішує, зводяться до таких:

- побудова моделі об'єкта й формування зображення;
- перетворення моделі й зображення;
- ідентифікація об'єкта й одержання необхідної інформації.

2) 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 й класифікації властивостей об'єктів (при обробці аерокосмічних знімків, уведені креслень, у системах навігації, виявлення і наведення).

3) 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.

4) Когнітивна комп'ютерна графіка тільки формується новий напрям, поки ще недостатньо чітко окреслений.

Це КГ для наукових абстракцій, що сприяє народженню нового наукового знання. Технічною основою для неї є потужні комп'ютери і високопродуктивні засоби візуалізації.

Одним з найбільш ранніх прикладів використання когнітивної комп'ютерної графіки є робота Ч.Страуса "Несподіване застосування ЕОМ в чистій математиці" (ТІІЕР, т. 62, № 4, 1974, с.96-99). У ній показано, як для аналізу складних алгебраїчних кривих використовується "п-мірна" дошка на основі графічного терміналу. Користуючись пристроями введення, математик може легко управляти поточними значеннями параметрів, "поглиблюючи тим самим своє розуміння ролі варіацій цих параметрів". В результаті отримано "кілька нових теорем і визначені напрямки подальших досліджень".

Комп'ютерна графіка застосовується для візуалізації даних у різних сферах людської діяльності: наука, індустрія розваг, медицина, будівництво, видавнича справа, авіація тощо.

Структура та методи комп'ютерної графіки засновані на досягненнях фундаментальних та прикладних наук: математики, фізики, хімії, біології, статистики, програмування тощо. Це стосується, як програмних, так і апаратних засобів створення та обробки зображень. Тому комп'ютерна графіка є однією з найважливіших ділянок інформатики та стимулює розвиток комп'ютерної індустрії.

## **1.2. Коротка історія розвитку комп'ютерної графіки.**

Перша офіційно визнана спроба використання дисплея для виводу зображення з ЕОМ - Массачусетський технологічний університет (МТІ),

машина Whirlwind-I в 1950 р. А сам термін "комп'ютерна графіка" вперше використав в 1960 р. співробітник компанії Boeing У. Феттер.

На початку свого розвитку комп'ютерну графіку розглядали, як частину системного програмування для ЕОМ чи один з розділів систем автоматизованого проектування (САПР).

Перше реальне застосування КГ пов'язують з ім'ям Дж. Уїтні. Він займався кіновиробництвом в 50-60-х роках і вперше використовував комп'ютер для створення титрів до кінофільму.

Наступний крок у розвитку КГ- робота Айвена Сазерленда, який в 1961р створив програму малювання, названу ним Sketchpad (альбом для малювання). Програма використовувала світлове перо для малювання найпростіших фігур на екрані. Отримані картинки можна було зберігати і відновлювати. У цій програмі було розширене коло основних графічних примітивів, зокрема, крім ліній і точок був введений прямокутник, який задавався своїми розмірами і розташуванням.

Спочатку комп'ютерна графіка була векторною, тобто зображення формувалося з тонких ліній. Ця особливість була пов'язана з технічною реалізацією комп'ютерних дисплеїв.

Перша комп'ютерна відеогра - Spacewar ("Зоряна війна") у 1961 р. створена студентом С. Расселом.

У зв'язку з успіхом КГ, великі корпорації зацікавлюються розробками КГ і починають комерційно підтримувати розвиток КГ. "Дженерал моторз", "Дженерал електрик", приступили до комерційних розробок в області КГ.

Це було використання графічної системи з розподілом часу для багатьох етапів проектування автомобілів.

Центр досліджень в області КГ - університет штату Юта (завдяки Д.Евансу і А.Сазерленду).

Учнем Сазерленда став Е.Кетмул, майбутній творець алгоритму видалення невидимих поверхонь з використанням Z-буфера (1978). Тут же працювали Дж.Варнок, автор алгоритму видалення невидимих граней на основі розбивки

області (1969) та засновник Adobe System (1982), Дж.Кларка, майбутній засновник компанії Silicon Graphics (1982).

Всі ці дослідники дуже сильно розвинули алгоритмічну сторону комп'ютерної графіки.

Щодо розвитку КГ в Україні, то слід зазначити, що перша інтерактивна графічна система автоматизованого проектування електронних схем була розроблена Київським політехнічним інститутом і Науково-дослідним інститутом автоматизованих систем керування і плануванні в будівництві (НДІАСК, Київ) і демонструвалася в 1969р.

Справжній широкий розвиток КГ пов'язаний з появою персональних комп'ютерів «Macintosh» (MAC) фірми Apple (спеціально для потреб поліграфії).



Рис.3. ПК «Macintosh».

Саме для платформи MAC почали з'являтися перші спеціалізовані операційні системи та графічні редактори. Але сталося так, що справжніми «масовими» комп'ютерами стали комп'ютери класу IBM/PC (PC). А більшість графічних оболонок та редакторів почали відтворюватися на базі графічного досвіду MAC та перекладені для комп'ютерів PC. Так з'явилася славнозвісна операційна система Windows, а також дуже велика кількість графічних пакетів, програм та редакторів (наприклад: QuickTime, Page Maker, майже всі продукти корпорації Adobe та багато інших).

У середині 1970-х років КГ продовжує розвиватися в бік все більшої реалістичності зображень. Е.Кетмул, Фонг, Ф.Кроу, Дж.Брезенхем створюють ефективні растрові алгоритми.

У 1980-ті роки з'являється цілий ряд компаній, що займаються прикладними розробками в області КГ. Silicon Graphics, Adobe System...

У 1982 році на екрані кінотеатрів вийшов фільм “Трон”, в якому вперше використовувалися кадри, синтезовані на комп'ютері (Рис.4).



Рис. 4. Кадр з фільму "Трон".

Ще в 1979 році Джордж Лукас, глава фірми “Lucasfilm” і творець серіалу “Зоряні війни”, організував в своїй фірмі відділ, який займався впровадженням останніх досягнень КГ в кіновиробництво.

У 1990-ті роки КГ застосовується масово в кіноіндустрії, мережі Інтернет...

Підводячи підсумки, виділяють окремі етапи розвитку КГ.

*Перший етап* - 1960-1970-і роки, коли КГ формувалася як наукова дисципліна. У цей час розроблялися основні методи і алгоритми: відсікання, растроva розгортка графічних примітивів, зафарбування візерунками, реалістичне зображення просторових сцен (видалення невидимих ліній і граней, трасування променів, випромінюють поверхні), моделювання освітленості.

*Другий етап* - 1980-ті, коли КГ розвивається більш як прикладна дисципліна. Розробляються методи її застосування в самих різних областях людської діяльності.

*Третій етап* – з 1990-тих років, коли методи комп'ютерної графіки стають основним засобом організації діалогу “людина-комп'ютер” і залишаються такими по теперішній час.

### 1.3. Дисплейні технології підтримки комп'ютерної графіки.

Розвиток КГ обумовлений розвитком технічних засобів її підтримки. Перш за все це пристрой виведення, якими є дисплеї. Дисплеї є найпоширенішим типом пристройв виведення графічної інформації. Вони застосовуються скрізь - від мобільних телефонів до рекламних щитів.

За принципом відображення усі дисплеї поділяються на векторні та растрові. З експерименту в МТІ почався етап розвитку векторних дисплеїв, а саме дисплеїв з довільним скануванням променя. Векторний дисплей відображає зображення з допомогою примітивів – векторів (ліній).

Відмінна риса векторних дисплеїв є можливість безпосереднього графічного діалогу, що полягає в простій вказівці за допомогою світлового пера об'єктів на екрані (ліній, символів і т.д.).

Дисплеї з растровим скануванням променя з середини 1970-х років одержують переважне поширення.

Растрове пристрій можна розглядати як матрицю дискретних точок, кожна з яких може бути підсвічена. Відрізок будується за допомогою послідовності точок, апроксимуючих ідеальну траєкторію відрізка. Відрізки можуть виглядати як послідовність "сходинок" (ступінчастий ефект). Растрові технології відображення є значно дешевими, ніж векторні.

У даний час існують декілька типів дисплеїв, що різняться технологіями виготовлення: дисплеї на електронно-променевій трубці (МИНУЛЕ), дисплеї на рідкокристалічних індикаторах (ТЕПЕРІШНЕ) і інші їх види (МАЙБУТНЕ).

Дисплеї на електронно-променевій трубці мають такі недоліки – мерехтіння зображення, вага та габарити. Перевагою залишається низька їх вартість, проте з часом ця перевага все менше стає істотною у порівнянні з наступним типом дисплеїв.

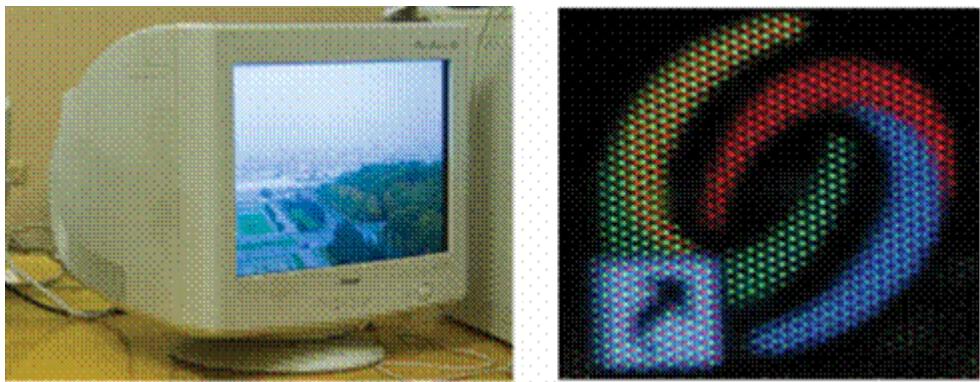


Рис. 5. Дисплей на електронно-променевій трубці та вигляд зображення

Дисплеї на рідкоクリсталічних індикаторах є растроюми пристроями (іх теж можна представити як матрицю елементів - рідких кристалів). LCD-монітори – домінуючі сьогодні. Мають найменші габарити і енергоспоживання, тому широко використовуються в портативних комп'ютерах.

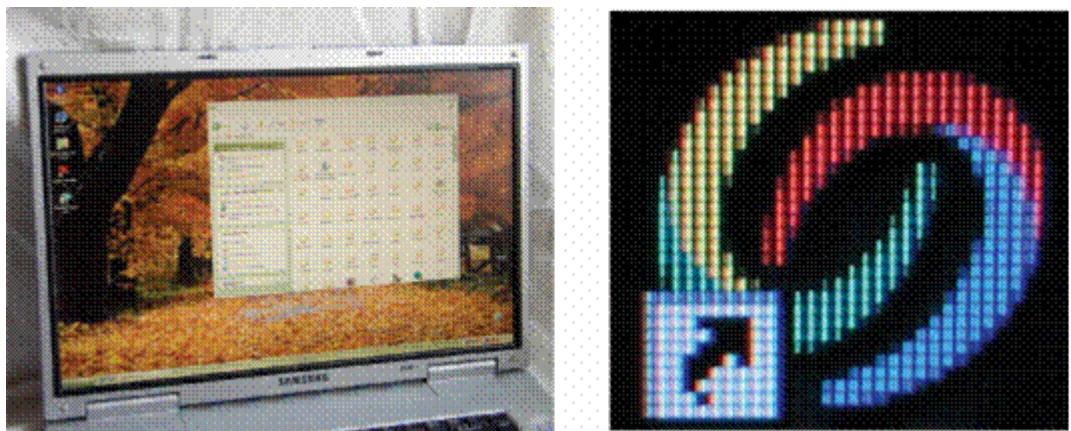


Рис.6. Дисплей на рідкоクリсталічних індикаторах та вигляд зображення.

Хоча історично такий спосіб виводу зображення з'явився раніше, ніж растрний дисплей з ЕПТ, але швидко розвиватися він почав значно пізніше.

Інші типи дисплеїв - це плазменні екрани, дисплеї на світлодіодах, електронний папір.

Основними параметрами монітора є розмір екрану, розмір зерна, швидкість оновлення зображень (частота кадрової розгортки), ступінь плоскості екрану (вища реалістичність зображення, менше відблисків).

Для професійної роботи з графікою використовують монітори з діагоналями 21 чи 22 дюйми.

Зерно – це мінімальна точка люмінофору (піксель), яка вимірюється в десятих частках міліметра.

Близьким параметром до розміру зерна є роздільна здатність. Показує, скільки пікселів може вміститися на екрані. Роздільну здатність описують два числа, кількість точок по горизонталі, кількість горизонтальних ліній, наприклад  $320 \times 200$ .

Іншою характеристикою моніторів є **частота вертикальної розгортки**, тобто частота оновлення кадрів, яку ще часто називають частотою регенерації. Для комфортної роботи необхідно, щоб частота верти кальної розгортки складала не менше 85 Гц.

Менша частота шкідлива для очей (миготіння на екрані швидко стомлює очі). Якщо частота вертикальної розгортки вища за 110 Гц, то людина вже не помічає ніякого мерехтіння зображення.

**Горизонтальна частота розгортки** показує кількість ліній, яку можна вивести на екран за 1 с. Для сучасних моніторів вона складає від 15 кГц до 100 кГц.

Параметри моніторів пов'язані між собою, наприк лад, якщо зменшити роздільну здатність, то зростає частота розгортки. Істотними характеристиками для роботи з моніторами є **ергономічність**.

#### **1.4. Елементи графічної підсистеми комп'ютерної графіки**

Відеоплата (відеокарта) – плата з мікросхемами, що забезпечують вивід зображення на екран. Кадровий буфер – частина основної пам'яті, в якій зберігається зображення. Відеопам'ять на відеокарті є двопортова – під'єднана як до шини, по якій передаються дані від центрального процесора, так і до мікросхеми, яка відповідає за вивід на дисплей, і вони можуть одночасно звертатися до неї.

Для підключення дисплеїв до комп'ютерів використовуються стандарти, що встановлюють логічні і фізичні параметри з'єднання. У даний час два найпоширеніших з них - це аналоговий VGA і цифровий DVI.

Графічний процесор, здатний швидко виконувати основні операції із зображеннями у відеопам'яті:

- *Робота з прямокутними блоками.* Мікросхема, яка забезпечує виконання такої операції, називається бліттер, тому що її основна операція - це BitBlt (Bit Block Transfer), тобто копіювання прямокутного блоку зображення в інше зображення з можливим застосуванням побітових логічних операцій (І, АБО, виключає АБО).
- *Растеризація примітивів* (відрізки, окружності, еліпси, прямокутники, багатокутники.) Також може підтримуватися заливання одноколірних зон іншим кольором або за шаблоном.
- *Підтримка виводу символів певним шрифтом.* Іноді шрифт можна завантажувати з основної пам'яті. У даний час цей режим використовується при завантаженні ПК, а також при роботі в режимі терміналу (частіше використовується в ОС Linux).
- Апаратне прискорення відео і фільтрація зображення. Кодування і декодування відео - дуже ресурсномістка операція, (обробка великих обсягів даних). Деякі відеокарти здатні апаратно декодувати відеопотік. У сучасних відеокартах також є підтримка телебачення високої чіткості (англ. HDTV - High Definition Television). Апаратне масштабування відео. Апаратне відображення відео в частині екрана носить назву оверлея (англ. overlay). Деякі відеокарти також можуть апаратно робити фільтрацію зображень.

Сучасні відеокарти містять також процесор опрацювання тривимірної графіки.

### **Програмний інтерфейс КГ**

Найперші ПК, що з'явилися на початку 1980-х років, працювали винятково в текстовому режимі. Потім з'явилися карти з графічними можливостями. У зв'язку з малим розміром адресного простору (1 Мб) процесора Intel 8086 доводилося відображати тільки частину відеопам'яті в адресний простір процесора і спеціальними командами задавати, яка саме це частина.

Команди відеокарті посилалися шляхом переривань або запису інформації безпосередньо в її апаратний порт. Для використання додаткових можливостей

відеокарт розробникам прикладних програм та ігор доводилося самим реалізовувати найпростіші операції для кожного їх типу.

Операційні системи з графічним інтерфейсом змінили механізм реалізації зображень на комп'ютері. Між прикладною програмою і апаратурою сформовано "прошарок", який дозволяє вирішити питання сумісності .

Безпосередньо на низькому рівні відеокартою управляє її драйвер - програма (фірма-розробник відеокарти), а прикладна програма звертається до нього через виклики чітко визначеного загального для всіх драйверів абстрактного інтерфейсу (англ. API - Application Programming Interface). Таким чином, з'явилася апаратна незалежність, що стало важливим кроком вперед, з урахуванням все більш зростаючої кількості відеокарт з обмеженою сумісністю.

В UNIX-подібних ОС графічний інтерфейс надається системою X Windows, що працює за принципом "клієнт-сервер". Програма-клієнт відправляє запит API по мережі (хоча для самої програми це виглядає як просто виклик функції); отримавши цей запит, програма-сервер відповідає за його виконання. Хоча така схема і є гнучкою (можна, наприклад, мати кілька дисплеїв в одного комп'ютера або, навпаки, багато комп'ютерів, підключених до одного дисплею), але в той же час вона вимагає і додаткових витрат на передачу даних по мережному протоколу.

## ЛЕКЦІЯ 2. Застосування КГ у задачах побудови GUI

На сьогоднішній день, комп'ютерна графіка стрімко розвивається в багатьох напрямках, та має широке застосування. З одного боку, слугує для автоматизації креслення технічної документації, з іншого - сприяє вирішенню проблем оперативної взаємодії людини і машини.

Починаючи з 90х років, методи комп'ютерної графіки стають основним засобом організації діалогу між користувачем та комп'ютером. Однією з основних точок доступу, де юзери взаємодіють з дизайном інтерфейсу - є графічний інтерфейс користувача (GUI). Якщо порівняти GUI зі звичайною командним рядком, то в першому варіанті перед користувачем відкривається повний доступ до всіх елементів, який він бачить на дисплеї. Реалізувати цей доступ можна з використанням різних пристрійв введення: миші, трекбола, клавіатури, джойстика і ін.

Зазвичай в GUI кожен графічний об'єкт передає сенс функції за допомогою зрозумілого образу, щоб користувачеві було простіше розібратися з певним програмним забезпеченням і легше взаємодіяти з ОС в цілому. Але важливо розуміти, що GUI - це лише складова частина графічного інтерфейсу. Функціонує він на рівні візуалізації даних і таким же чином взаємодіє з користувачем.

Взаємодія відбувається з візуальними зображеннями на цифрових панелях управління, наприклад, робочий стіл комп'ютера - це є графічний інтерфейс.

Головним правилом GUI є - “Направляти користувача”, це допоможе підвищити продуктивність взаємодії. Забезпечити це можна застосовуючи наступні евристики Нільсена, для створення зручних інтерфейсів.

- Сповіщення про стан системи (зворотній зв'язок, індикатор прогресу, Message Box повідомлення)
- Схожість з реальним світом
- Контроль та свобода дій
- Послідовність та стандартизація
- Попередження помилок (створення підказок Hint)
- Впізнаваність замість згадування
- Гнучкість та ефективність використання
- Естетика та мінімалізм
- Розпізнавання та вирішення помилок
- Довідка та документація

Користувач повинен керувати всім, що відбувається на екрані. Завжди мати можливість втрутитись в автоматичний процес виконання програми, перервати його чи перезапустити. А додатки в свою чергу повинні завжди повідомляти користувача, що відбувається у системі.

Суттєвим моментом в інтерфейсі GUI є використання метафор “Документ”, “Ключ”, “Корзина”, це певне ототожнення з реальним світом що допомагає впізнавати а не вивчати.

Допомога користувачу забезпечується наявністю підказок, довідкової інформації, та діалогових вікон які повідомляють користувачу про можливі наслідки виконуваних дій.

Простота інтерфейсу полягає в його лаконічності. Форми не повинні бути переобтяжені зайвими елементами та написами. Також слід передбачити зникнення елементів інтерфейсу в тому разі коли потреба в них відпала.

Екранні форми повинні бути виконані акуратно з акуратним розташуванням елементів на екрані, створюючи правильні акценти які спонукатимуть користувача до дій.

Всі форми інтерфейсу повинні бути витримані в одному стилі і своїм оформленням не відволікати від основної роботи зі системою.

В минулому користувацькі інтерфейси розвивалися лише шляхом еволюції технологій та систем, на базі яких розроблялось ПЗ. Такий підхід називають системно-керованою або технологічно-керованою розробкою. Побажання користувачів абсолютно не враховувались, і їм відповідно надавались програмні функції з інтерфейсом, який розробники могли розробити своїми силами.

Однією з головних причин створення невдалого програмного забезпечення є недостатнє залучення потенційних користувачів до проекту. Не менше значення мають і наслідки недостатньо активної участі користувачів, зокрема відсутність знань про фактичних користувачів продукту та середовище (ринок) використання розробленого ПЗ. Детальна інформація про користувачів та їх середовище допомагає встановити рамки, в яких повинно здійснюватись проектування інтерфейсу та забезпечуватись його практичність.

Ефективність інтерфейсу обумовлена швидким засвоєнням користувачем простої моделі взаємодії з комп'ютером. Цього можна досягти через узгодженість, існує три аспекти узгодженості:

- 1) Фізична узгодженість - відноситься до узгодженості технічних засобів.
- 2) Синтаксична узгодженість - забезпечується певною послідовністю і порядком розташування елементів на екрані та послідовністю дій користувача.
- 3) Семантична узгодженість - визначена значеннями елементів, що утворюють інтерфейс.

*Ось декілька порад, які допоможуть створити візуально правильний UI:*

- 1) Зробіть кнопки та інші поширені елементи інтерфейсу передбачуваними (включаючи адаптивність та масштабування), щоб користувачі могли їх несвідомо використовувати скрізь. Форма повинна відповідати функції.
- 2) Робіть правильні акценти та ієрархію. Слід чітко розуміти на чому повинен зосередитись користувач, який основний фокус його уваги і куди зміститься даний фокус при певній дії.
- 3) Всі елементи повинні бути вирівняні, пам'ятайте про сітку.
- 4) Зверніть увагу на ключові функції, використовуючи:

Колір, яскравість і контраст. Уникайте використання широкого спектру кольорів, пропрацюйте свою колірну гамму, адже колір тільки посилює дизайн а не формує його.

- 5) Створіть ієрархію шрифтів, які ви будете використовувати. Текст подається через розмір шрифту, його вагу, динаміку, розмір літер та відстань між ними. Користувачі повинні знаходити потрібну інформацію просто скануючи, без зайвих зусиль.

- 6) Мінімізуйте кількість дій для виконання завдань, зосередьтеся на головній функції на сторінці. Направляйте користувачів, вказуючи бажані дії. Полегшувати складні завдання, використовуючи прогресивне розкриття інформації.
- 7) Розташуйте елементи керування поблизу об'єктів, якими користувачі хочуть керувати. Наприклад, кнопка для подання форми повинна знаходитись поруч із формою.
- 8) Тримайте діалог між користувачем та системою. Інформуйте користувачів про відповіді чи дії системи, підтримуйте зворотній зв'язок.
- 9) Допомагайте користувачу якщо є така можливість. Наприклад, користувачу потрібно заповнити форму, ви можете додати автозаповнення вже наявною інформацією.
- 10) Завжди надавайте наступні кроки, які користувачі можуть зробити природним чином, незалежно від їхнього контексту.

На сьогоднішній день практично всі системи побудовані на певних патернах, і мають схожу структуру, відповідно принести унікальність в ту чи іншу систему ми можемо завдяки кастомізованому і і мікро-інтеракціям які впливають на відчуття під час роботи з тим чи іншим продуктом.

Мікро-інтеракції - є деталізованим підходом, що робить досвід користувача більш персоналізованим. Призначений для підвищення зручності користувачів за допомогою інноваційних рішень, що спрощують цифрову взаємодію.

Основними аспектами в роботі з візуальним дизайном є:

**Композиція** - допомагає контролювати та концентрувати увагу користувачів. Користувачі витрачатимуть менше часу на пошук та засвоєння інформації.

**Якірні об'єкти** - це найпомітніші об'єкти на сторінці: ілюстрації, заголовки, логотипи та піктограми. Простий абзац набірного тексту теж виступає якірним об'єктом, якщо він оточений мінус-простором. Правило говорить, що будь-який якірний об'єкт повинен бути вписаний в візуальний прямокутник, і тяжіти або розташовуватися в одному з кутів чи в візуальному центрі свого прямокутника.

**Колір** - кожен колір має візуальну вагу, і використовується, щоб встановити ієрархію контенту. У UI найяскравіший колір часто використовується для елементів, що взаємодіють з користувачем.

Правило полягає в тому, що, якщо один елемент є важливішим за інший, то він повинен мати більшу візуальну вагу. Це дозволяє користувачеві швидко переглядати сторінки і розрізняти важливу і другорядну інформацію.

**Розмір** - чим більший елемент, тим більше уваги йде до нього. Ідея ієрархії за допомогою розміру полягає в намірі дати фокус. Погляд чіпляється за великі об'єкти, але необхідно пам'ятати, що важливі елементи дизайну не повинні бути занадто великими, тому що таким чином може виникнути небажаний дисбаланс.

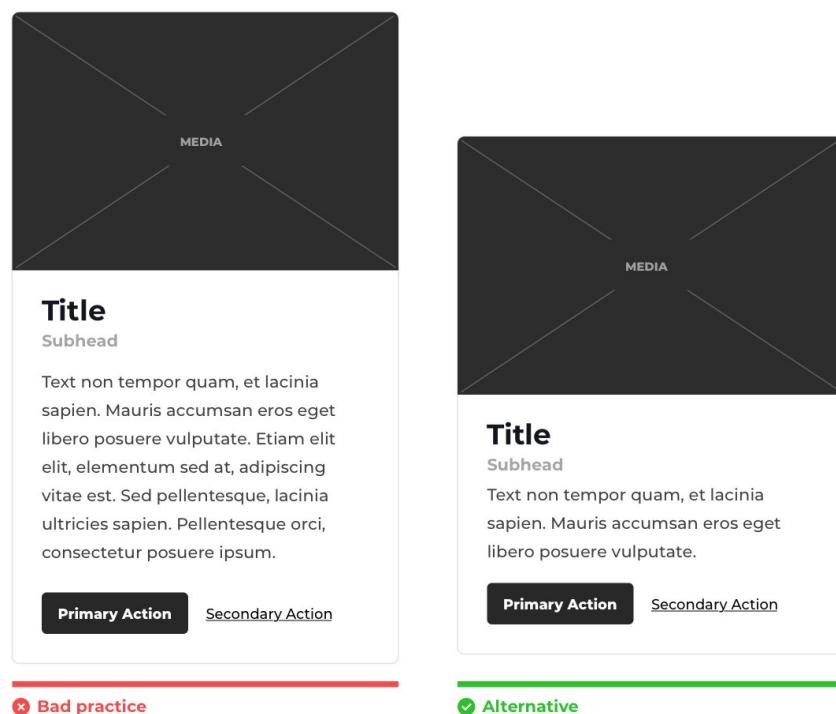
**Негативний простір (white space)** - або мінус-простір це порожній простір на інтерфейсі, це простір між елементами вашої композиції. Цей простір залишається не використаним, що робить макет більш читабельним і

виразним. Користувач може зосередити увагу на краще представлених елементах.

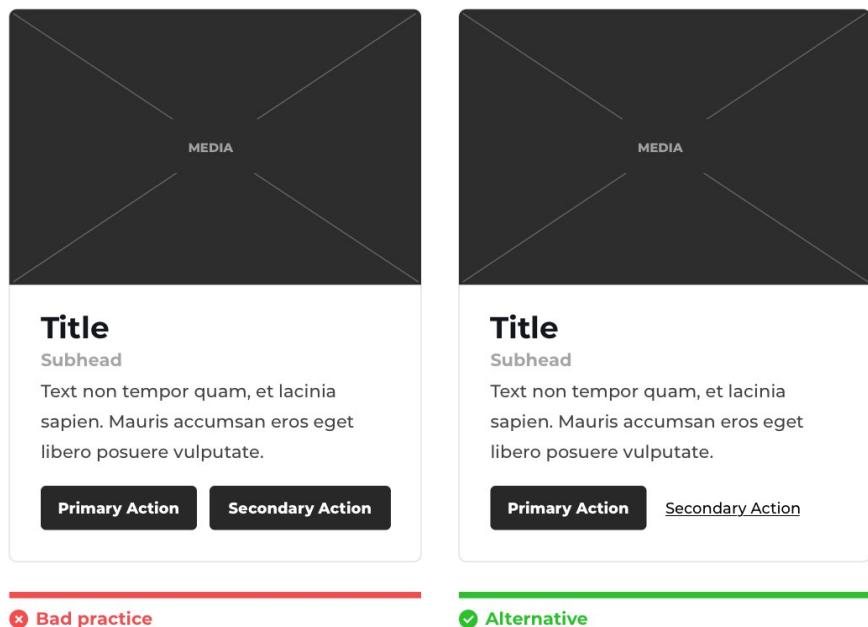
**Типографія** - Ваш шрифт повинен підтримувати процес прийняття рішення, яке визначає зміст таким чином, щоб не створювати додаткове навантаження на читача. Вдала типографія фокусує увагу читача саме на змісті, а не на шрифті, як такому.

Розглянемо приклади створення технічно правильних карток в інтерфейсі:

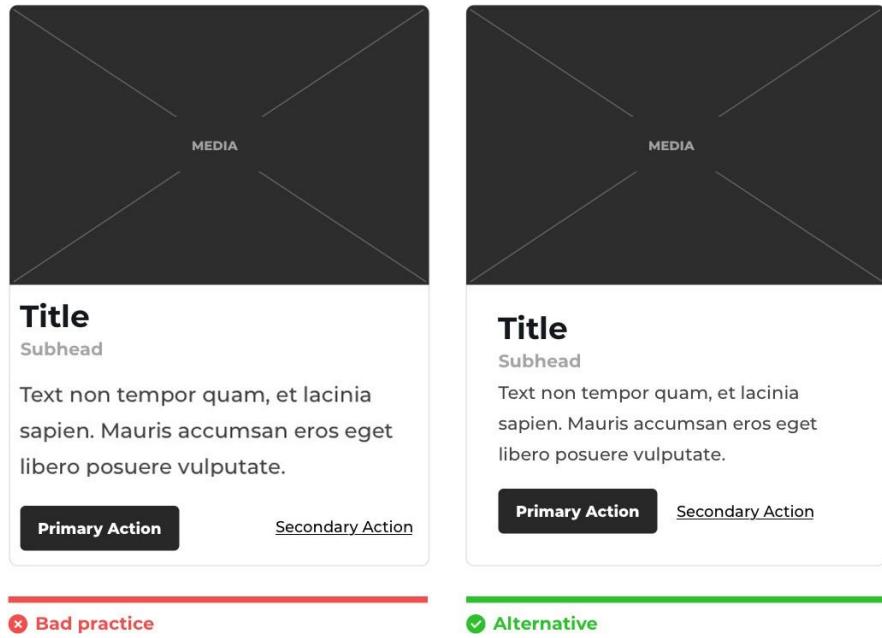
1. *Обмежити вміст* - обмежте контент до 100 символів або не більше двох коротких речень. Карточка повинна містити достатньо інформації, щоб допомогти користувачеві визначити, чи слід продовжувати взаємодію, та не відволікати від зчитування інформації з інших карток які можуть розташовуватися поруч.



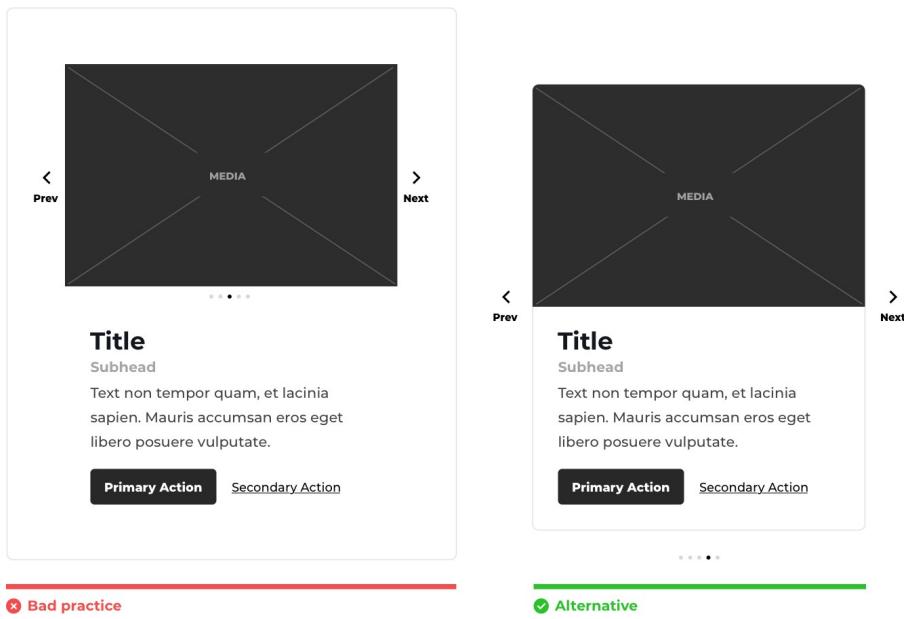
2. *Організуйте контент* - вміст картки поділяється на смислові блоки - за рахунок чого економиться екранний простір. Схоже до того, як текст групується в параграфи і формує смислові блоки, різні елементи, зібрані на картці, створюють цілісну одиницю контенту.
3. *Розмежуйте дії* - картки які дають можливість виконати декілька дій, повинні бути візуально диференційовані. У наведеному прикладі другорядна дія є менш помітною, і має стиль посилання замість стилю кнопки.



4. *Слідкуйте за простором* - потрібно щоб контент дихав, не потрібно нехтувати негативним простором та відступами в середині карток. Не створюйте хаотичних відступів, користуйтесь правилами сіток.

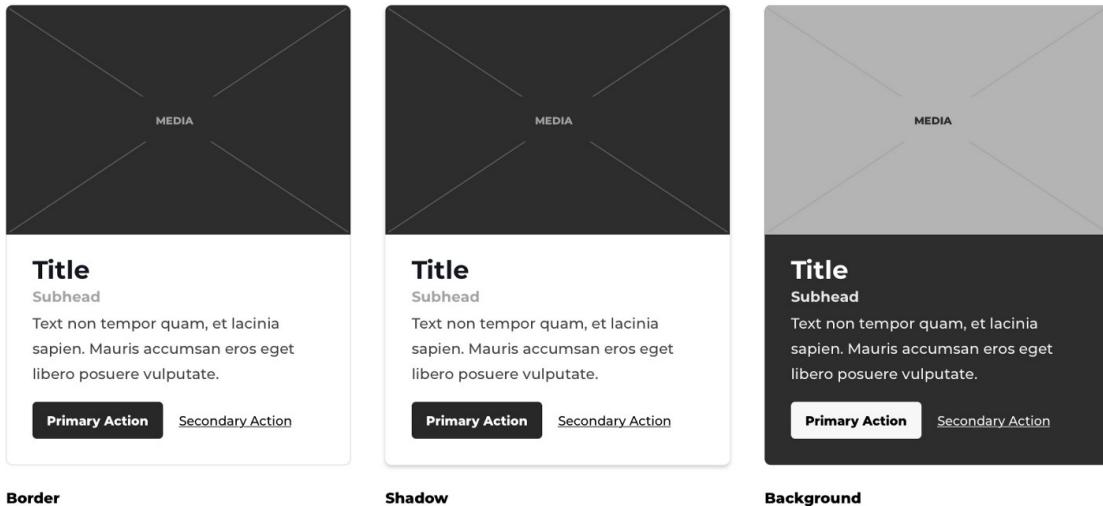


5. Одна ідея на картку - не нагромаджуйте одну картку великою кількістю функціональних ідей, часто це може створювати додатковий стрес для користувача.



6. Відокремте карту від фону - за допомогою обведення, тіні або кольору.

Це допоможе створити додатковий контраст та фокус на вмісті картки.



Будь-який процес розробки інтерфейсів повинен бути ітераційним, оскільки вдалий інтерфейс неможливо одержати без періодичного повернення до попередніх етапів. Критерієм для завершення ітераційної розробки є той факт, що усі вимоги користувачів задоволені, а сам продукт відповідає запланованим цілям.

## ЛЕКЦІЯ 3. ВЕКТОРНА ГРАФІКА

Моделлю зображень будемо вважати його опис за певними правилами, враховуючи деякі припущення. Спосіб має бути зручним для комп'ютерного представлення.

У залежності від способу формування зображень КГ поділяють на:

- растрову;
- векторну;
- **фрактальну;** \*
- **тривимірну.** \*

Кожен з цих способів є комп'ютерною моделлю зображень, яка має свої області застосування і, відповідно, переваги і недоліки.

Розглядаємо векторну графіку. Історично це перший спосіб формування комп'ютерного зображення.

### 3.1. Спосіб формування зображень у векторній графіці

Зображення у векторній КГ описується з допомогою базових примітивів. Базовий примітив – команда пристрою побудувати певну лінію (вектор). Вектор – це фрагмент прямої чи кривої лінії, замкнений або ні. Вектор – звичайний відрізок, коло/еліпс, прямокутник, багатокутник, крива... Примітиви визначаються типом пристрою відображення.

Лінія є елементарним об'єктом, якому притаманні певні особливі властивості: форма, товщина, колір, стиль тощо.

Будь-який об'єкт (прямокутник, еліпс, текст і навіть пряма лінія) сприймається як криві лінії. Це все приклади векторів.

Векторні зображення є незамінні там, де принципове значення має збереження чітких контурів, а саме:

- повноколірні ілюстрації;

- складні креслення;
- логотипи та емблеми;
- графічні зображення для Web;
- мультиплікація;
- рисунки на основі оригіналів.

Векторну графіку називають об'єктно-орієнтованою. Термін "об'єктно-орієнтована" слід розуміти в тому сенсі, що всі операції, що виконуються в процесі створення і зміни зображень, користувач проводить не із зображенням у цілому і не з його найдрібнішими, атомарними частинками (пікселями точкового зображення), а з об'єктами – семантично навантаженими елементами зображення. Починаючи із стандартних об'єктів (кругів, прямокутників, текстів і т. д.), користувач може будувати складені об'єкти і маніпулювати з ними як з єдиним цілим. Таким чином, зображення стає ієрархічною структурою, на самому верху якої знаходиться ілюстрація в цілому, а в самому низу - стандартні об'єкти.

Основними конструктивами векторного об'єкту є

- *Шлях* – це маршрут, що з'єднує початкову та кінцеву точку.
- *Сегмент* - окрема частина шляху, може бути як прямою, так і кривою лінією.
- *Вузол* - початкова або кінцева точка сегмента.

Кожен елемент векторної графіки містить ці три основні елементи і дозволяє їх редагування. Векторні об'єкти завжди мають шлях, що визначає їх форму. Всі шляхи містять дві компоненти: сегменти та вузли. У основі моделі ліній у векторній графіці лежать два поняття: вузол і сегмент.

Вузлом - це точка на площині зображення, що фіксує положення одного з кінців сегменту. Сегментом називається частина лінії, що сполучає два суміжні вузли.

Якщо шлях є замкненим, тобто кінцева точка співпадає з початковою, об'єкт

має внутрішню ділянку, яка може бути заповненою кольором або іншими об'єктами.

Заповнення замкненої області можна виконати одним з 4 способів:

- *однорідне заповнення* одним кольором або штрихуванням;
- *градієнтне*, при якому кольори або тіні поступово змінюються (лінійна, радіальна, конічна, прямокутна закономірність тощо);
- *візерункове*, при якому об'єкт заповнюється повторювальними зображеннями (двохколірними або повноколірними);
- *текстурне* заповнення (художні зображення). Призначення такого заповнення відобразити реалістично поверхні об'єктів. Для представлення текстури використовуються растрові зображення.

Математичні основи векторної графіки складають точка (вузол), відрізок (сегмент) прямої лінії, сегменти кривих другого та третього порядків.

- Точка. Об'єкт на площині представляється двома числами (x, y) відносно початку координат.
- Пряма лінія. Їй відповідає рівняння  $y=kx+b$ . Вказавши параметри k та b можна створити пряму лінію у відомій системі координат.
- Сегмент прямої. Для опису потрібно додатково вказати параметри x1 та x2, відповідно початок та кінець відрізку.
- Крива лінія II порядку. До них належать еліпси, круги, параболи, гіперболи тощо. Пряма лінія є також випадком кривої II порядку. Крива II порядку описується рівнянням  $a_0x^2+a_1y^2+a_2xy+a_3x+a_4y+a_5=0$ . Для побудови відрізка кривої додатково потрібні ще два параметри початку та кінця відрізку.
- Крива лінія III порядку. Важлива наявність точки перегину, що дозволяє відобразити різноманітні об'єкти. Рівняння кривої III порядку  $a_0x^3+a_1y^3+a_2x_2y+a_3xy^2+a_4x^2+a_5y^2+a_6xy+a_7x+a_8y+a_9=0$ . Для опису відрізка

потрібні ще два параметри початку та кінця відрізку. Зауважимо, що пряма та криві II порядку є частковим випадком кривих III порядку.

- Криві Без'є. Параметричний вид кривих III порядку. Метод побудови кривих Без'є заснований на використанні пари дотичних, що проведені до лінії в крайніх точках. На форму кривої лінії впливає кут нахилу дотичних та довжина її відрізка. Таким чином, дотичні відіграють роль віртуальних важелів, за допомогою яких керують формою кривої.

### 3.2. Переваги та недоліки векторної графіки

Переваги:

- невеликі за розміром файли, оскільки зберігається не зображення, а лише його основні дані, використовуючи які, програма відновлює зображення;
- об'єкти легко трансформуються, ними легко маніпулювати. Редагуючи векторний об'єкт, можна змінити властивості ліній, з яких складається зображення. Можна пересувати об'єкт, змінювати його розміри, форму та колір, не впливаючи на якість зображення;
- векторна графіка не залежить від роздільності, тобто векторні об'єкти відтворюють на пристроях з різною роздільністю без втрати якості зображення.

Недоліки:

- Низька фотoreалістичність
- Обмежена область застосування
- Низька швидкість промальовування на растрових пристроях

### 3.3. Популярні програми та формати векторної графіки

Прикладні програми векторної графіки призначені для технічних креслень, зображень ділового характеру, рекламних ілюстрацій.

Найпопулярнішими прикладними програмами є продукти фірм

- Corel - CorelDraw,
- Adobe - Illustrator,
- Macromedia - FreeHand,
- стандартний додаток у MS Office - Word Editor.

Серед векторних форматів, на відміну від растрових, ідея розумної стандартизації проявляється значно слабше.

Розробники практично всіх векторних графічних програм хочуть мати справу тільки зі своїми власними форматами, що пов'язано, швидше за все, зі специфікою алгоритмів формування векторного зображення.

Але, так як можливість перенесення файлів між різними додатками в векторній графіці не менш актуальна, ніж у растрової, тоного роду стандартом стали файлові формати двох найбільш популярних професійних графічних пакетів.

Досить суперечливим є формат CDR, основний робочий формат популярного пакета CorelDRAW, що є незаперечним лідером у класі векторних графічних редакторів на платформі PC.

Маючи порівняно невисоку стійкість і проблеми із сумісністю файлів різних версій формату, тим не менш формат CDR можна назвати професійним.

EPS – використовується у видавничих системах, поєднує растрову і векторну графіку. Надійний та універсальний формат, “рідна” програма – Adobe Illustrator. Серед професійних форматів Adobe - AI, PDF.

WMF - рідний векторний формат Windows. Сприймається практично усіма програмами Windows, так чи інакше пов'язаними з векторною графікою. Однак, незважаючи на простоту і універсальність, користуватися форматом WMF варто тільки в крайніх випадках, оскільки він не може зберігати деякі параметри, які

можуть бути присвоєні об'єктам в різних векторних редакторів, не сприймається Macintoshами, і, найголовніше, спотворює колірну схему зображення.

# Комп'ютерна графіка

[Інформаційна панель](#) / [Мої курси](#) / [Комп'ютерна графіка](#) / [Розділ 2. Лекція 4](#) / [Лекція 4](#)

## Лекція 4

### Лекція 4. Крива Без'є - основний інструмент векторної графіки

Одним із базових розділів комп'ютерної графіки є розділ, який досліджує зображення математичних кривих на площині і в просторі. Існує безліч способів побудови кривих вручну за допомогою олівця, ручки, пензлика та різноманітних інструментів: лінійки, лекала, циркуля, шаблону і т.д. Кожен інструмент відповідає своїй певній меті, але серед них немає жодного універсального. Так і в комп'ютерній графіці криві будуються за допомогою різних методів та інструментів.

Під час конструювання математичних моделей кривих ліній найчастіше зустрічаються такі задачі:

- апроксимації;
- інтерполяції;
- згладжування.

Задача *апроксимації* (наближеного представлення) виникає під час заміни кривої, що описується рівнянням функцій складної природи (наприклад з точки зору швидкодії обчислень її значень і похідних, інтегрування, диференціювання), іншою кривою, в деякому наближенні близької до заданої, рівняння якої більш просте.

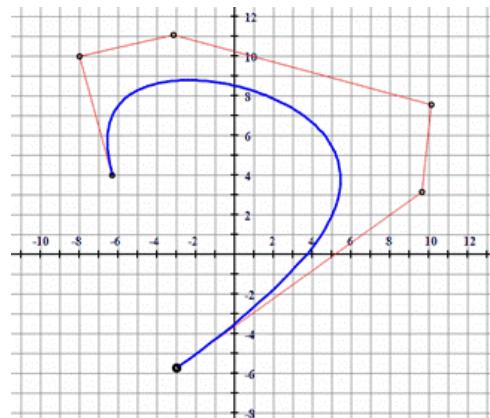


Рис. 1 Приклад апроксимації функції, що задана п'ятьма точками

Задача *інтерполяції* (наближеного відновлення) виникає, коли дана скінчена множина точок, через які потрібно провести криву. До форми і гладкості відновлюваної кривої можуть висуватись додаткові вимоги.

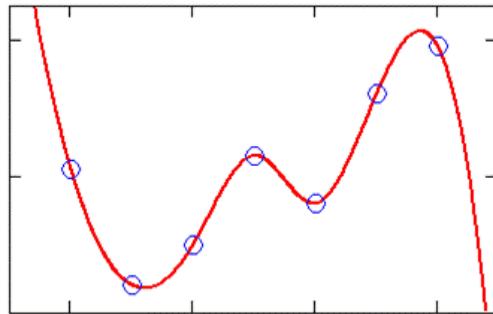


Рис. 2 Приклад інтерполяції функції, що задана сімома точками

Задача згладжування кривих полягає у відновленні гладкої функції, тобто потрібно побудувати криву, що проходить поблизу заданих точок із заданим допустимим відхиленням. Така задача має багато розв'язків, однак, наклавши на задану функцію додаткові умови, можна досягти необхідної однозначності.

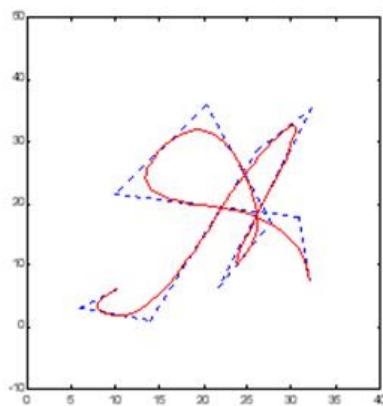


Рис. 3 Приклад згладжування кривої

Існує велика кількість методів розв'язку згаданих вище задач. Кожний з них є ефективним для свого класу об'єктів. У багатьох підсистемах комп'ютерної графіки і геометричних розрахунках перевага надається кусково-поліноміальним методам і представленням.

Сплайн є одним зі способів побудови кривих, поверхонь складної форми, які були вперше використані для математичного опису форми нових автомобілів, літаків, космічних кораблів, побутових приладів під час їхнього проектування.

Сплайн – це гладка крива, яка будується з використанням дуг і проходить через кілька контрольних точок, що керують формою сплайна. Іншими словами, сплайн – функція, область визначення якої розбита на фрагменти (сегменти), на кожному з яких функція є деяким поліномом (многочленом). Максимальний степінь поліномів в сплайні називається степенем сплайна.

Один із найбільш загальних типів сплайнів є криві Без'є, розроблені математиком П'єром Без'є. Криві та поверхні Без'є використовувалися в 60-х роках компанією "Рено" для комп'ютерного проектування форми кузовів автомобілів. Хоча варто зауважити, що в 1959 р. криві Без'є використовувалися Полем де Кастельє з компанії [«Сітроен»](#), але його дослідження не публікувались і приховувались компанією як [комерційна таємниця](#) до кінця 1960-х. Згодом це відкриття стало одним з найважливіших інструментів систем автоматизованого проектування та програм комп'ютерної графіки.

## 1. Крива Без'є – основа векторної графіки

Застосування кривої Без'є є досить широким: спосіб управління рухом, засіб створення анімаційних рисунків, метод опису художніх шрифтів, спосіб передачі виділених об'єктів, інтерактивний інструмент дизайну форми різноманітних об'єктів.

Без'є розробив метод зображення кривої, який створює у користувача більш природне сприйняття. Крива Без'є визначається вершинами багатокутника, який задає форму кривої. Кривій належить перша та остання вершина, в той час як інші вершини характеризують похідні, порядок та вид кривої. Таким чином, крива задається за допомогою відкритого багатокутника (ламаної) сформованого заданими точками, як показано на рис. 4.

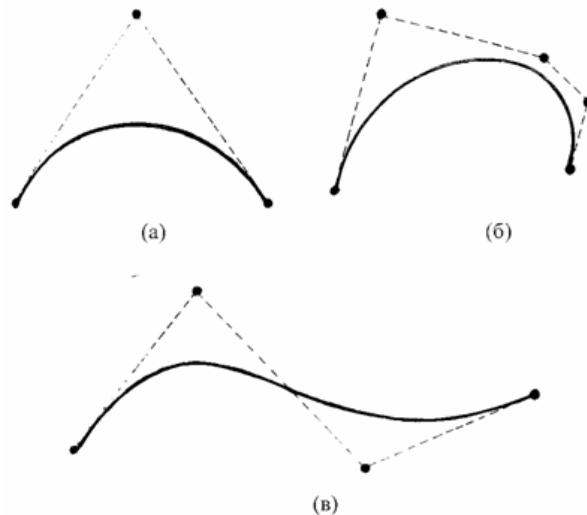


Рис 4. Приклади кривих Без'є

Так як вигляд кривої залежить від форми багатокутника, то зміна положення вершин цього багатокутника створює у користувача значно більше явного відчуття сприймання зв'язку між входом і виходом. Для того щоб збільшити порядок будь-якого криволінійного сегменту, потрібно тільки задати додаткові вершини. Така методика відрізняється значною гнучкістю і позбавлена багатьох недоліків.

Для побудови кубічної кривої Без'є (яку найчастіше використовують) необхідно мати 4 контрольні точки. Але крива проходить лише через дві з них, які називаються **опорними**, у свою чергу перша є початковою, друга – кінцевою. Інші дві точки, через які не проходить крива, називаються **керуючими**.

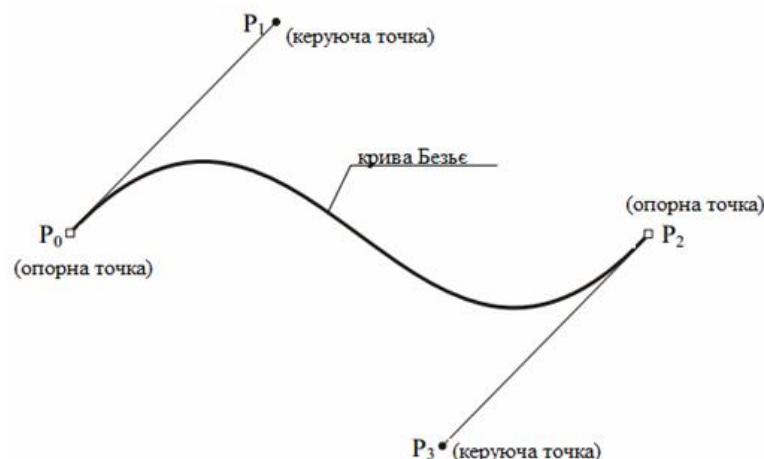


Рис 5. Вигляд кривої Безье

Якщо зміщаються початкова та кінцева точки, то крива теж відповідно змінюватиме свій вигляд. Зміщення керуючих точок змінить кривизну відповідної частини кривої Без'є. Таким чином за допомогою переміщення контрольних точок можна отримати безмежну кількість форм кривої Без'є.

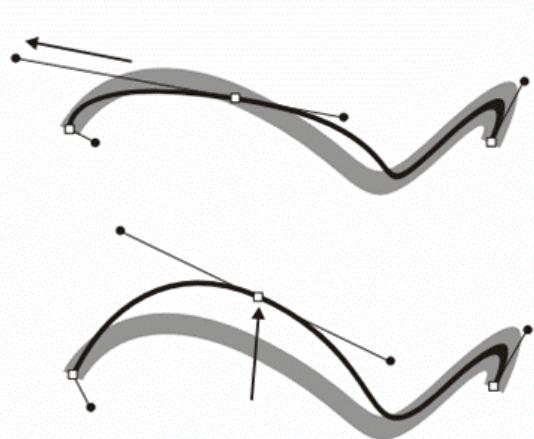


Рис. 6 Зміщення точок кривої

Для побудови кривої Без'є в комп'ютерній графіці використовують три основні способи: параметричний, рекурсивний, матричний.

## 2. Параметричний вигляд кривої Без'є

На початку ХХ ст. Сергій Наташевич Бернштейн, що закінчив Харківський університет, під час доведення теореми Стоуна-Вейєрштрасса вперше вивів поліноми, що будуються в явному вигляді, які і стали основою для отримання сплайнів, в тому числі і кривих Без'є.

Тобто крива Без'є – це лінійна комбінація базисних поліномів Бернштейна степені  $n$ , що задається виразом:

$$B(t) = \sum_{i=0}^n P_i b_{i,n}(t), \quad 0 < t < 1. \quad (1)$$

$n$  – степінь полінома, який характеризується  $n+1$  вершинами,

$i$  – порядковий номер вершини,

$P_i$  – функція компонент векторів опорних вершин,

$b_{i,n}(t)$  – базисні функції кривої Без'є (поліноми Бернштейна), що мають вигляд:

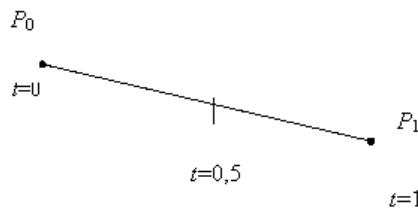
$$b_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}, \quad (2)$$

де

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}. \quad (3)$$

Розглянемо криві Без'є, що класифікуються за значенням  $n$ :

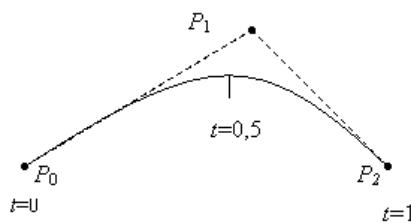
- $n=1$  (за двома точками). Крива вироджується у відрізок, що визначається опорними точками  $P_0$  і  $P_1$ .  
Параметричне представлення:



$$B(t) = (1-t)P_0 + tP_1.$$

Рис. 6 Вигляд кривої Без'є, побудованої при  $n=1$

- $n=2$  (за трьома точками). Параметричне рівняння має вигляд:



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

Рис. 7 Вигляд кривої Без'є, побудованої при  $n=2$

- $n=3$  (за чотирима точками – кубічна). Використовується досить часто, особливо в сплайнах.  
Параметричне рівняння:

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

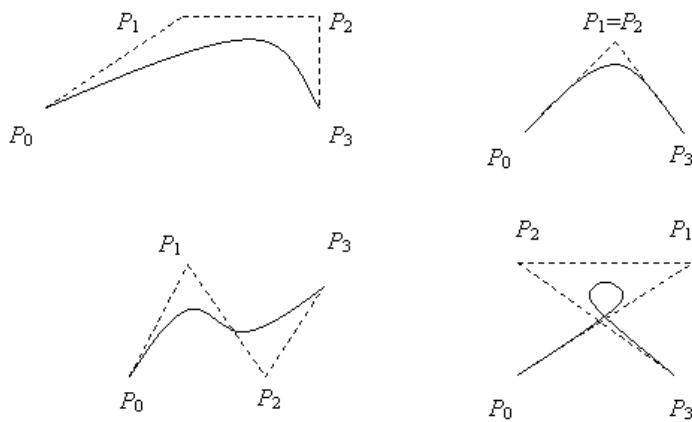


Рис. 8 Вигляд кубічної кривої Без'є

Властивості многочленів Бернштейна суттєво впливають на поведінку кривих Без'є. Наведемо деякі з них:

1. Многочлени Бернштейна набувають невід'ємних значень.
2. У сумі вони дають одиницю.
3. Многочлени не залежать від вершин масиву  $P$ , а залежать лише від кількості точок у масиві.

Геометричний алгоритм для кривої Без'є дозволяє обчислити координати  $(x, y)$  точки кривої Без'є за значенням параметра  $t$ .

1. Кожна сторона опорної ламаної ділиться у відношенні  $t:(1-t)$ , де  $t$  – аргумент точки поділу.
2. Точки поділу з'єднуються відрізками прямих і утворюють нову ламану (багатокутник). Кількість вузлів нового контуру на одиницю менша, ніж кількість вузлів попереднього контуру.
3. Сторони нового контуру знову діляться пропорційно значенню  $t$  і т.д. Це продовжується до тих пір, поки не буде отримана єдина точка поділу – точка кривої Без'є.

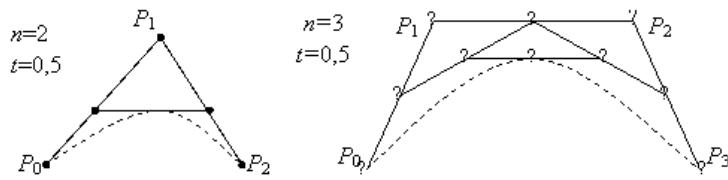


Рис. 9 Побудова кривої Без'є

Метод побудови кривої Без'є заснований на використанні пари дотичних (керуючих ліній), проведених до кінців сегмента кривої. На форму кривої впливають кут нахилу дотичної і довжина її відрізка.

### 3. Матричний та рекурсивний вигляд кривої Без'є

Рівняння кривої Без'є також можна записати у матричній формі.

Наприклад, для  $n=3$  (задано чотири точки) формула матиме вигляд:

$$B(t) = \begin{bmatrix} (1-t)^3 & 3t(1-t)^2 & 3t^2(1-t) & t^3 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}. \quad (4)$$

Групуючи коефіцієнти, отримаємо:

$$B(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}.$$

Загальний матричний вигляд кривої Без'є:

$$B(t) = T \cdot N \cdot P,$$

$$T = [t^n \ t^{n-1} \ \dots \ t \ 1],$$

$$N = \begin{bmatrix} \binom{n}{0} \binom{n}{n} (-1)^n & \binom{n}{1} \binom{n-1}{n-1} (-1)^{n-1} & \dots & \binom{n}{n} \binom{n-n}{n-n} (-1)^0 \\ \binom{n}{0} \binom{n}{n-1} (-1)^{n-1} & \binom{n}{1} \binom{n-1}{n-2} (-1)^{n-2} & \dots & 0 \\ \dots & & & \\ \binom{n}{0} \binom{n}{1} (-1)^1 & \binom{n}{1} \binom{n-1}{0} (-1)^0 & \dots & 0 \\ \binom{n}{0} \binom{n}{0} (-1)^0 & 0 & \dots & 0 \end{bmatrix},$$

$$P^T = [P_0 \ P_1 \ \dots \ P_{n-1} \ P_n].$$

Також існує рекурсивна формула побудови кривих Без'є, автором якої

$$\epsilon \text{ Поль де Кастельє: } B_{R, R, \dots, R_n}(t) = (1-t) B_{R, R, \dots, R_{n-1}}(t) + t B_{R, R, \dots, R_n}(t). \quad (5)$$

#### 4. Властивості кривої Без'є

У процесі інтерактивного конструювання ламана Без'є є засобом управління формою. Ефективність методу обумовлена такими *властивостями кривих Без'є*:

- 1) Порядок точок у масиві  $P$  суттєво впливає на вигляд кривої Без'є.
- 2) Форма кривої Без'є повторює хід ламаної  $P_0P_1\dots P_n$ . При зміні порядку точок повністю змінюється форма кривої.

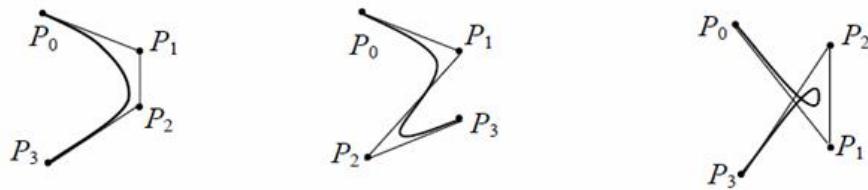


Рис.10 Кубічні криві Без'є при зміні порядку точок

- 3) Перша та остання точки кривої збігаються з відповідними точками масиву  $P$ , тобто  $B(0) = P_0$ ,  $B(1) = P_n$ . Усі інші вершини ламаної в загальному випадку знаходяться поза кривою.
- 4) Нахил дотичних векторів в крайніх точках кривої співпадає з нахилом, відповідно першої та останньої ланки ламаної Без'є.
- 5) У рівнянні, що описує елементарну криву Без'є, немає вільних параметрів, тобто заданий масив однозначно визначає криву Без'є. Тому немає можливостей якимось чином впливати на форму кривої Без'є.

6) Крива Без'є інваріантна відносно афінних перетворень. Тобто побудувавши криву Без'є, над нею можна здійснити афінні перетворення, а можна вчинити інакше: спочатку здійснити афінні перетворення над опорними вершинами, а потім побудувати криву Без'є на нових вершинах. Якщо результати будуть однаковими, то кажуть, що крива є інваріантною відносно цього афінного перетворення. При цьому легко помітити, наприклад, що поворот опорних вершин і подальша побудова кривої займає менше часу, ніж поворот самої кривої. Нагадаємо, що афінні перетворення включають у себе поворот, розтягування/стиск, паралельне перенесення та іх можливі комбінації.

7) Крива Без'є лежить в опуклій оболонці вершин масиву  $P$ .

Змістовоно многочлен Без'є можна представити як деяку намагнічену еластичну стрічку, закріплена в першій і останній точках. Стрічка притягується до кожної точки, причому чим вища напруженість магнітного поля в точках, тим більше буде притягнута до них. Під час прямування кратності до нескінченості многочлен Без'є прямує до ламаної кривої, точками спряження для якої служать точки-орієнтири.

Практичне конструювання кривих за методом Без'є є евристичною процедурою. Спочатку конструктор вручну накидує бажану криву. Пізніше він вказує вершини ламаної Без'є, яка на його думку, створить перше наближення до необхідної кривої. Наступним кроком є пересування вершин таким чином, щоб поступово покращити співпадання чи наближення. Якщо необхідно, деякі вершини викидаються або додаються нові.

Поряд з перевагами, криві Без'є мають і ряд недоліків:

1. Степінь функціональних поліномів напряму залежить від кількості точок у заданому векторі. Степінь многочлена  $B(t)$ , що визначає криву Без'є, на одиницю менший від кількості точок масиву, тобто крива Без'є побудована на  $n+1$  точці задається многочленом степеня  $n$ .

2. Якщо в масив  $P$  додати хоча б одну вершину, то необхідно повністю перерахувати параметричні рівняння кривої Без'є

3. Зміна хоча б однієї точки в масиві  $P$  приводить до помітної зміни всієї кривої Без'є.

Незважаючи на недоліки, крива Без'є є зручною для використання в задачах комп'ютерної графіки, оскільки формула є достатньо простою з точки зору математичних обчислень, універсальною з точки зору програмування, геометрично наглядною з точки зору користувача – художника, дизайнера, проектувальника.

## 5. Приклади побудови кривої Без'є

**Приклад 1.** Нехай маємо набір точок  $P_0=[1,1]$ ,  $P_1=[2,3]$ ,  $P_2=[4,3]$ ,  $P_3=[3,1]$ , що є вершинами багатокутника.

Використаємо параметричну формулу побудови кривої Без'є (1):

Тут  $n=3$ , оскільки маємо 4 вершини. Звідси

$$P(t) = P_0 b_{3,0} + P_1 b_{3,1} + P_2 b_{3,2} + P_3 b_{3,3},$$

$$b_{3,3}(t) = t^3, \quad b_{3,2}(t) = 3t^2 \cdot (1-t), \quad b_{3,1}(t) = 3t \cdot (1-t)^2.$$

$$P(0) = P_0 = [1,1], \quad P(1) = P_3 = [3,1].$$

Таким чином:

$$\begin{aligned}
 P(0.15) &= 0.614 \cdot P_0 + 0.325 \cdot P_1 + 0.0574 \cdot P_2 + 0.0034 \cdot P_3 = [1.5, 1.765], \\
 P(0.35) &= 0.275 \cdot P_0 + 0.444 \cdot P_1 + 0.239 \cdot P_2 + 0.043 \cdot P_3 = [2.248, 2.367], \\
 P(0.5) &= 0.125 \cdot P_0 + 0.375 \cdot P_1 + 0.375 \cdot P_2 + 0.125 \cdot P_3 = [2.75, 2.5], \\
 P(0.65) &= 0.043 \cdot P_0 + 0.239 \cdot P_1 + 0.444 \cdot P_2 + 0.275 \cdot P_3 = [3.122, 2.36], \\
 P(0.85) &= 0.0034 \cdot P_0 + 0.514 \cdot P_1 + 0.325 \cdot P_2 + 0.614 \cdot P_3 = [3.248, 1.75].
 \end{aligned}$$

Коефіцієнти для кривої Без'є наведені у таблиці:

Таблиця 1.

$t$	$b_{3,0}$	$b_{3,1}$	$b_{3,2}$	$b_{3,3}$
0	1	0	0	0
0.15	0.614	0.325	0.0574	0.0034
0.35	0.275	0.444	0.239	0.043
0.5	0.125	0.375	0.375	0.125
0.65	0.043	0.239	0.444	0.275
0.85	0.0034	0.0574	0.325	0.614
1	0	0	0	1

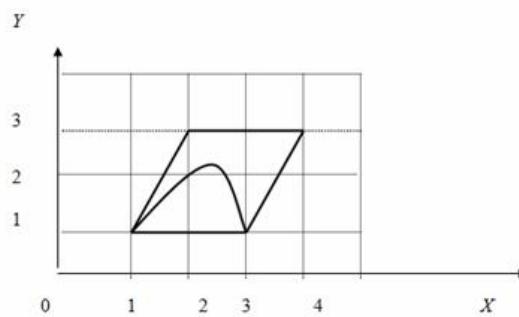


Рис. 11 Вигляд кривої Без'є (приклад 1).

**Приклад 2.** Складена кубічна крива Без'є – це неперервна крива  $\gamma$ , що є об'єднанням елементарних кубічних кривих  $\gamma_0, \gamma_1, \dots, \gamma_l$ , тобто

$$\gamma = \gamma_0 \cup \gamma_1 \cup \dots \cup \gamma_l.$$

Для побудови складеної кривої Без'є розбиваємо масив точок  $P_0, P_1, \dots, P_n$  ( $n$  має бути кратним 3) на  $l$  підмножин, кожна з яких містить чотири точки так, що остання точка попередньої підмножини – це перша точка наступної підмножини, тобто

$$\{P_0, P_1, P_2, P_3\}, \{P_3, P_4, P_5, P_6\}, \dots, \{P_{3l}, P_{3l+1}, P_{3l+2}, P_{3l+3}\}.$$

На кожній такій підмножині можна побудувати елементарну криву Без'є, тобто окремий фрагмент складеної кривої Без'є. Об'єднання окремих фрагментів дає складену кубічну криву Без'є, яка належить до класу неперервних функцій. Якщо тепер змінити якусь точку масиву, то складена крива Без'є зміниться тільки локально. Щоб крива була гладкою, необхідно забезпечити виконання умови на краях двох сусідніх сегментів:

$$P'(1) = gQ'(0).$$

$P'(1)$  – похідна першого сегменту в останній точці;

$Q'(0)$  – похідна другого (сусіднього) сегменту в першій точці;

$g$  – деяка стала.

Отже, нехай задано точки  $P_0, P_1, P_2, P_3, P_4, P_5$ .

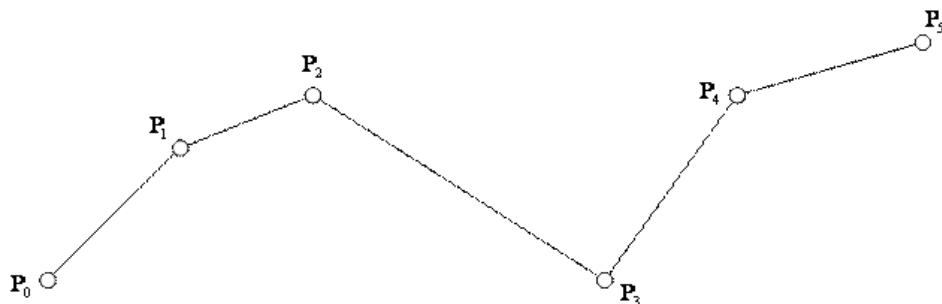


Рис. 12 Вхідні дані для побудови кривої Без'є

Побудуємо криву третього порядку. Як бачимо, з набору точок можна розглянути дві криві (два сегменти) з точками  $B_i$  та  $C_i$ . Виберемо середину відрізка  $B_0 = C_0$ , що знаходиться між точками  $P_2, P_3$ . Тоді перша крива буде мати:  $B_0 = P_0, B_1 = P_1, B_2 = P_2, B_3 = C_0$ , друга крива –  $C_0 = P_3, C_1 = P_4, C_2 = P_5, C_3 = P_5$  (рис.5).

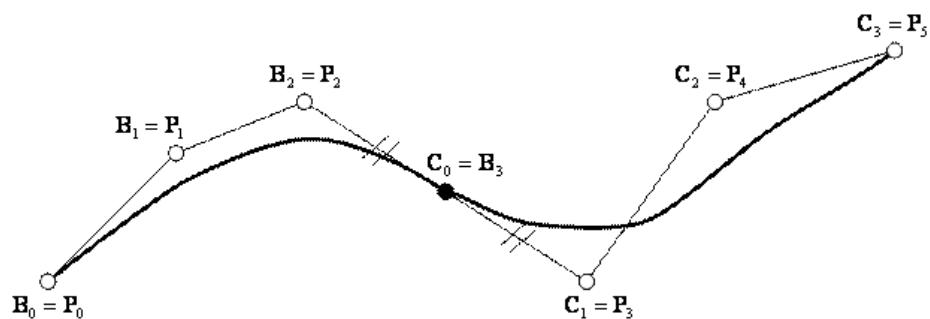


Рис.13. Побудова кривої Без'є

З прикладу випливає, що

$$P'(1) = n(B_3 - B_2) = 3(B_3 - B_2),$$

$$Q'(0) = m(C_1 - C_0) = 3(C_1 - C_0).$$

Тобто умова гладкості виконується.

$$P'(1) = Q'(0), \text{ тобто } |B_3 - B_2| = |C_1 - C_0|.$$

*Примітка.* Для того, щоб складена кубічна крива Без'є була C1-гладкою необхідно, щоб трійки вершин  $P_{3i-1}, P_{3i}, P_{3i+1}$  лежали на одній прямій. Тобто розбиваємо масив точок  $P_0, P_1, \dots, P_n$  на  $l$  підмножин таким чином:

$$\{P_0, P_1, P_2, P_3\}, \{P_1, P_2, P_3, P_4\}, \dots, \{P_{3l}, P_{3l+1}, P_{3l+2}, P_{3l+3}\}.$$

Відрізок  $\{P_0, P_1, P_2, P_3\}$ , описує криву між точками  $P_1$  та  $P_2$ , а відрізок  $\{P_1, P_2, P_3, P_4\}$  між точками  $P_2$  та  $P_3$  і т.д. Гладкість зшивання відрізків буде залежати від порядку полінома (чим вищий порядок, тим гладше будуть зшиватись відрізки). На границях необхідно вибрати умову, наприклад, що відповідні похідні рівні нулю.

**Зauważення!** Метод побудови кривої Без'є в графічних пакетах базується на використанні дотичних управлюючих ліній, що проведені до сегментів кривої на його кінцях (вузлах). При цьому форму кривої Без'є моделюють шляхом зміни нахилу дотичних і довжини відрізка, які визначаються контрольними точками.

## Висновки

Завдяки простоті задачі та маніпуляції, криві Без'є знайшли широке застосування в комп'ютерній графіці для моделювання гладких ліній. Крива цілком лежить в опуклій оболонці своїх контрольних точок. Ця властивість кривих Без'є з одного боку значно полегшує завдання знаходження точок перетину кривих (якщо не перетинаються опуклі оболонки опорних точок, то не перетинаються і самі криві), а з іншого боку дозволяє здійснювати інтуїтивно зрозуміле управління параметрами кривої в графічному інтерфейсі за допомогою її опорних точок. Крім того афінне перетворення кривої (перенесення, масштабування, обертання та ін) також можуть бути здійснені шляхом застосування відповідних трансформацій до опорних точок.

Найбільше значення мають криві Без'є другого та третього ступенів (квадратичні і кубічні). Криві вищих ступенів під час обробки вимагають більшого обсягу обчислень і для практичних цілей використовуються рідше. Для побудови ліній складних форм окремі криві Без'є можуть бути послідовно з'єднані один з одним. Для того, щоб забезпечити гладкість лінії у місці з'єднання двох кривих, три суміжні контрольні точки обох кривих повинні лежати на одній прямій. У програмах векторної графіки на зразок Adobe Illustrator або Inkscape подібні фрагменти відомі під назвою «шляхів» (path).

Ви зайшли під ім'ям Бірбан Юрій (Вихід)  
КГ ПЗ ПІ-31з

Українська (uk)  
Українська (uk)  
English (en)

Data retention summary  
Get the mobile app

# Комп'ютерна графіка

[Інформаційна панель](#) / [Мої курси](#) / [Комп'ютерна графіка](#) / [РОЗДІЛ 2. Лекція 5](#) / [Лекція 5](#)

## Лекція 5

### Лекція 5. Растроva модель зображень

Загалом, усі зображення КГ за принципом побудови можна поділити на растроvi, векторнi, фрактальнi, тривимiрнi. Моделлю називають опис за певними правилами, враховуючи деякi припущення. Прийнято вiд способу формування розглядати растроvу та векторну моделi зображень.

**5. 1. Поняття про растроvу модель.** Растроva модель представляє зображення у виглядi комбiнацiї точок (пiкселiв), яким притаманнi свiй колiр та яскравiсть i якi певним чином розташованi у координатнiй сiтцi. Пiксель (Pixel) – це скорочення, утворене вiд термiну Picture Element (англ.). Растром називають матрицю пiкселiв. Поняття матрицi вказує на розмiщення пiкселiв за координатами по горизонталi та вертикалi.

Такий пiдхiд є ефективним у випадку, коли графiчне зображення має багато напiвтонiв i iнформацiя про колiр важливiша за iнформацiю про форму (фотографiї та полiграфiчнi зображення). При редагуваннi растроvих об'ектiв, користувач змiнює колiр точок, а не формi лiнiй. Растроvi зображення називають точковими.



Рис.1. Приклад растроvого зображення.

Для опису розташування пiкселiв використовують систему координат. Координати пiкселiв утворюють дискретний ряд значень, найчастiше використовують цiлi числа для нумерацiї.

Пiксел (0,0) розмiщено в лiвому верхньому кутi площини. Растроva графiка базується на вiдомих алгоритмах, у яких зручно використовуються координати точок. Адреса пiкселя (2 координати) лiнеризується, тобто за двома значеннями (положення по осi X та по осi Y) обчислюється одне значення.

#### 5.2. Кiлькiснi та якiснi характеристики раstru

Растр має кiлькiсну та якiсну характеристики. Кiлькiсна характеристика раstru – це розмiр раstra, тобто кiлькiсть пiкселiв по вертикалi та горизонталi. Записують цю характеристику як NxM, де N, M – вiдповiднi кiлькостi пiкселiв.

Якiсна характеристика раstru – це роздiльна здатнiсть, яка вказує кiлькiсть точок на одиницю довжини, вимiрюється у точках на дюйм (dpi - dots per inch). Роздiльна здатнiсть залежить вiд вимог до якостi зображення та розмiру файла, способу оцифрування або методу створення готового зображення, вибраного форматu файла та iнших

параметрів. Чим вище вимоги до якості, тим більша має бути роздільності.



Рис.2. Зображення з різною роздільною здатністю.

Крім dpi, використовують ще інші різновиди роздільної здатності lpi, ppi, spi. Форма пікселів раstra визначається особливостями пристрою графічного виводу. Піксели можуть мати прямокутну чи квадратну форми. Розмір пікселя можуть дорівнювати кроку раstra.

**5.3. Типи зображень за глибиною кольору.** Глибина кольору - важливий параметр растрових зображень характеризує максимальну кількість кольорів, які використані у зображенні. задається кількістю біт.

Існує декілька типів зображень із різною глибиною кольору:

- чорно-білі;
  - у відтінках сірого;
  - з індексованими кольорами;
  - повноколірні;
- Чорно-білі зображення. На один піксель зображення відводиться 1 біт інформації - чорний та білий. Глибина кольору - 1 біт.
  - Зображення у відтінках сірого. Піксел сірого зображення кодується 8 бітами (1 байт). Глибина кольору - 8 біт, піксел може приймати 256 різних значень - від білого (255) до чорного (0 яскравості).
  - Зображення з індексованими кольорами. Перші кольорові монітори працювали з обмеженою колірною гамою (16, згодом 256 кольорів). Такі кольори називаються індексованими і кодуються 4 або 8 бітами у вигляді колірних таблиць. У такій таблиці всі кольори вже визначені і можна використовувати лише їх. Індексовані кольори використовують з метою економії обчислювальних ресурсів.
  - Повноколірні зображення. Глибина кольору не менше як 24 біти, що дає можливість відтворити понад 16 мільйонів відтінків. Повноколірні зображення називаються True Color (правдивий колір). Бітовий об'єм кожного пікселя розподіляється по основних кольорах обраної колірної моделі, по 8 бітів на колір. Колірні складові організуються у вигляді каналів, спільне зображення каналів визначає колір зображення. Повноколірні зображення на відміну від вище розглянутих є багатоканальними і залежать від колірної моделі (RGB, CMY, CMYK, Lab, HBS), які різняться за глибиною кольорів і способом математичного опису кольорів.

Для вимірювання глибини кольору використовується величина bpp (bit per pixel). Тобто, для чорно-білого зображення bpp=1, для індексованого зображення, коли індекс коротке ціле bpp=16 і т.д.

Обробка растрових зображень стосується, в першу чергу, кольорів та їх атрибутів. Ці задачі розв'язуються найефективніше. Якщо обробка растрових зображень стосується зміни форми, то спостерігаються спотворення.

При збільшенні раstroвого зображення відбувається пікселізація, тобто при масштабуванні збільшується розмір точок і стають помітними елементи раstra. Для усунення цього, потрібно заздалегідь оцифрувати оригінал із роздільністю, достатньої для якісного відтворення при масштабуванні. Або при масштабуванні застосовують метод інтерполяції, коли при збільшенні зображення, додається необхідне число проміжних точок. При зменшенні раstroвого зображення відбуваються спотворення, коли елементи раstra накладаються один на одного.

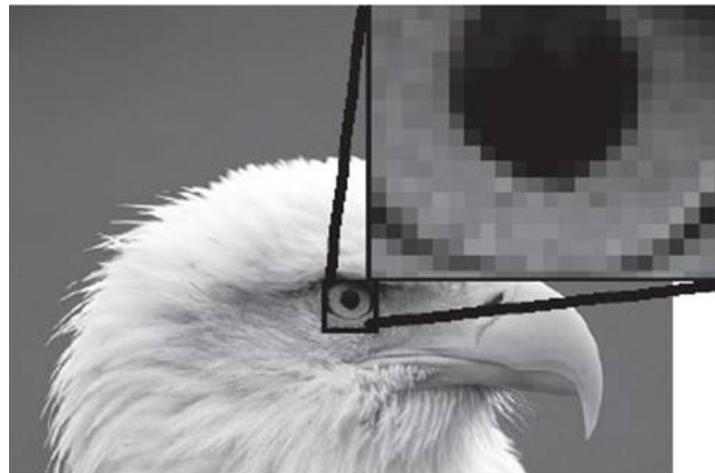


Рис. 3. Ефект пікселізації при збільшенні фрагмента зображення.

5.3. Програмні системи та формати файлів раstroвої графіки. Прикладні програми раstroвої графіки призначені для обробки оцифрованих фотографій, відеокадрів, кадрів мультиплікаційних фільмів, створення книжкових та журнальних ілюстрацій.

Найпопулярнішими прикладними програмами є продукти фірм

- Adobe - програма PhotoShop,
- Corel – програма PhotoPaint,
- Macromedia – програма FireWorks,
- Fractal Design – програма Painter,
- стандартний додаток у Windows - Paint.



Рис.4. Вигляд вікна найпростішого раstroвого редактора.

## Популярні формати растрової графіки

- Найпростіший раstralовий формат BMP є рідним форматом системи Windows, він підтримується всіма графічними редакторами, які працюють під її управлінням. Цей формат застосовується для збереження раstralових зображень, призначених для використання в Windows (і, по суті, більше ні на що не придатний).
- Найпопулярнішим форматом на інтернетівських просторах є достатньо вже давній формат GIF, запропонований CompuServe в 1987 році. Його особливістю є використання режиму індексованих кольорів (не більш 256), що обмежує область застосування формату зображень, які мають різкі кольорові переходи. Цей формат не може застосовуватися для поліграфії
- Найпопулярнішим форматом для збереження фотографічних зображень є JPEG (або JPG) JPEG може зберігати тільки 24-бітові повнокольорові зображення. Хоча JPEG добре стискає фотографії, але цей стиск відбувається з втратами і псує якість, тим не менше, він може бути легко налаштований на мінімальні, практично непомітні для ока, втрати. Проте не варто використовувати JPEG для збереження зображень, які будуть опрацьовуватися, так як при кожному збереженні в цьому форматі процес погіршення якості зображення носить лавиноподібний характер. Таким чином, можна зберегти якість зображення при мінімальному розмірі файлу.
- Як універсальний формат для зберігання раstralових зображень пропонується формат TIFF, який досить широко використовується, в першу чергу, у видавничих системах, що вимагають зображення найкращої якості. До речі, можливість запису зображень у форматі TIFF є однією з ознак високого класу сучасних цифрових фотокамер. Завдяки своїй сумісності з більшістю професійного ПЗ для обробки зображень, формат TIFF дуже зручний при перенесенні зображень між комп'ютерами різних типів (наприклад, з PC на Mac і назад).

### 5.4. Переваги та недоліки раstralової графіки.

#### Переваги раstralової графіки

- простота автоматизованого вводу (оцифрування) зображень, фотографій, слайдів, рисунків за допомогою сканерів, відеокамер, цифрових фотоапаратів;
- фотorealістичність. Можна отримувати різні ефекти, такі як туман, розмитість, тонко регулювати кольори, створювати глибину предметів тощо.

#### Недоліки раstralової графіки

- складність управління окремими фрагментами зображення. Потрібно самостійно виділяти ділянку, що є складним процесом;
- ефекти спотворення при масштабуванні (“розмитість” при збільшенні, “ступінчастість” при зменшенні)
- розмір файлу є пропорційним до площини зображення, роздільноті і типу зображення, і, переважно, при хорошій якості є великим.
- раstralове зображення має певну роздільність і глибину представлення кольорів. Ці параметри можна змінювати лише у визначених межах і, як правило, із втратою якості;

Недолік, пов’язаний з масштабуванням, може бути вирішений спеціальними методами, що базуються на методах інтерполяції. Недолік, що стосується великих витрат пам’яті, може бути вирішений, якщо застосовувати спеціальні методи стиску зображень.

[◀ План лекції 5](#)[Перейти до...](#)[Презентація до Лекція 5 ►](#)

Ви зайшли під ім'ям Бірбан Юрій (Вихід)

КГ ПЗ ПІ-31з

Українська (uk)

Українська (uk)

English (en)

Data retention summary

Get the mobile app

# Комп'ютерна графіка

[Інформаційна панель](#) / [Мої курси](#) / [Комп'ютерна графіка](#) / [РОЗДІЛ 2. Лекція 6](#) / [Лекція 6](#)

## Лекція 6

### Лекція 6.

#### Фрактальна графіка (Частина 1)

*Математика, якщо на неї правильно подивитися,*

*відображає не тільки істину,*

*але і незрівнянну красу.*

Крім растрової та векторної графіки окремо розглядають ще фрактальну графіку та тривимірну графіку.

##### 6.1. Поняття про фрактальну графіку.

Слово фрактал утворено від латинського *fractus* і в перекладі означає той, що *складається з фрагментів (подрібнений)*.

Термін “фрактал” запропоноване Бенуа Мандельбротом в 1975 році для позначення самоподібних структур. Народження фрактальної геометрії прийнято пов'язувати з виходом в 1977 році книги Мандельброта ‘*The Fractal Geometry of Nature*’.

Фракталом називається структура, що складається з частин, які в якомусь сенсі подібні до цілого.

В основі цього явища лежить дуже проста ідея : нескінчену по красі і різноманітності множину фігур можна отримати з відносно простих конструкцій за допомогою всього двох операцій - копіювання і масштабування.





Рис. 1. [Приклади](#) фракталів у природі

Фракали знаходять все більше й більше практичних застосувань у науці. Основна причина цього полягає в тому, що вони описують реальний світ іноді навіть краще, ніж традиційна фізика чи математика. Більшість просторових систем у природі є нерегулярним і фрагментарним, форма цих систем погано піддається опису апаратом евклідової геометрії.

Масштабна інваріантність, що спостерігається у фракталах, може бути або точною, або наближеною. Інваріантність - незмінність якої-небудь величини по відношенню до деяких перетворень.

Однією з основних властивостей фракталів є самоподібність. У найпростішому випадку невелика частина фрактала містить інформацію про весь фрактал. Серед важливих властивостей фракталів з точки зору програмної реалізації ітераційність та рекурсивність фрактальних структур.

Фактично знайдений спосіб легкого представлення складних об'єктів, які схожі на природні. Тобто за допомогою декількох коефіцієнтів можна задати ліній і поверхні дуже складної форми. Фрактальна геометрія незамінна при генерації штучних хмар, гір, поверхні морів тощо.

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 за формулами, рівняннями.

## 6.2. Класифікація фракталів може бути проведена за різними критеріями.

Найчастіше використовують класифікацію за **способом побудови**

- Геометричні (конструктивні);
- Алгебраїчні (рекурентні, динамічні);
- Стохастичні (випадкові)

- IFS-фракали

Геометричні та алгебраїчні фракали називають детермінованими. Окремо розглядають підвид фракталів, що утворюється через систему ітеративних функцій (Iterated Functions System) – IFS-фракали.

Крім вище наведеної класифікації, можна навести поділ фракталів за **рівнем самоподібності**. За цим критерієм виділяють три типи фракталів:

- **фракали точної самоподібності;**
- **фракали з майже самоподібністю;**
- **фракали з статистичною самоподібністю.**

Фракталами точної самоподібності є геометричні фрактали, масштабна інваріантність яких є у явному вигляді. До фракталів з майже самоподібністю належать алгебраїчні фрактали. Майже самоподібність означає, що частинки фракталу у певному сенсі (за деяким законом) подібні до цілого зображення. Статистична самоподібність використовує випадкові величини.

Ще одна класифікація враховує поняття **детермінованості**. До детермінованих фракталів належать геометричні, алгебраїчні фрактали, а до недетермінованих - стохастичні.

### 6.3. Геометричні фрактали

Фрактали цього типу будуються поетапно. Спочатку зображується основа . Потім деякі частини основи замінюються на фрагмент . На кожному наступному етапі частини вже побудованої фігури , аналогічні заміненим частинам основи , знову замінюються на фрагмент , взятий у відповідному масштабі. Кожного разу масштаб зменшується. Коли зміни стають візуально непомітними , вважають , що побудована фігура добре наближає фрактал і дає уявлення про його форму . Однак насправді для отримання фрактала потрібно нескінченне число етапів. Міняючи основу і фрагмент , можна отримати багато різних геометричних фракталів .У двомірному випадку їх отримують за допомогою деякої ламаної (або поверхні в тривимірному випадку), яку наз. *генератором*.

За один крок алгоритму кожен з відрізків, складових ламаної, замінюється на ламану-генератор, у відповідному масштабі.

У результаті повторення цієї процедури, виходить геометричний фрактал.

Найвідоміші фрактали такого типу - Сніжинка Коха, крива Коха, крива Гільберта-Пеано, множина Кантора, килим Серпінського, трикутник Серпінського, крива дракона, Т-Квадрат, губка Менгера...

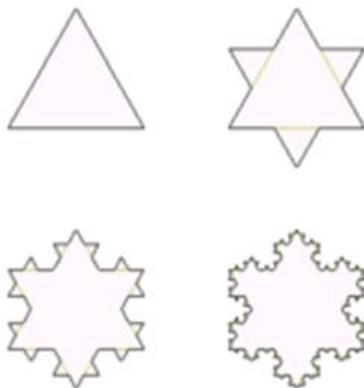


Рис.2. Кроки побудови сніжинки Коха

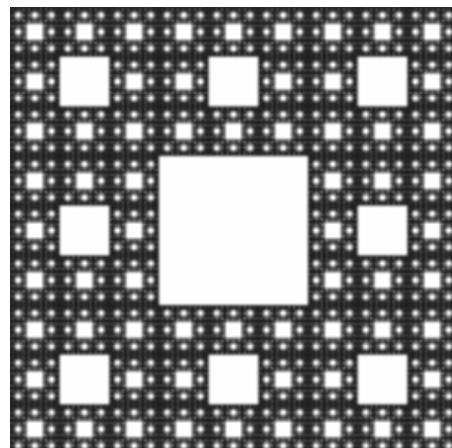


Рис. 3. Килим Серпінського

Цей фрактал є межею нескінченної конструкції, що починається з трикутника та доповнюється рекурсивною заміною кожного сегменту набором із чотирьох сегментів, які утворюють трикутний «виступ».

#### 6.4. Теорія динамічних систем - основа алгебраїчних фракталів

Алгебраїчні фракталі належать до найбільшої групи фракталів. Отримують їх за допомогою нелінійних процесів в n-мірних просторах.

Для побудови алгебраїчних фракталів використовуються ітерації нелінійних відображень, що задаються простими алгебраїчними формулами.

Найбільш вивчені двомірні процеси.

Інтерпретуючи нелінійний ітераційний процес, як дискретну динамічну систему, можна користуватися термінологією теорії цих систем: *фазовий портрет, стацийний процес, аттрактор* і т.д. Дискретною динамічною системою є система, що має визначені стани в конкретні моменти часу

Відомо, що нелінійні динамічні системи володіють декількома стійкими станами. Той стан, в якому опинилася динамічна система після деякого числа ітерацій, залежить від її початкового стану. Тому кожен стійкий стан (або як говорять – *аттрактор*) володіє деякою областю початкових станів, з яких система обов’язково потрапить в дані кінцеві стани. Таким чином, фазовий простір системи розбивається на *області тяжіння* аттракторів. Якщо фазовим є двомірний простір, то забарвлюючи області тяжіння різними кольорами, можна отримати *колірний фазовий портрет* цієї системи (ітераційного процесу). Міняючи алгоритм вибору кольору, можна отримати складні фрактальні картини з химерними багатоколірними узорами.

Алгебраїчні фракталі визначаються рекурентним відношенням в кожній точці простору (плошина комплексних чисел).

$$\begin{aligned} Z_{k+1} &= F(Z_k) \\ Z_k &= x + iy, \quad i^2 = -1. \end{aligned}$$

Причому  $F$  - будь-яка нелінійна функція.

Несподіванкою для математиків стала можливість за допомогою примітивних алгоритмів породжувати дуже складні нетривіальні структури.

#### 6.5. Алгоритм алгебраїчних фракталів. [Приклади](#).

У загальному випадку є така задача – заповнити область  $A_x * B_y$  (матриця раству) кольорами, які генеруються за законом

$$Z_{k+1} = F(Z_k, c)$$

де  $z_k, z_{k+1}, c$  - комплексні числа,  $z_k = x_k + i * y_k$ , тобто  $(x_k, y_k)$  – точка раству,  $h$  – крок раству;  $F$  – нелінійна функція,  $k$  – крок ітерації (відповідає за колів),  $0..K_{\max}$  (велике число, н-ад, 500).

Обчислення відбуваються доти, поки виконається умова

$$|z_k| \leq R \quad i \quad k \leq K_{\max},$$

де  $R$  – межа множини, певна константа  $i$ , як тільки  $k > K_{\max}$ , колір точки – чорний.

Якщо виконалася умова  $|z_k| > R \quad i \quad k > K_{\max}$ , необхідно взяти  $k$  як колір. Наприклад, як залишок від ділення 256.

Умова (2) може мати вигляд  $\operatorname{Re}(z) \leq R_1, \operatorname{Im}(z) \leq R_2, R_1, R_2$  - певні константи.

Прикладами фракталів цього типу є множина Мандельброта, палаючий корабель, фрактал Ляпунова, басейни Ньютона, біоморфи...

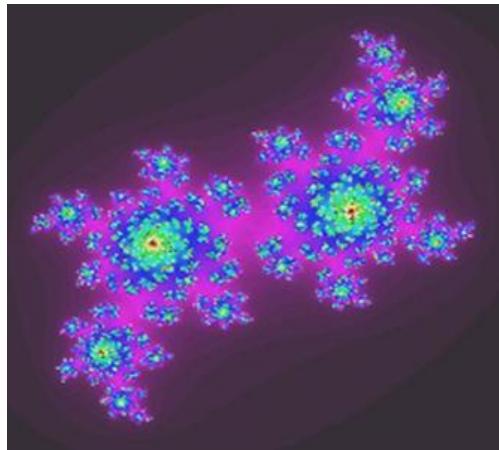


Рис. 4. Множина Жуліа

Для фракталів Мандельброта правило  $F$  задано формулою :  $F(z) = z^2 + c$  , де  $z$  і  $c$  - комплексні числа. Кожній точці на прямій відповідає те чи інше дійсне число , а кожній точці на площині - комплексне . Для комплексних чисел, як і для дійсних , визначені операції додавання і множення. Множина Мандельброта - чорна область на ілюстраціях - складається з усіх таких  $c$  , що послідовність точок  $z_0 = 0$  ;  $z_1 = 0^2 + c = c$  ;  $z_2 = c^2 + c$  ;  $z_3 = (c^2 + c)^2 + c$ ; .. і т.д. не прямує на нескінченність , тобто всі ці точки лежать всередині деякого кола з центром на початку координат. На збільшених зображеннях видно вкрай складна структура поблизу кордону.

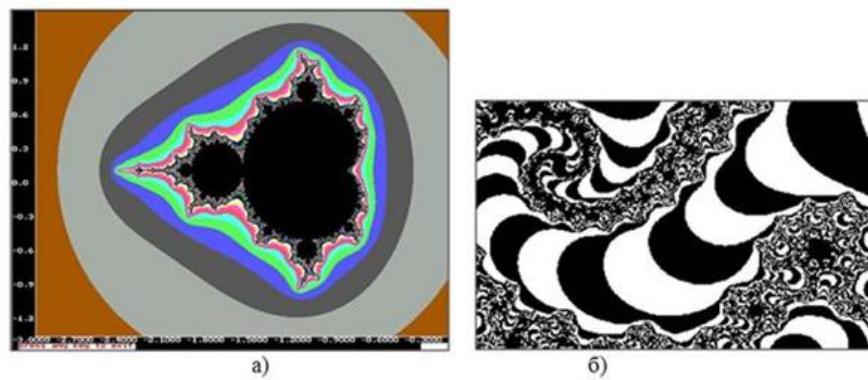


Рис. 5. а) Множина Мандельброта;

б) Збільшена ділянка границі множини Мандельброта

Формули для побудови фракталів Ньютона засновані на методі вирішення нелінійних рівнянь , який був придуманий великим математиком ще в XVII столітті.

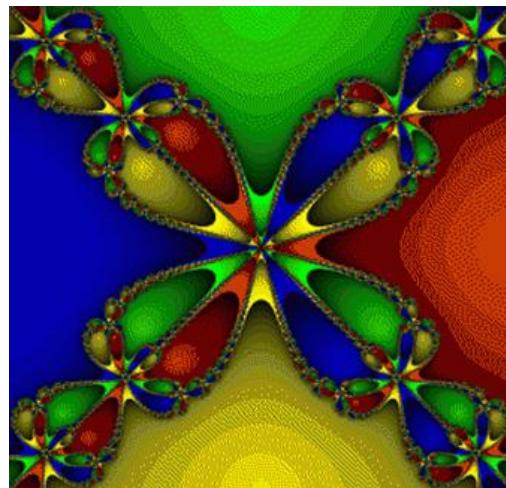


Рис.6. Приклад фракталу Ньютона.

Застосовуючи загальну формулу Ньютона  $z_{n+1} = z_n - f(z_n)/f'(z_n)$ ,  $n = 0, 1, 2, \dots$  для вирішення рівняння  $f(z) = 0$  до многочлену  $z^k - a$ , отримаємо послідовність точок:  $z_{n+1} = ((k-1)z_n^k - a) / kz_n^{k-1}$ ,  $n = 0, 1, 2, \dots$ . Вибираючи як початкові наближення різні комплексні числа  $z_0$ , будемо отримувати послідовності, які сходяться до корення цього многочлена. Оскільки коренів у нього рівно  $k$ , то вся площину розбивається на  $k$  частин - областей тяжіння коренів. Межі цих частин мають фрактальну структуру.

Остання зміна: Wednesday 23 June 2021 18:35 PM

[◀ План лекції 6](#)

Перейти до...

[Презентація до Лекція 6 ►](#)

Ви зайшли під ім'ям Бірбан Юрій (Вихід)

КГ ПЗ ПІ-31з

Українська (uk)

Українська (uk)

English (en)

Data retention summary

Get the mobile app

# Комп'ютерна графіка

[Інформаційна панель](#) / [Мої курси](#) / [Комп'ютерна графіка](#) / [РОЗДІЛ 2. Лекція 7](#) / [Лекція 7](#)

## Лекція 7

### Лекція 7.

#### Фрактальна графіка (Частина 2)

*Математика, якщо на неї правильно подивитися,*

*відображає не тільки істину,*

*але і незрівнянну красу.*

Крім растрової та векторної графіки окремо розглядають ще фрактальну графіку та тривимірну графіку.

##### 7.1. Стохастичні фрактали

Такі фрактали утворюються в тому випадку, якщо в ітераційному процесі випадковим чином міняти які-небудь його параметри. При цьому виходять об'єкти дуже схожі на природні – несиметричні дерева, порізані берегові лінії, рельєф місцевості і поверхні моря і т.д.

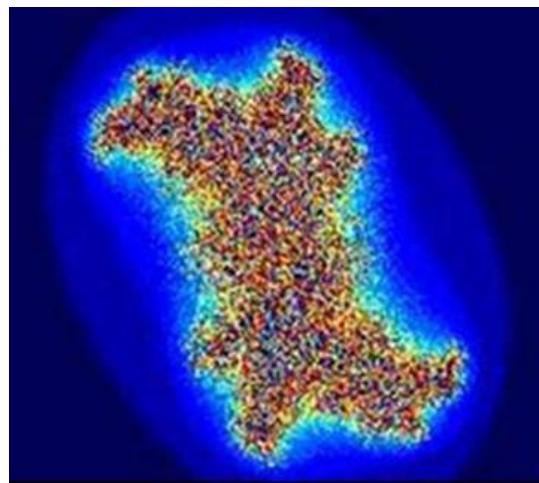


Рис. 7.1 Приклад стохастичного фракталу на основі множини Жюліа

Представниками цього класу можна вважати траекторію броунівського руху, різні види рандомізованих фракталів, тобто таких, які можна отримати за допомогою рекурсивної процедури, в яку на кожному кроці введений випадковий параметр.

Змоделювати броунівський рух на комп'ютері досить просто: частинку потрібно зміщувати на задану відстань (у моделі одинакових кроків) в довільному напрямку.

Спроможність фрактальної графіки застосовують для автоматичної генерації незвичайних ілюстрацій. Типовий представник даного класу фракталів «Плазма».

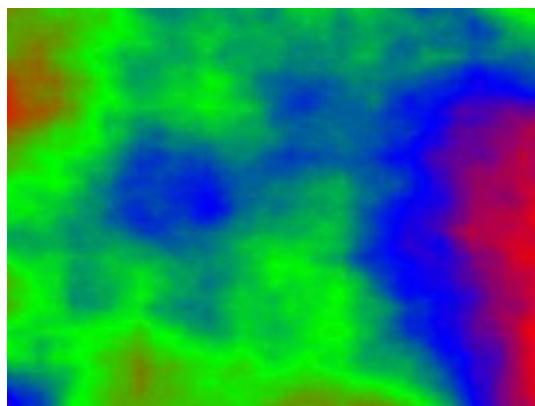


Рис.7.2. Фрактал «Плазма»

Для його побудови використовують прямокутник і для кожного його кута визначається колір. Далі знаходиться центральна точка прямокутника і розфарбовується у колір рівний середньому арифметичному кольорів по кутах прямокутника плюс деяке випадкове число. Чим більше випадкове число - тим більш «рваним» буде малюнок. Якщо, наприклад, вважати, що колір точки - це висота над рівнем моря, то отримається замість плазми - гірський масив. Саме на цьому принципі моделюються гори в більшості програм.

Гранична крива рандомізованої сніжинки Коха може служити моделлю, наприклад, контуру хмари або острова.

Поверхня гір моделюється з використанням фракталів. Початок з трикутника в тривимірному просторі та з'єднання центральних точок кожного ребра відрізками, отримуючи 4 трикутники. Далі центральні точки зсуваються догори або донизу на випадкову відстань у фіксованому діапазоні. Процедура повторюється зі зменшенням діапазону на кожній ітерації вдвічі.

Рекурсивна природа алгоритму гарантує, що ціле є статистично подібним до кожної з деталей.

## 7.2. Система ітеративних функцій (Iterated Functions System )

IFS-метод з'явився в середині 80-х років як простий засіб отримання фрактальних структур.

IFS є системою функцій деякого фіксованого класу функцій, що відображають одну багатовимірну множину на іншу.

$$\begin{cases} x_{k+1} = F_x(x_k, y_k) \\ y_{k+1} = F_y(x_k, y_k) \end{cases} \quad (1)$$

Найпростіша IFS складається з афінних перетворень :

$$\begin{cases} X' = A * X + B * Y + C \\ Y' = D * X + E * Y + F \end{cases} \quad (2)$$

A, B, C, D, E, F – деякі коефіцієнти.

Іншими різновидами перетворень є квадратичні, кубічні, проектуючі.

Наприклад, проектні:

$$X' = (A_1X + B_1Y + C_1)/(D_1X + E_1Y + F_1); \quad (3)$$

$$Y' = (A_2X + B_2Y + C_2)/(D_2X + E_2Y + F_2);$$

квадратичні:

$$X' = A_1X^2 + B_1XY + C_1Y^2 + D_1X + E_1Y + F_1; \quad (4)$$

$$Y' = A_2X^2 + B_2XY + C_2Y^2 + D_2X + E_2Y + F_2.$$

Можна побудувати IFS для кривої Коха. Виділимо в структурі подібні частини і для кожної з них обчислимо коефіцієнти афінного перетворення. У афінний колаж буде включено стільки афінних перетворень, скільки існує частин подібних цілому зображення. Неважко бачити, що ця крива має чотири частини, подібні до цілої кривої (рис. 7.3). Для знаходження IFS знову розташуємо перше покоління цього фрактала на сітці координат дисплея 640 × 350.

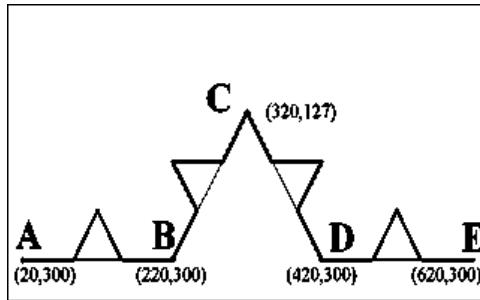


Рис. 7.3. Заготовка для побудови IFS кривої Коха

Для її побудови потрібний набір афінних перетворень, що складається з чотирьох перетворень:

$$X' = 0.333X + 13.333; \quad (6)$$

$$Y' = 0.333Y + 200;$$

$$X' = 0.333X + 413.333;$$

$$Y' = 0.333Y + 200;$$

$$X' = 0.167X + 0.289Y + 130;$$

$$Y' = -0.289X + 0.167Y + 256;$$

$$X' = 0.167X - 0.289Y + 403;$$

$$Y' = 0.289X + 0.167Y + 71.$$

Результат застосування цього афінного колажа після десятої ітерації можна побачити на рис. 7.4.

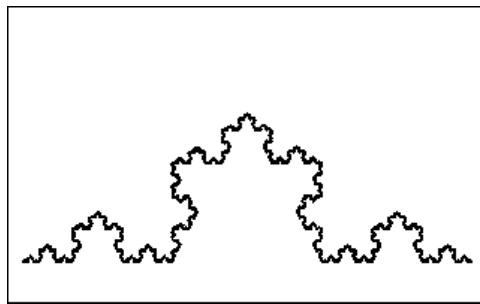


Рис. 7.4. Крива Коха, побудована за допомогою IFS у прямокутнику 640 × 350

У 1988 році відомі американські фахівці в теорії динамічних систем Барнслі і Слоан запропонували деякі ідеї, засновані на міркуваннях теорії динамічних систем, для стискування і зберігання графічної інформації.

Вони назвали свій метод фрактальним стискуванням інформації. Походження назви пов'язане з тим, що геометричні фігури, що виникають в цьому методі, зазвичай мають фрактальну природу в сенсі Мандельброта.

На підставі цих ідей Барнслі і Слоан створили алгоритм, який, за їх твердженням, дозволить стискувати інформацію в 500-1000 разів.

За цим методом зображення кодується декількома простими перетвореннями (у нашому випадку афінними), тобто коефіцієнтами цих перетворень (A,B,C,D,E,F).

Наприклад, закодувавши якесь зображення двома афінними перетвореннями, ми однозначно визначаємо його за допомогою 12-ти коефіцієнтів.

Якщо тепер задатися якою-небудь початковою точкою (наприклад  $X=0 Y=0$ ) і запустити ітераційний процес, то ми після першої ітерації отримаємо дві точки, після другої – чотири, після третьої – вісім і так далі. Через декілька десятків ітерацій сукупність отриманих точок описуватиме закодоване зображення.

Але проблема полягає в тому, що дуже важко знайти коефіцієнти IFS, які кодували б довільне зображення.

Використання IFS для стиску звичайних зображень (наприклад, фотографій) засноване на виявленні локальної самоподібності, на відміну від фракталів, де спостерігається глобальна самоподібність. За алгоритмом Барнслі відбувається виділення в зображенні пар областей, менша з яких подібна більшій, і збереження декількох коефіцієнтів, що кодують перетворення та переводять велику область в меншу. Потрібно, щоб безліч “менших” областей покривали все зображення. При цьому у файл, що кодує зображення будуть записані не тільки коефіцієнти, що характеризують знайдені перетворення, але і місцеположення та лінійні розміри “великих” областей, які разом з коефіцієнтами описуватимуть локальну самоподібність кодованого зображення. Алгоритм відновлення у цьому випадку повинен застосовувати кожне перетворення не до всієї множини точок, що вийшли на попередньому кроці алгоритму, а до деякої підмножини, що належить області, відповідній перетворенню.

### 7.3. Програмні системи та формати файлів фрактальної графіки.

Фрактальний метод стиску використовується для економного збереження зображень без втрати якості. FIF (Fractal Image Format) – спеціальний формат файлів для фрактальних зображень.

Для синтезу зображень фрактальної графіки існує не дуже багато програмних продуктів. Серед них немає визнаного лідера, як в програмах растроїв і векторної графіки. Також особливістю цих програм є те, що вони використовують фрактальний підхід до синтезу зображень разом із звичним набором інструментів растроїв і векторної графіки. Відомі фрактальні редактори – Fractal Design Painter, Bruce, Fractint.

Фрактальна графіка у пакетах наукової візуалізації використовується для побудови ілюстрацій, що імітують природні процеси.

Серед таких програмних засобів продукти фірми Golden SoftWare:

*Surfer* - створення дво-, тривимірних поверхонь;

*Map Viewer* - побудова кольорових карт.

*Surfer* дозволяє опрацьовувати та візуалізувати двовимірні набори даних, що описані функцією  $z=f(x,y)$ . Можна побудувати цифрову модель поверхні, застосувати допоміжні операції і візуалізувати результат.

*Map Viewer* дозволяє вводити та корегувати карти - змінювати масштаб, перетворювати координати, обробляти й виводити у графічному вигляді числову інформацію, пов'язану з картами.

Пакет *Iris Explorer* (фірма Graphics) призначено для створення моделей погодних умов та поверхонь океану.

Пакет *Earth Watch* (фірма Earth Watch) призначений для моделювання та демонстрації тривимірного зображення метеоумов над Землею, дозволяє будувати топологічні поверхні і прогнозувати погоду.

### 7.4. Генератори фрактальних зображень

*XaosPro* – безкоштовний генератор фрактальних зображень (анімований рух у фракталі).

*Apophysis* – інструмент для генерації фракталів на основі базових фрактальних формул.

*XenoDream* – створення різноманітних об'ємних структур шляхом комбінування простих форм і фрактальних зображень, отриманих із застосуванням IFS-методів.

Fractracker – створення тривимірних зображень на базі фрактальної геометрії, що представляє собою щось середнє між генератором фракталів і 3D-редактором.

Incendia - це мультипроцесорний генератор тривимірних фракталів, розроблений іспанським програмістом Ramiro Perez, який вивчав фракали з 1989 року.

## 7.5. Області застосування фракталів

Фрактальний світ добре відображає реальний, оскільки властивості фракталів демонструють багато природних об'єктів.

Фрактальна математика сьогодні дуже широко застосовується, вона може бути використана для аналізу змін цін і заробітної платні, статистики помилок на телефонних станціях і опису Всесвіту у цілому.

Фрактальним підходом можна описувати:

- структури неживої природи: лінії берегів, рельєф місцевості, обриси хмар, структури корисних копалин,
- структури живої природи: системи кровообігу людини, будови нирок і легенів, які нагадують по структурі дерева з кроною,
- процеси: економічні, турбулентні, які використовуються при прогнозі погоди.

Остання зміна: Friday 16 July 2021 15:17 PM

[◀ План лекції 7](#)

Перейти до...

[Презентація до Лекція 7 ►](#)

Ви зайдли під ім'ям Бірбан Юрій (Вихід)

КГ ПЗ ПІ-31з

Українська (uk)

Українська (uk)

English (en)

Data retention summary

Get the mobile app

# Комп'ютерна графіка

[Інформаційна панель](#) / [Мої курси](#) / [Комп'ютерна графіка](#) / [РОЗДІЛ 2. Лекція 8](#) / [Лекція 8](#)

## Лекція 8

### Лекція 8.

#### Тривимірна графіка

Крім растрової та векторної графіки окрім розглядають ще фрактальну графіку та тривимірну графіку.

#### 8.1. Поняття про тривимірну графіку.

3D графіка (3 Dimensions) — один з розділів комп'ютерної графіки, який призначений для зображення об'єктів (на площині).

Дана графіка застосовується в кінематографі, архітектурній справі, комп'ютерних іграх, поліграфії, науці та промисловості.

Основна відмінність від 2D графіки — опис і представлення зображення в трьох площинах, яке в подальшому перетворюється в 2D для виводу на екран або папір.

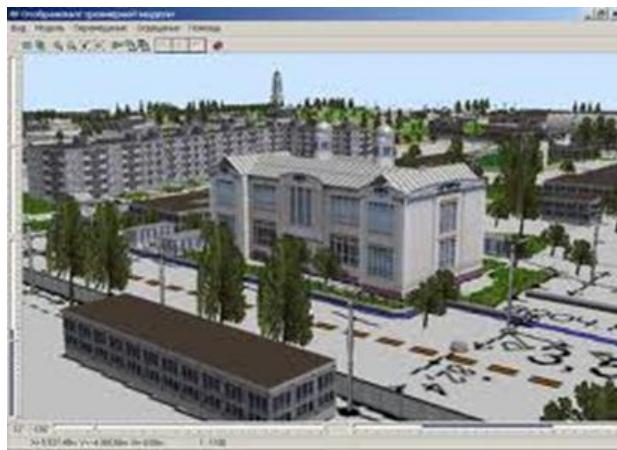


Рис. 8. Приклад тривимірного зображення.

3D і 2D графіка тісно взаємопов'язані між собою, оскільки тривимірна графіка використовує алгоритми 2D векторної і 2D растрової графіки.

Здебільшого для побудови 3D об'єкта достатньо мати уявлення про його лицевий вигляд, вигляд збоку та вигляд зверху.

У тривимірній комп'ютерній графіці всі об'єкти зазвичай представляються як набір поверхонь або частинок.

Мінімальну поверхню називають полігоном. В якості полігону зазвичай вибирають трикутники.

## Тривимірна графіка для реалістичних зображень

3D-графіка призначена для імітації тривимірних образів об'єктів, які попередньо створюються в пам'яті комп'ютера в такій послідовності:

- попередня підготовка,
- створення геометричної моделі сцени,
- налаштування освітлення і знімальних камер,
- підготовка і призначення матеріалів,
- візуалізація сцени.

Таким чином створюється уявний світ, який часто називають віртуальним.

Тривимірна графіка використовує віртуальний простір, який називають сценою. У цьому просторі проектувальник розміщує необхідні об'єкти, призначає для них певний матеріал (дерево, залізо, скло тощо), розміщує джерела світла, а також віртуальні камери, що визначають точки перегляду сцени.

Під час відтворення тривимірної графіки на екрані комп'ютера будується геометрична проекція тривимірної моделі на площину екрану. Цей процес називають рендерингом або візуалізацією.

*Попередня підготовка* передбачає продумування складу сцени, розміщення об'єктів і їх деталей, які будуть видимими з передбачуваних напрямів спостереження.

На етапі *створення геометричної моделі сцени* за допомогою різноманітних інструментальних засобів будується тривимірні геометричні моделі об'єктів сцени, після чого сцену можна розглядати і “фотографувати” з будь-якого потрібного ракурсу.

*Налаштування освітлення і знімальних камер* передбачає правильний вибір джерел світла, що дозволяє виконувати імітацію фотографування сцени в будь-яких умовах освітленості. Освітленість всіх об'єктів, тіні від них і бліки світла розраховуються автоматично.

Моделі знімальних камер дають можливість розглядати тривимірну сцену і виконувати її знімання під будь-яким вибраним кутом зору.

*Підготовка і призначення матеріалів* надає сцені візуальної правдоподібності. Працюючи з матеріалами, можна налаштовувати такі їх якості, як сила блиску, прозорість, дзеркальність, рельєфність та інші. Реальні фотографії можна включати в склад матеріалів або використовувати для імітації фону.

*Візуалізація сцени* полягає в проведенні програмою розрахунків і нанесення на зображення всіх тіней, бліків, взаємних відблисків об'єктів і т. п.

Тобто для отримання тривимірного зображення потрібно виконати *моделювання* - створення математичної моделі сцени і об'єктів в ній та *рендеринг* (rendering) - побудова проекції відповідно до вибраної фізичної моделі .

### 8.2. Рендеринг

На етапі рендерингу математична (векторна) просторова модель перетворюється на плоску (растрову) картинку.

Існує декілька технологій візуалізації часто комбінованих разом.

**Z-буфер** – алгоритм, який відповідає за створення зображень 3D-об'єктів, спираючись на глибину елементів зображення. Використовується в «OpenGL» і «DirectX».

При створенні (візуалізації) 3d- об'єкту його глибина генерується на осі Z-координат і зберігається у Z-буфери. Z-буфер є двомірним масивом (X,Y- координати) із глибиною для кожного екранного пікселя. Коли інший об'єкт сцени повинен бути відображенний у цьому пікселі зараз, тоді порівнюється дві глибини та перекривається поточний піксель, якщо об'єкт знаходиться більше до спостерігача. Обрізування

глибина зберігається в Z-буфері і замінює попередню. Z-буфер дозволяє правильно відтворювати звичне для нас сприйняття об'єктів, розміщених на різних відстанях.

Сканлайн («кидання променю», спрощений алгоритм зворотного трасування променів) - розрахунок кольору кожної точки картинки на основі побудови променю з точки зору спостерігача через уявний отвір в екрані на місці цього пікселя «в сцену» до перетину з першою поверхнею. Колір пікселя буде таким же, як колір цієї поверхні;

Це спрощений, нерекурсивний варіант трасування променів, не відбувається подальша обробка відбитих чи заломлених променів, а враховується лише перша поверхня-перешкода на шляху променів.

Трасування променів (Ray tracing, рейтрейсинг) - те ж, що і сканлайн, але колір пікселя уточнюється за рахунок побудови додаткових променів від точки перетину променю погляду. Ідея алгоритму полягає в тому, щоб простежити хід променя від уявного ока глядача крізь кожен піксель на уявному екрані і обчислити колір об'єкта, видимого оком крізь нього.

Глобальне освітлення— розрахунок взаємодії поверхонь і середовищ у видимому спектрі випромінювання за допомогою інтегральних рівнянь.

Такі алгоритми враховують не лише пряме світло, що надходить безпосередньо від джерела світла, але також і відбиті промені світла від інших поверхонь об'єктів сцени.

### 8.3. Програмні системи та формати файлів тривимірної графіки.

Програмні пакети, що дозволяють створювати тривимірну графіку, тобто моделювати об'єкти віртуальної реальності і створювати на основі цих моделей зображення (3d-редактори), дуже різноманітні. Останні роки стійкими лідерами в цій галузі є комерційні продукти: такі як 3D Studio Max, Maya, Lightwave 3D, Softimage, Sidefx Houdini, Maxon Cinema 4D, Rhinoceros 3D, Nevercenter Silo, пакет Blender.

Щоб отримати зображення тривимірного об'єкту, необхідно створити в програмі його об'ємну модель.

Модель об'єкту в багатьох 3d-редакторах відображається в чотирьох вікнах проекцій, що дає якнайповніше уявлення про геометрію об'єкту.

У тривимірній графіці використовується мова VRML (Virtual Reality Modelling Language), яка призначена для опису тривимірних зображень і оперує об'єктами, що описують геометричні фігури та їх розташування в просторі.

Мова була створена в 1994 році консорціумом на чолі з Silicon Graphics для застосування в мережах INTERNET. Чинним стандартом є VRML 97, відомий як ISO / IEC 1477.

Серед відомих форматів файлів *WRL*-файл – звичайний текстовий файл зі списком об'єктів, які названі вузлами (створений через VRML), *3DS* – файл пакету 3D Studio MAX, який зберігає 3D-моделі у формі каркасних сіток, текстуру, ефекти тощо. *MAX* – ще один формат 3D Studio MAX, а *X* – “рідний” формат DirectX, структурно- та контекстнонезалежний формат.

Остання зміна: Friday 16 July 2021 15:34 PM

[◀ План лекції 8](#)

Перейти до...

[Презентація до лекції 8 ►](#)

Ви зайшли під ім'ям Бірбан Юрій (Вихід)

КГ ПЗ ПІ-31з

Українська (uk)

Українська (uk)

English (en)

Data retention summary

Get the mobile app

# Комп'ютерна графіка

[Інформаційна панель](#) / [Мої курси](#) / [Комп'ютерна графіка](#) / [Розділ 2. Лекція 9](#) / [Лекція 9](#)

## Лекція 9

### **Лекція 4. Комп'ютерні моделі кольорів.**

*Всі кольори - друзі своїх сусідів  
й любителів іх пропилежностей.*

#### **4.1. Поняття про колір. Психофізичні аспекти**

Колір - це властивість матеріальних об'єктів, яка сприймається як усвідомлене зорове відчуття та виникає в результаті дії на око потоків видимого електронно-магнітного випромінювання (з довжинами хвиль від 380 до 760 нм).

Той або інший колір «привласнюється» людиною об'єкту в процесі зорового сприйняття цього об'єкту.

Світло сприймається або безпосередньо від джерела, наприклад, від освітлювальних приладів, або як відбиття від поверхонь об'єктів або заломлення при проходженні крізь прозорі і напівпрозорі об'єкти.

Амплітуда, що визначає енергію хвилі, відповідає за яскравість кольору.

Око людини - складна оптична система. Фоторецептори поділяються на два види: палички і колбочки.

Палички є високочутливими елементами і працюють в умовах слабкого освітлення. Вони нечутливі до довжини хвилі і тому не "розвірзняють" кольору.

Колбочки, навпаки, "розвірзняють" колір.

Паличок існує тільки один тип, а колбочки поділяються на три види, кожен з яких чутливий до певного діапазону довжин хвиль (довгі, середні або короткі).

Саме поняття кольору є особливістю людського "бачення" навколошнього середовища.

Перехід від одного кольору до іншого здійснюється безперервно, поступово. Кожному кольору співставляється не якось одна довжина хвилі світла, а довжини хвиль, що потрапляють в деякий інтервал значень.

Таблиця 1. Відповідність довжини хвилі кольорам.

	КОЛІР	ДОВЖИНА ХВИЛІ (МКМ)
	Фіолетовий	0,38 - 0,45
	Синій	0,45 - 0,5
	Блакитний	0,5 - 0,53
	Зелений	0,53- 0,57
	Жовтий	0,57 - 0,59
	Оранжевий	0,59 - 0,62
	Червоний	0,62 - 0,76

Максимум спектральної чутливості ока знаходитьться в жовто-зеленій частині спектру (0,555 мкм).

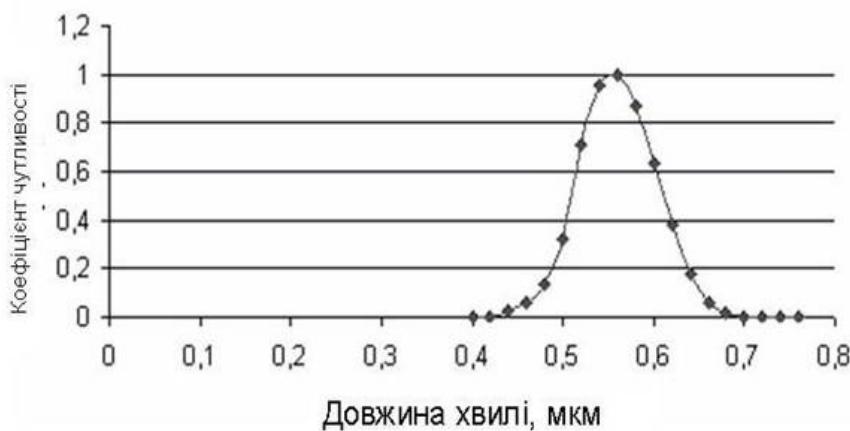


Рис.1. Крива спектральної чутливості ока при денному світлі

Сучасні моделі кольорів базуються на відомих теоріях фізики.

До дослідів Ньютона (кінець 17 ст.) вважалося, що білий колір – найпростіший. Ньютон це заперечив, білий колір – це суміш безлічі кольорів. Ним було висловлено припущення – будь-який колір отримується змішуванням основних.

Згідно фізики природа світла носить двоїстий характер. Світло – потік частинок (корпускулярна теорія). Світло – потік електромагнітних хвиль різної довжини і амплітуди (теорія Гука і Гюйгенса).

## 4.2. Поняття моделі кольорів

Призначення кольорної моделі – дати засоби опису кольору в межах деякого колірного діапазону, у тому числі і для виконання інтерполяції кольорів.

Існують різні моделі, оскільки із зображенням виконуються різні дії: відображення на екран, видрук на принтер, опрацювання кольорів, перетворення в сірі тони, корекція яскравості, інтенсивності і т.п.

Кожна модель має своє призначення, тобто ефективна для виконання окремих операцій.

Розглянемо апаратно-орієнтовані триколірні моделі RGB, CMY та триатрибутні HSV, HSL.

**4.2.1. Адитивна модель RGB** – це апаратно-орієнтована модель. в якій кольори описуються за допомогою змішування трьох базових кольорів – червоного (Red), зеленого (Green), синього (Blue) – в різних пропорціях. Тому модель RGB наз. адитивною (від англ. «add» складати, додавати).

Схема адитивних кольорів працює на основі принципу випромінювання світла. RGB використовується в дисплеях для формування кольорів.

У цій схемі відсутністю всіх кольорів є чорний колір, а присутністю всіх кольорів – білий.

Простір RGB геометрично представляють кубом в координатному просторі з початком відліку (0,0,0), вісь X – компонента червоного кольору, вісь Y – зелена компонента, вісь Z – синя компонента.

Будь-який колір у такому випадку представляється точкою з трьома координатами (r,g,b), де  $r,g,b \in [0;1]$ . Можливе також трактування, що де  $r,g,b \in [0;255]$ .

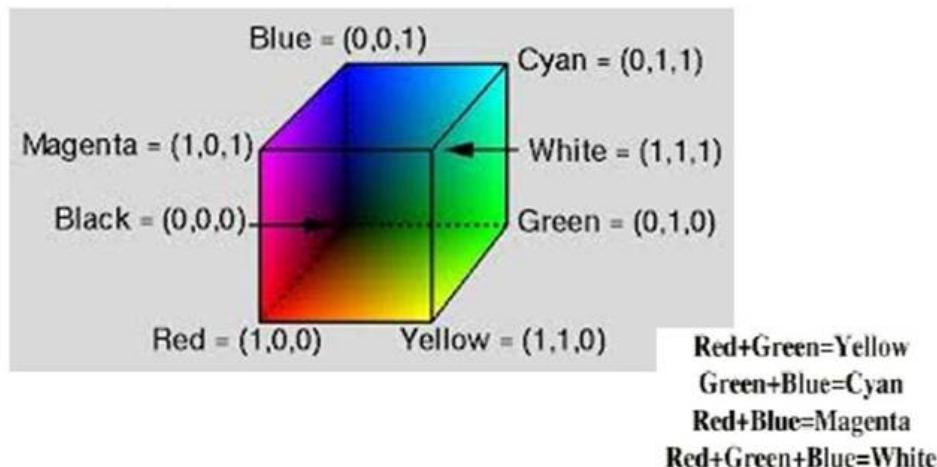


Рис.2. Колірний куб моделі RGB

Недолік RGB - обмеженість у застосуванні, лише на пристроях, які працюють за принципом випромінювання, неможливість відображення деяких кольорів, а саме насичених зелено-синіх.

**4.2.2. Субтрактивна модель CMY** - апаратно-орієнтована модель, яка використовується для формування кольорів на основі принципу віднімання від падаючого світлового потоку частини, яка формується шаром фарби з трьох компонент (блакитний/Cyan, Magenta/пурпурний, Yellow/жовтий).

Тому модель CMY називають субтрактивною (від англ. «sub» віднімати).

Колір в CMY утворюється при відніманні інших кольорів від загального променя світла ( $r+g+b$ ).

У цій схемі білий колір утворюється в результаті відсутності всіх кольорів, тоді як їх присутність дає чорний колір.

Схема субтрактивних кольорів працює на основі принципу відбиття та поглинання світла.

Дана система була широко відома задовго до того, як комп'ютери почали використовуватися для створення графічних зображень. Її область застосування – поліграфія.

У просторі RGB навпроти базових кольорів розміщено базові кольори CMY. Кажуть, що вони доповнюють один одного, поглинають. Кольори моделі CMY є додатковими до кольорів моделі RGB, тобто доповнюючими їх до білого.

Таким чином система координат CMY – той же куб, що і для RGB, але з початком відліку в точці з RGB-координатами (1,1,1), відповідною білому кольору.

Насправді на практиці користуються системою CMYK (а не CMY). Компонента K (від "Black") введена додатково, оскільки це не економно змішувати 3 "непрості" фарби для отримання "простого" чорного кольору, тим більше ідеального чорного кольору змішуванням трьох базових не отримується (модель допускає певні неточності).

Зв'язок адитивної та субтрактивної моделей можна представити математичним виразом.

$$\begin{pmatrix} r \\ g \\ b \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} c \\ m \\ y \end{pmatrix}$$

З цієї формулі легко визначаються пари доповнюючих кольорів.

Перетворення кольорів з системи RGB в систему CMYK стикається з рядом проблем. Основна складність полягає в тому, що в різних системах кольори можуть змінюватися. В даний час більшість графічних редакторів дозволяють працювати безпосередньо в кольорах CMYK.

**4.2.3. Моделі кольорів HSV і HSL** використовуються, щоб позбутися обмежень, що накладаються апаратним забезпеченням. Ці моделі використовуються, коли коректують яскравість, насиченість, перетворюють зображення в сіре та ін.

HSV задається трьома атрибутами:

Hue – колірний тон,

Saturation - насиченість ,

Value – інтенсивність.

Ця модель побудована на основі суб'єктивного сприйняття кольору людиною і добре узгоджується з нею.

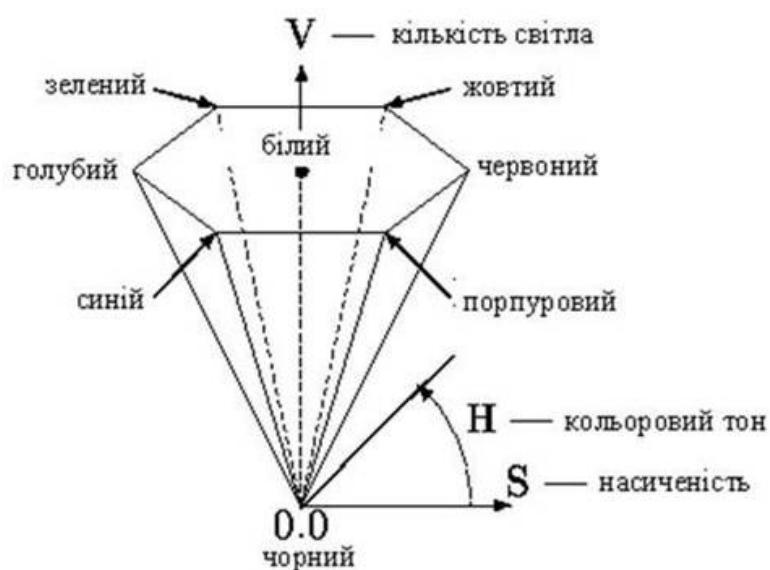


Рис. 3. Колірний простір HSV.

Компонента H визначається довжиною хвилі домінуючої компоненти в спектрі, ("чистий пігмент").

Колірний тон вимірюється в градусах.

Доповнюючі кольори відрізняються на  $180^\circ$ . Червоний  $0^\circ$  – блакитний  $180^\circ$ .

Жовтий  $60^\circ$  – синій  $240^\circ$ . Зелений  $120^\circ$  – пурпурний  $300^\circ$ .

Компоненти S і V змінюються від 0 до 1.

S – насиченість або чистота кольору, визначається часткою білого кольору.

S вказує наскільки колір близький до "чистого" пігменту і визначається відстанню до осі піраміди.

V – інтенсивність, вказує на загальну кількість світлового потоку, що потрапляє в око, (яскравість).

V=0 відповідає вершині і задає чорний колір.

Вісь піраміди задає сірі кольори (S=0) від чорного до білого (центр основи).

HSL – модель аналогічна до попередньої. Відмінність в компоненті L – освітленість, вказує величину чорного відтінку, доданого до кольору.

Простір HSL – це подвійна піраміда, яку отримують витягуванням у HSV-піраміді точки S=0, V=1 (білий колір) вгору.

Інші моделі XYZ, CIE Luv, (Y,Cb,Cr).

#### 4.3. Кодування кольорів в комп'ютері.

**4.3.1. Повноколірне кодування** передбачає, що компоненти в моделях задаються числами побайтно. Наприклад, для Windows API таке компонентне кодування кольору в адитивній схемі (4 байти)

00000000 bbbbbbbb gggggggg rrrrrrrr. Перший байт використовується для задання прозорості і тоді використовується система RGB $\alpha$ .

Моделі дозволяють працювати з усіма можливими кольорами. Проте це не завжди потрібно. Можна зекономити пам'ять, якщо працювати з певним набором кольорів, які є актуальними для зображень.

**4.3.1. Палітра кольорів** – це набір актуальних кольорів для певних зображень, представлений у формі таблиці. Палітра застосовується, якщо потреба в економії ресурсів.

Таблиця палітри визначає колір через індекс (порядковий номер). Тобто кольору відповідає з точки зору комп'ютерної реалізації всього-навсього один байт.

Таблиця 2. Представлення палітри кольорів.

Індекс	R	G	B	Колір
0	0	0	0	Чорний
255	255	255	255	Білий

Крім стандартних палітр кольорів, можна використати спеціально під потреби сформовані палітри. Як знати, які кольори актуальні? Відповідь на це питання дає процес, який називається оптимізацією палітри.

1) Найпростіший підхід полягає у переборі всіх пікселів картинки, і обрахунку, скільки разів зустрічається кожен колір. Палітра складається з тих кольорів, які зустрічаються частіше за інших.

Якщо деякий відтінок синього кольору зустрічається 100 разів, а відтінок червоного тільки 20, то перевага віддається синьому кольору. Основний недолік - деякі кольори будуть зовсім виключені з палітри.

2) Рівномірна розподіленість кольорів - це вибір комплекту кольорів для палітри з рівномірно розподіленими червоною, зеленою і синію компонентами. Такий підхід забезпечує широкий вибір кольорів.

Але при цьому не враховується, що в більшості картинок немає рівномірного колірного розподілу.

3) Метод квантування кольорів медіанним перетином

Колірний простір розглядається як тривимірний куб. Кожна вісь куба відповідає одному з трьох основних кольорів: червоному, зеленому, синьому. Кожна з трьох сторін розбивається на 255 рівних частин, поділки на осіх нумеруються від 0 до 255.

Перший крок полягає у відображені усіх колірних точок.

Другий крок полягає у відсіканні "країв" куба, які не містять пікселів.

Наприклад, якщо всіх пікселів значення червоної компоненти не менші, ніж 8 і не більше, ніж 250, то відкидаються частини куба від K=0 до K=7 і від K=251 до K=255.

Третій крок - розрізання отриманого паралелепіпеда на два в серединній точці (медіані) найдовшої сторони.

Тепер паралелепіпед розділений на два паралелепіпеди меншого розміру.

Наступні кроки такі ж. Далі попередній процес - відсікання порожніх "країв" і розрізання найдовшої сторони в серединній - повторюється для двох менших паралелепіпедів. Тепер початковий куб роздільний на чотири паралелепіпеди.

Медіанний перетин повторно застосовується для того, щоб розділити куб на 8, 16, 32, 64, 128 і 256 паралелепіпедів.

Формування палітри відбувається за центрами паралелепіпедів. Кожен з 256 паралелепіпедів містить піксели приблизно однакового кольору і центр кожного паралелепіпеда представляє оптимальне значення кольору для палітри.

Маючи координати вершин, дуже просто обчислити координати центральної точки.

Альтернативним методом є формування палітри за усередненими значеннями. До палітри потрапляє середнє значення всіх пікселів, які знаходяться усередині паралелепіпеда.

На обчислення йде більше часу, але отримана палітра буде точнішою.

Остання зміна: Friday 27 August 2021 16:16 PM

[◀ План лекції](#)

Перейти до...

[Презентація 8 ►](#)

Ви зайшли під ім'ям Бірбан Юрій (Вихід)

КГ ПЗ ПІ-31з

Українська (uk)

Українська (uk)

English (en)

Data retention summary

Get the mobile app

# Комп'ютерна графіка

[Інформаційна панель](#) / [Мої курси](#) / [Комп'ютерна графіка](#) / [Розділ 3. Лекція 11](#) / [Лекція 10](#)

## Лекція 10

### [Лекція 6](#)

#### [Лекція 5.](#) Основи створення рухомих зображень

Математичною основою задачі створення рухомих зображень у КГ є афінні перетворення.

#### 5.1. Визначення перетворень координат

Нехай задана  $n$ -мірна система координат  $(k_1, k_2, \dots, k_n)$ , яка описує положення точки в просторі за допомогою числових значень  $k_i$ . Якщо створити іншу,  $N$ -вимірну, систему координат в базисі  $(m_1, m_2, \dots, m_N)$  і поставити задачу визначення координат в новій системі, знаючи координати в старій, то рішення (якщо воно існує) можна записати в такому вигляді:

$$\begin{cases} m_1 = f_1(k_1, k_2, \dots, k_n) \\ m_2 = f_2(k_1, k_2, \dots, k_n) \\ \dots \\ m_N = f_N(k_1, k_2, \dots, k_n) \end{cases}, \quad (5.1)$$

де  $f_i$  – функція перетворення  $i$ -ої координати, аргументи – координати в системі  $k_i$ .

Існує й зворотня задача до (5.1) – за відомими координатами  $(m_1, m_2, \dots, m_N)$  визначити координати  $(k_1, k_2, \dots, k_n)$ . Розв'язок зворотної задачі записується так:

$$\begin{cases} k_1 = F_1(m_1, m_2, \dots, m_N) \\ k_2 = F_2(m_1, m_2, \dots, m_N) \\ \dots \\ k_n = F_n(m_1, m_2, \dots, m_N) \end{cases}, \quad (5.2.)$$

де  $F_j$  – функції оберненого перетворення.

У випадку, коли розмірності координат не співпадають ( $n \neq N$ ), здійснити однозначне перетворення координат не вдається. Наприклад, за двовимірними екранними координатами не можна без додаткових умов однозначно визначити тривимірні координати об'єктів. Якщо розмірності систем співпадають ( $n = N$ ), то також можливі випадки, коли не можна однозначно вирішити пряму чи зворотну задачі.

Перетворення координат класифікують:

- ✓ за системами координат – наприклад, перетворення з полярної системи координат в прямокутну;
- ✓ за видом функції перетворення.

#### 5.2. Лінійні перетворення координат. Афінне перетворення.

За видом функції перетворення розрізняють лінійні та нелінійні перетворення. Якщо при всіх  $i = 1, 2, \dots, N$  функції  $f_i$  – лінійні відносно аргументів  $(k_1, k_2, \dots, k_n)$ , тобто:

$$f_i = a_{i1}k_1 + a_{i2}k_2 + \dots + a_{in}k_n + a_{in+1}, \quad (5.3)$$

де  $a_{ij}$  ( $j=1, 2, \dots, n+1$ ) – константи, то такі перетворення називають лінійними, а при  $n=N$  – афінними.

Якщо хоча б для одного і функція  $f_i$  – нелінійна відносно  $(k_1, k_2, \dots, k_n)$ , тоді перетворення координат в цілому не є лінійним.

Наприклад, перетворення:

$$X = 3x + 5y;$$

$$Y = 4xy + 10y,$$

нелінійне, оскільки у виразі для  $Y$  присутнє  $xy$ .

Лінійні перетворення зручно для наочності записують у матричній формі:

$$\begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n-1,1} & a_{n-1,2} & \dots & a_{n-1,n-1} \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{pmatrix} * \begin{pmatrix} k_1 \\ k_2 \\ \dots \\ k_{n-1} \\ k_n \end{pmatrix} + \begin{pmatrix} a_{1,n+1} \\ a_{2,n+1} \\ \dots \\ a_{n-1,n+1} \\ a_{n,n+1} \end{pmatrix}$$

(5.4).

Тут матриця коефіцієнтів ( $a_{ij}$ ) множиться на матрицю – стовпець ( $k_i$ ) і в результаті отримаємо матрицю – стовпець ( $m_i$ ).

Властивості афінних перетворень:

- $n$ -вимірний об'єкт відображається в  $n$ -вимірний, точка – в точку, лінія – в лінію, поверхня – в поверхню;
- зберігається паралельність ліній і площин;
- зберігаються пропорції паралельних об'єктів (довжин відрізків на паралельних прямих і площ на паралельних площинах).

Ці властивості дозволяють будувати прообрази полігонів на площині й поліедрів у просторі за скінченим набором точок – їх вершинами.

**5.3. Афінні перетворення трьох видів** є основою для рухомих зображень:

- переміщення/зсув;
- масштабування (збільшення/зменшення);
- поворот на кут.

Ці перетворення можна проводити як відносно системи координат, так і відносно самого об'єкту зображення. Задамо певну двовимірну систему координат ( $x, y$ ).

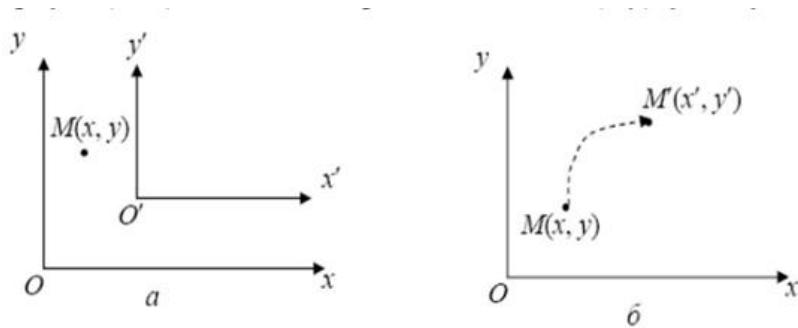


Рис. 1. а) Перетворення системи координат. б) Перетворення об'єкту

Причому ці перетворення є оберненими один до одного.

### 5.3.1. Переміщення/зсув – перетворення зображене на Рис.2.

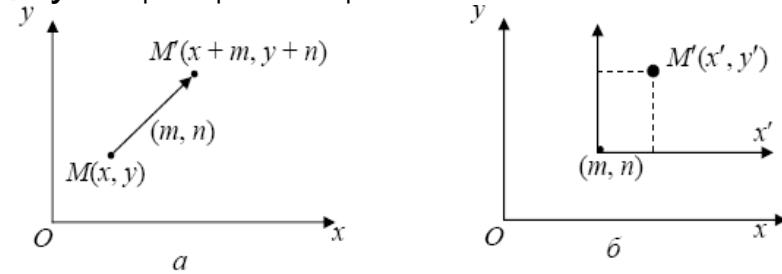


Рис. 2. а) Переміщення об'єкту б) Зсув системи координат

а) паралельний зсув координат точки на вектор  $(m, n)$  в даній системі координат задається формулами :

$$\begin{cases} x' = x + m \\ y' = y + n \end{cases} \quad (5.5)$$

б) зсув системи координат на вектор  $(m, n)$  задається формулами :

$$\begin{cases} x' = x - m, \\ y' = y - n \end{cases} \quad (5.5')$$

### 5.3.2. Масштабування системи координат

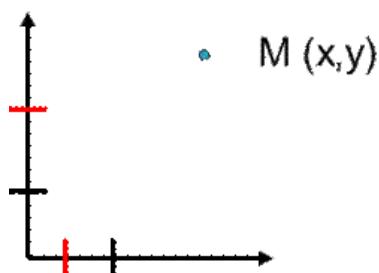


Рис.5. Масштабування системи координат

Тоді координати точки  $M$  в новій системі  $(0X'Y')$  будуть визначатися за формулами

$$\begin{cases} x' = ax \\ y' = dy \end{cases} \quad (5.6)$$

де  $a=1/i_1$ ,  $d=1/i_2$ ,  $i_1$ ,  $i_2$  – значення нових одиничних відрізків у старій системі координат (0XY). Наприклад, для рис. 3 коефіцієнти масштабування обчислюються як  $a=1:i_1(1:1/2)$ ,  $d=1:j(1:4)$ .

Аналогічно масштабування розтяг/стиск об'єкту вздовж координатних осей можна записати такими ж формулами (5.6).

Якщо  $a = d$ , то маємо пропорційне масштабування, якщо  $a \neq d$ , то масштабування – непропорційне. При  $a = d > 1$  відбувається збільшення зображення, при  $a = d < 1$  – рівномірний стиск.

При  $a = 1$ ,  $d = -1$  одержуємо дзеркальне відображення відносно осі  $x$ ,

при  $a = -1$ ,  $d = 1$  – дзеркальне відображення відносно осі  $y$ .

**5.3.3. Поворот на кут  $\varphi$  проти годинникової стрілки** зображенено на Рис.4.

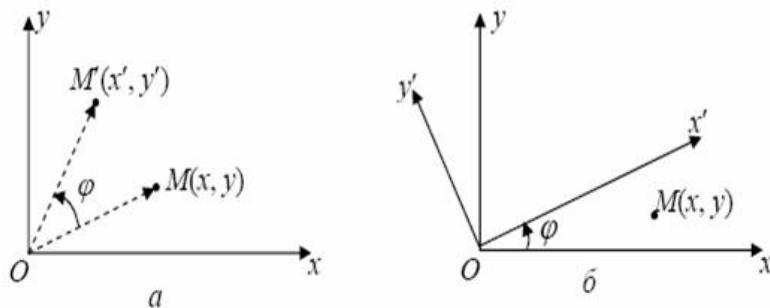


Рис. 4. а) Поворот об'єкту

б) Поворот системи координат

а) поворот точки відносно початку координат на кут  $\varphi$  проти годинникової стрілки задається формулами

$$\begin{cases} x' = x \cos \varphi - y \sin \varphi \\ y' = x \sin \varphi + y \cos \varphi \end{cases} \quad (5.7)$$

б) поворот системи координат на кут  $\varphi$  проти годинникової стрілки задається формулами

$$\begin{cases} x' = x \cos \varphi + y \sin \varphi \\ y' = -x \sin \varphi + y \cos \varphi \end{cases} \quad (5.7')$$

Для виведення формул (5.7-5.7') необхідно обчислити відповідні значення нових координат, використавши формули косинус/синус у прямокутному трикутнику (Рис.5).

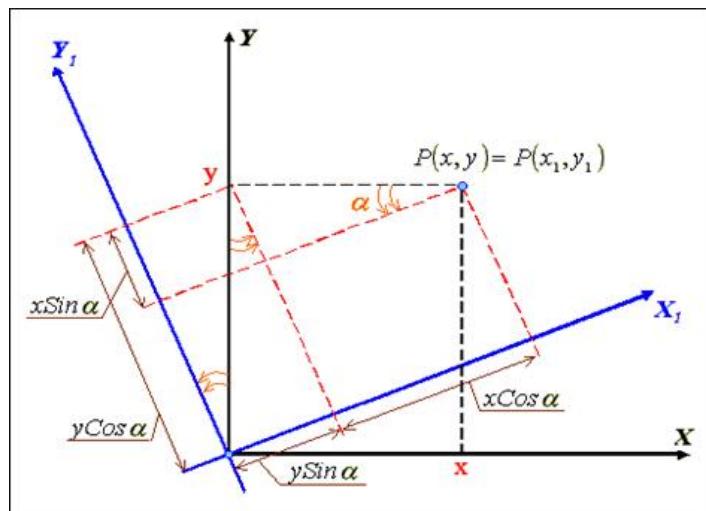


Рис.5. Поворот системи координат на кут  $\varphi$ .

Довільне афінне відображення (1) можна задати за допомогою композиції елементарних відображень (5.5) – (5.7)/ (5.5') – (5.7').

Для ефективного використання цих формул у задачах КГ переходят до **матричного запису**.

Афінне перетворення координат  $(x, y)$  описується формулами:

$$X = Ax + By + C;$$

$$Y = Dx + Ey + F,$$

де  $A, B, \dots, F$  – константи. Значення  $(X, Y)$  можна трактувати як координати в новій системі координат.

Афінне перетворення зручно записувати в матричному вигляді. Константи  $A, B, \dots, F$  утворюють матрицю перетворення, яка, будучи помноженою на матрицю-стовпець координат  $(x, y)$ , дає матрицю-стовпець  $(X, Y)$ . Однак для того, щоб врахувати константи  $C$  і  $F$ , необхідно перейти до так званих однорідних координат.

**5.4. Однорідні координати** довільної точки  $M(x, y)$  площини – це координати, які ставляться у відповідність точки  $M'(x, y, 1)$  або  $M'(hx, hy, h)$ ,  $h \neq 0$  в просторі. Фіктивна  $z$ -координата має значення скалярної константи, найчастіше  $h=1$ .

Практичне значення введення однорідних координат:

- ▶ Над однорідними координатами за спеціальними формулами можна проводити операції, і тільки на останньому етапі перед виведенням зображення на екран необхідно перейти до декартових координат.
- ▶ Застосування однорідних координат дає багато зручностей при розв'язуванні графічних задач, для пристрій, що працюють з цілими координатами. Наприклад, для точки  $M(0,5; 0,1)$  можна вибрати  $h = 10$  і працювати з однорідними цілими координатами  $M'(5; 1; 10)$ .
- ▶ Однорідні координати дозволяють записувати безмежно віддалені точки простору, уникаючи переповнення розрядної сітки комп'ютера, завдяки нормалізації чисел. За допомогою однорідних координат можна оперувати точкою, що знаходиться в нескінченності.
- ▶ зручність при заданні геометричних перетворень у матричній формі. **За допомогою однорідних координат і матриць третього порядку можна описати довільне афінне перетворення.**

Це означає, що довільне афінне перетворення може мати вигляд матричного виразу.

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

## 5.5. Матриці основних елементарних перетворень (для об'єктів)

Матриця перенесення (translation) точки на вектор  $(m, n)$  має вигляд:

$$T = T(m, n) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{pmatrix}; \quad (5.8)$$

Матриця розтягу (стиску) (dilation) відносно початку координат має вигляд:

$$D = D(a, d) = \begin{pmatrix} a & 0 & 0 \\ 0 & d & 0 \\ 0 & 0 & 1 \end{pmatrix}; \quad (5.9)$$

Матриця повороту (rotation) точки відносно початку координат у додатному напрямку

$$R = R(\varphi) = \begin{pmatrix} \cos \varphi & \sin \varphi & 0 \\ -\sin \varphi & \cos \varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}; \quad (5.10)$$

Перетворення системи координат задається відповідними оберненими матрицями. Таким чином,

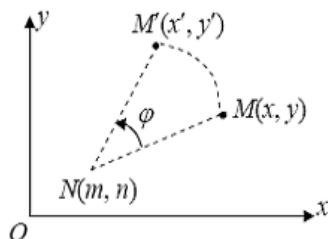
- ▶  $R(-\varphi)$  – поворот системи координат на кут  $\varphi$  у додатному напрямі;
- ▶  $T(-m, -n)$  – зсув системи координат на вектор  $(m, n)$ .

Будь-яке афінне перетворення може бути представленим як послідовність операцій з числа вказаних простіших: зсув, розтяг/стискання та повертання.

Елементи довільної матриці афінного перетворення не несуть в собі явно вираженого геометричного змісту. Для того, щоб реалізувати відображення, потрібно знайти елементи матриці, виходячи з геометричного опису перетворення. Як правило, побудову такої матриці в залежності від складності задачі розбивають на кілька етапів, що відповідають основним елементарним перетворенням.

Щоб отримати матрицю результату комбінації елементарних перетворень, потрібно перемножити відповідні матриці елементарних перетворень у правильному порядку.

Приклад. Побудувати матрицю повороту точки  $M(x, y)$  відносно довільної точки  $N(m, n)$  на кут  $\varphi$  у додатному напрямку.



**Розв'язування.** Матриця  $R$  задає поворот точки відносно початку координат. Однорідні координати дають можливість знайти матрицю повороту відносно довільної точки. У загальному випадку поворот відносно довільної точки може бути реалізований шляхом таких перетворень:

1) переміщення точки  $N(m, n)$  на вектор  $(-m, -n)$  так, щоб центр повороту сумістився з початком координат;

2) поворот точки на кут  $\varphi$  у додатному напрямку відносно початку координат;

3) переміщення одержаного результату назад так, щоб центр повороту сумістився з точкою  $N$ .

Отже, для знаходження результуючого повороту точки  $M(x, y)$  відносно точки  $N(m, n)$  потрібно перемножити матриці  $T$ ,  $R$ ,  $T$  за вказаним порядком.

$$\begin{aligned}
 (x' \ y' \ 1) &= (x \ y \ 1) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -m & -n & 1 \end{pmatrix} \begin{pmatrix} \cos \varphi & \sin \varphi & 0 \\ -\sin \varphi & \cos \varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{pmatrix} = \\
 &= (x \ y \ 1) \begin{pmatrix} \cos \varphi & \sin \varphi & 0 \\ -\sin \varphi & \cos \varphi & 0 \\ -m \cos \varphi + n \sin \varphi + m & -m \sin \varphi - n \cos \varphi + n & 1 \end{pmatrix}.
 \end{aligned}$$

◀ ▶

Остання зміна: Monday 14 December 2020 19:13 PM

◀ План лекції 10

Перейти до...

Презентація 10 ►

Ви зайшли під ім'ям Бірбан Юрій (Вихід)

КГ ПЗ ПІ-31з

Українська (uk)

Українська (uk)

English (en)

Data retention summary

Get the mobile app

# Комп'ютерна графіка

[Інформаційна панель](#) / [Мої курси](#) / [Комп'ютерна графіка](#) / [Розділ 3. Лекція 13](#) / [Лекція 6](#)

## Лекція 6

### [Лекція 6](#)

#### Лекція 6. Моделі опису поверхонь (частина 1). Аналітична, векторна полігональна моделі.

Поняття "тривимірна графіка" ("3D-графіка") може використовуватися у різних контекстах. У даному випадку мова піде про створення зображення на площині, а не в об'ємі. Тривимірні способи відображення поки що недостатньо широко поширені. Серед найпоширеніших моделей опису поверхонь - аналітична модель, векторна полігональна модель, воксельна модель, модель рівномірної сітки, модель нерівномірної сітки.

##### 6.1. Аналітична модель

Аналітичною моделлю називають опис поверхні математичними формулами. У комп'ютерній графіці можна використовувати різновиди такого описання - явна, алгебрична та параметрична форми представлення.

Явна форма аналітичної моделі має вигляд функції двох аргументів  $z=f(x,y)$ .

Алгебрична форма аналітичної моделі має вигляд рівняння  $F(x, y, z) = 0$ .

Здебільшого використовується параметрична форма опису поверхні:

$$x = F_x(s, t);$$

$$y = F_y(s, t);$$

$$z = F_z(s, t);$$

де  $s$  і  $t$  – параметри, які змінюються у визначеному діапазоні, а функції  $F_x, F_y, F_z$  визначають форму поверхні.

Перевагою параметричного описання є простота для опису поверхні, які відповідають неоднозначним функціям, є замкненими поверхнями. Опис можна провести так, що формула не буде істотно змінюватися при повертанні та масштабуванні поверхні.

Приклад. Аналітичний опис поверхні кулі.

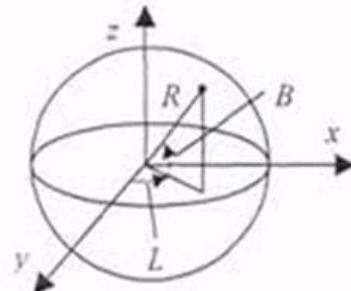


Рис.1. Зображення сфери у декартовій системі.

Явна форма має вигляд:  $z = \pm \sqrt{R^2 - x^2 - y^2}$ .

Алгебрична форма запишеться як  $x^2 + y^2 + z^2 - R^2 = 0$ .

У параметричній формі поверхня кулі задається рівняннями:

$$x = R \sin s \cos t,$$

$$y = R \sin s \sin t,$$

$$z = R \cos s.$$

Для опису складних поверхонь часто використовуються сплайнами. Сплайн – це спеціальна функція, більш за все придатна для апроксимації окремих фрагментів поверхні. Декілька сплайнів утворюють модель складної поверхні. Іншими словами, сплайн – це теж поверхня, але така, для якої можна просто

обчислити координати її точок. Здебільшого використовують кубічні сплайні. Третій степінь – найменший зі степенів, які дозволяють описати будь-яку форму, ѹ при стикуванні сплайнів можна забезпечити неперервну першу похідну – така поверхня буде без зламів в місцях стику. Прикладом сплайнів є криві Без'є.

$$P(s, t) = \sum_{i=0}^m \sum_{j=0}^n C_m^i C_n^j s^i (1-s)^{m-i} t^j (1-t)^{n-j} P_{ij},$$

де  $C_m^i, C_n^j$  – коефіцієнти біному Ньютона, параметри  $0 \leq s \leq 1, 0 \leq t \leq 1$ ,  $P_{ij}$  – опорні точки,  $(m+1) \cdot (n+1)$  – кількість точок.

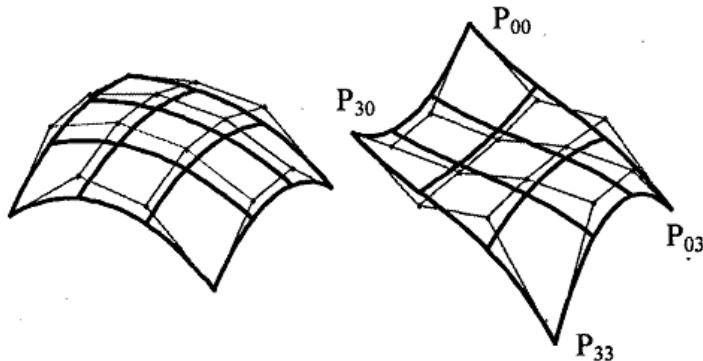


Рис. 2. Кубічні сплайнни Без'є.

6.2. Переваги та недоліки використання аналітичної моделі. Позитивна характеристика аналітичної моделі – це, в першу чергу, її економічність з точки витрат пам'яті комп'ютера. Оскільки з допомогою математичної формули описується як завгодно велика множина точок. Відповідно не виникатиме потреба в інтерполяції значень. Проте, якщо мова йде про процесорний час, то ситуація кардинально протилежна. Обчислення у формулах вимагають витрат процесорного часу, що є суттєвим недоліком у порівнянні з іншими моделями.

### 6.3. Векторна полігональна модель

Векторна полігональна модель є найбільш пошиrenoю в сучасних системах тривимірної комп'ютерної графіки. Її використовують в системах автоматизованого проектування, в комп'ютерних іграх та тренажерах, в геоінформаційних системах та ін.

Базовими примітивами, з допомогою яких конструюється будь-яке зображення є такі елементи: вершина, вектор, полілінія, полігон, полігональна поверхня.

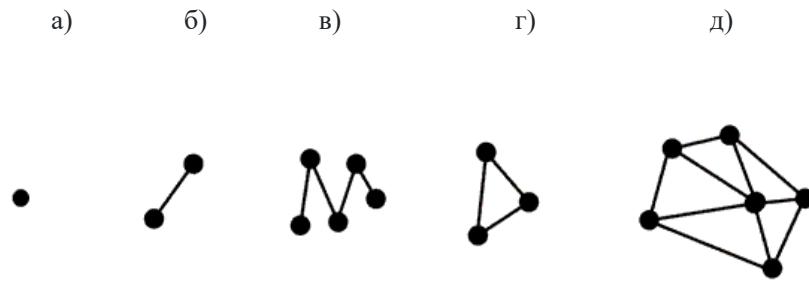


Рис.3: а) вершина; б) вектор; в) полілінія; г) полігон; д) полігональна поверхня.

Елемент «вершина» (vertex) – головний елемент опису, всі інші є похідними. При використанні тривимірної декартової системи координати вершин визначаються як  $(x_i, y_i, z_i)$ . Кожен об'єкт однозначно визначається координатами власних вершин.

Вершина може моделювати окремий точковий об'єкт, розмір якого не має значення, а також може використовуватися в якості кінцевих точок для лінійних об'єктів та полігонів. Двома вершинами задається вектор. Декілька векторів складають полілінію. Полілінія може моделювати окремий лінійний об'єкт, товщина якого не враховується, а також може представляти контур полігона. Полігон моделює плоский об'єкт. Один полігон може описувати плоску грань об'ємного об'єкта. Найпростіший полігон – це трикутник. Декілька граней складають об'ємний об'єкт у вигляді полігональної поверхні – багатогранника чи незамкненої поверхні (в літературі часто використовується назва „полігональна сітка“).

Приклад опису тривимірного об'єкта з допомогою векторної полігональної моделі, а саме – через трикутну та чотирикутну полігональну сітку представлена на рис.4.

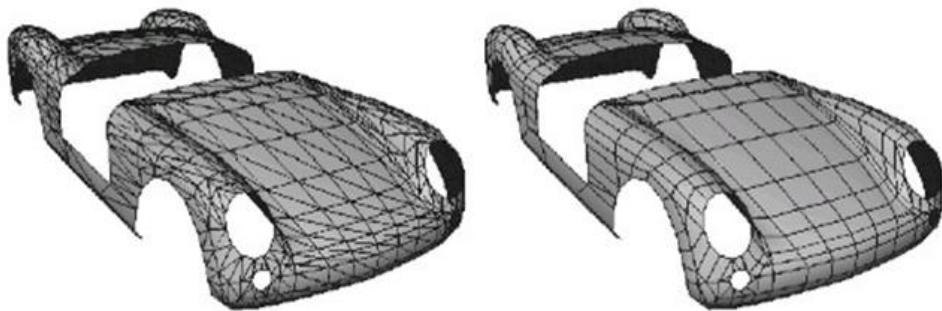


Рис. 4. Корпус автомобіля.

Крім того, що можна вибрати для опису поверхні різні базові полігони, побудова векторної полігональної моделі для полігонів одного ж і того типу може мати декілька варіантів.

**6.4. Приклад опису об'єкту через векторну полігональну модель.** Побудувати векторну полігональну модель для куба.

1) Зберігання окремо усіх граней.

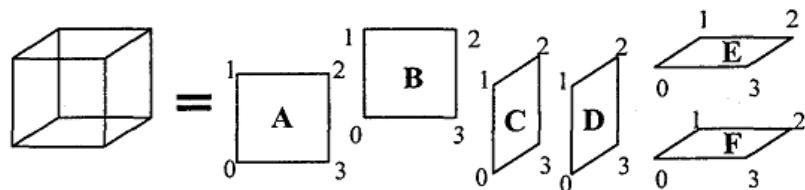


Рис.5. Представлення куба через грані.

Грань A = { (x<sub>A0</sub>, y<sub>A0</sub>, z<sub>A0</sub>), (x<sub>A1</sub>, y<sub>A1</sub>, z<sub>A1</sub>), (x<sub>A2</sub>, y<sub>A2</sub>, z<sub>A2</sub>), (x<sub>A3</sub>, y<sub>A3</sub>, z<sub>A3</sub>) }.

Грань B = { (x<sub>B0</sub>, y<sub>B0</sub>, z<sub>B0</sub>), (x<sub>B1</sub>, y<sub>B1</sub>, z<sub>B1</sub>), (x<sub>B2</sub>, y<sub>B2</sub>, z<sub>B2</sub>), (x<sub>B3</sub>, y<sub>B3</sub>, z<sub>B3</sub>) }.

Грань C = { (x<sub>C0</sub>, y<sub>C0</sub>, z<sub>C0</sub>), (x<sub>C1</sub>, y<sub>C1</sub>, z<sub>C1</sub>), (x<sub>C2</sub>, y<sub>C2</sub>, z<sub>C2</sub>), (x<sub>C3</sub>, y<sub>C3</sub>, z<sub>C3</sub>) }.

Грань D = { (x<sub>D0</sub>, y<sub>D0</sub>, z<sub>D0</sub>), (x<sub>D1</sub>, y<sub>D1</sub>, z<sub>D1</sub>), (x<sub>D2</sub>, y<sub>D2</sub>, z<sub>D2</sub>), (x<sub>D3</sub>, y<sub>D3</sub>, z<sub>D3</sub>) }.

Грань E = { (x<sub>E0</sub>, y<sub>E0</sub>, z<sub>E0</sub>), (x<sub>E1</sub>, y<sub>E1</sub>, z<sub>E1</sub>), (x<sub>E2</sub>, y<sub>E2</sub>, z<sub>E2</sub>), (x<sub>E3</sub>, y<sub>E3</sub>, z<sub>E3</sub>) }.

Грань F = { (x<sub>F0</sub>, y<sub>F0</sub>, z<sub>F0</sub>), (x<sub>F1</sub>, y<sub>F1</sub>, z<sub>F1</sub>), (x<sub>F2</sub>, y<sub>F2</sub>, z<sub>F2</sub>), (x<sub>F3</sub>, y<sub>F3</sub>, z<sub>F3</sub>) }.

Програмно реалізувати такий варіант векторної полігональної моделі можна різними способами: масивами, структурами, класом.

Незважаючи на це, витрати пам'яті будуть визначатися як

$$\Pi_1 = 6 \times 4 \times 3 \times P_8,$$

де  $P_8$  – розрядність числа для задання координат вершини.

Очевидно, що такий опис збитковий, адже кожна вершина записується тричі.

2) Опис через вершини без повтору.

Для цього потрібно пронумерувати вершини, а грані описуватимуться через номери (індекси) вершин.

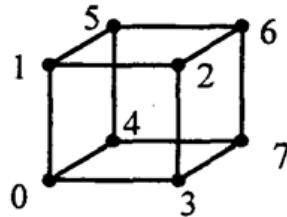
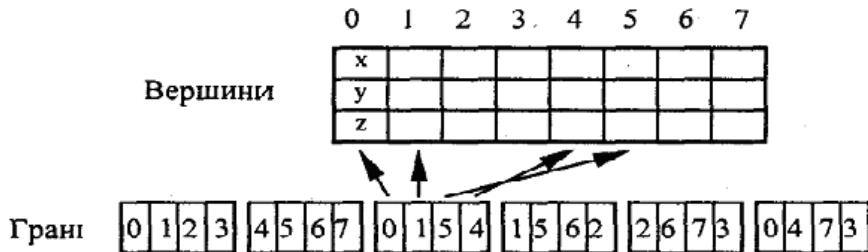


Рис. 6. Представлення куба з індексами вершин.

У такому випадку куб описеться такими структурами:



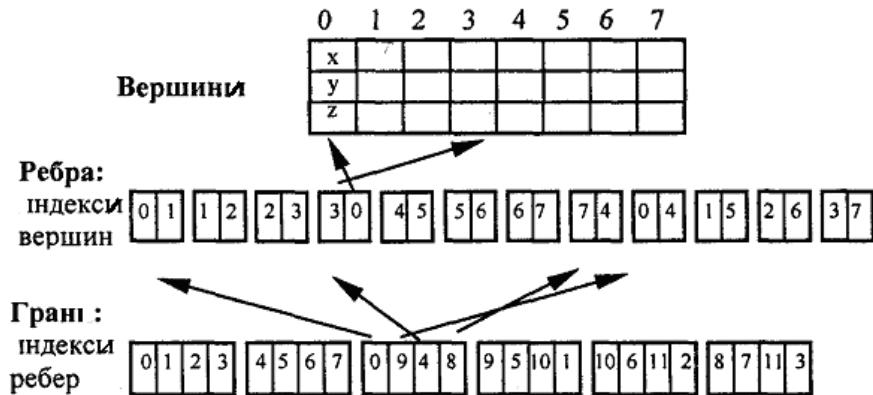
Витрати пам'яті обчислюються як:

$$P_2 = 8 \times 3 \times P_e + 6 \times 4 \times P_i$$

де  $P_e$  – розрядність числа для задання координати вершини,  $P_i$  – розрядність індексу вершин.

3) Лінійно-вузлова модель.

Для такого способу використовуються, крім вершин, ще ребра. Ребра задаються через індекси.



Витрати пам'яті обчислюються як:

$$P_3 = 8 \times 3 \times P_e + 12 \times 2 \times P_{i.e} + 6 \times 4 \times P_{i.p}$$

де  $P_e$  – розрядність числа для задання координати вершини,  $P_{i.e}$  – розрядність індексу вершин,  $P_{i.p}$  – розрядність індексу ребер.

Очевидно, що другий спосіб є найекономічніший. Якщо аналізувати швидкість виведення, то третій спосіб є найефективнішим. Розглядаючи реалізацію різних операцій для куба, можна вибрати той чи інший спосіб представлення куба. Наприклад, якщо змістити вершину грані, то перший спосіб без додаткових алгоритмів буде неефективним. В той же час операція роз'єму граней є найефективнішою для першого способу.

Обираючи той чи інший спосіб опису поверхні, розробник графічної системи має проаналізувати переваги та недоліки, які проявляються при виконанні різних топологічних операцій, а також витрати обчислювальних ресурсів.

#### 6.5. Переваги та недоліки векторної полігональної моделі.

Позитивні риси векторної полігональної моделі:

- зручність масштабування об'єктів; при збільшенні чи зменшенні об'єкти виглядають більш якісно, ніж при растрових моделях описання; діапазон масштабування визначається точністю апроксимації й розрядністю чисел для представлення координат вершин;
- невеликий обсяг даних для описання простих поверхонь, які адекватно апроксимуються плоскими гранями;
- необхідність обчислювати тільки координати вершин при перетвореннях систем координат або при пересуванні об'єктів;
- апаратна підтримка багатьох операцій в сучасних графічних відеосистемах, яка обумовлює достатню швидкість при анімації.

Недоліки полігональної моделі:

- складні алгоритми візуалізації для створення реалістичних зображень;
- складні алгоритми виконання топологічних операцій, таких, наприклад, як розрізи;
- апроксимація плоскими гранями призводить до похибок в моделюванні.

[◀ План лекції 6](#)[Перейти до...](#)[Список питань для самоперевірки 6 ►](#)

Ви зайшли під ім'ям Бірбан Юрій (Вихід)

КГ ПЗ ПІ-31з

Українська (uk)

Українська (uk)

English (en)

[Data retention summary](#)

[Get the mobile app](#)

# Комп'ютерна графіка

[Інформаційна панель](#) / [Мої курси](#) / [Комп'ютерна графіка](#) / [Розділ 3. Лекція 14](#) / [Лекція 7](#)

## Лекція 7

[Лекція 7](#). Моделі опису поверхонь (частина 2). Воксельна модель. Сіткова модель.

### 7.1. Воксельна модель.

Voxel – volume element, елемент об'єму, аналогічний пікселю, але є тривимірний. Цей термін утворено зі слів: об'ємний (англ. *volumetric*) і піксел (англ. *pixel*). Кожен воксель також має свій колір та прозорість. Повна прозорість вокселя означає порожнину відповідної точки об'єма. Воксельна модель – це тривимірний растр. Чим більше вокselів у визначеному об'ємі та менше їх розмір, тим точніше моделюються тривимірні об'єкти – збільшується роздільна здатність.

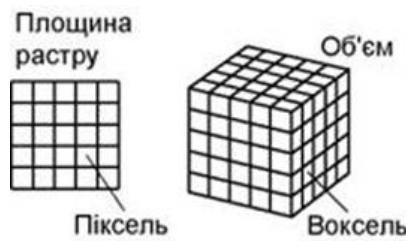


Рис. 7.1. Піксель і воксель

Для сучасної комп'ютерної графіки воксельний метод вважається одним з перспективних. Його використовують в комп'ютерних системах для медицини. Наприклад, при скануванні томографом виходять зображення зрізів об'єкта, які потім об'єднуються у вигляді об'ємної моделі для подальшого аналізу.

Медичні пристрої видають пошарово інформацію при скануванні. А потім створюється остаточне зображення. На Рис.7.2. використано 150 шарів.

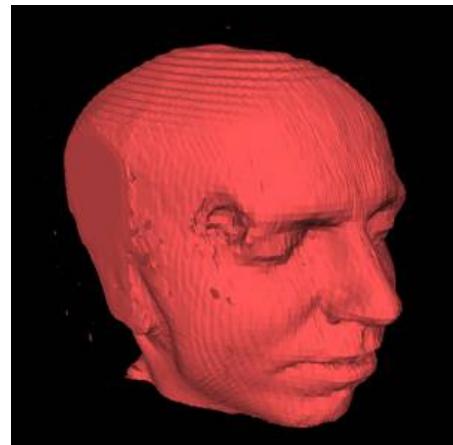


Рис. 7.2. Зображення результатів магнітно-резонансної томографії.

Воксельний метод використовується в геології, сейсмології, в комп'ютерних іграх. Воксели також використовуються для графічних пристрій, які створюють дійсно об'ємні зображення.

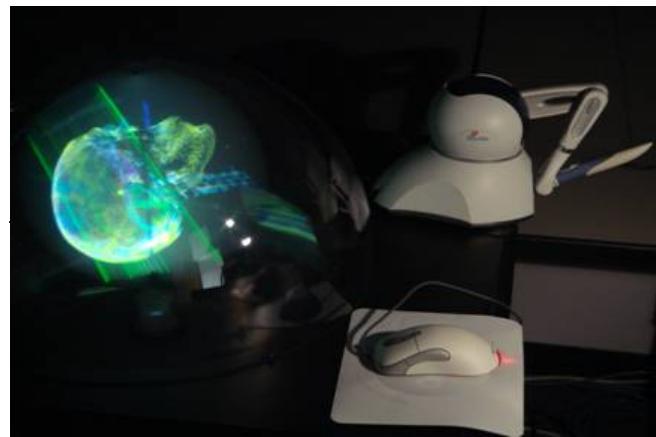


Рис. 7.3. Візуалізація воксельної моделі на тримірному дисплеї Perspecta volumetric

Позитивні риси воксельної моделі:

- дозволяє достатньо просто описувати складні об'єкти та сцени;
- проста процедура відображення об'ємних сцен;
- просте виконання топологічних операцій над окремими об'єктами та сценою в цілому.

Наприклад, просто виконується показ розрізу – для цього відповідні воксели можна зробити прозорими.

Недоліки воксельної моделі:

- велика кількість інформації, необхідної для представлення об'ємних даних;
- значні витрати пам'яті обмежують роздільну здатність, точність моделювання;
- велика кількість вокселів обумовлює незначну швидкість створення зображень об'ємних сцен;
- як і для будь-якого раstra, виникають проблеми збільшення чи зменшення зображення (наприклад, при збільшенні погіршується роздільна здатність зображення).

## 7.2. Рівномірна сітка

Рівномірна сітка – це різновид сіткової моделі, коли сітка складається з рівновіддалених між собою вузлів.

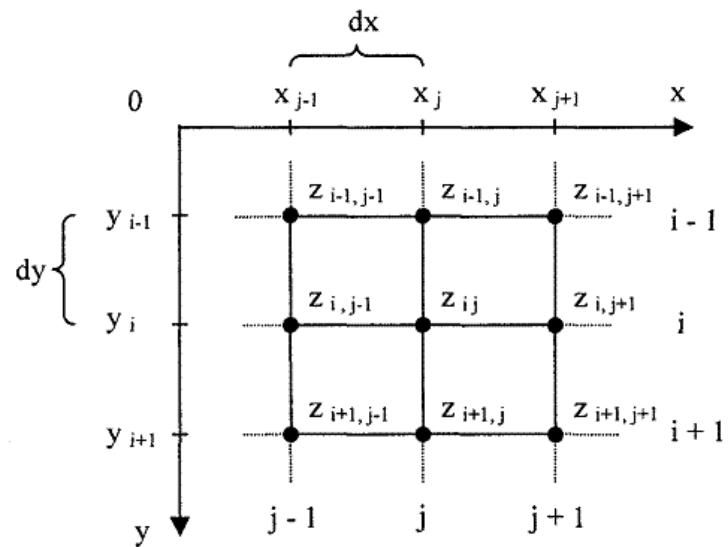


Рис. 7.4. Схема рівномірної сітки.

Кожному вузлу сітки з індексами  $(i, j)$  приписується значення висоти  $z_{ij}$ , тобто  $z_{ij}=f(x_i, y_j)$ . Індексам  $(i, j)$  відповідають визначені значення координат  $(x, y)$ . Відстань між вузлами є однаковою:  $dx$  по осі  $x$  та  $dy$  по осі  $y$ .

Фактично, така модель – двомірний масив, растр, матриця, кожний елемент якої зберігає значення висоти.

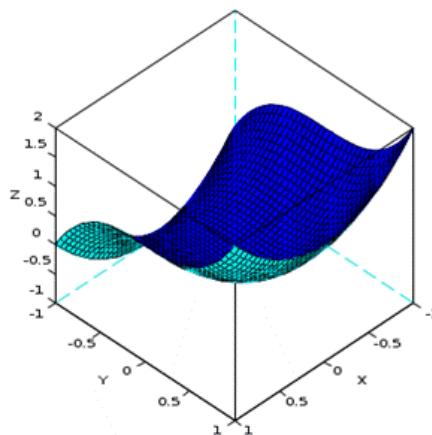


Рис. 7.5. Приклад тримірної сітки.

Не кожна поверхня може бути представлена такою моделлю. Якщо в кожному вузлі записується тільки одне значення висоти, то це означає, що поверхня описується однозначною функцією  $z = f(x, y)$ . Інакше кажучи, це така поверхня, яку будь-яка вертикаль перетинає тільки один раз. Не можуть моделюватися також вертикальні грані. Необхідно відмітити, що для сітки не обов'язково використовувати тільки декартові координати. Наприклад, для того щоб описати поверхню кулі однозначною функцією, можна використовувати полярні координати. Рівномірна сітка часто використовується для описання рельєфу земної поверхні.

Якщо необхідно визначити значення в точці, що не співпадає з вузлом сітки (Рис.7.5), то виконують наближені обчислення одним з відомих методів інтерполяції.

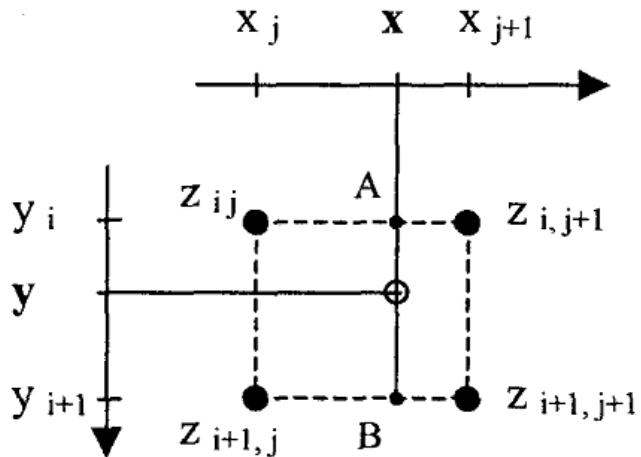


Рис. 7.5. Точка поза вузлами рівномірної сітки.

#### Позитивні риси рівномірної сітки:

- простота опису поверхонь;
- можливість швидко дізнатися висоту у вузлі сітки (непотрібно проводити обчислення, значення задано явно)

#### Недоліки рівномірної сітки:

- поверхні, які відповідають неоднозначній функції висоти у вузлах сітки, не можуть моделюватися;
- для обчислення значення в будь-якій точці поверхні, що не належить сітці, потрібно виконувати інтерполяцією;
- для описання складних поверхонь необхідна велика кількість вузлів, яка може бути обмежена обсягом пам'яті комп'ютера;
- описання окремих типів поверхонь може бути значно складнішим, ніж в інших моделях (наприклад, багатогранна поверхня вимагає додаткового обсягу даних для описання порівняно з полігональною моделлю).

## 7.2. Нерівномірна сітка

Нерівномірною сіткою є модель опису поверхні у вигляді множини окремих точок  $\{(x_0, y_0, z_0), (x_1, y_1, z_1), \dots, (x_{n-1}, y_{n-1}, z_{n-1})\}$ , які належать поверхні і не є логічно пов'язаними між собою. Ці точки можуть бути отримані, наприклад, в результаті вимірювання поверхні якого-небудь об'єкта за допомогою певного обладнання.

Таку модель можна вважати узагальненням для деяких розглянутих вище моделей. Наприклад, векторна полігональна модель та рівномірна сітка можуть вважатися різновидами нерівномірної сітки.

Нерівномірність задання опорних точок ускладнює визначення координат для інших точок поверхні, які не співпадають з опорними точками.

Цю задачу можна вирішити декількома способами, у тому числі тріангуляцією.

Описання поверхні трикутними гранями можна вважати різновидом векторної полігональної моделі. В англомовній літературі для неї зустрічається така назва: TIN (Triangulated Irregular Network). Найвідомішим описаним методом тріангуляції є метод Делоне, який базується на дуальному графі розбиття Вороного (український і російський математик). Після тріангуляції (лат. triangulatio = покриття трикутниками) отримуємо полігональну поверхню, відображення якої зробити вже достатньо просто.

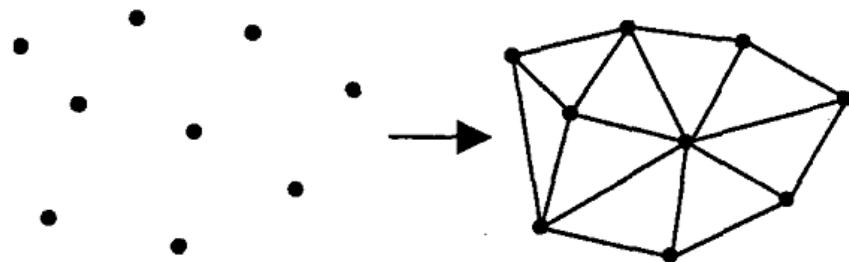


Рис.7.6. Тріангуляція нерівномірної сітки.

Розглянемо ще один з варіантів опису поверхонь – ізолінії висоти. Будь – яка ізолінія складається з точок, які представляють одне числове значення певного показника, в даному випадку – значення висоти.

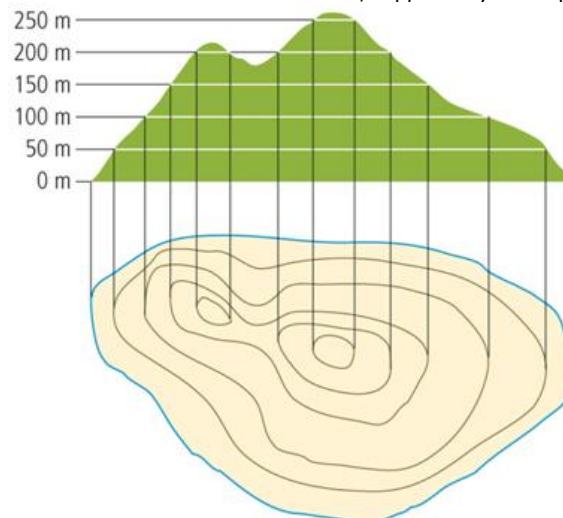


Рис. 7.7. Представлення поверхні за ізолініями.

Ізолінії висоти також можна уявити собі як контури розрізу поверхні горизонтальними площинами (тому для ізоліній висоти часто застосовується назва „горизонталі“).

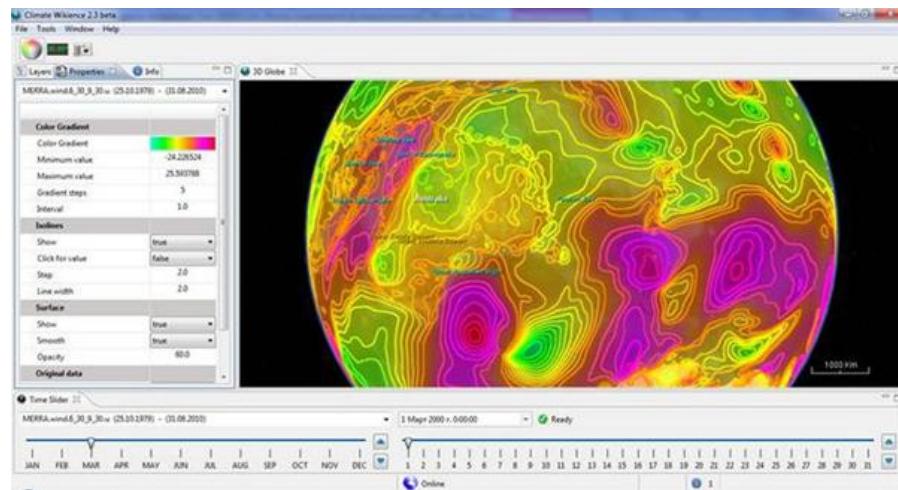


Рис. 7.8. Приклад відображення карти ізоліній у системі Climate Wikience

Опис поверхні ізолініями висоти часто використовується в картографії. Звичайно, для опису поверхні можна використовувати не тільки ізолінії висоти, але й інші ізолінії, наприклад х- чи у-ізолінії. У комп'ютерних системах ізолінії часто описуються векторно - полілініями. Використовуються також ізолінії у вигляді сплайнів кривих.

Точки, які складають ізолінії, їх окрім опорні точки розташовуються нерівномірно. Це ускладнює розрахунок координат точок поверхні. В графічних комп'ютерних системах для виконання багатьох операцій, в першу чергу – для показу поверхні, здебільшого необхідно перетворити описання поверхні в іншу форму. Перетворення ізоліній в полігональну модель також виконується методами тріангуляції (тут алгоритми тріангуляції більш складні, ніж для тріангуляції окремих точок). Для перетворення нерівномірної сітки в рівномірну використовують спеціальну інтерполяцію.

Позитивні риси нерівномірної сітки:

- використання окремих опорних точок, найбільш важливих для заданої форми поверхні, обумовлює менший обсяг інформації порівняно з іншими моделями, наприклад, з рівномірною сіткою;
- наочність показу рельєфу поверхні ізолініями на мапах, планах.

Недоліки:

- неможливість чи складність виконання багатьох операцій над поверхнями;
- складні алгоритми перетворення в інші форми опису поверхонь.

Остання зміна: Wednesday 28 May 2014 00:44 AM

[◀ План лекції 7](#)

Перейти до...

[Список питань для самоперевірки 7 ►](#)

Ви зайшли під ім'ям Бірбан Юрій (Вихід)

КГ ПЗ ПІ-31з

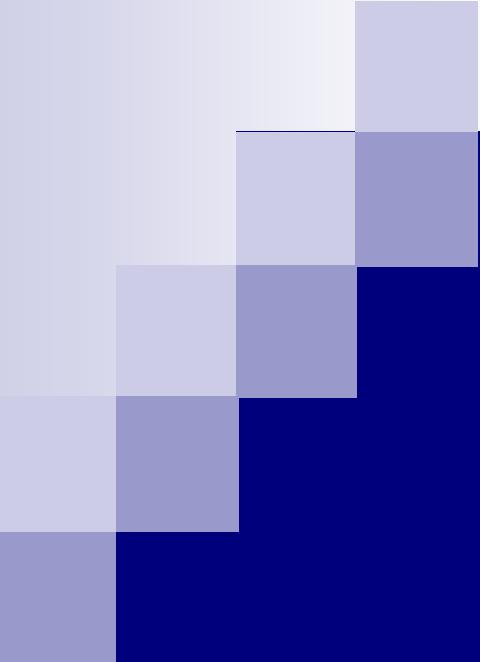
Українська (uk)

Українська (uk)

English (en)

Data retention summary

Get the mobile app

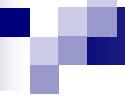


# Комп'ютерна графіка

Лектор Левус Є.В.  
Керівники лабораторних занять  
Левус Є.В.  
Джумеля Е.А.  
Коцун В.І.  
5 кредитів, ЗАЛІК

# Оцінювання

- Лабораторна робота  
(комплексне завдання,  
виконується командою з 2 осіб  
або одноосібно) - **30 б**
- Контрольні заходи **70 б**



# Методичні матеріали

- НМК у ВНС

# Лекція 1. Вступ до КГ

- *Предмет і галузі застосування комп'ютерної графіки*
- *Коротка історія розвитку комп'ютерної графіки*
- *Дисплейні технології підтримки комп'ютерної графіки*

# Перспективні комп'ютерні графічні технології:

- 3D-графіка на базі спеціалізованих 3D-карт та графічних робочих станцій, програмування графіки на базі інструментальних засобів візуальних компонентних об'єктно-орієнтованих середовищ мов програмування високого рівня;

- швидкий синтез графічних зображень на базі мов програмування низького рівня;
- каркасне, поверхневе та твердотільне 3D-моделювання та реалістична візуалізація;
- анімаційні графічні технології;
- мультимедійні та гіpermедійні графічні технології;

- інтерактивне відео та віртуальна реальність;
- видавничі графічні комп'ютерні технології;
- технології комп'ютерної графіки на Web-сторінках глобальної мережі Internet.

# Визначення

**Комп'ютерна графіка** - це спеціальна галузь комп'ютерних дисциплін, що вивчає методи і засоби створення та обробки зображень за допомогою програмно-апаратних обчислювальних комплексів.

Сфера застосування КГ включає  
чотири основні області

1. Відображення інформації
2. Проектування
3. Моделювання (імітація)
4. Сфера розваг
5. Графічний інтерфейс користувача

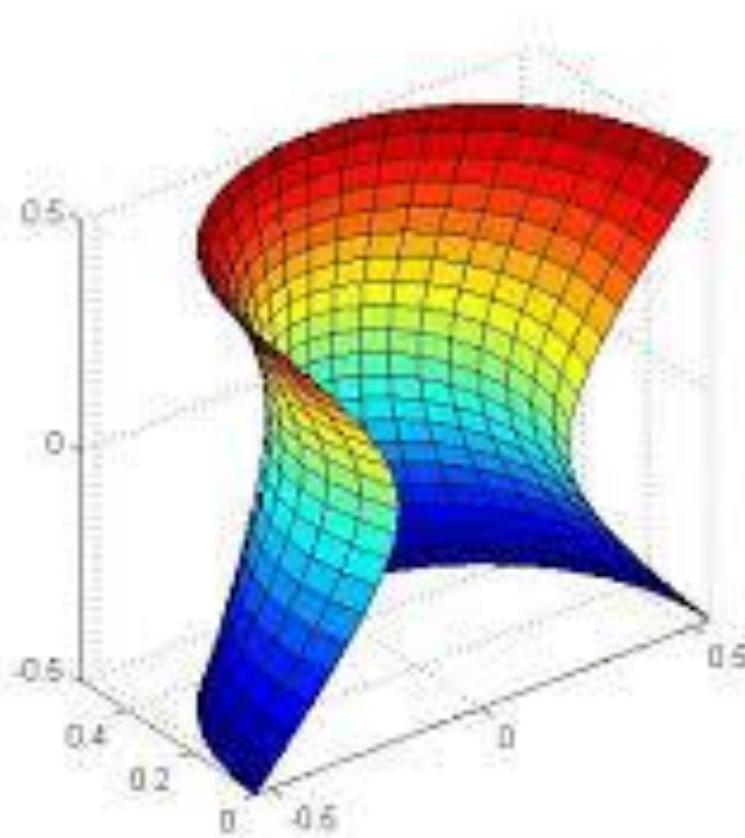
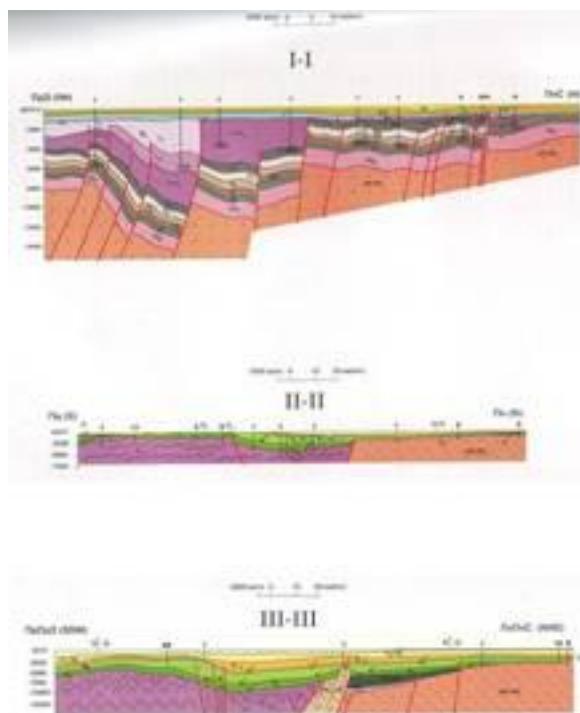
# 1. Відображення інформації

- Проблема подання накопиченої інформації найкраще може бути вирішена за допомогою графічного відображення.

Наприклад, дані про кліматичні зміни за тривалий період, динаміка популяцій тваринного світу, екологічний стан різних регіонів

# Сучасна наука використовує графічне представлення інформації

- візуалізація результатів експериментів
- аналіз даних натурних спостережень
- наочні картини результатів  
математичного моделювання процесів і  
явищ

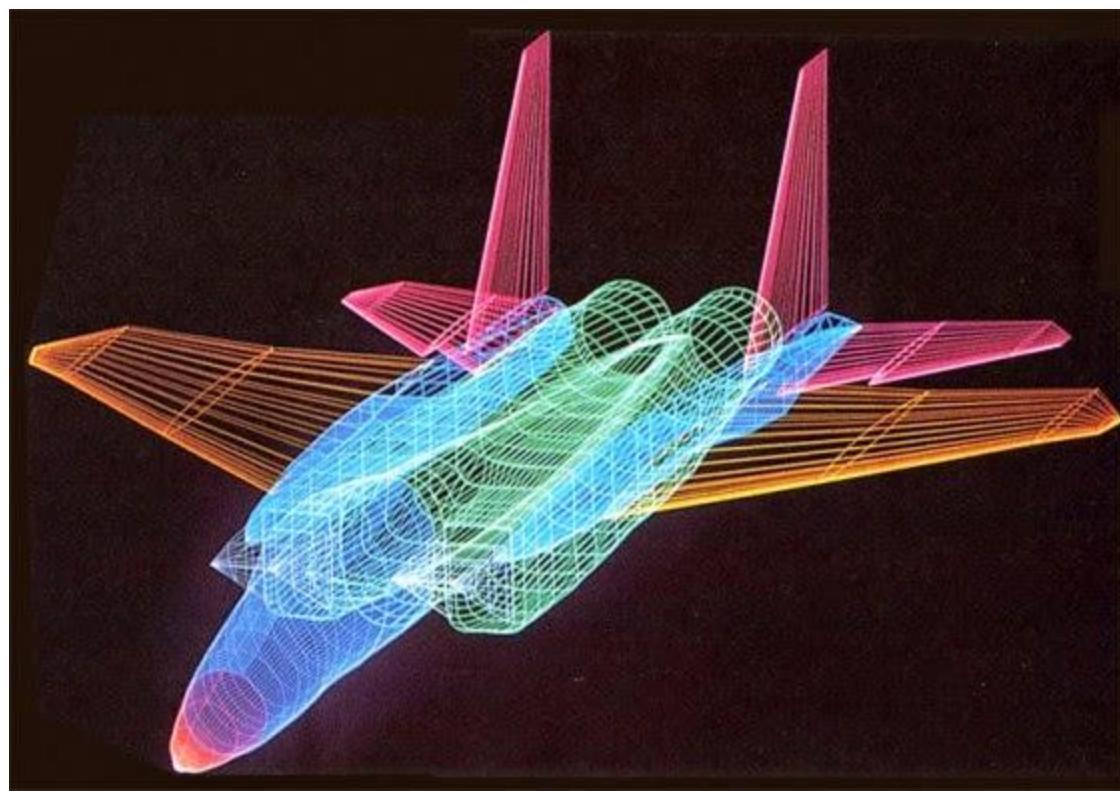


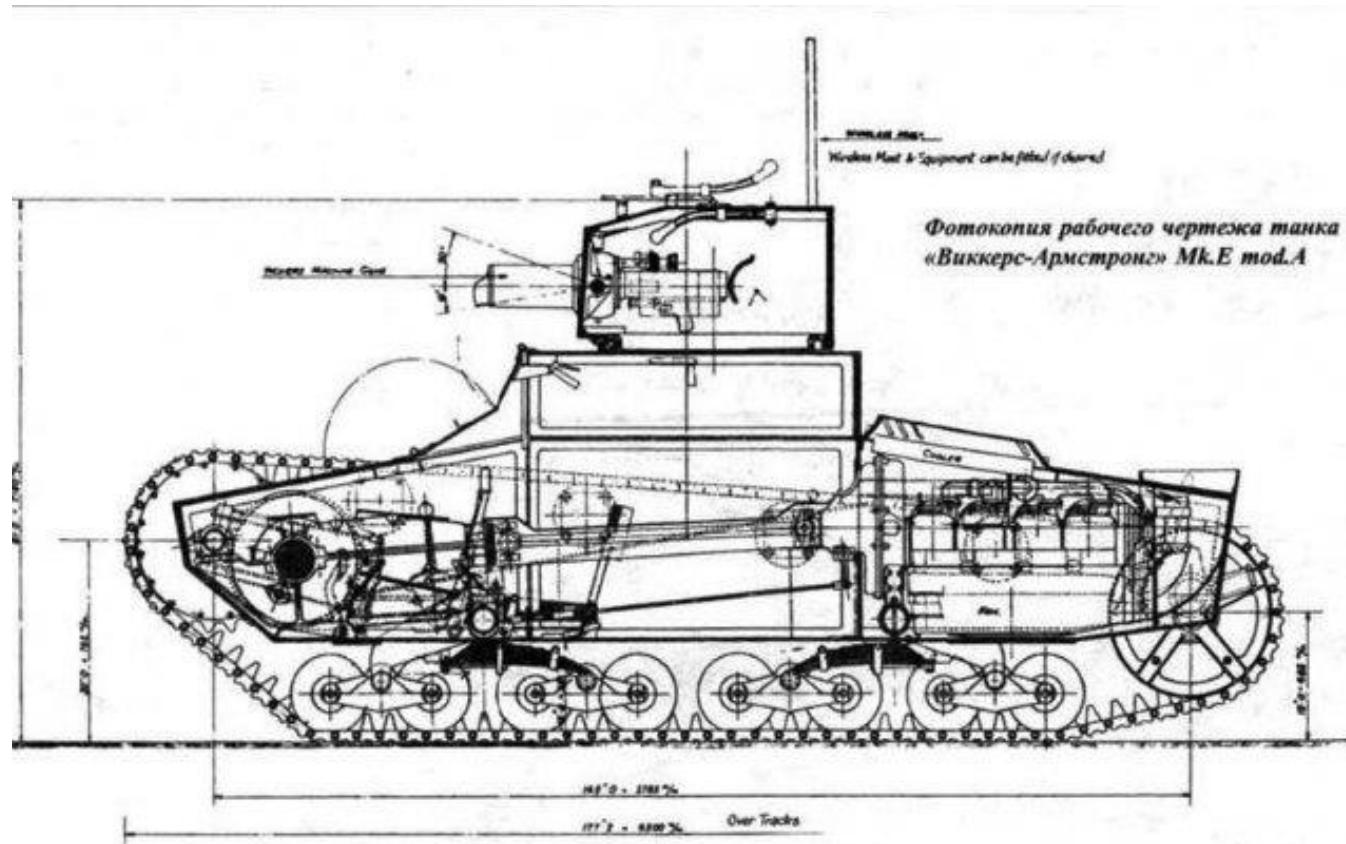
# Наприклад

- В геології в результаті обробки тривимірних натурних даних можна отримати геометрію пластів, що залягають на великій глибині.
- У медицині широко використовуються методи діагностики, що використовують комп'ютерну візуалізацію внутрішніх органів людини. Томографія, енцефалограми ...

## 2. Проектування (дизайн)

- У будівництві та техніці креслення є основою проектування нових споруд або виробів.
- Процес проектування є ітеративним, тобто конструктор перебирає безліч варіантів з метою вибору оптимального за певними параметрами.
- Існують розвинені програмні засоби автоматизації проектно-конструкторських робіт (САПР), що дозволяють швидко створювати креслення об'єктів, виконувати розрахунки на міцність і т.п.





Фотокопия рабочего чертежа танка  
«Виккерс-Армстронг» Mk.E mod.A

### 3. Імітація

- Тренажери
- Віртуальна реальність
- Доповнююча реальність





## 5. Кіноіндустрія, ігри







# 5. Графічний інтерфейс користувача

- Розділ “Людино-машинна взаємодія”

# Основні напрями КГ:

- **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, сприяюча пізнанню).

# 1) Образотворча комп'ютерна графіка

своїм предметом має **синтезовані зображення**.

Основні види завдань, які вона вирішує, зводяться до таких:

- побудова моделі об'єкта й формування зображення;
- перетворення моделі й зображення;
- ідентифікація об'єкта й одержання необхідної інформації.

## 2) 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 й класифікації властивостей об'єктів (при обробці аерокосмічних знімків, уведені креслень, у системах навігації, виявлення і наведення).

### 3) 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.

## 4) Когнітивна комп'ютерна графіка

- тільки формується новий напрям, поки ще недостатньо чітко окреслений.

Це КГ для наукових абстракцій, що сприяє народженню нового наукового знання. Технічною основою для неї є потужні комп'ютери і високопродуктивні засоби візуалізації.

Одним з найбільш ранніх прикладів використання когнітивної комп'ютерної графіки є робота Ч.Сtraуса "Несподіване застосування ЕОМ в чистій математиці" (TIIER, т. 62, № 4, 1974, с.96-99).

У ній показано, як для аналізу складних алгебраїчних кривих використовується "п-мірна" дошка на основі графічного терміналу. Користуючись пристроями введення, математик може легко управляти поточними значеннями параметрів, "поглиблюючи тим самим своє розуміння ролі варіацій цих параметрів". В результаті отримано "кілька нових теорем і визначені напрямки подальших досліджень".

# Історія КГ

- Перша офіційно визнана спроба використання дисплея для виводу зображення з ЕОМ - **Массачусетський технологічний університет, машина Whirlwind-I в 1950 р.**
- Термін "**комп'ютерна графіка**" вперше використав в 1960 р. співробітник компанії Boeing У. Феттер.

На початку свого розвитку комп'ютерну графіку розглядали, як частину системного програмування для ЕОМ чи один з розділів систем автоматизованого проектування (САПР).

- Перше реальне застосування КГ пов'язують з ім'ям Дж. Уїтні. Він займався кіновиробництвом в 50-60-х роках і вперше використовував комп'ютер для створення титрів до кінофільму.
- Наступний крок у розвитку КГ- робота **Айвена Сазерленда**, який в 1961р створив програму малювання, названу ним **Sketchpad** (альбом для малювання). Програма використовувала світлове перо для малювання найпростіших фігур на екрані. Отримані картинки можна було зберігати і відновлювати. У цій програмі було розширено коло основних графічних примітивів, зокрема, крім ліній і точок був введений прямокутник, який задавався своїми розмірами і розташуванням.

Спочатку комп'ютерна графіка була **векторною**, тобто зображення формувалося з тонких ліній (а не точок). Ця особливість була пов'язана з технічною реалізацією комп'ютерних дисплеїв.

# Перша комп'ютерна відеогра

Spacewar ("Зоряна війна") у 1961 р.  
створена студентом С. Расселом

# Великі корпорації підтримують КГ

“Дженерал моторз”, “Дженерал електрик”,  
приступили до комерційних розробок в  
області КГ.

Це було використання графічної системи  
з розподіленим часом для багатьох  
етапів **проектування автомобілів**.

# Центр досліджень в області КГ

університет штату Юта (завдяки **Д.Евансу і А.Сазерленду**).

Учнем Сазерленда став **Е.Кетмул**, майбутній творець алгоритму видалення невидимих поверхонь з використанням Z-буфера (1978). Тут же працювали **Дж.Варнок**, автор алгоритму видалення невидимих граней на основі розбивки області (1969) та засновник Adobe System (1982), **Дж.Кларка**, майбутній засновник компанії Silicon Graphics (1982).

**Всі ці дослідники дуже сильно розвинули алгоритмічну сторону комп'ютерної графіки.**

В Україні перша інтерактивна графічна система автоматизованого проектування електронних схем була розроблена Київським політехнічним інститутом і Науково-дослідним інститутом автоматизованих систем керування і плануванні в будівництві (НДІАСК, Київ) і демонструвалася в 1969р.

# Справжній широкий розвиток КГ

пов'язаний з появою персональних комп'ютерів **«Macintosh» (MAC)** фірми Apple (спеціально для потреб поліграфії).

Саме для платформи MAC почали з'являтися перші спеціалізовані операційні системи та графічні редактори.

Але сталося так, що справжніми «масовими» комп'ютерами стали комп'ютери класу IBM/PC (PC). А більшість графічних оболонок та редакторів почали відтворюватися на базі графічного досвіду MAC та перекладені для комп'ютерів PC.

Так з'явилася славнозвісна операційна система Windows, а також дуже велика кількість графічних пакетів, програм та редакторів (наприклад: QuickTime, Page Maker, майже всі продукти корпорації Adobe та багато інших).

# У середині 1970-х років

КГ продовжує розвиватися в бік все більшої реалістичності зображень.

Е.Кетмул, Фонг, Ф.Кроу, Дж.Брезенхем створюють ефективні растрові алгоритми.

# У 1980-ті роки

з'являється цілий ряд компаній, що займаються прикладними розробками в області КГ.

***Silicon Graphics, Adobe System...***

У 1982 році на екрані кінотеатрів вийшов фільм **“Трон”**, в якому вперше використовувалися кадри, синтезовані на комп'ютері.

Ще в 1979 році Джордж Лукас, глава фірми “Lucasfilm” і творець серіалу “Зоряні війни”, організував в своїй фірмі відділ, який займався впровадженням останніх досягнень КГ в кіновиробництво.



# У 1990-ті роки

Кіноіндустрія, Інтернет...

- Індустрія розваг
- Освіта
- Торгівля
- ...

# Етапи розвитку

- У 1960-1970-і роки вона формувалася як **наукова дисципліна**. У цей час розроблялися основні методи і алгоритми: відсікання, растроva розгортка графічних примітивів, зафарбування візерунками, реалістичне зображення просторових сцен (видалення невидимих ліній і граней, трасування променів, випромінюють поверхні), моделювання освітленості.
- У 1980-ті графіка розвивається більш як **прикладна дисципліна**. Розробляються методи її застосування в самих різних областях людської діяльності.
- У 1990-ті роки методи комп'ютерної графіки стають основним засобом організації діалогу "людина-комп'ютер" і залишаються такими по теперішній час.

# Дисплейні технології підтримки КГ

Розвиток КГ обумовлений розвитком технічних засобів ТІ підтримки.

Перш за все це пристрой виведення, якими є дисплеї.

Розрізняють декілька типів дисплеїв, що використовують електронно-променеву трубку (МИНУЛЕ), а також дисплеї на рідкокристалічних індикаторах (ТЕПЕРІШНЄ) і інші їх види (МАЙБУТНЄ).

Ми розглянемо їх функціональні особливості, а не внутрішню будову.

Дисплейна графіка на першому етапі свого розвитку використовувала електронно-променеві трубки (ЕПТ) з довільним скануванням променя для виведення у вигляді зображення інформації з ЕОМ.

З експерименту в МТІ почався етап розвитку **векторних дисплеїв** (дисплеїв з довільним скануванням променя).

# Найпростіший із пристройів на ЕПТ

є дисплей на запам'ятовуючій трубці з прямим копіюванням зображення.

Запам'ятовуюча трубка має властивість тривалого часу післясвітіння: зображення залишається видимим протягом тривалого часу (до однієї години). Складність зображення практично не обмежена. Стирання відбувається шляхом подання на всю трубку спеціального напруги, при якому світіння зникає, і ця процедура займає приблизно 0,5 с. Тому зображення, отримані на екрані, можна стерти частково, а, анімація на такому дисплеї неможливі.

Його досить легко програмувати, але рівень інтерактивності в нього нижче, ніж у дисплеїв інших типів.

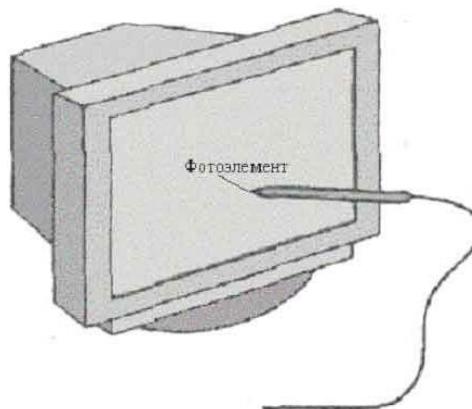
# Векторні дисплеї з регенерацією зображення

- це наступний тип векторної технології.

Для того щоб зображення було постійно видимим, доводиться його "перемальовувати" (регенерувати зображення) 50 або 25 разів на секунду. Необхідність регенерації зображення вимагає збереження його опису в спеціально виділеній пам'яті (пам'ять регенерації). Сам опис зображення називається дисплейним файлом. Зрозуміло, що такий дисплей вимагає досить швидкого процесора для обробки дисплейного файлу і управління переміщенням променя по екрану.

# Відмінна риса векторних дисплеїв

є можливість безпосереднього графічного діалогу, що полягає в простій вказівці за допомогою світлового пера об'єктів на екрані (ліній, символів і т.д.).



# Дисплей з растровим скануванням променя

з середини 1970-х років одержують переважне поширення.

Растрове пристрій можна розглядати як матрицю дискретних точок, кожна з яких може бути підсвічена.

Відрізок будується за допомогою послідовності точок, апроксимуючих ідеальну траєкторію відрізка.

Відрізки можуть виглядати як послідовність "сходинок" (ступінчастий ефект).

# Дисплей на рідкокристалічних індикаторах

є растровими пристроями (іх теж можна представити як матрицю елементів - рідких кристалів). LCD-монітори – домінуючі сьогодні.

Мають найменші габарити і енергоспоживання, тому широко використовуються в портативних комп'ютерах.

Хоча історично такий спосіб виводу зображення з'явився раніше, ніж растровий дисплей з ЕПТ, але швидко розвиватися він почав значно пізніше.

# Основними параметрами монітора є

**розмір екрану, розмір зерна, швидкість оновлення зображень (частота кадрової розгортки), ступінь плоскості екрану (вища реалістичність зображення, менше відблисків).**

Для професійної роботи з графікою використовують монітори з діагоналями 21 чи 22 дюйми.

- **Зерно** – це мінімальна точка люмінофору (піксель), яка вимірюється в десятих частках міліметра.
- Близьким параметром до розміру зерна є **роздільна здатність**. Показує, скільки пікселів може вміститися на екрані. Роздільну здатність описують два числа, кількість точок по горизонталі, кількість горизонтальних ліній, наприклад  $320 \times 200$ .

- Іншою характеристикою моніторів є **частота вертикальної розгортки**, тобто частота оновлення кадрів, яку ще часто називають частотою регенерації. Для комфортної роботи необхідно, щоб частота вертикальної розгортки складала не менше 85 Гц.

Менша частота шкідлива для очей (миготіння на екрані швидко стомлює очі). Якщо частота вертикальної розгортки вища за 110 Гц, то людина вже не помічає ніякого мерехтіння зображення.

■ **Горизонтальна частота розгортки** показує кількість ліній, яку можна вивести на екран за 1 с. Для сучасних моніторів вона складає від 15 кГц до 100 кГц.

Параметри моніторів пов'язані між собою, наприклад, якщо зменшити роздільну здатність, то зростає частота розгортки. Істотними характеристиками для роботи з моніторами є **ергономічність**.

**Це сукупність властивостей, які характеризують пристосованість монітору до взаємодії з користувачем з урахуванням фізико-біологічних особливостей людини.**

# Типи дисплеїв (вивчаємо самостійно !!!)

- PDP
- IPS
- TFT
- AMOLED
- LCD

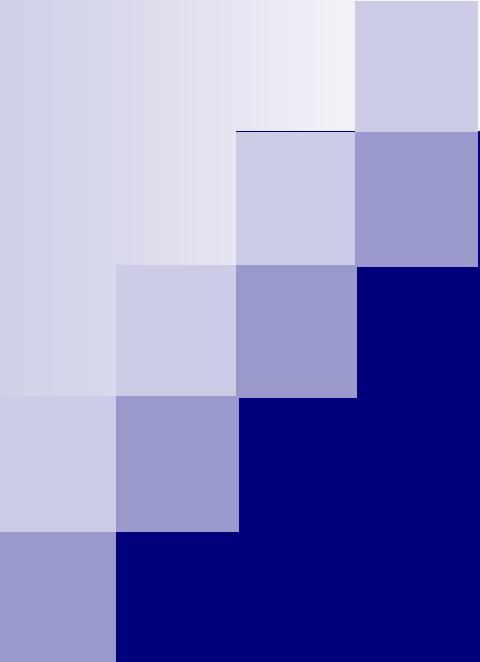
# Графічний процесор, здатний швидко виконувати основні операції із зображеннями у відеопам'яті:

- Робота з прямокутними блоками. Ця мікросхема наз. бліттер, тому що її основна операція - це BitBlit (Bit Block Transfer), тобто копіювання прямокутного блоку зображення в інше зображення з можливим застосуванням побітових логічних операцій (І, АБО, виключає АБО).
- Растеризація примітивів (відрізки, окружності, еліпси, прямокутники, багатокутники.) Також може підтримуватися заливання одноколірних зон іншим кольором або за шаблоном.

- Підтримка виводу символів певним шрифтом. Іноді шрифт можна завантажувати з основної пам'яті. В даний час цей режим використовується при завантаженні ПК, а також при роботі в режимі терміналу (частіше використовується в ОС Linux).
- Апаратне прискорення відео і фільтрація зображення. Кодування і декодування відео - дуже ресурсномістка операція, (обробка великих обсягів даних). Деякі відеокарти здатні апаратно декодувати відеопотік. У сучасних відеокартах також є підтримка телебачення високої чіткості (англ. HDTV - High Definition Television). Апаратне масштабування відео. Апаратне відображення відео в частині екрана носить назву оверлея (англ. overlay). Деякі відеокарти також можуть апаратно робити фільтрацію зображень.



Сучасні відеокарти містять також  
процесор опрацювання тривимірної  
графіки



# Фрактальна графіка.

**Геометричні та  
алгебраїчні фрактали**

Слово **фрактал** утворено від латинського **fractus** і в перекладі означає той, що складається з фрагментів (подрібнений).



Створення фрактальної композиції полягає не в промальовуванні готових об'єктів, а в **програмуванні**.

У програмних засобах зображення автоматично генеруються шляхом математичних розрахунків.

# В основі концепції фрактальності

лежить властивість виділяти об'єкти різного масштабу згідно ієрархічного принципу організації Всесвіту.

Основна гіпотеза, що лежить в основі фракталів – це **самоподібність**, тобто вигляд фрактальної структури не змінюється при обмежених масштабних перетвореннях.

# *Книга природи написана мою фракталів*

*Фрактальний світ добре відображає реальний, оскільки властивості фракталів демонструють багато природних об'єктів.*

*Фрактальна математика сьогодні дуже широко застосовується, вона може бути використана для аналізу змін цін і заробітної платні, статистики помилок на телефонних станціях і опису Всесвіту у цілому.*

# Алгоритми фрактальної геометрії

використовують для стиснення зображень, дистанційному зондуванні і радіолокації, моделюванні фракталоподібних розсіювальних систем, еволюційних обчислень, тощо.

Термін “фрактал” запропоноване Бенуа Мандельбротом в 1975 році для позначення *самоподібних структур*.

Народження фрактальної геометрії прийнято пов'язувати з виходом в 1977 році книги Мандельброта `*The Fractal Geometry of Nature*'.

- *Бенуа Мандельброт* – вчений, аналітик-математик компанії IBM, досліджував шуми в електронних схемах, засновник фрактальної геометрії.
- Хоча поняття **ФРАКТАЛ** придумав сам Бенуа Мандельброт, до того часу було багато інших досліджень, праць, що стосувалися фрактальних структур (математики, починаючи ще з 16 століття!)

Однією з основних  
властивостей фракталів є  
самоподібність.

У найпростішому випадку невелика  
частина фрактала містить інформацію  
про весь фрактал.

*Ітераційність, рекурсивність* –  
властивості фрактальних структур.

*Фракталом називається структура, що складається з частин, які в якомусь сенсі подібні до цілого.*

Є три типи самоподібності у фракталах:  
*Точна, майже самоподібність, статистична.*

**ЗАУВАЖЕННЯ!** Не всі самоподібні структури є фракталами в строгому сенсі цього терміну.  
Обов'язково - топологічна розмірність є **нечіле число.**

$D = \ln n / \ln N$ ,

$n$  – стільки разів вміститься зменшена в  $N$  фігура у вихідній

Масштабна інваріантність, що спостерігається у фракталах, може бути або точною, або наближеною.

Інваріантність - незмінність якої-небудь величини по відношенню до деяких перетворень.



# Корисна властивість фракталів

- за допомогою декількох коефіцієнтів можна задати лінії і поверхні дуже складної форми.

Більшість просторових систем у природі є нерегулярним і фрагментарним, форма цих систем погано піддається опису апаратом евклідової геометрії.

Фрактальна геометрія незамінна при генерації штучних хмар, гір, поверхні морів тощо.

Фактично знайдений спосіб легкого представлення складних об'єктів, які схожі на природні.

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 за формулами, рівняннями.

# Класифікація фракталів за способом побудови

- **Геометричні** (конструктивні) фрактали
- **Алгебраїчні** (рекурентні, динамічні) фрактали
- **Стохастичні** (випадкові) фрактали
- **IFS** - фрактали

## Зауваження.

Класифікація є дещо умовною. Залежно від інтерпретації методу побудови, один і той же фрактал може належати до різних класів.

# Детерміновані і недетерміновані фрактали

- Недетерміновані – стохастичні.
- Детерміновані – класичні (геометричні, алгебраїчні)

# Ще одна класифікація

Є три **типи самоподібності** у фракталах:

- *точна,*
- *майже самоподібність,*
- *статистична.*

# Геометричні фрактали

У двомірному випадку їх отримують за допомогою деякої ламаної (або поверхні в тривимірному випадку), яку наз. генератором.

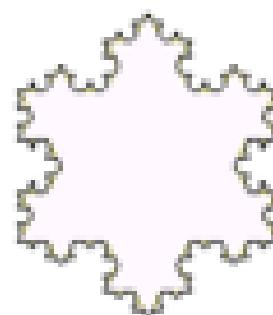
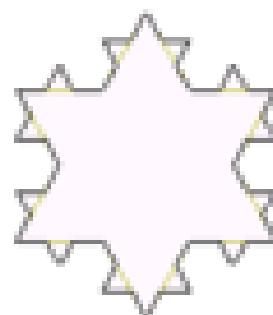
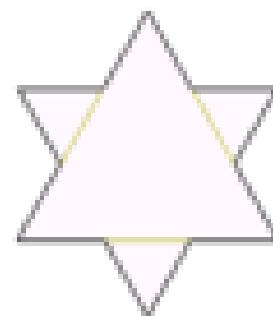
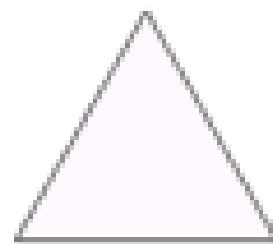
За один крок алгоритму кожен з відрізків, складових ламаної, замінюється на ламану-генератор, у відповідному масштабі.

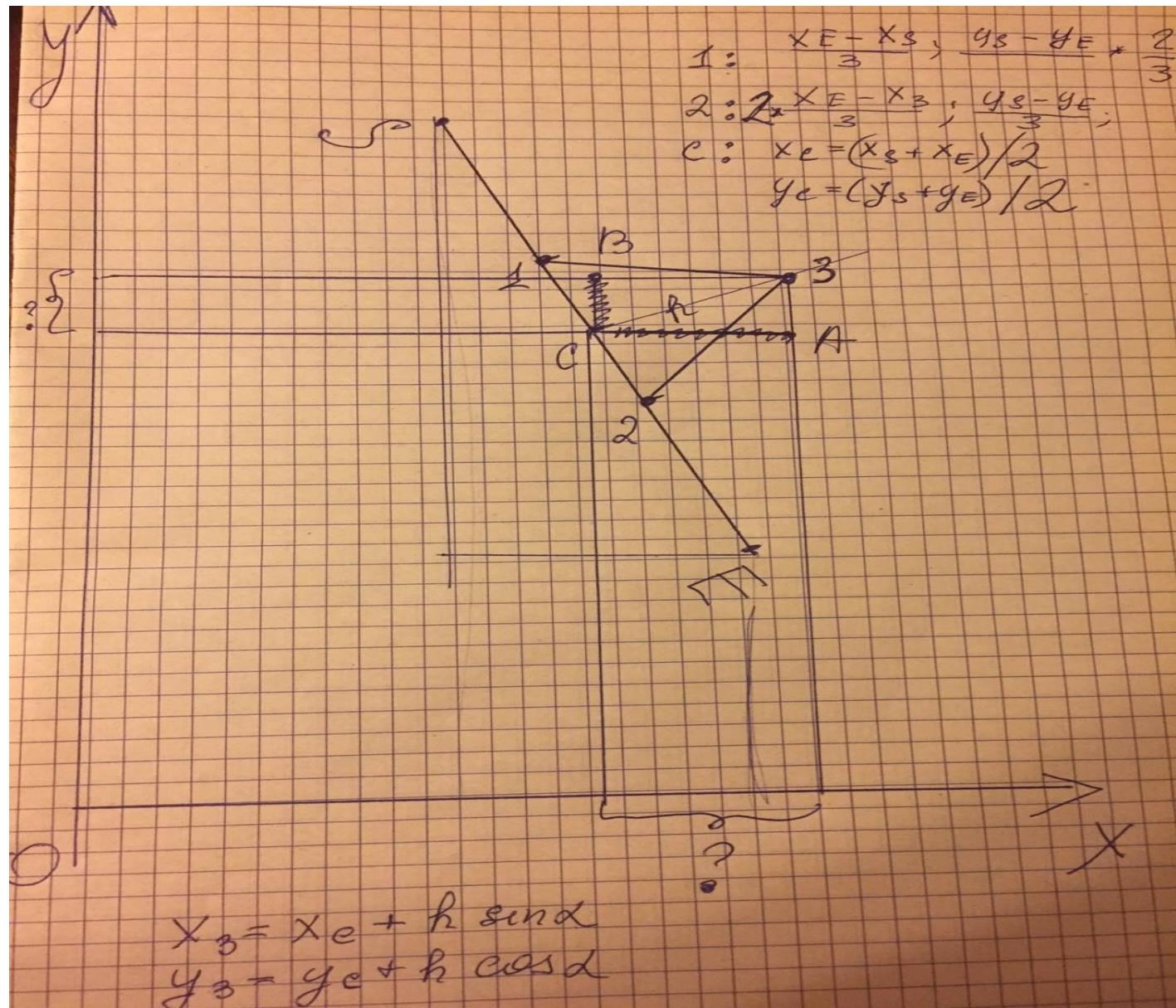
У результаті повторення цієї процедури, виходить геометричний фрактал.

# Приклади

Сніжинка Коха, крива Коха, крива  
Гільберта-Пеано, множина Кантора,  
килим Серпінського, трикутник  
Серпінського, крива дракона, Т-Квадрат,  
губка Менгера...

# Сніжинка Коха



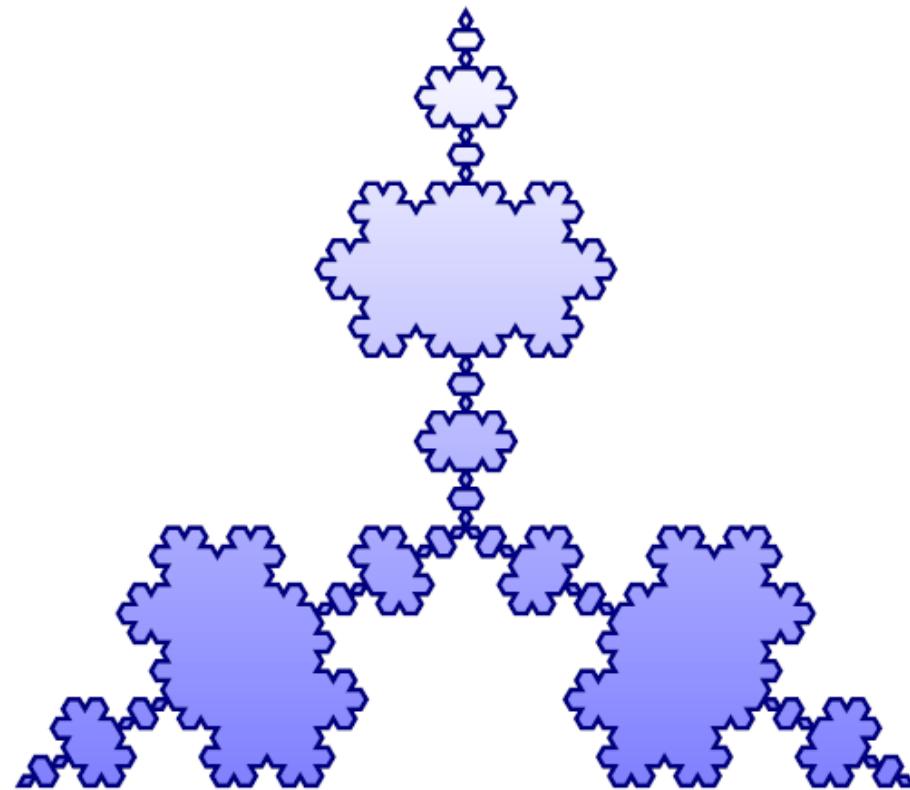


# Узагальнення лінії Коха

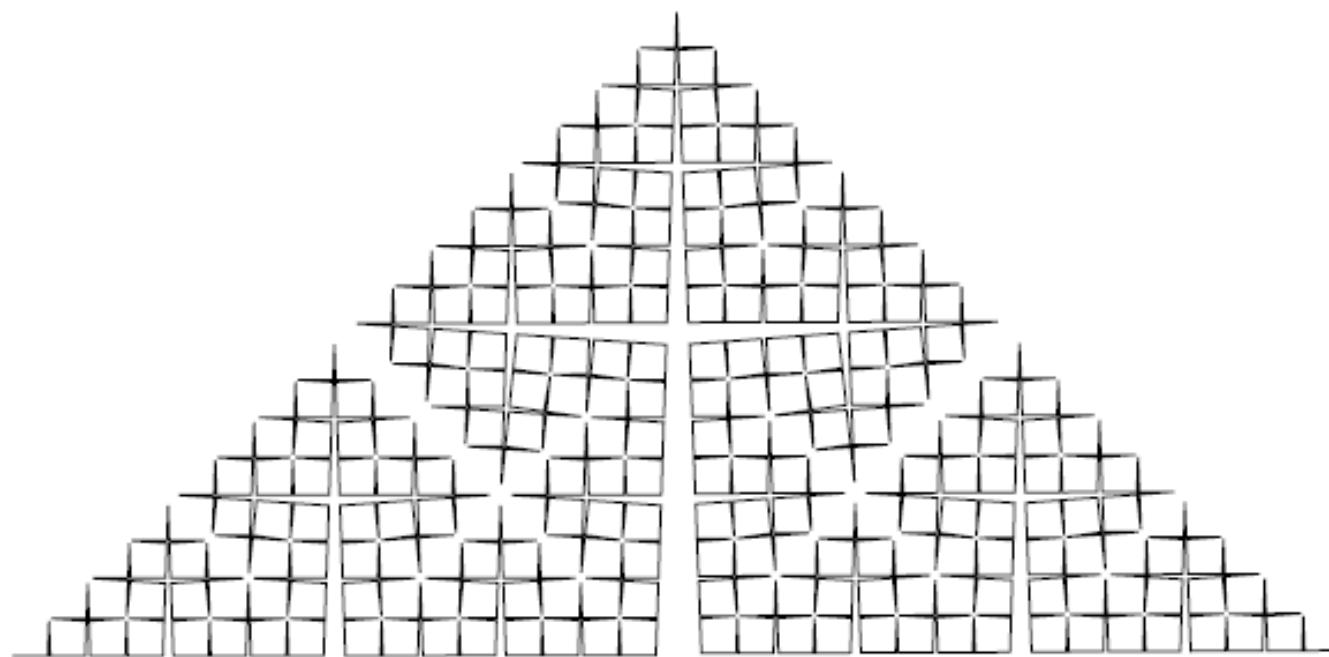
1. Замкнена ламана – квадрат/трикутник
2. Генератор – лінія з поділом на частини у певному відношенні.

Острів Коха, Острів Мінковського,  
Льодовий квадрат.

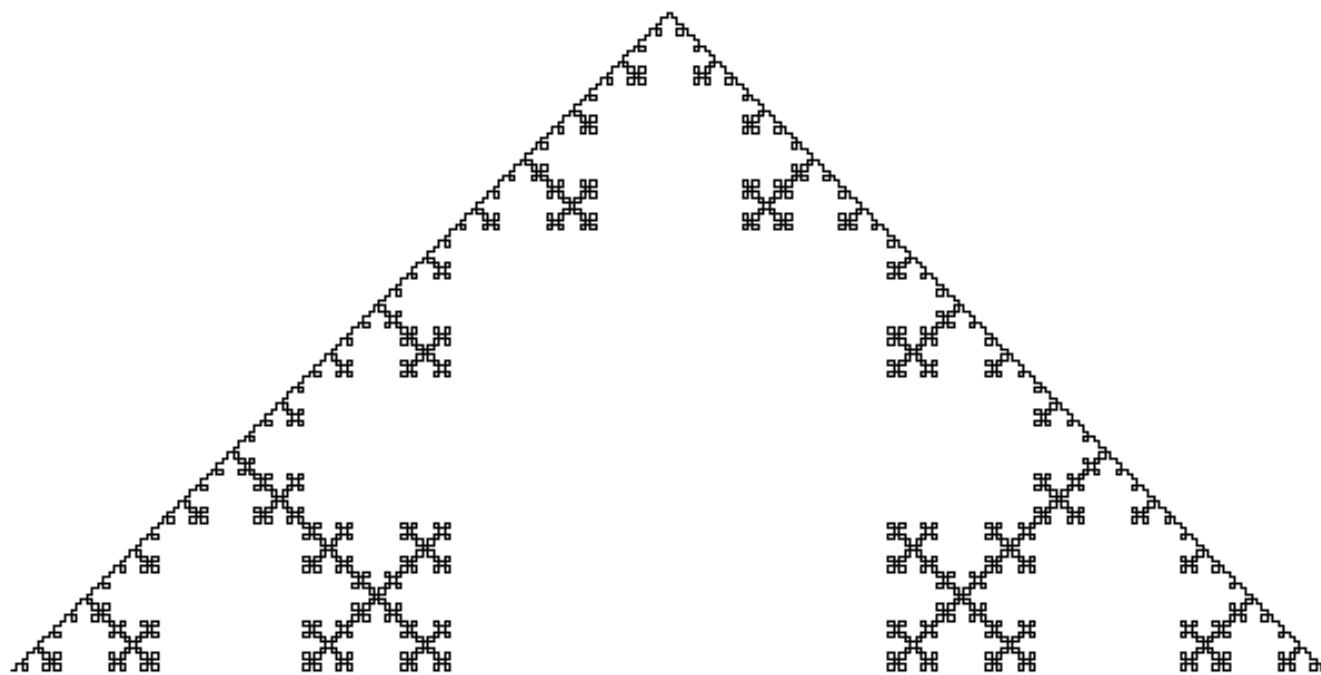
# Сніжинка Коха «навпаки»



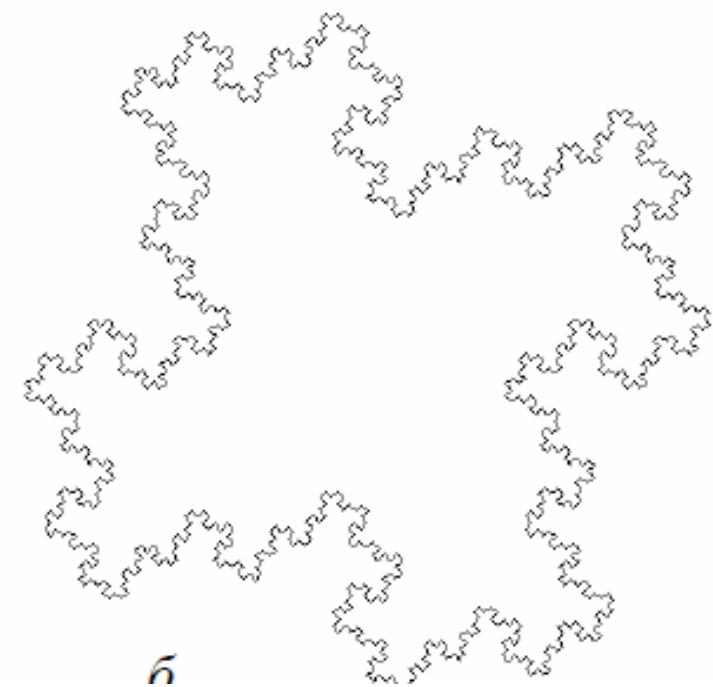
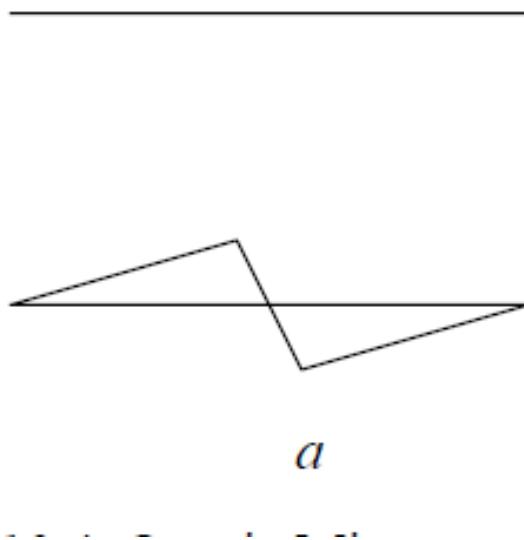
# Лінії Чезаро. Рівнобедрені трикутники від 60 °



# Замість трикутників квадрати

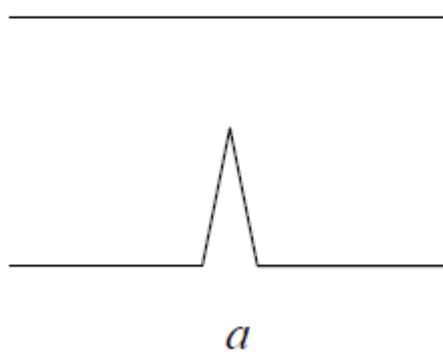


# Острів Міньковського

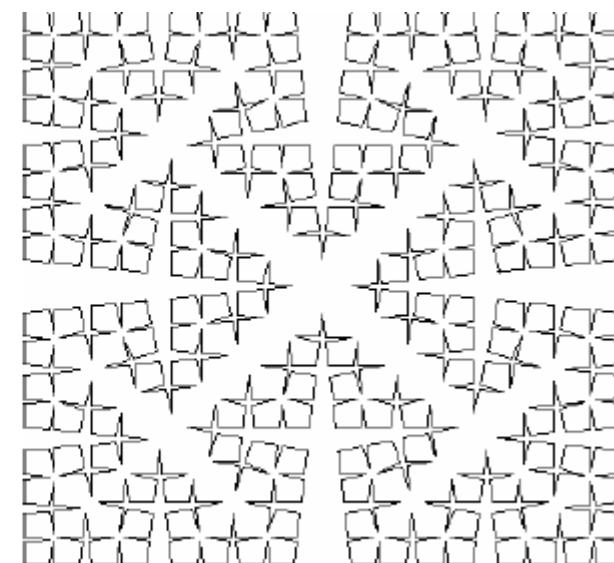


Проміжні значення  $(0,4; 0,2)$   $(0,6;-0,2)$

# Льодовиковий фрактал

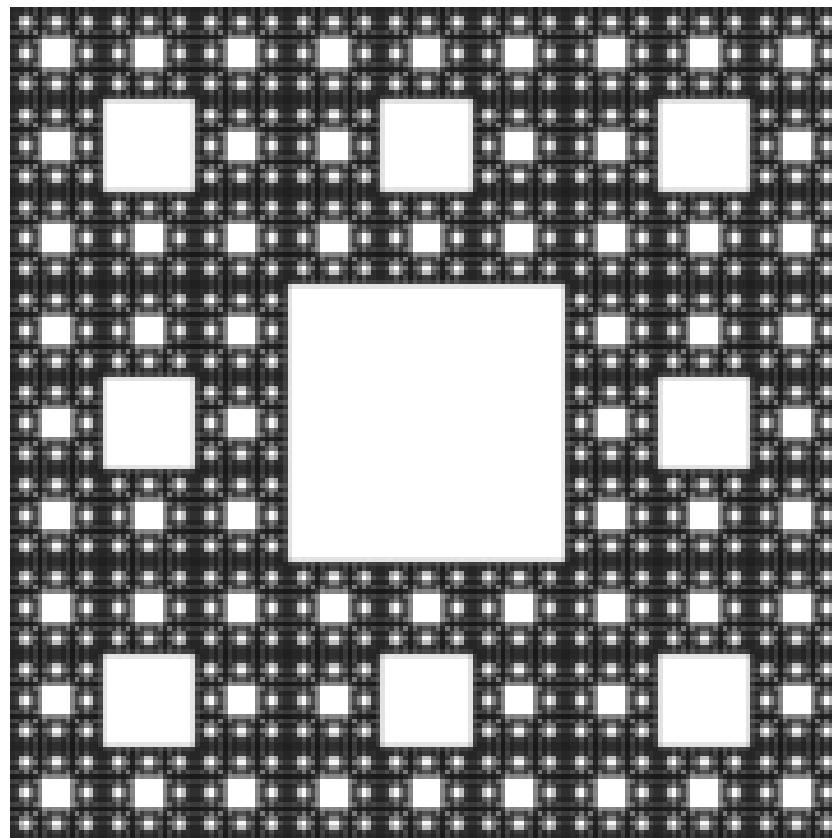


*б*



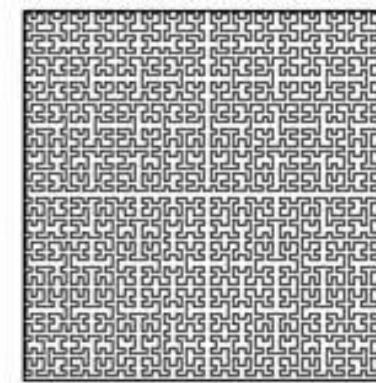
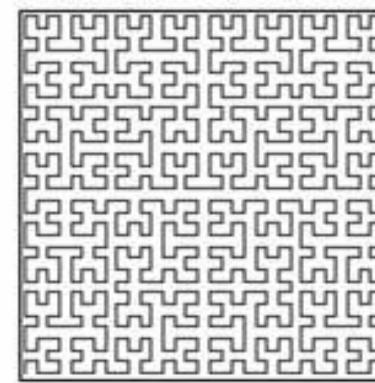
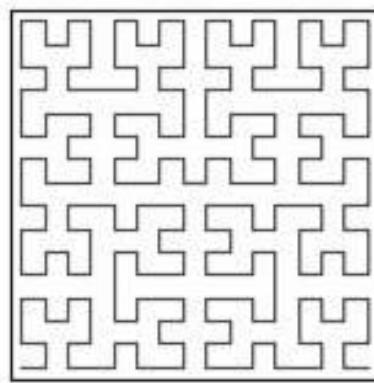
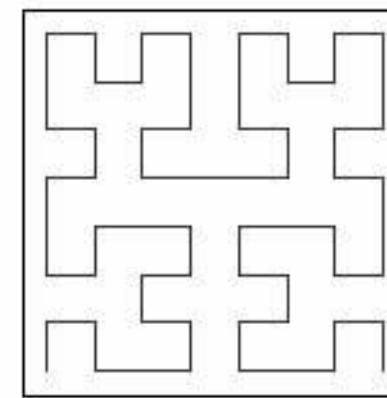
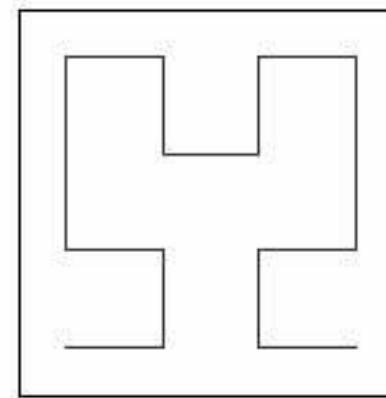
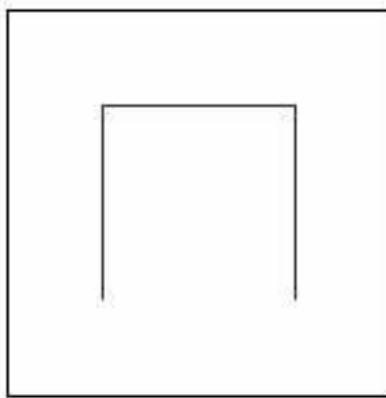
Проміжні значення  $(0,47; 0)$ ,  $(0,5; 0,47)$ ,  $(0,53; 0)$

# Килим Серпінського



Цей фрактал є межею нескінченної конструкції, що починається з квадрата та доповнюється рекурсивною заміною кожного сегменту набором із дев'яти сегментів.

# Крива Гільберта-Пеано



# Алгебраїчні фрактали

Це найбільша група фракталів.

Отримують їх за допомогою нелінійних процесів в **n-мірних** просторах.

Для побудови алгебраїчних фракталів використовуються ітерації нелінійних відображень, що задаються простими алгебраїчними формулами.

Найбільш вивчені двомірні процеси.

# Теорія динамічних систем

## Означення

- Детермінованість
- Фазовий простір
- Атрактор

- **Динамічна система** – математична абстракція, призначена для опису і вивчення систем, що еволюціонують у часі.
- **Детермінованість** - передбаченість поведінки за станом у початковий момент часу.
- **Фазовий простір** – множина усіх можливих станів системи у фіксований момент часу.

# Атрактор -

стабільний стан, який володіє деякою областю початкових станів, із яких система обов'язково потрапить в цей кінцевий стан.

Таким чином фазовий простір системи розбивається на області тяжіння атракторів.

Якщо фазовим є двовимірний простір, то присвоюючи кожній області тяжіння свій колір можна отримати кольоровий фазовий портрет цієї системи (ітераційного процесу).

Фрактали, що визначаються рекурентним відношенням в кожній точці простору (площина комплексних чисел).

$$Z_{k+1} = F(Z_k, C)$$
$$Z_k = x_k + iy_k, i^2 = -1,$$

$F$  – **нелінійна** функція,  
k – номер ітерації

Несподіванкою для математиків стала можливість за допомогою примітивних алгоритмів породжувати дуже складні нетривіальні структури.

Прикладами фракталів цього типу є множина Мандельброта, палаючий корабель, фрактал Ляпунова, басейни Ньютона, біоморфи...

# Вивчаємо

- Узагальнений алгоритм для динамічних фракталів
- Алгоритм Мандельброта
- Алгоритм Жюліа

# Алгоритм динамічних фракталів

У загальному випадку є така задача – заповнити область  $A_x * B_y$  (матриця растроу) кольорами, які генеруються за законом

$$z_{k+1} = F(z_k, c), \quad (1)$$

де  $z_k, z_{k+1}, c$  – комплексні числа,  $z_k = x_k + i * y_k$ , тобто  $(x_k, y_k)$  – точка растроу,  $h$  – крок растроу;

$F$  – нелінійна функція,

$k$  – крок ітерації (відповідає за колір),  $0..K_{\max}$  (велике число, н-ад, 500).

Обчислення відбуваються доти, поки виконаються умови

$$|z_k| \leq R \text{ i } k \leq K_{\max}, \quad (2)$$

де  $R$  – межа множини, певна константа і, як тільки  $k > K_{\max}$ , колір точки – чорний.

Якщо виконалася умова  $|z_k| > R$  і  $k \leq K_{\max}$ , необхідно взяти  $k$  як колір.

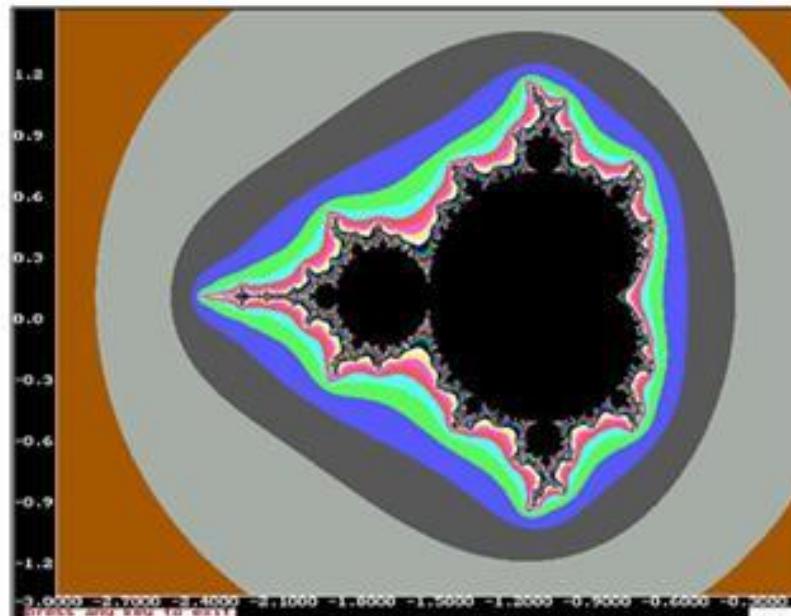
Наприклад, як залишок від ділення 256.

Умова (2) може мати вигляд  $\operatorname{Re}(z) \leq R_1, \operatorname{Im}(z) \leq R_2$ .

За допомогою описаного метода, використовуючи різні функції комплексної змінної, можна отримати безліч різноманітних алгебраїчних фракталів.

а) Множина Мандельброта;

б) Збільшена ділянка границі множини Мандельброта



а)



б)

$$z_{k+1} = z_k^2 + c, z_0 = c.$$

$c$  – комплексне число:  $c_x \in [-2; 1,5]$ ,  $c_y \in [-2; 2]$  – множина  $C$ .

Умова приналежності  $|z_k| \leq 2$  і для  $k \leq K_{\max}$

Ітераційний процес для першої точки растроу

$$\begin{cases} x_0 = c_x^0 \\ y_0 = c_y^0 \end{cases}$$

$$\begin{cases} x_1 = (c_x^0)^2 - (c_y^0)^2 + c_x^0 \\ y_1 = 2c_x^0 c_y^0 + c_y^0 \end{cases}$$

---

$$\begin{cases} x_{k+1} = x_k^2 - y_k^2 + c_x^0 \\ y_{k+1} = 2x_k y_k + c_y^0 \end{cases}$$

Беремо наступну точку растроу і виконуємо ітераційні обчислення для неї. Так проходимо всі точки растроу.

Тобто, необхідно перетворити множину комплексних чисел  $C$  у растрові точки  $A_x * B_y$ . Важливим є крок!

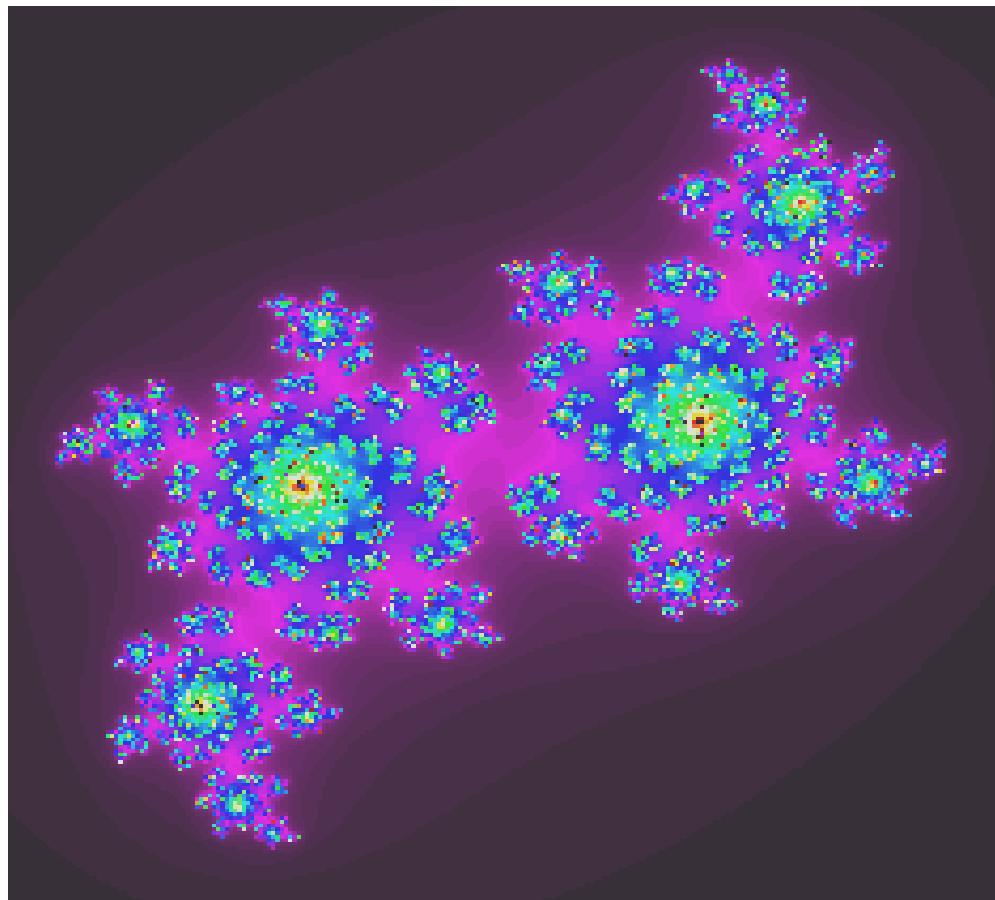
- Ітераційний процес продовжується до тих пір, поки не вийде за межі кола радіусу 2, центр якого знаходиться в точці  $(0,0)$  В такому випадку це означає, що атрактор динамічної системи знаходиться в нескінченності.
- Якщо протягом великої кількості ітерацій сходиться до якої-небудь точки кола, то це означає що система перейшла в стабільний стан ( знайдено атрактор в межах кола).

- Змінювати колір можна в залежності від кількості ітерацій, протягом яких точка знаходилась в колі 2. якщо система перейшла в стабільний стан, то колір точки можна позначити чорним.

# До множини Мандельброта

належать ті точки, які протягом нескінченної кількості ітерацій не „вилітають” у нескінченність.

# Множина Жюліá



Множина Мандельброта є частковим випадком множини Жюліа. У загальному випадку множину Жюліа можна записати у вигляді

$$J(f) = \lim_{n \rightarrow \infty} (f_n(z))$$

- де  $f_n$  – деяка комплексна функція (як правило, квадратична),
- $z$  – комплексна змінна.

$$z_{k+1} = z_k^2 + c, c^* - \text{фіксована константа}$$

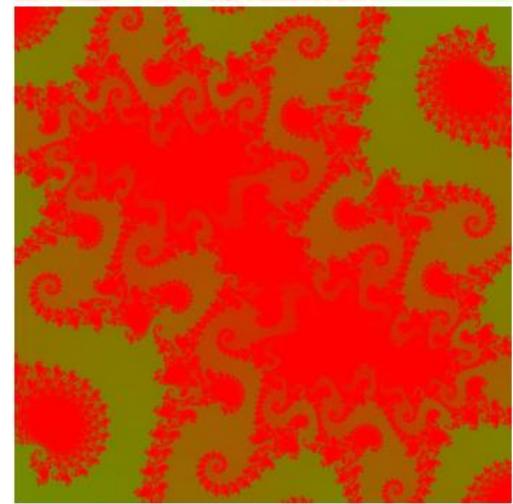
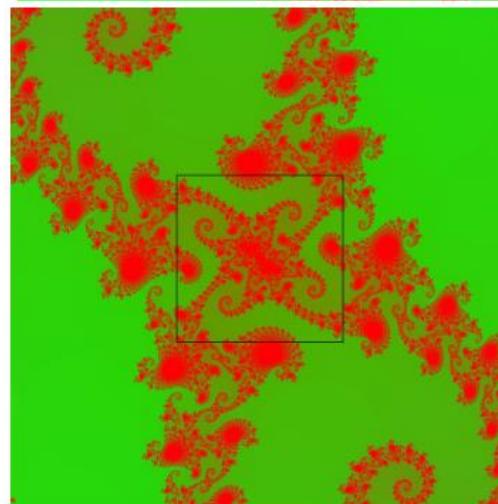
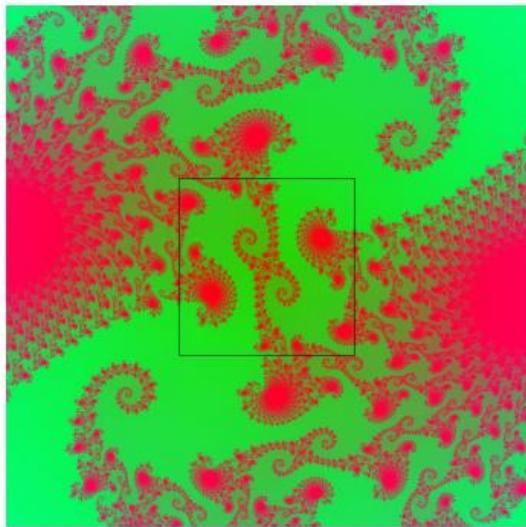
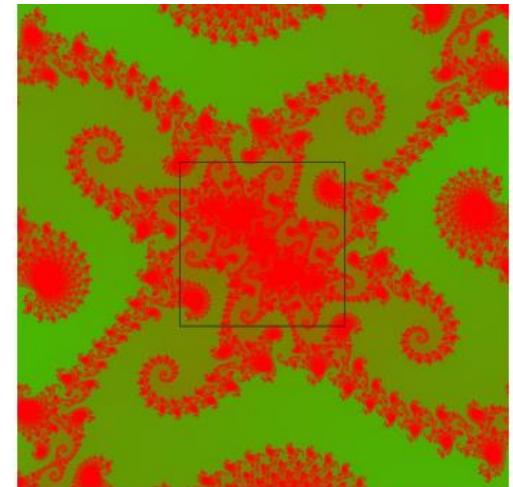
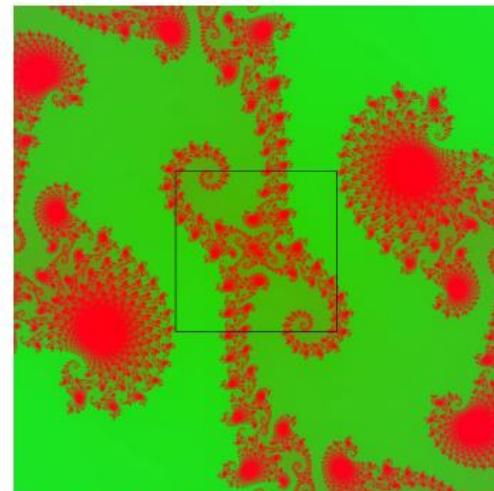
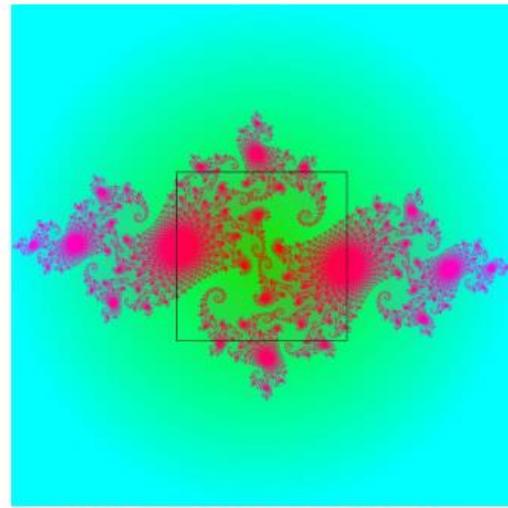
(не змінюється для точок й ітерацій), комплексне число.

$$z_0 \in [-1; 1] + i[-1; 1]$$

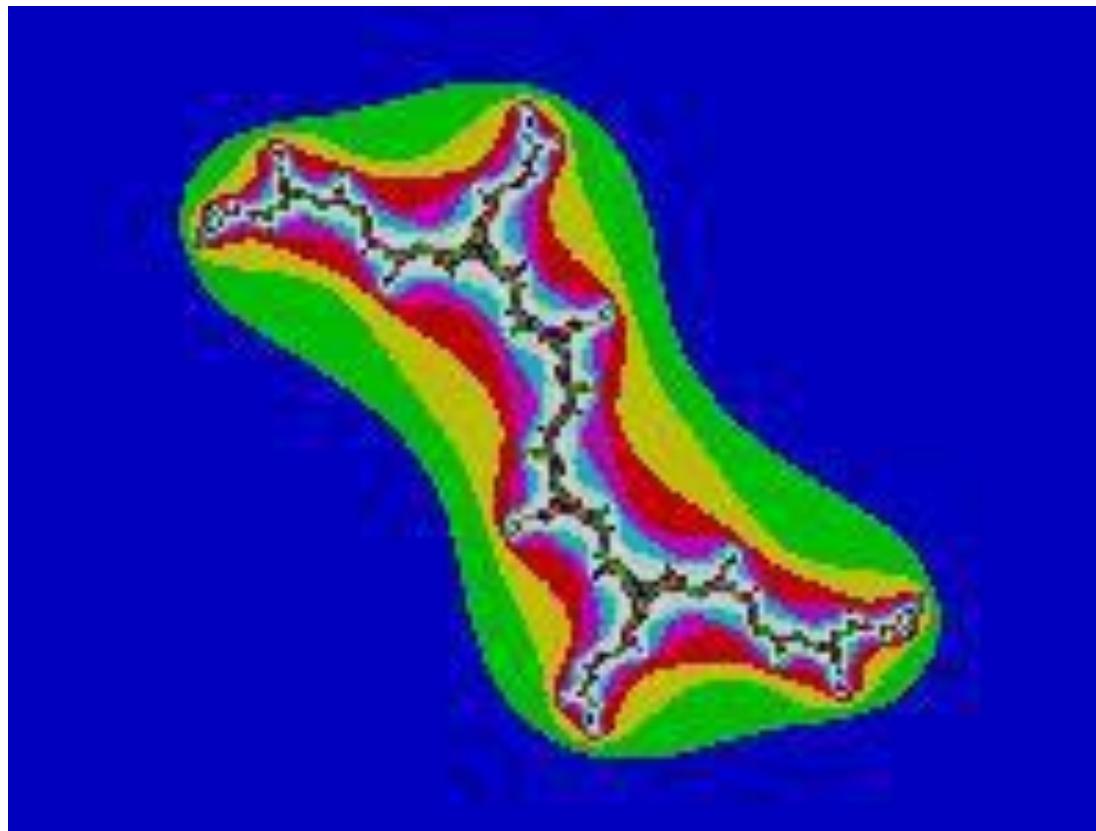
# Фрактал Жюліа

$$f(z) = z^2 - 0.74543 + 0.11301i$$

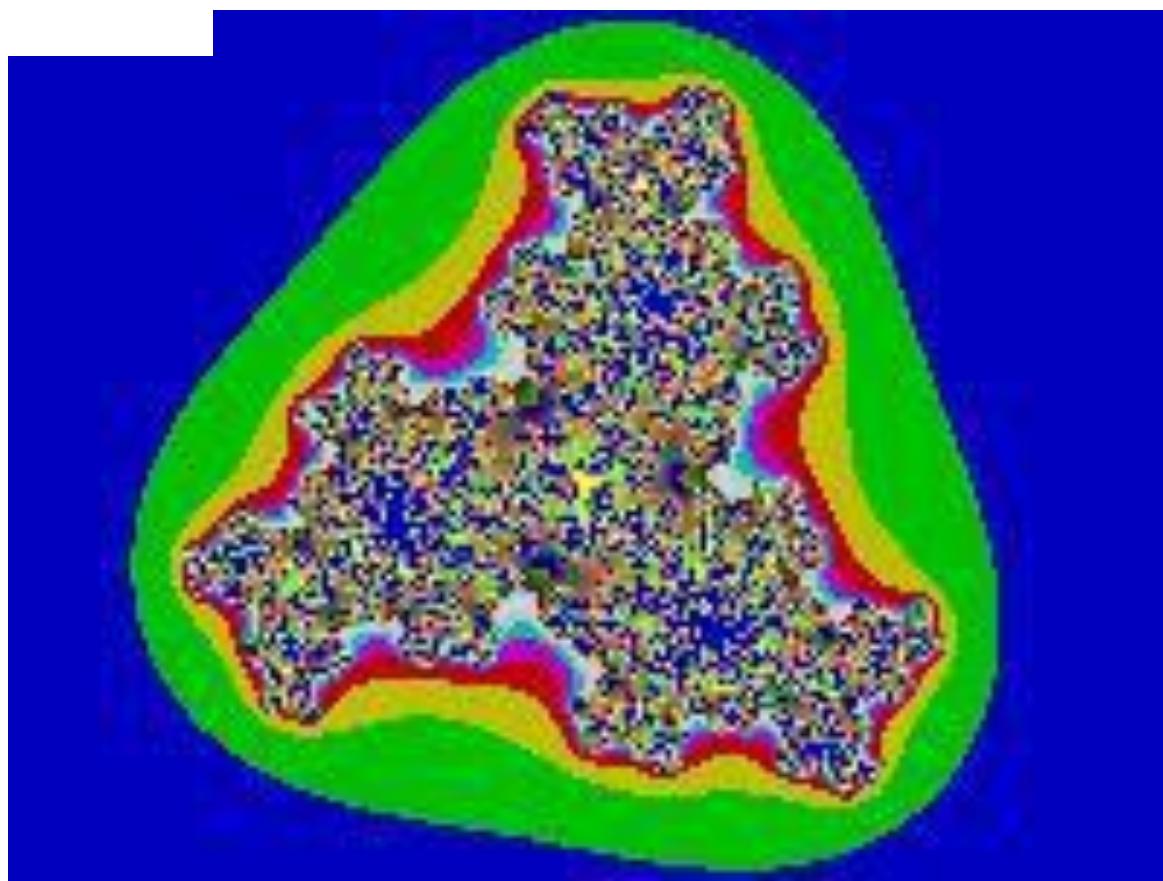
# Масштабування фракталу



# Фрактал Жюліа, $c=i$



$Z^3 + C$



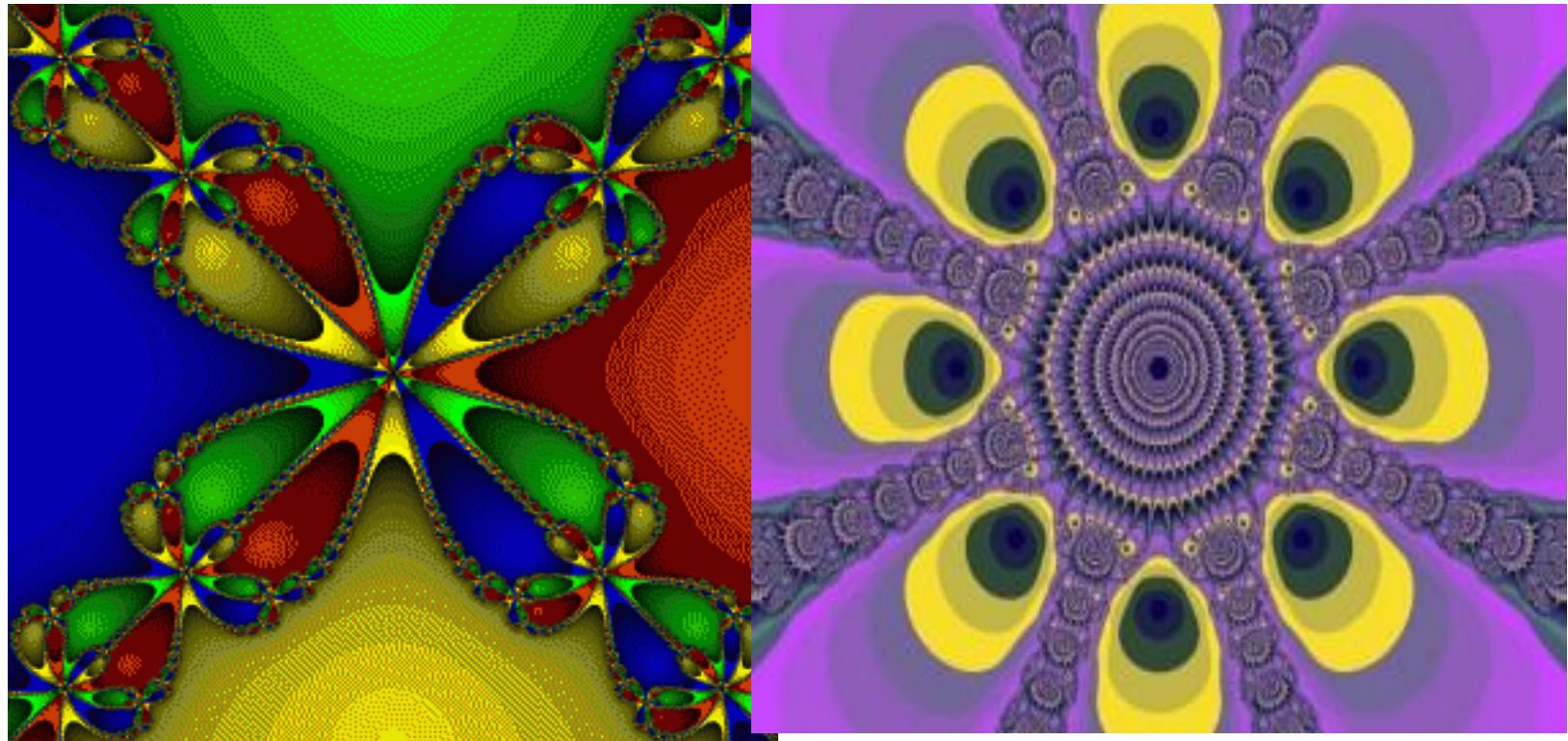
Фрактали Ньютона базуються на методі розв'язування нелінійних рівнянь.

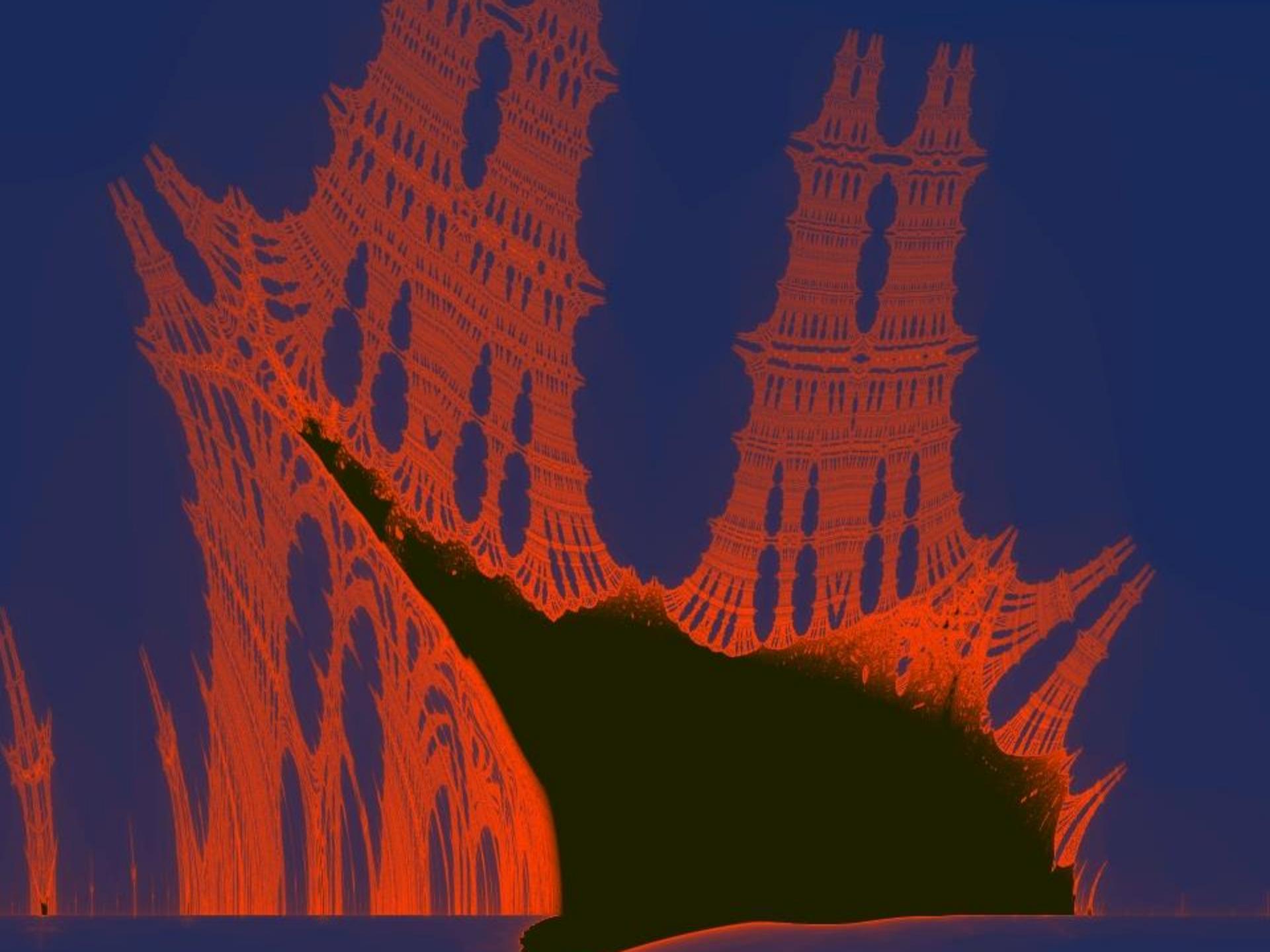
$$Z_{k+1} = Z_k - f(Z_k) / f'(Z_k),$$

$$|\operatorname{Re} Z| \leq C_1, |\operatorname{Im} Z| \leq C_2$$

$$f(z) = z^n - a$$

# Фрактали Ньютона

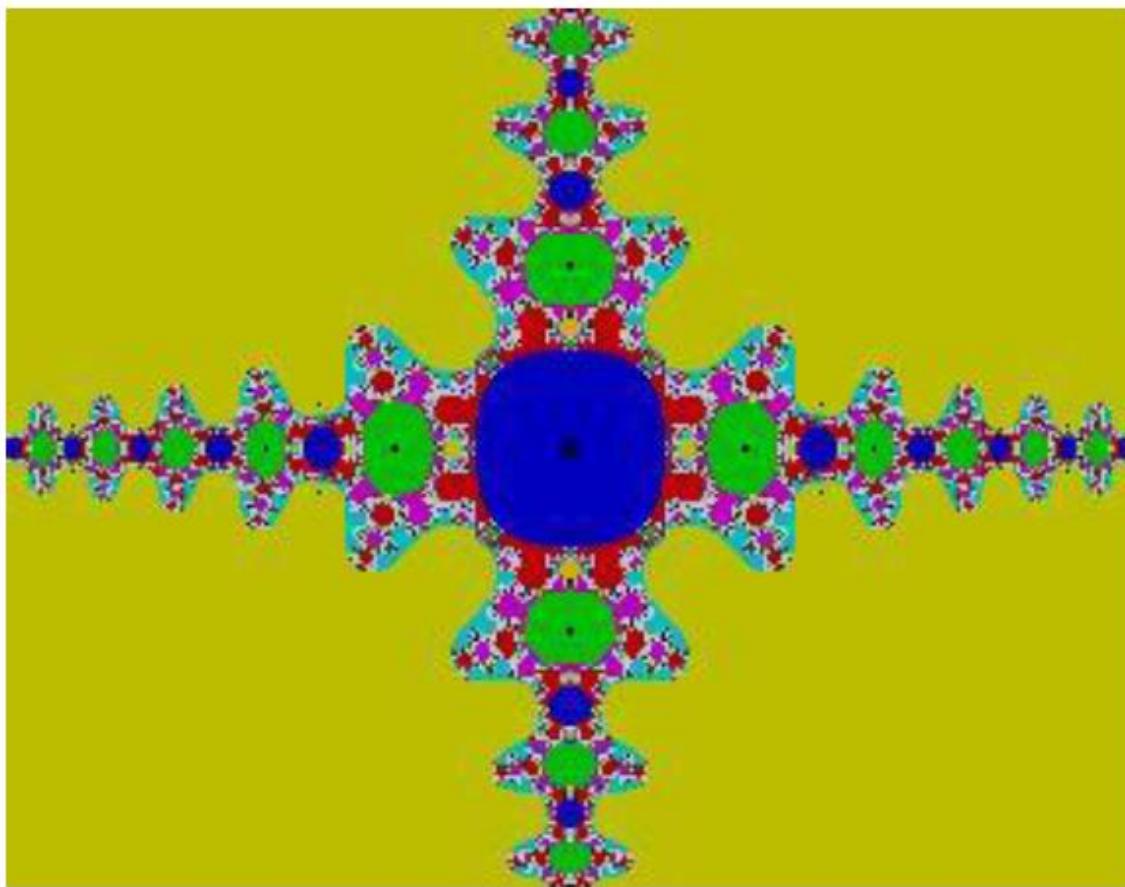




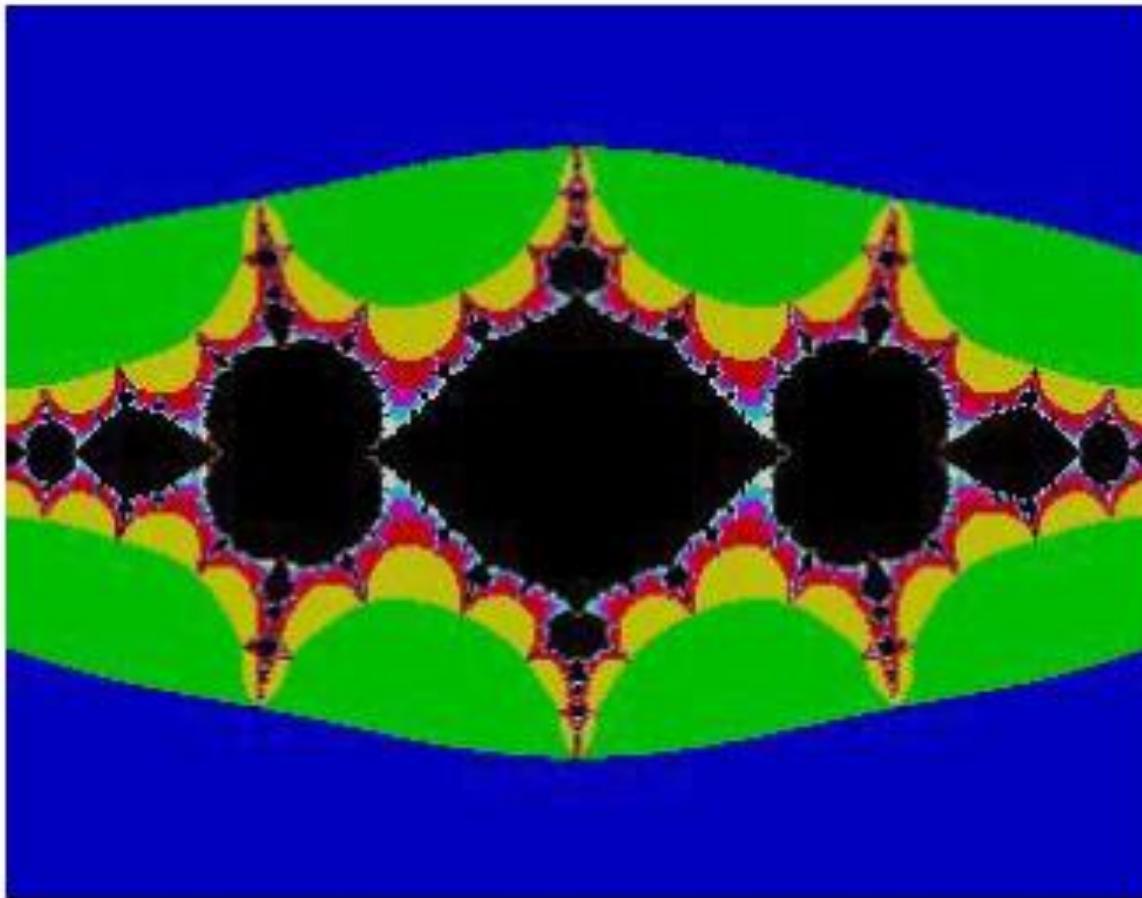
# Зображення алгебраїчного фракталу

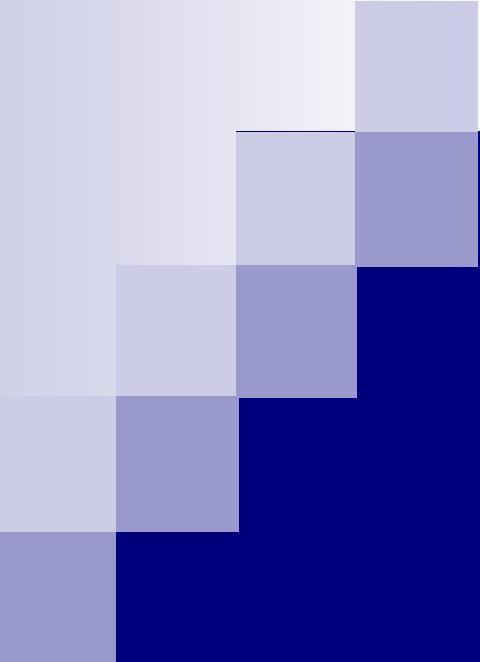
– це точки на комплексній площині, які визначаються атракторами нелінійної динамічної системи, що представляється ітеративним обчисленням деякої функції від комплексного числа

$\operatorname{ctg}(z)$



$z * \sin(z)$





# Фрактальна графіка.

## Стохастичні та IFS-фрактали

**Фракталом називається структура, що складається з частин, які в якомусь сенсі подібні до цілого.**

Є три **типи самоподібності** у фракталах:

- точна,
- майже самоподібність,
- статистична.

# Корисна властивість фракталів

- за допомогою декількох **коефіцієнтів (формул)**  
можна задати лінії і поверхні дуже складної форми.

Більшість просторових систем у природі є  
нерегулярним і фрагментарним, форма цих систем  
погано піддається опису апаратом евклідової  
геометрії.

Фрактальна геометрія незамінна при генерації штучних  
хмар, гір, поверхні морів тощо.

Фактично знайдений спосіб легкого представлення  
складних об'єктів, які схожі на природні.

# Класифікація фракталів за способом побудови

- **Геометричні** (конструктивні) фрактали
- **Алгебраїчні** (рекурентні, динамічні) фрактали
- **Стохастичні** (випадкові) фрактали
- **IFS** - фрактали

# Стохастичні фрактали

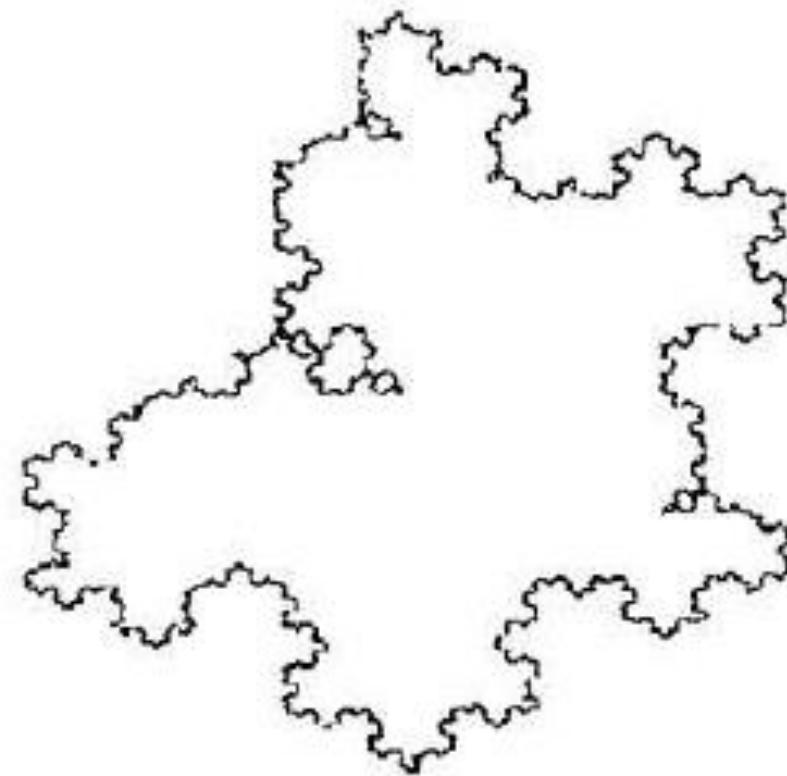
утворюються в тому випадку, якщо в ітераційному процесі випадковим чином міняти які-небудь його параметри.

При цьому виходять об'єкти дуже схожі на природні – несиметричні дерева, порізані берегові лінії і т.д.

Двовимірні стохастичні фрактали використовуються при моделюванні рельєфу місцевості і поверхні моря.

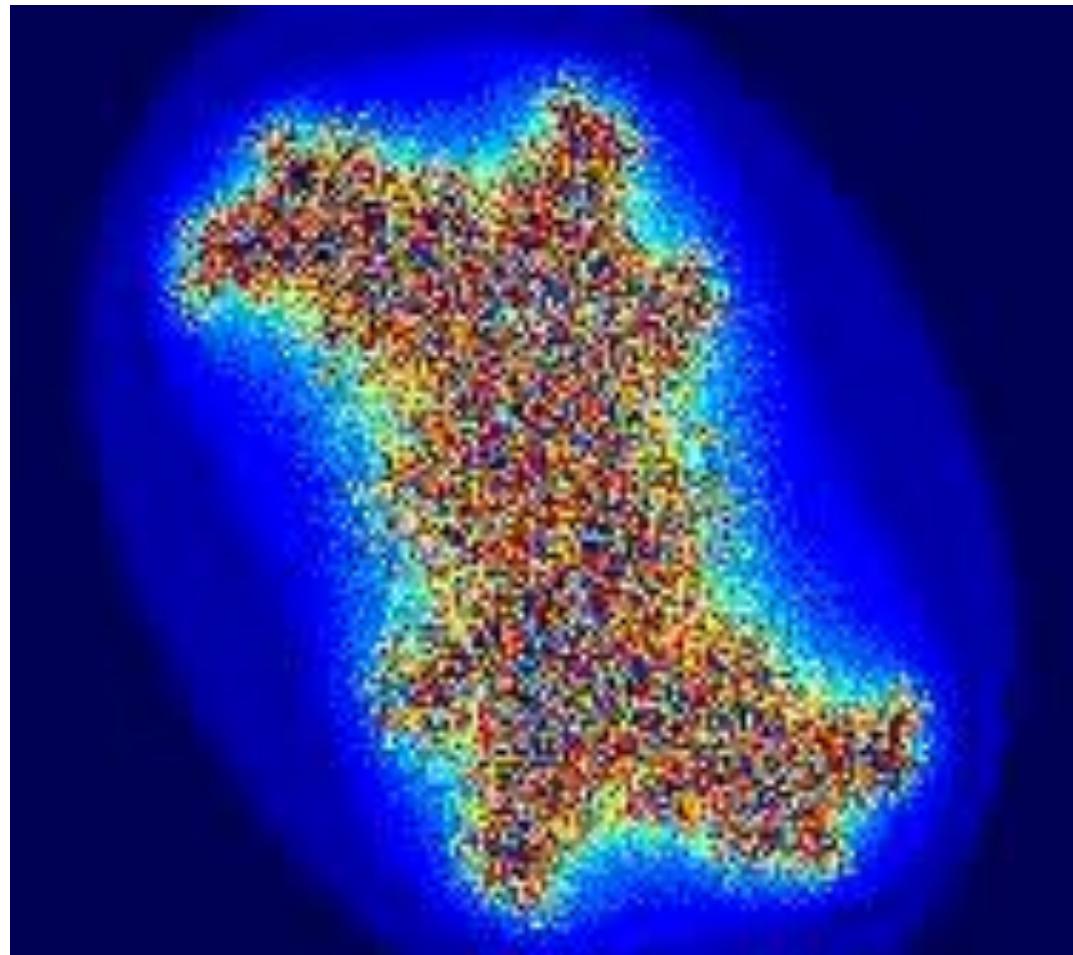
Представниками цього класу можна вважати траекторію броунівського руху, плазму та різні види *рандомізованих фракталів, тобто таких, які можна отримати за допомогою рекурсивної процедури, в яку на кожному кроці введений випадковий параметр.*

# Рандимізована сніжинка Коха



- Границя крива рандомізованої сніжинки Коха може служити моделлю, наприклад, контуру хмари або острова.

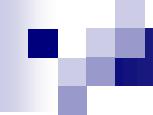
# Броунівський рух



Стохастичним природним процесом є броунівський рух.

Змоделювати броунівський рух на комп'ютері досить просто:

**частинку потрібно зміщувати на задану відстань (у моделі однакових кроків) в довільному напрямку.**



Такі процеси задаються досить просто:

потрібно **задати** послідовність  
випадкових чисел і налаштувати  
**відповідний алгоритм.**

# Ітераційний процес

$$x_n = x_{n-1} \pm 1$$

де  $x_n$  - точки прямої.

Точнішим наближенням до реального броунівського руху є заміна кроків  $\pm 1$  випадковими величинами , які мають розподіл Гауса.

$$x_n = x_{n-1} \pm g_n$$

Двовимірний фрактальний вінеровський процес (поверхня) визначається в такий спосіб: – це функція двох аргументів  $X(x, y)$ , що володіє наступними властивостями:

1)  $X(0,0) = 0$  і майже всі реалізації процесу безперервні;

2) Приросту процесу  $X(x, y)$  відповідає:

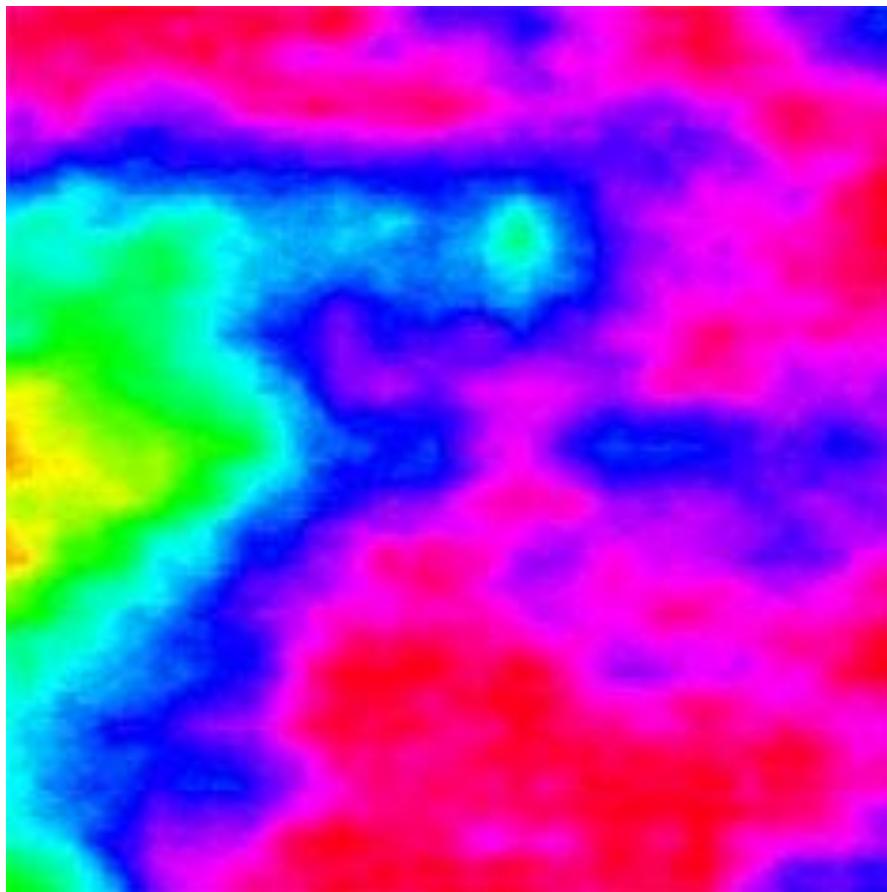
$$\Delta X = X(x + \Delta x, y + \Delta y) - X(x, y),$$

де  $\Delta x = x(t_2) - x(t_1)$ ,  $\Delta y = y(t_2) - y(t_1)$  – є випадковою величиною, що має нормальній інтегральний закон розподілу.

Фрактальна розмірність двовимірного вінеровського процесу дорівнює:

$$d_f = 3 - H.$$

# Плазма



# Алгоритм «Плазма»

Нехай потрібно заповнити плазмою **квадрат** точок.

1. Задається яким-небудь чином, наприклад, **випадково, кольори кутів квадрата.**
2. Визначається **колір середини кожної сторони** як середнє між кольорами інцидентних їй вершин плюс / мінус деяка випадкова величина.
3. Визначається **колір центру квадрата** як середнє між кутами плюс / мінус деяка випадкова величина.
4. Виходить 4 квадрати із заданими вершинами – для кожного з них **повторюється алгоритм з кроку 2.**

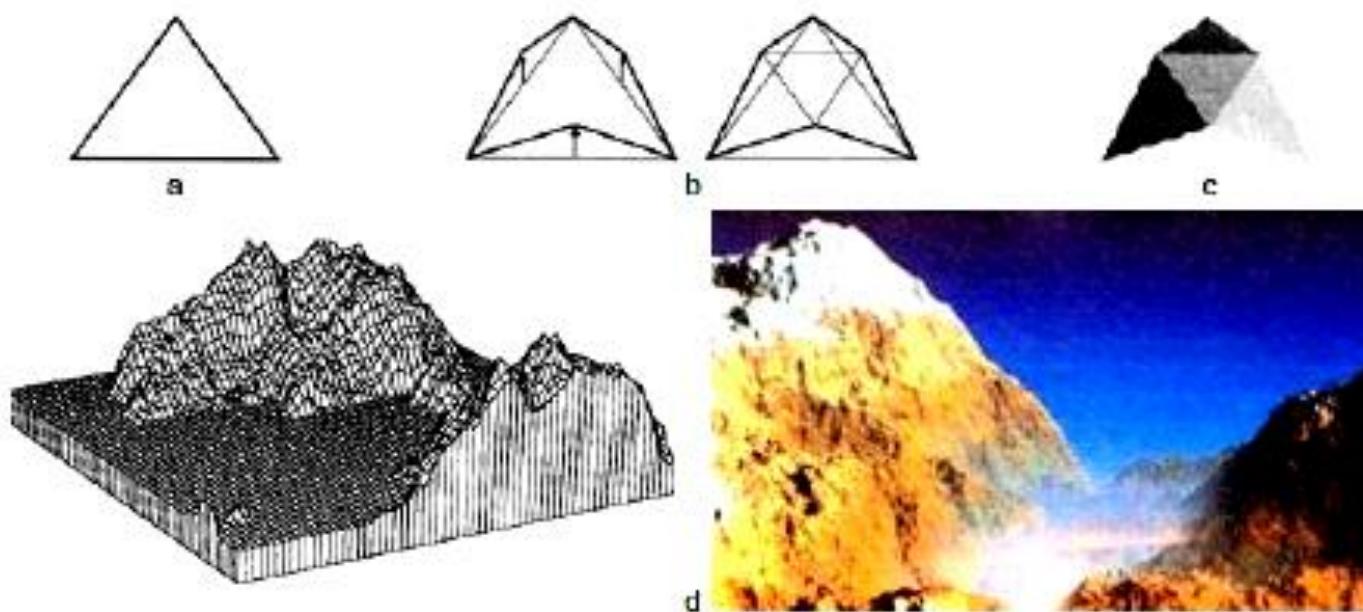
- Розмах випадкової величини має залежати від розміру квадрата – чим менший квадрат, тим менше відхилення.

# Поверхня гір моделюється з використанням фракталів

- початок з трикутника в тривимірному просторі та з'єднання центральних точок кожного ребра відрізками, отримуючи 4 трикутники;
- центральні точки потім зсуваються догори або донизу на випадкову відстань у фіксованому діапазоні.

Процедура повторюється зі зменшенням діапазону на кожній ітерації вдвічі.

Рекурсивна природа алгоритму гарантує, що ціле є статистично подібним до кожної з деталей.

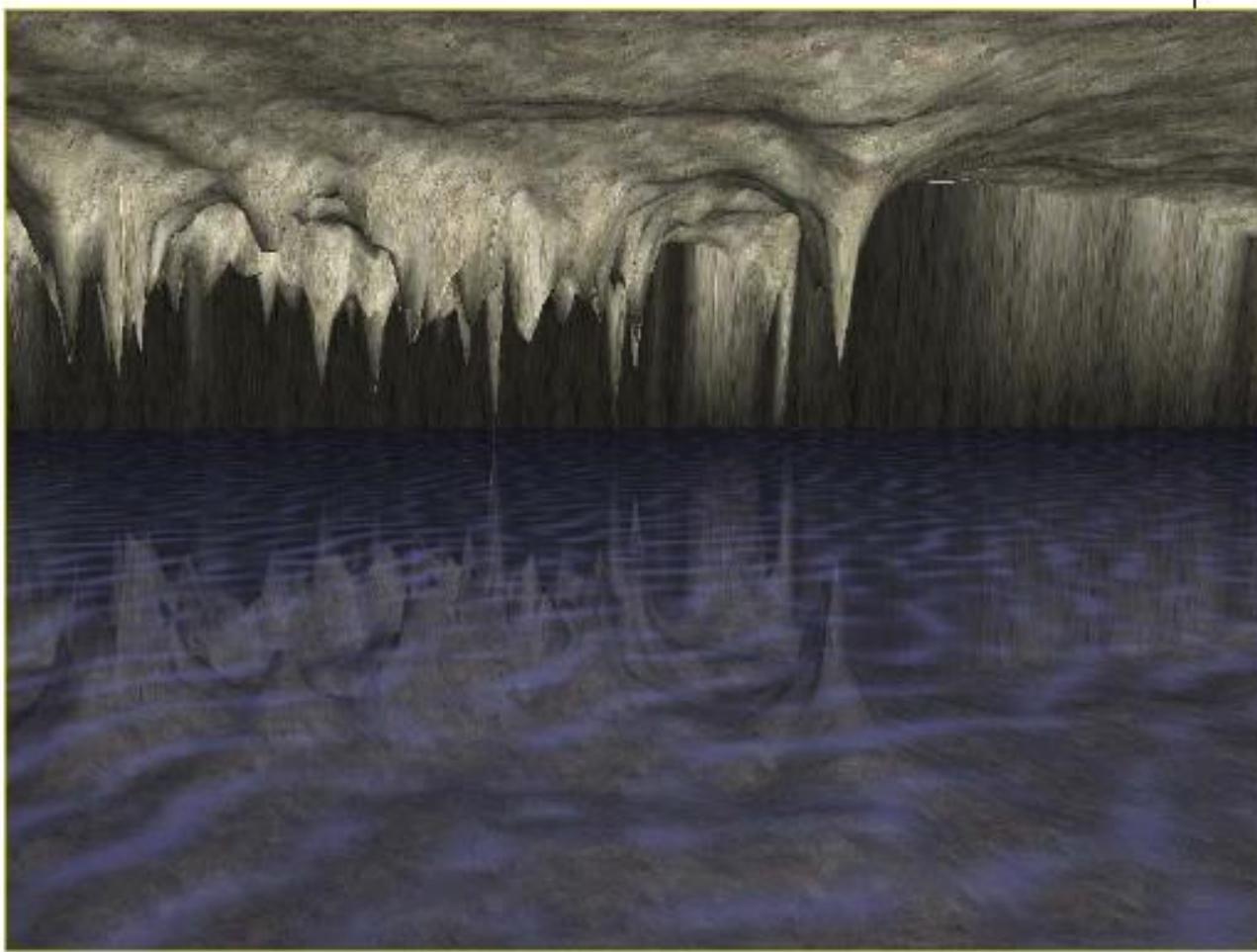


- [https://commons.wikimedia.org/wiki/File:Animated\\_fractal\\_mountain.gif#/media/Файл:Animated\\_fractal\\_mountain.gif](https://commons.wikimedia.org/wiki/File:Animated_fractal_mountain.gif#/media/Файл:Animated_fractal_mountain.gif)

Якщо колір точки у плазмі - це висота над рівнем моря, то отримаємо замість плазми - гірський масив.

За допомогою алгоритму, схожого на плазму будується карта висот, до неї застосовуються різні фільтри, накладається текстура.





# Лорен Карпентер (Loren C. Carpenter)

- майбутній співзасновник легендарної студії Pixar, який спочатку працював у компанії Boeing Computer Services, що займається розробкою нових літаків, Перший візуалізував природні об'єкти (гірський пейзаж), використавши теорію Мандельброта.



Принцип, який використовував  
Лорен Карпентер для  
досягнення мети,  
був дуже простий. Він полягав у тому,  
щоб розділяти більшу геометричну фігуру  
на дрібні елементи, а ті, в свою чергу,  
ділити на аналогічні фігури меншого  
розміру, використовуючи певні випадкові  
параметри

# Фрактали, побудовані методом

## Iterated Functions System

IFS-метод з'явився в середині 80-х років як простий засіб отримання фрактальних структур.

IFS є системою функцій деякого фіксованого класу функцій, що відображають одну багатовимірну множину на іншу.

# Загальна формула ітераційного процесу

$$\begin{cases} x_{k+1} = F_x(x_k, y_k) \\ y_{k+1} = F_y(x_k, y_k) \end{cases}$$

# Фіксований клас функцій

- Афінні перетворення
- Квадратичні перетворення
- Проектуюче перетворення
- ...

Найпростіша IFS складається з афінних перетворень :

$$\left\{ \begin{array}{l} X^* = A \cdot X + B \cdot Y + C \\ Y^* = D \cdot X + E \cdot Y + F \end{array} \right.$$

# Що таке афінне перетворення?

- 1.
- 2.

# Загальні вигляди перетворень

$$X' = A_1 X^2 + B_1 X Y + C_1 Y^2 + D_1 X + E_1 Y + F_1;$$

$$Y' = A_2 X^2 + B_2 X Y + C_2 Y^2 + D_2 X + E_2 Y + F_2.$$

$$X' = (A_1 X + B_1 Y + C_1) / (D_1 X + E_1 Y + F_1);$$

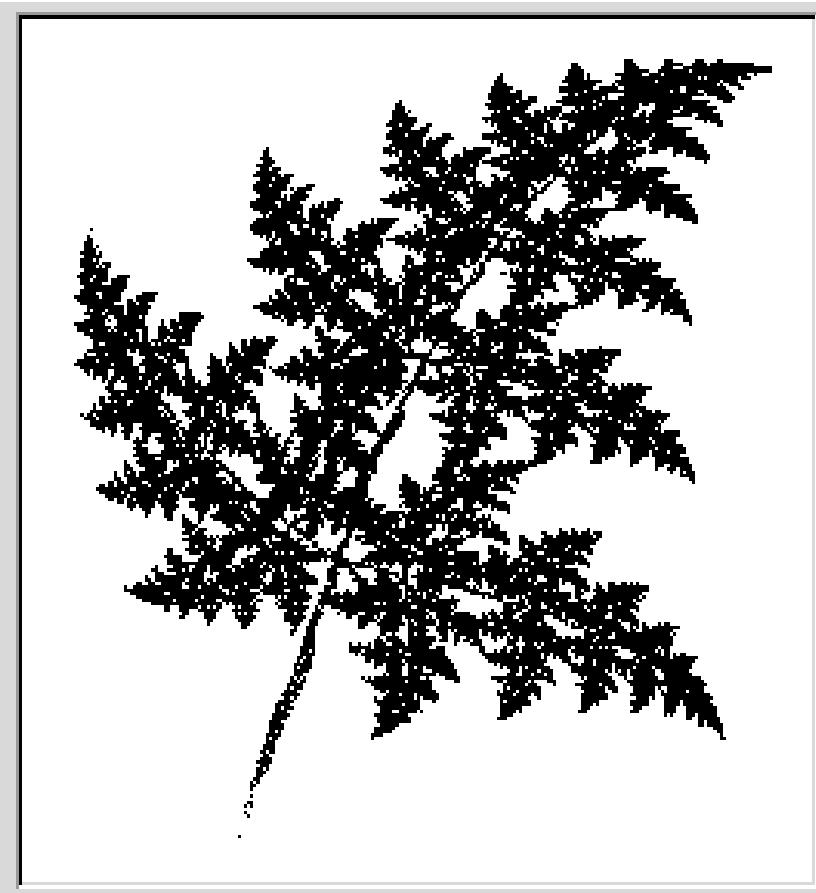
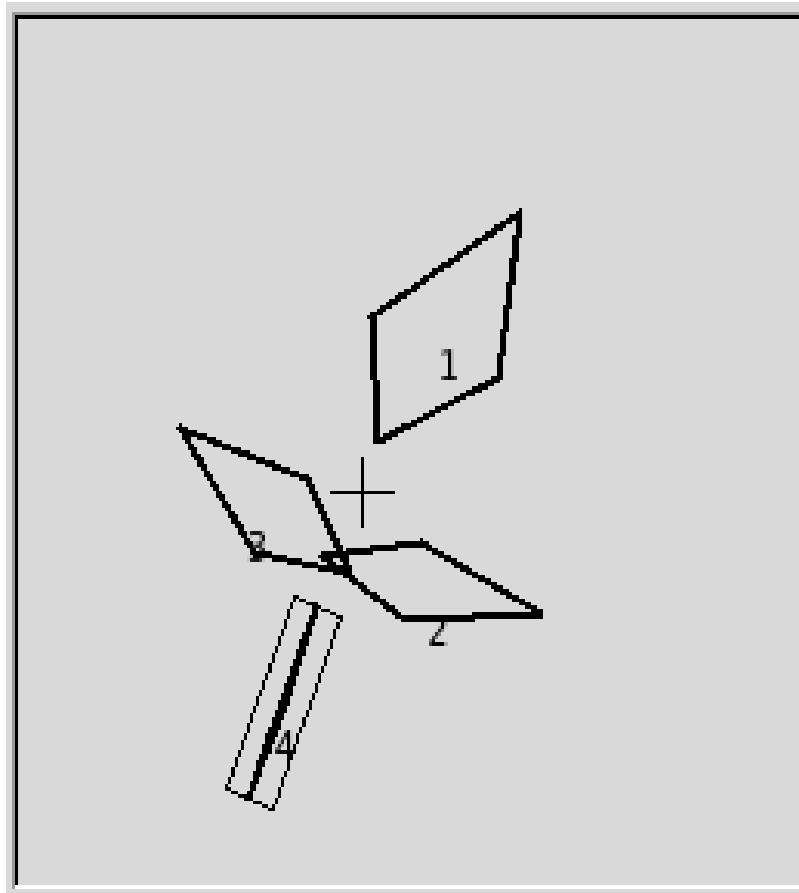
$$Y' = (A_2 X + B_2 Y + C_2) / (D_2 X + E_2 Y + F_2);$$

# Папороть Барнслі

- 4 області і 4 перетворення  $R_1, R_2, R_3, R_4$
- $$\begin{pmatrix} x^* \\ y^* \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} c \\ f \end{pmatrix}$$
  - формула для IFS

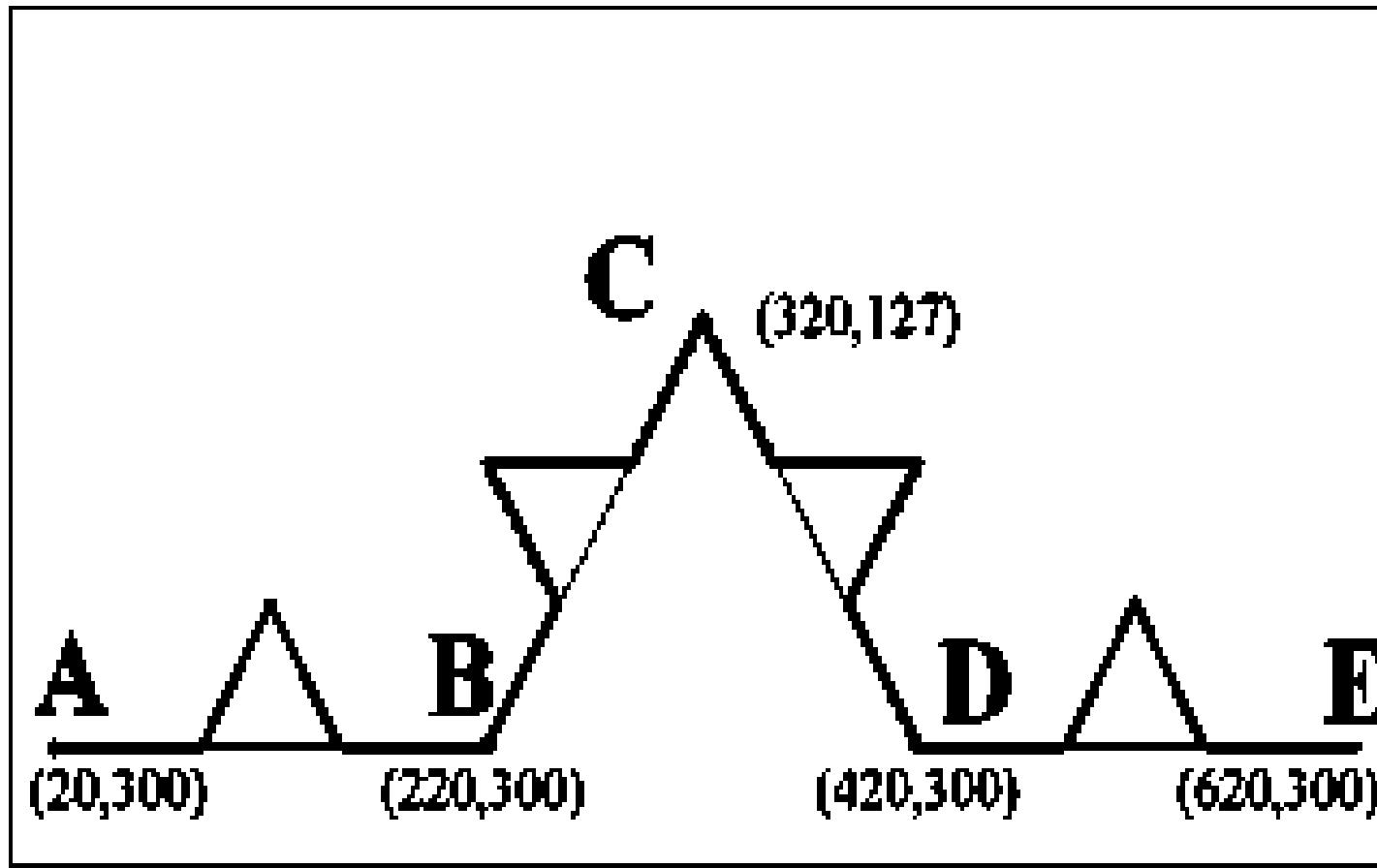
# Значення коефіцієнтів беруться з певною імовірністю $P$

	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>	<b>e</b>	<b>f</b>	<b>P</b>
$R_1$							0,85
$R_2$							0,07
$R_3$							0,07
$R_4$							0,01



# Приклади

- Лінія Коха, трикутник Серпінського, дракон "Хартера-Хейтуя..."
- Використовуються афінні перетворення: зсув (переміщення), масштабування, поворот



Для побудови лінії Коха на растрі 640x350 потрібний набір афінних перетворень, що складається з чотирьох перетворень:

$$X' = 0.333X + 13.333 ;$$

$$Y' = 0.333Y + 200 ;$$

$$X' = 0.333X + 413.333 ;$$

$$Y' = 0.333Y + 200 ;$$

$$X' = 0.167X + 0.289Y + 130 ;$$

$$Y' = -0.289X + 0.167Y + 256 ;$$

$$X' = 0.167X - 0.289Y + 403 ;$$

$$Y' = 0.289X + 0.167Y + 71 .$$

У 1988 році відомі американські фахівці в теорії динамічних систем Барнслі і Слоан запропонували деякі ідеї, засновані на міркуваннях теорії динамічних систем, для стискування і зберігання графічної інформації.

Вони назвали свій метод методом фрактального стискування інформації. Походження назви пов'язане з тим, що геометричні фігури, що виникають в цьому методі, зазвичай мають фрактальну природу в сенсі Мандельброта.

# Використання фракталів

На підставі цих ідей Барнслі і Слоан створили алгоритм, який, за їх твердженням, дозволить стискувати інформацію в 500-1000 разів.

# Коротко про метод

Зображення кодується декількома простими перетвореннями (у нашому випадку афінними), тобто коефіцієнтами цих перетворень (A,B,C,D,E,F).

Наприклад, закодувавши якесь зображення двома афінними перетвореннями, ми однозначно визначаємо його за допомогою 12-ти коефіцієнтів.

Якщо тепер задатися якою-небудь початковою точкою (наприклад  $X=0 Y=0$ ) і запустити ітераційний процес, то ми після першої ітерації отримаємо дві точки, після другої – чотири, після третьої – вісім і так далі. Через декілька десятків ітерацій сукупність отриманих точок описуватиме закодоване зображення.

Але проблема полягає в тому, що дуже важко знайти коефіцієнти IFS, які кодували б довільне зображення.

Фрактальний метод стиску  
використовується для економного  
збереження зображень без втрати  
якості.

FIF (Fractal Image Format) – спеціальний  
формат файлів

# Фрактальна графіка у пакетах наукової візуалізації

використовується для побудови ілюстрацій, що імітують природні процеси.

Серед таких програмних засобів продукти фірми Golden SoftWare: *Surfer* - створення дво-, тривимірних поверхонь;

*Map Viewer* - побудова кольорових карт.

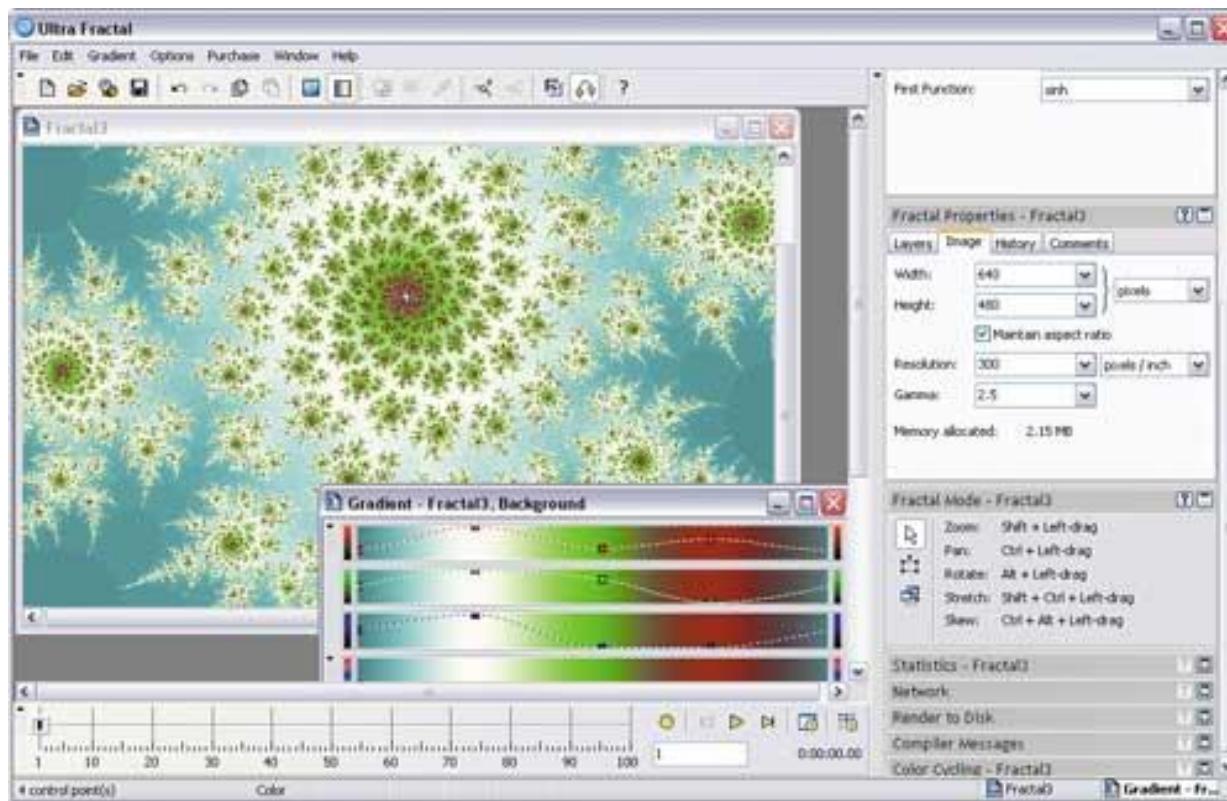
*Surfer* дозволяє опрацьовувати та візуалізувати двовимірні набори даних, що описані функцією  $z=f(x,y)$ . Можна побудувати цифрову модель поверхні, застосувати допоміжні операції і візуалізувати результат.

*Map Viewer* дозволяє вводити та корегувати карти - змінювати масштаб, перетворювати координати, обробляти й виводити у графічному вигляді числову інформацію, пов'язану з картами.

Пакет *Iris Explorer* (фірма Graphics) призначена для створення моделей погодних умов та поверхонь океану.

Пакет *Earth Watch* (фірма Earth Watch) призначений для моделювання та демонстрації тривимірного зображення метеоумов над Землею, дозволяє будувати топологічні поверхні і прогнозувати погоду.

# Программа *Fractal Explorer*



# Генерація фракталів 2 способами

- на основі базових фрактальних зображень, побудованих по вхідних формулах,
- з нуля.

- Перший варіант дозволяє отримати результати просто (вибрати формулу не важко). Можна змінити колірну палітру, добавити до нього фонове зображення. Можна фрактальне зображення трансформатувати, при необхідності масштабувати, змінити розміри зображення.

- Створення зображення з нуля набагато складніше і передбачає вибір одного з двох способів. Можна вибрати тип фрактала майже з 150 варіантів. А потім вже перейти до зміни різноманітних параметрів: налаштуванню палітри, фону і ін. А можна спробувати створити свою призначену для користувача формулу, скориставшись вбудованим компілятором.

- **XaosPro** – безкоштовний генератор фрактальних зображень (анімовагий рух у фракталі).
- **Apophysis** – інструмент для генерації фракталів на основі базових фрактальних формул.
- **XenoDream** – створення різноманітних об'ємних структур шляхом комбінування простих форм і фрактальних зображень, отриманих із застосуванням IFS-методів.
- **Fractracker** – створення тривимірних зображень на базі фрактальної геометрії, що представляє собою щось середнє між генератором фракталів і 3D-редактором.

- Incendia - це мультипроцесорний генератор тривимірних фракталів, розроблений іспанським програмістом Ramiro Perez, який вивчає фрактали з 1989 року.
- Fractal Zoomer - компактний фрактальний генератор

# *Книга природи написана мою фракталів*

Фрактальний світ добре відображає реальний, оскільки властивості фракталів демонструють багато природних об'єктів.

Фрактальна математика сьогодні дуже широко застосовується, вона може бути використана для аналізу змін цін і заробітної платні, статистики помилок на телефонних станціях і опису Всесвіту у цілому.

# Фрактальним підходом можна описувати

- структури неживої природи: лінії берегів, рельєф місцевості, обриси хмар, структури корисних копалин,
- структури живої природи: системи кровообігу людини, будови нирок і легенів, які нагадують по структурі дерева з кроною,
- процеси: економічні, турбулентні, які використовуються при прогнозі погоди.

# Алгоритми фрактальної геометрії

використовують для стиснення зображень, дистанційному зондуванні і радіолокації, моделюванні фракталоподібних розсіювальних систем, еволюційних обчислень, тощо.



# Що таке 3D-графіка?

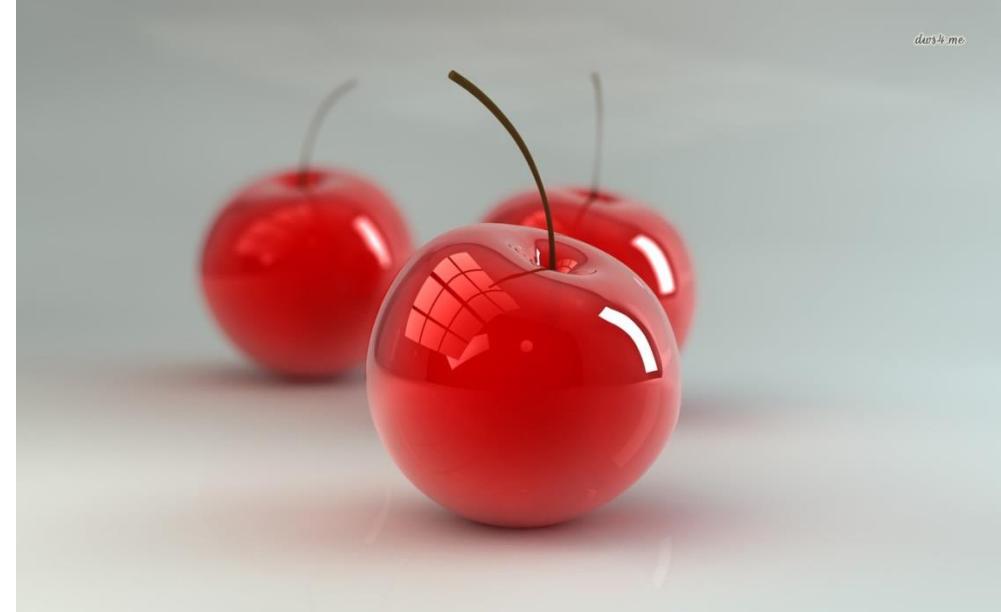
## **2 трактування:**

- Симуляція тривимірності на плоских об'єктах
- Відображення 3D- об'єктів на 3D-пристроях

# Тривимірна графіка

**3D графіка** (3 Dimensions) – один з розділів комп’ютерної графіки, який призначений для зображення об’ємних об’єктів (на площині).

Дана графіка застосовується в кінематографі, архітектурній справі, комп’ютерних іграх, поліграфії, науці та промисловості.



- З появою 3D-пристроїв, розвитком віртуальної, доповненої реальності виокремилося поняття 3D-графіки (у «чистому» вигляді)



[https://www.youtube.com/watch?v=2\\_UyVjCNbc0](https://www.youtube.com/watch?v=2_UyVjCNbc0)

<https://hightech.plus/2018/07/25/predstavlen-golograficheskii-displei-dlya-3d-razrabotchikov>

<http://hi-news.pp.ua/tehnika-tehnologiyi/12425-golografchniy-ekran-opis-pristry-princip-roboti.html>

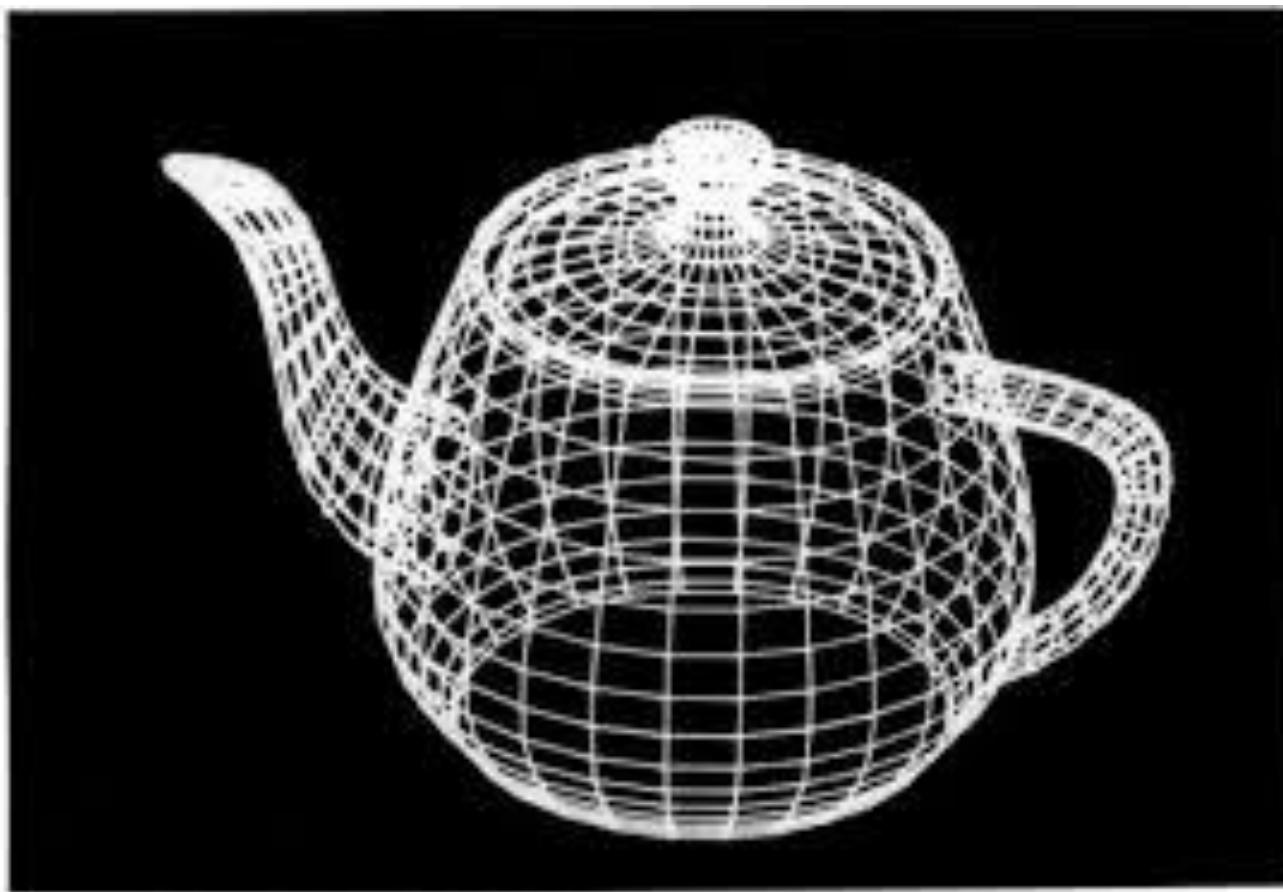
Основна відмінність від 2D графіки – опис і представлення зображення в трьох площинах, яке в подальшому перетворюється в 2D для виводу на екран або папір.

3D і 2D графіка тісно взаємоповязані між собою, оскільки тривимірна графіка використовує алгоритми 2D векторної і 2D растрової графіки.

Здебільшого для побудови 3D об'єкта достатньо мати уявлення про його лицевий вигляд, вигляд збоку та вигляд зверху.

У тривимірній комп'ютерній графіці всі об'єкти зазвичай представляються як набір поверхонь або частинок.

Мінімальну поверхню називають **полігоном**.  
В якості полігона зазвичай вибирають трикутники.



# Тривимірна графіка для реалістичних зображень

3D-графіка призначена для імітації тривимірних образів об'єктів, які попередньо створюються в пам'яті комп'ютера в такій послідовності:

- 1. попередня підготовка,**
- 2. створення геометричної моделі сцени,**
- 3. настроювання освітлення і знімальних камер,**
- 4. підготовка і призначення матеріалів,**
- 5. візуалізація сцени.**

Таким чином створюється уявний світ, який часто називають *віртуальним*.

Тривимірна графіка використовує віртуальний простір, який називають **сценою**.

У цьому просторі проектувальник розміщує необхідні об'єкти, призначає для них певний матеріал (дерево, залізо, скло тощо), розміщує джерела світла, а також віртуальні камери, що визначають точки перегляду сцени.

Під час відтворення тривимірної графіки на екрані комп'ютера будується геометрична проекція тривимірної моделі на площину екрану. Цей процес називають **рендерингом** або **візуалізацією**.

Перші **чотири** пункти є *підготовчими*,  
а **останній** пункт – це формування  
зображення (*рендеринг*).

# 1. Попередня підготовка

передбачає продумування складу сцени, розміщення об'єктів і їх деталей, які будуть видимими з передбачуваних напрямів спостереження.

На цьому етапі складається вміст сцени.

Треба передбачити всі об'єкти та їх деталі, тому бажано мати намальований ескіз.

## 2. Створення геометричної моделі сцени

- це **формалізований** опис тривимірної геометричної моделі об'єктів, що мають ширину, довжину та висоту.

Об'єкти розташовуються у тривимірному просторі, причому їх можна вкладати у середину інших об'єктів.

Набір інструментів по створенню геометричних моделей називається **геометричним конструктором сцен**.

Після створення геометричної моделі, сцену можна розглядати з довільного ракурсу.

### 3. Робота над композицією: світло та камери

- це налаштування моделей джерел освітлення та розставлення зйомочних камер.

Правильний вибір джерел світла дозволяє виконувати імітацію фотографування сцени в будь-яких умовах освітленості.

Освітленість всіх об'єктів, тіні від них і бліки світла розраховуються автоматично.

Моделі знімальних камер дають можливість розглядати тривимірну сцену і виконувати її знімання під будь-яким вибраним кутом зору.

## 4. Підготовка та призначення матеріалів

- надання сцені візуальної правдоподібності, наближує якість зображення до реальної фотографії.

Працюючи з матеріалами, можна настроювати такі їх характеристики, як сила блиску, прозорість, дзеркальність, рельєфність та інші. Реальні фотографії можна включати в склад матеріалів або використовувати для імітації фону.

Є інструменти зі створення нових матеріалів або вибір готових матеріалів із бібліотек

# ОТЖЕ, СЦЕНА (віртуальний простір моделювання) включає в себе такі категорії :

- **Геометрія** ( побудована за допомогою різних технік модель, наприклад будівля);
- **Матеріали** (інформація про візуальні властивості моделі, наприклад колір стін і відбиваюча / заломлююча здатність вікон)
- **Джерела світла** (налаштування напрямків, потужності, спектра освітлення);
- Віртуальні камери (вибір точки та кута побудови проекції);
- **Сили та дії** (налаштування динамічних спотворень об'єктів, застосовується в основному в анімації);
- **Додаткові ефекти** (об'єкти, що імітують атмосферні явища: світло у тумані, хмари, полум'я і пр.)

Завдання тривимірного моделювання — описати ці об'єкти і розмістити їх у сцені з допомогою геометричних перетворень відповідно вимогам до майбутнього зображення.

## 5. Візуалізація сцени

полягає в проведенні програмою обчислень і нанесення на зображення всіх тіней, бліків, взаємних відблисків об'єктів і т. п.

- Час візуалізації сцени, тобто формування сцени, залежить від складності сцени та швидкодії комп'ютера.

Для отримання тривимірного зображення потрібно виконати такі етапи:

- **моделювання** - створення математичної моделі сцени і об'єктів в ній (2,3,4).
- **рендерінг** (rendering) - побудова проекції відповідно до вибраної фізичної моделі (5).

# Рендеринг

На цьому етапі математична (векторна) просторова модель перетворюється на плоску (растрову) картинку.

Існує декілька технологій візуалізації, часто комбінованих разом.

- *Z-буфер* – алгоритм, який відповідає за створення зображень 3D-об'єктів, спираючись на глибину елементів зображення.

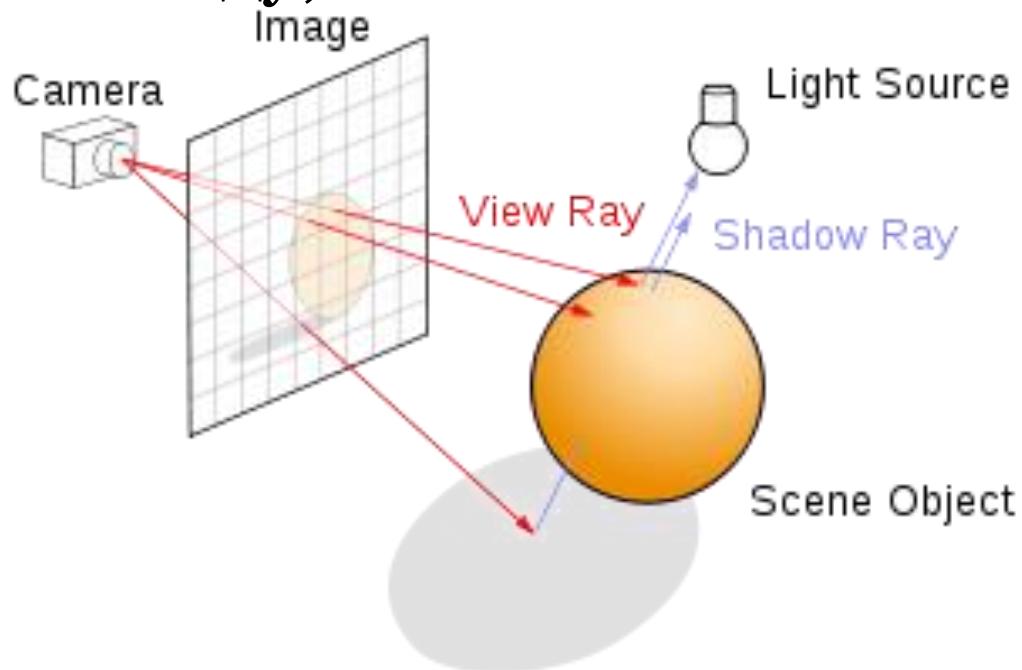
Використовується в «OpenGL» і «DirectX».

При створенні (візуалізації) 3d- об'єкту його глибина генерується на осі Z-координат і зберігається у Z-буфери. Z-буфер є двомірним масивом (X, Y- координати) із глибиною для кожного екранного пікселя. Коли інший об'єкт сцени повинен бути відображенний у цьому пікселі зараз, тоді порівнюється дві глибини та перекривається поточний піксель, якщо об'єкт знаходиться більше до спостерігача. Обрана глибина зберігається в Z-буфери і замінює попередню. Z-буфер дозволяє правильно відтворювати звичне для нас сприйняття об'єктів, розміщених на різних відстанях.

- *Сканлайн* («кидання променю», спрощений алгоритм зворотного трасування променів) - розрахунок кольору кожної точки картинки на основі побудови променю з точки зору спостерігача через уявний отвір в екрані на місці цього пікселя «в сцену» до перетину з першою поверхнею. Колір пікселя буде таким же, як колір цієї поверхні;

Це спрощений, нерекурсивний варіант трасування променів, не відбувається подальша обробка відбитих чи заломлених променів, а враховується лише перша поверхня-перешкода на шляху променів.

- *Трасування променів (Ray tracing, рейтрейсинг)* - те ж, що і сканлайн, але колір пікселя уточнюється за рахунок побудови додаткових променів від точки перетину променю погляду;



- Ідея алгоритму полягає в тому, щоб простежити хід променя від уявного ока глядача крізь кожен піксел на уявному екрані і обчислити колір об'єкта, видимого оком крізь нього.

- *Глобальне освітлення*— розрахунок взаємодії поверхонь і середовищ у видимому спектрі випромінювання за допомогою інтегральних рівнянь.

Такі алгоритми враховують не лише *пряме світло* , що надходить безпосередньо від джерела світла, але також і *відбиті промені світла* від інших поверхонь об'єктів сцени.

# Питання

Чи не простіше сфотографувати реальну сцену?

Є випадки, коли використання тривимірної графіки є єдиним можливим засобом рішення.

# Області застосування тривимірної графіки:

## Комп'ютерне проектування:

- швидке вирішення задач проектування інтер'єрів;
- будовування вигаданої сцени у зображення реального світу;
- будовування зображення реального об'єкта у тривимірну сцену як складової (віртуальна галерея);

## Автоматизоване проектування:

- синтез зовнішнього вигляду складних деталей, візуальний вигляд автомобілів, літаків, пароплавів.

## Комп'ютерні ігри:

- найпопулярніша ділянка використання тривимірної графіки. По мірі удосконалення програмних засобів моделювання, зросту продуктивності та збільшення ресурсів пам'яті комп'ютерів віртуальні світи стають більш складними й подібними до реальності.

## Комбіновані зйомки:

- тривимірну графіку застосовують там, де зробити реальні фотографії просто неможливо, або потребує великих витрат (внутрішня будова працюючого двигуна, науково-фантастичні сюжети, нереальні світи, відеомонтаж, реклама тощо). Практично застосовується у книжковій та журнальній графіці і є популяризацією науки, реклами, художньої творчості.

# Формати файлів 3d графіки

Мова VRML (Virtual Reality Modelling Language) призначена для опису тривимірних зображень і операє об'єктами, що описують геометричні фігури та їх розташування в просторі.

Мова була створена в 1994 році консорціумом на чолі з Silicon Graphics для застосування в мережах INTERNET. Чинним стандартом є VRML 97, відомий як ISO / IEC 1477.

WRL-файл – звичайний текстовий файл зі списком об'єктів, які названі вузлами.

3DS – файл пакету 3D Studio MAX, зберігає 3D-моделі у формі каркасних сіток, текстуру, ефекти тощо.

MAX – ще один формат 3D Studio MAX.

X – “рідний” формат DirectX, структурно- та контекстнонезалежний формат.

# Програмні пакети, що дозволяють створювати тривимірну графіку

- Autodesk 3ds Max
- Autodesk Maya
- Autodesk Softimage
- Blender
- Cinema 4D
- Zbrush
- ...

# Доповнена реальність

- (з англ. augmented reality або AR) – це доповнення фізичного світу за допомогою цифрових даних, яке забезпечується комп'ютерними пристроями (смартфонами, планшетами та окулярами AR) в режимі реального часу.

# VR (Virtual Reality) і AR

- VR вимагає **повного занурення** у віртуальне середовище,
- AR використовує **середовище навколо нас** та просто **накладає** поверх нього певну **частину віртуальної інформації** (наприклад, графіку, звуки та реакцію на дотики).

Оскільки віртуальний та реальний світи гармонійно співіснують в AR, користувачі мають змогу спробувати цілком новий, покращений світ, де **віртуальна інформація** використовується як **додатковий** корисний інструмент, що забезпечує допомогу в повсякденній діяльності.

# Знайомимося з AR

- <http://thefuture.news/page1837780.html>

Є декілька різних технологій, які використовуються для роботи AR

- базується на **маркерах**
- **безмаркерна**
- базується на **проекції**
- базується на **VIO** (візуальна інерціальна одометрія)

# Доповнена реальність (AR)

— проектування будь-якої цифрової інформації (зображення, відео, текст, графіка і т.д.) поверх екрану будь-яких пристрій. В результаті реальний світ доповнюється штучними елементами і новою інформацією. Може бути реалізована за допомогою додатків до звичайних смартфонів і планшетів, окулярів доповненої реальності, стаціонарних екранів, проекційних пристрій та інших технологій.

# Змішана реальність (MR)

— проектування тривимірних віртуальних об'єктів чи голограм на фізичний простір. Дозволяє переміщуватись навколо віртуального об'єкту, оглядати його з усіх боків і, за потребою, всередині. Вимагає, як правило, спеціального обладнання (окулярів чи шоломів).

- <https://nv.ua/ukr/project/smart-okulyari-hololens-2-vid-microsoft-dopomagayut-medikam-ryatuvati-zhittya-podrobici-50150051.html>

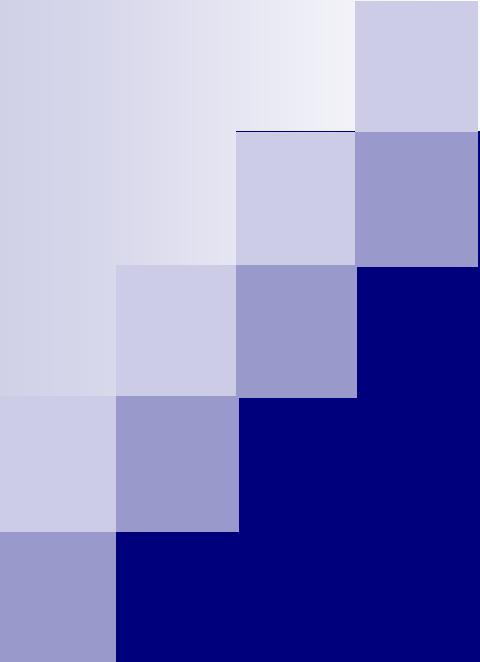
# Загальна схема створення доповненої реальності

- камера пристрою AR **знімає** зображення реального об'єкта;
- програмне забезпечення пристрою **проводить** ідентифікацію отриманого зображення та візуальне доповнення,
- **послидує** реальне зображення з його доповненням
- **виводить** кінцеве зображення на пристрій візуалізації.

# Самостійна робота!!!

- Проаналізувати сучасні 3d-технології відображення і сформувати коротку інформаційну довідку (1-2 стор. А4)

Зокрема, **стереоскопічні, голографічні, мультипланарні** ... дисплеї



# Комп'ютерні моделі кольорів. Частина 2.

## Лекція 7

# Класифікація моделей

Розглянемо

**апаратно-орієнтовані** триколірні моделі

RGB (aRGB, sRGB), CMY (CMYK)

**апаратно-незалежні** триатрибутні моделі,

які поділяються на

**перцепційні** HSV, HSL...

та

**модифіковані** XYZ, Lab...

# Перцепційні моделі

- HSV
- HSL
- HSB
- HIS
-

# Моделі кольорів HS\*

використовуються, щоб позбутися обмежень, що накладаються апаратним забезпеченням.

Ці моделі використовуються, коли коректують яскравість, насиченість, перетворюють зображення в сіре та ін.

HSV задається трьома атрибутами:

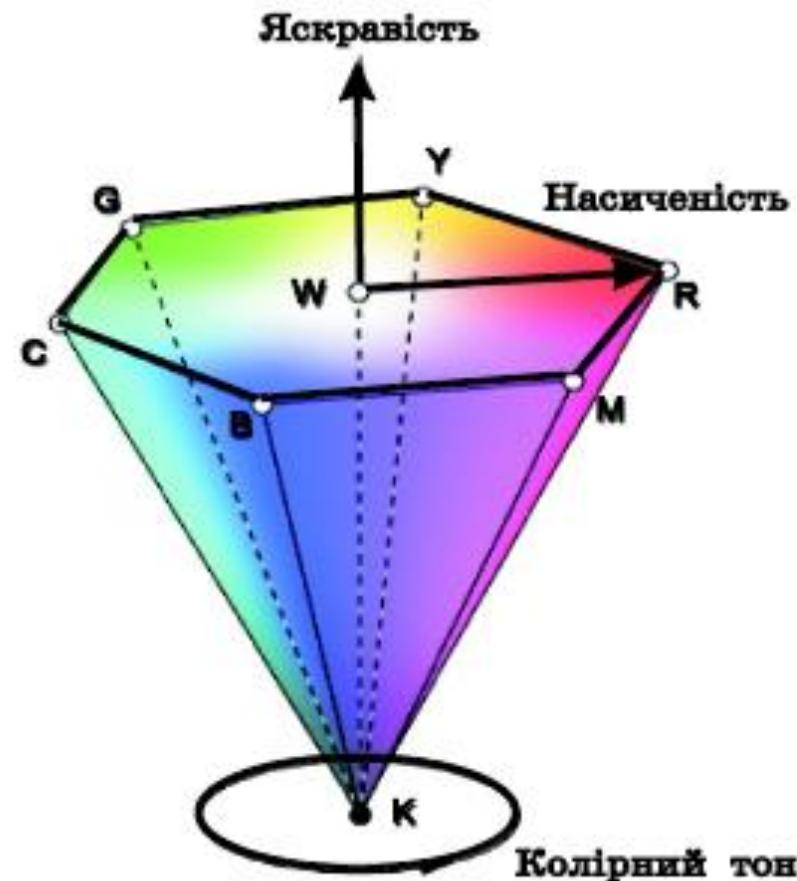
**Hue** – колірний тон,

**Saturation** - насиченість ,

**Value** – інтенсивність.

Ця модель побудована на основі суб'єктивного сприйняття кольору людиною і добре узгоджується з нею.

# Колірний простір HSV



# Компонента Н

визначається довжиною хвилі домінуючої компоненти в спектрі, (“чистий пігмент”).

Колірний тон вимірюється в градусах кута.

Доповнюючі кольори відрізняються на  $180^\circ$

**Червоний**  $0^\circ$  – **блакитний**  $180^\circ$

**Жовтий**  $60^\circ$  – **синій**  $240^\circ$

**Зелений**  $120^\circ$  – **пурпурний**  $300^\circ$

- Чому Пінгвіни Живуть Зимою Без Своїх Фантазій?
- Чому Пан Жабу З"їв, Бо Страва Французька.

# Компонента S

змінюються від 0 до 1.

S – *насиченість або чистота кольору*, співвідношення між домінуючою компонентою кольору і сумішшю решти кольорів.

S вказує наскільки колір близький до “чистого” пігменту і визначається відстанню до осі піраміди, тобто наскільки далеко знаходиться в просторі від рівного за ним за яскравістю білого кольору.

Пастельні кольори мають .... насиченість.

Кольорова фотографія з насиченістю 0 – це ...

# Компонента V

змінюються від 0 до 1.

V – інтенсивність, вказує на загальну кількість світлового потоку, що потрапляє в око, (яскравість), тобто наскільки сильно колір буде сприйматися оком.

V=0 відповідає вершині і задає чорний колір.

Вісь піраміди задає сірі кольори (S=0) від чорного до білого (центр основи).

Максимальна яскравість для довільного кольору дає відчуття ... кольору.

# V і H є залежними.

- Зміна V дає зміну H .
- При зменшенні V зелені кольори синіють, сині стають фіолетовими, жовті – оранжевими, оранжеві – червоними.

# RGB to HSV

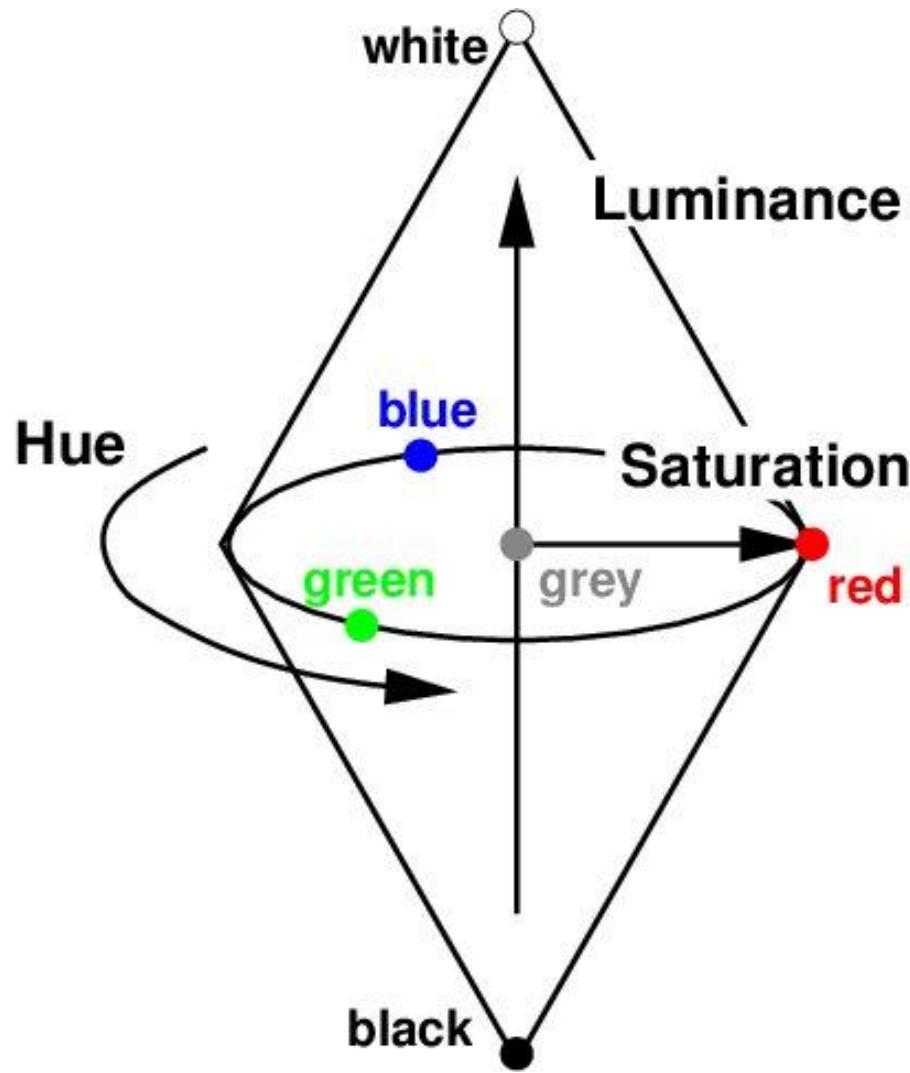
- *Divide r, g, b by 255*
- *Compute cmax, cmin, difference*
- *Hue calculation :*
  - *if cmax and cmin equal 0, then h = 0*
  - *if cmax equal r then compute  $h = (60 * ((g - b) / diff) + 360) \% 360$*
  - *if cmax equal g then compute  $h = (60 * ((b - r) / diff) + 120) \% 360$*
  - *if cmax equal b then compute  $h = (60 * ((r - g) / diff) + 240) \% 360$*
- *Saturation computation :*
  - *if cmax = 0, then s = 0*
  - *if cmax does not equal 0 then compute  $s = (diff/cmax)*100$*
- *Value computation :*
  - $v = cmax * 100$

HSL – модель аналогічна до попередньої.

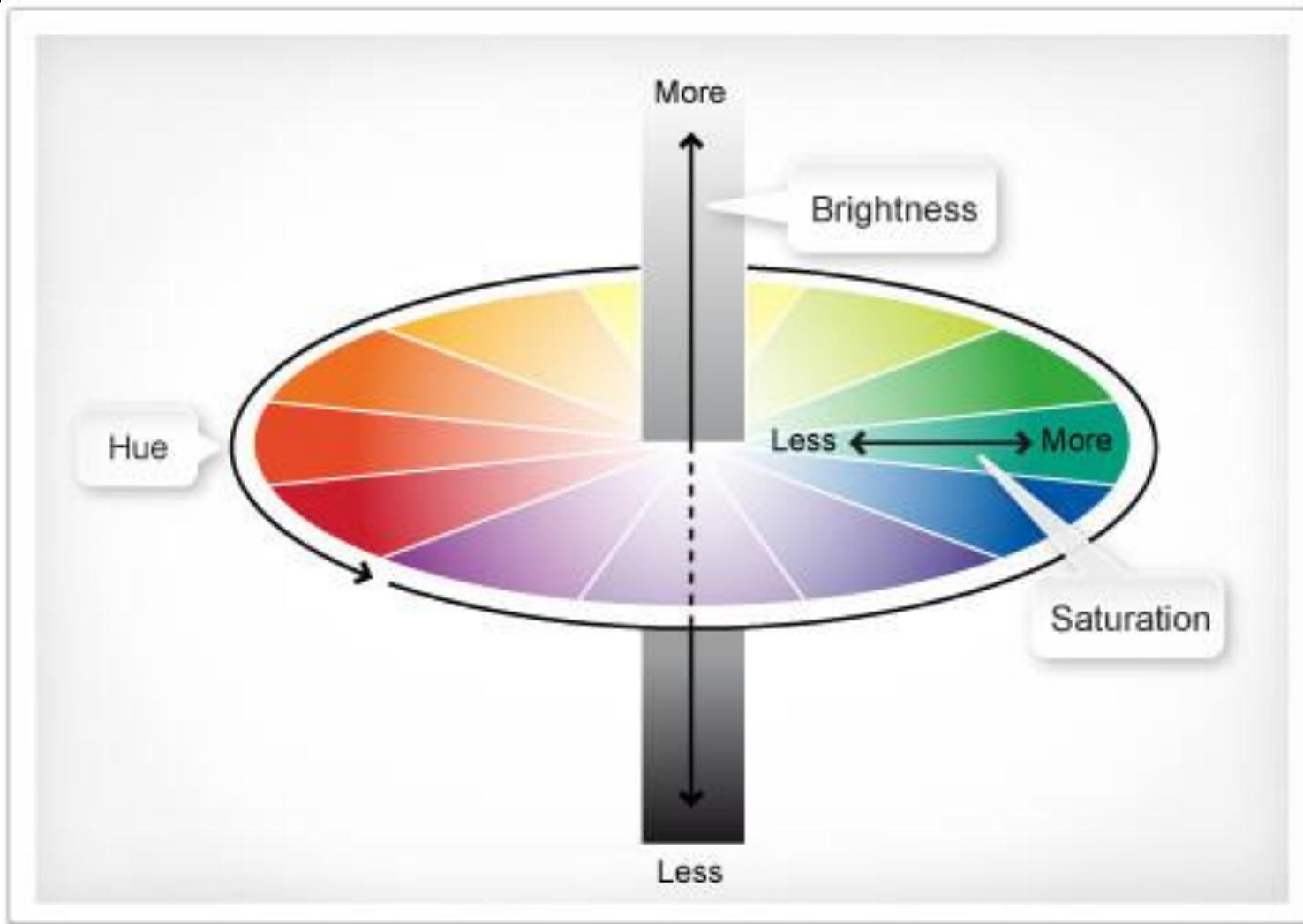
Відмінність в компоненті L – освітленість, вказує величину чорного відтінку, доданого до кольору.

Простір HSL – це подвійна піраміда, яку отримують витягуванням у HSV-піраміді точки  $S=0$ ,  $V=1$  (білий колір) вгору .

# HSL



# HSB



# Інші моделі

- CIE Lab
- CIE XYZ
- CIE Luv
- (Y,Cb,Cr)
- ...

- Міжнародна комісія з освітленості
- Commission internationale de l'éclairage

# Модель Lab

базується на людському сприйнятті кольору. При однаковій інтенсивності око людини сприймає промені **зеленого** кольору найбільш яскравими, дещо менш яскравими – **червоного** кольору, і ще менш яскравими – **синього**.

**Яскравість** при цьому є характеристикою сприйняття, а не характеристикою самого кольору.

При розробці Lab переслідувалася мета математичного коректування **нелінійності сприйняття кольору людиною**.

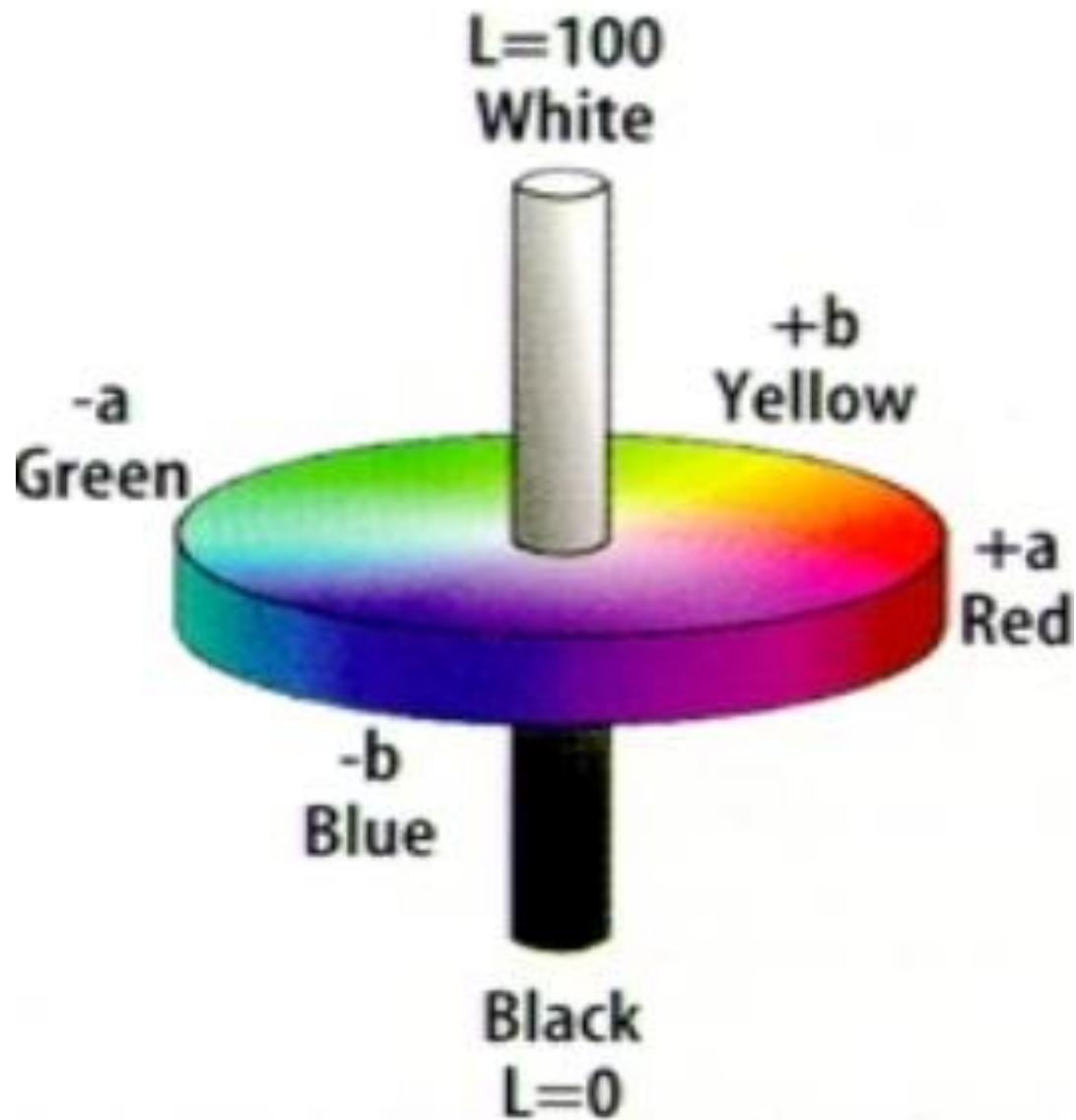
**L\*** – Lightness (світлосила, яскравість),

в межах [0-100],

**a\*** – канал кольорів від зеленого до червоного,

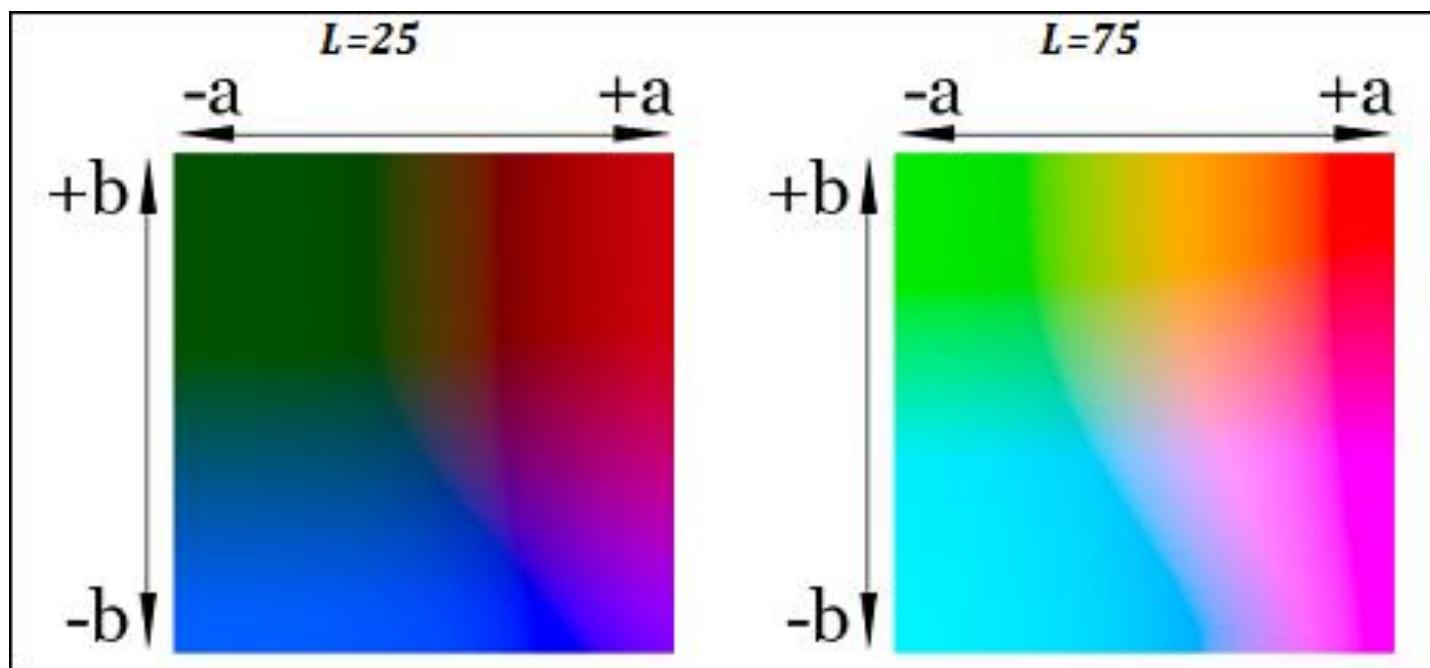
**b\*** – канал кольорів від синього до жовтого,

в межах [-200;200].



- Якщо  $a$  і  $b$  рівні 0, то змінюючи  $L$ , отримуємо зображення, що містить градації сірого.

представлено зрізи колірного тіла СІЕ  $L^*a^*b^*$  для двох значень светлоти:



- Оскільки, **яскравість** у моделі CIELab цілком відділена від **кольору**, то це робить модель зручною для регулювання *тонових характеристик* (підвищення контрасту, виправлення похибки тонових діапазонів), *видалення кольорового шуму*, ....
- Враховуючи **величезний колірний обсяг**, модель знайшла широке застосування в програмному забезпеченні для обробки зображень, через неї відбувається конвертація даних між іншими кольоровими просторами (наприклад, з RGB сканера в CMYK друкованого пристрою).

Перевагою колірної моделі  $L^*a^*b^*$  МКО (CIE Lab) є те, що вона не тільки ефективно вирішила проблему розробки рівноконтрастного колірного простору, але й фактично моделює процес представлення кольору апаратом людського зору.

Модель  $L^*a^*b^*$  МКО широко використовується у колориметрії та промисловості.

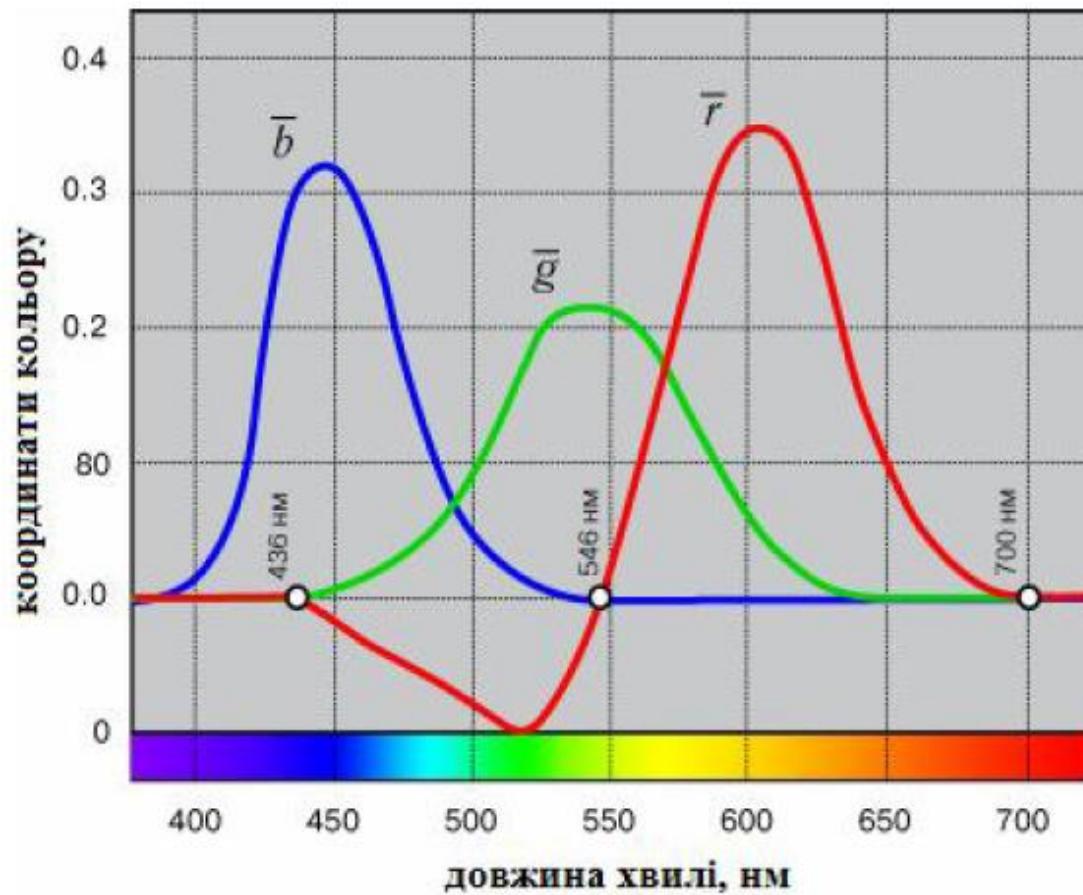
# CIE XYZ – еталонна модель

## Причини створення

Усі кольори, які були отримані Максвеллом шляхом змішування трьох основних кольорів із **додатними коефіцієнтами**, розташовуються **усередині трикутника**.

Колірний трикутник **не показує** нам всіх кольорів, які є **видимі людському оку**.

Це саме і є ті кольори, які в дослідах зі зрівнювання кольорів привели до появи **від'ємних коефіцієнтів** у кривих складання.



2. Криві накладання кольорів  $\bar{r}(\lambda)$ ,  $\bar{g}(\lambda)$ ,  $\bar{b}(\lambda)$  для стандартного колориметричного спостерігача за  $R = 700$  нм,  $G = 546,1$  нм,  $B = 435,8$  нм, які отримані на основі даних дослідів Гілда і Райта.

Для того, щоб уникнути від'ємних значень кривих додавання, до них були застосовані лінійні перетворення, в результаті яких було отримано нові криві додавання, які позначаються як  $x(\lambda)$  ,  $y(\lambda)$  ,  $z(\lambda)$  .

Ці криві відомі, як криві додавання кольорів для стандартного колориметричного спостерігача МКО 1931 р.

**Ці кольори є тільки фізичною абстракцією і виконують допоміжну математичну роль.**

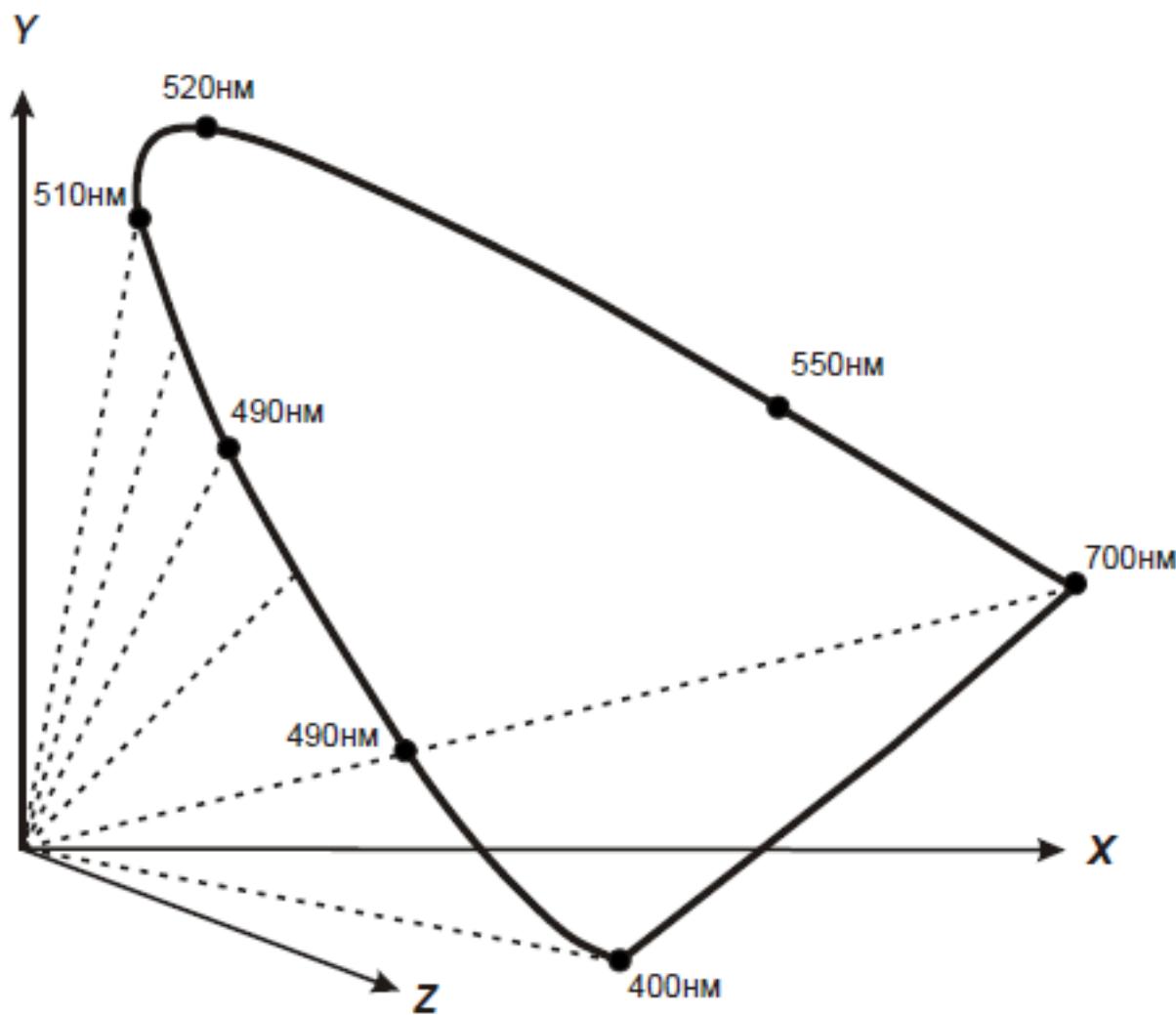
- Якщо світловий потік записати у вигляді  $I(\lambda)$  , то за допомогою функцій  $x(\lambda)$  ,  $y(\lambda)$  ,  $z(\lambda)$  знаходимо числа

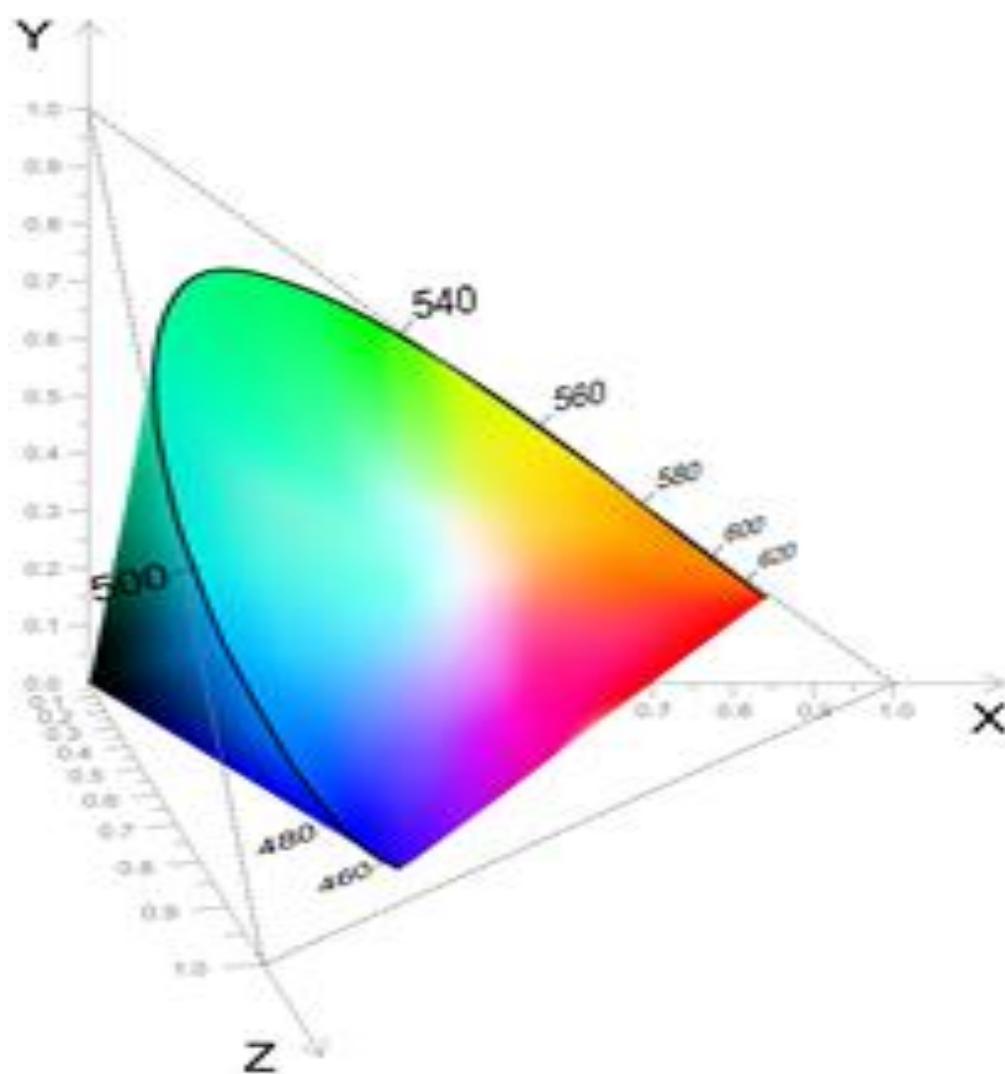
$$X = \int I(\lambda) x(\lambda) d\lambda ,$$

$$Y = \int I(\lambda) y(\lambda) d\lambda ,$$

$$Z = \int I(\lambda) z(\lambda) d\lambda .$$

# XYZ





У цьому колірному тілі будь-який колір, що сприймається оком, можна охарактеризувати однозначно.

Можна відзначити, що крива  $Y(\lambda)$  нагадує *криву чутливості*.

Величина  $Y$  виражає інтенсивність з урахуванням спектральної чутливості ока і називається **люмінантністю** (CIE Luminance).

З метою опису кольору введено так звані хроматичні координати

$$x = \frac{X}{X + Y + Z}, \quad y = \frac{Y}{X + Y + Z}.$$

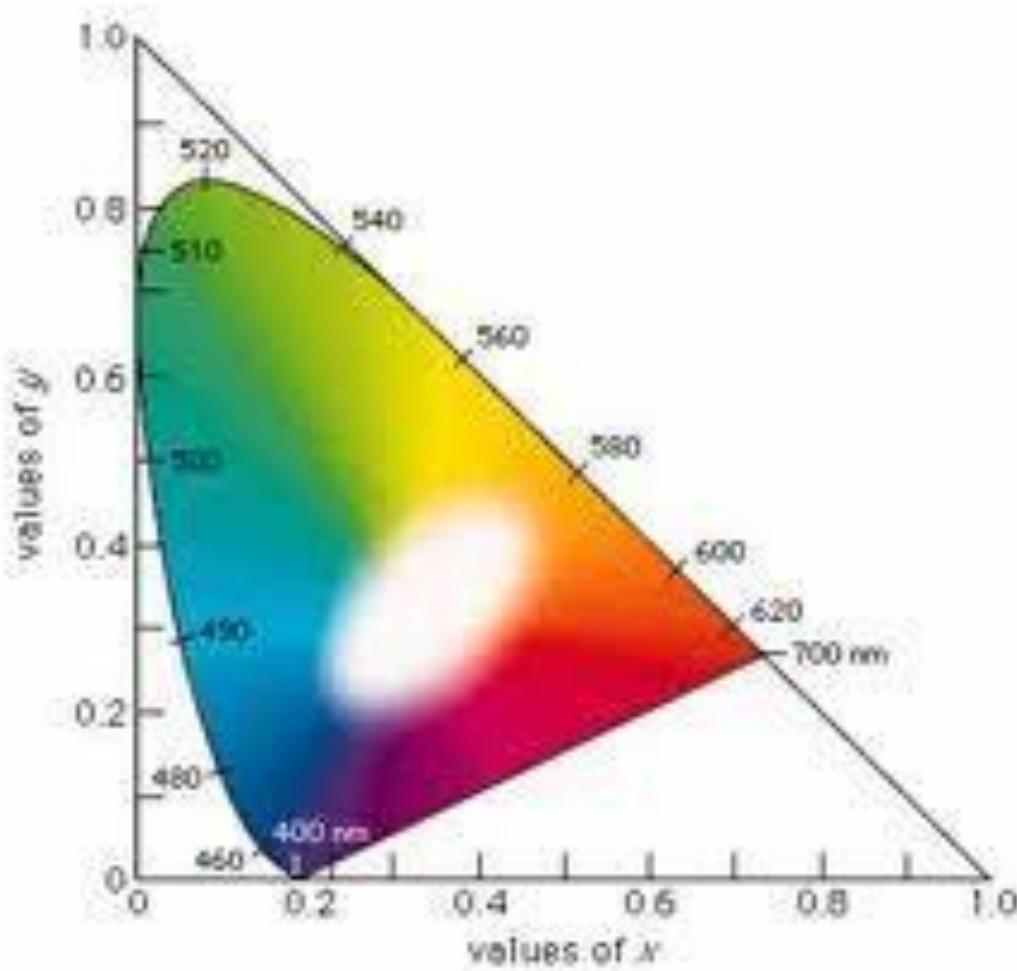
Будь-який колір, що сприймається оком, можна охарактеризувати трійкою чисел  $(x, y, Y)$ .

Оскільки працювати з об'ємним поданням колірного простору у вигляді неправильного конуса не дуже зручно, на практиці частіше користуються нормованим колірним простором, який отримав назву  $xyY$ .

# Хроматична діаграма (діаграма колірності)

У нормованому варіанті координати  $x$  і  $y$  зберігаються, а координата  $z$  зникає, оскільки цей варіант колірного простору двомірний. Вона будується шляхом проектування трикутника кольору на площину  $xy$ , тобто через січення простору площиною

$$x+y+z=\text{const}$$



- Система СІЕ XYZ створена шляхом математичних трансформацій системи СІЕ RGB і базується на тому ж принципу - будь-який колір можна точно специфікувати кількістю трьох випромінювань.
- Основна відмінність системи XYZ - колір її основних «випромінювань» існує тільки в колориметричних рівняннях, і отримати їх фізично неможливо.
- Основна причина створення системи XYZ - полегшення розрахунків. Координати кольору і колірності всіх можливих світлових випромінювань будуть додатніми.
- Також, координата кольору Y означає фотометричну яскравість.

# Основним недоліком

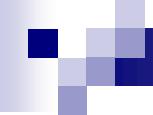
цієї системи є те, що використовуючи її, ми можемо констатувати тільки збіг чи розходження двох кольорів, але відстань між двома точками цього колірного простору **не відповідає** зоровому сприйняттю відмінності кольорів.

# Кодування кольорів

Компоненти в моделях задаються числами побайтно.

Наприклад, Windows API таке компонентне кодування кольору в адитивній схемі (4 байти)

00000000 bbbbbbbb gggggggg rrrrrrrr



Моделі дозволяють працювати з усіма можливими кольорами.

Проте це не завжди потрібно. Можна зекономити пам'ять, якщо працювати з певним набором кольорів, які є актуальними для зображень.

Палітра кольорів – це набір актуальних кольорів для певних зображень, представлений у формі таблиці.

Таблиця містить індекси кольорів. Тобто кольору відповідає індекс (як правило, байт).

Індекс	R	G	B	Колір
0	0	0	0	Чорний
255	255	255	255	Білий

# Як знати, які кольори актуальні?

Відповідь на це питання дає процес, який називають оптимізацією палітри.

# 1) Найпростіший підхід

полягає у переборі всіх пікселів картинки, і обрахунку, скільки разів зустрічається кожен колір. Палітра складається з тих кольорів, які зустрічаються частіше за інших.

Якщо деякий відтінок синього кольору зустрічається 100 разів, а відтінок червоного тільки 20, то перевага віддається синьому кольору. Основний недолік - деякі кольори будуть зовсім виключені з палітри.

## 2) Рівномірна розподіленість кольорів

Це вибір комплекту кольорів для палітри з рівномірно розподіленими червоною, зеленою і синьою компонентами. Такий підхід забезпечує широкий вибір кольорів.

Але при цьому не враховується, що в більшості картинок немає рівномірного колірного розподілу.

### 3) Метод квантування кольорів медіанним перетином

Колірний простір розглядається як тривимірний куб. Кожна вісь куба відповідає одному з трьох основних кольорів: червоному, зеленому, синьому. Кожна з трьох сторін розбивається на 255 рівних частин, поділки на осях нумеруються від 0 до 255.

Перший крок полягає у відображенні усіх колірних точок.

Другий крок полягає у відсіканні "країв" куба, які не містять пікселів.

Наприклад, якщо всіх пікселів значення червоної компоненти не менші, ніж 8 і не більше, ніж 250, то відкидаються частини куба від  $K=0$  до  $K=7$  і від  $K=251$  до  $K=255$ .

Третій крок - розрізання отриманого паралелепіпеда на два в серединній точці (медіані) найдовшої сторони.

Тепер паралелепіпед розділений на два паралелепіпеди меншого розміру.

# Наступні кроки

Далі попередній процес - відсікання порожніх "країв" і розрізання найдовшої сторони в серединній - повторюється для двох менших паралелепіпедів.

Тепер початковий куб роздільний на чотири паралелепіпеди.

Медіанний перетин повторно застосовується для того, щоб розділити куб на 8, 16, 32, 64, 128 і 256 паралелепіпедів.

# Формування палітри за центрами

Кожен з 256 паралелепіпедів містить піксели приблизно однакового кольору і центр кожного паралелепіпеда представляє оптимальне значення кольору для палітри.

Маючи координати вершин, дуже просто обчислити координати центральної точки.

# Формування палітри за усередненими значеннями

Це альтернативний метод до методу за центрами.

До палітри потрапляє середнє значення всіх пікселів, які знаходяться усередині паралелепіпеда.

На обчислення йде більше часу, але отримана палітра буде краще.

# Основи синтезу рухомих зображень

# Особливості людського зору

# Означення афінного перетворення

# Афінні перетворення мають такі властивості:

- ▶  $n$ -вимірний об'єкт відображається в  $n$ -вимірний, точка – в точку, лінія – в лінію, поверхня – в поверхню;
- ▶ зберігається паралельність ліній і площин;
- ▶ зберігаються пропорції паралельних об'єктів (довжин відрізків на паралельних прямих і площ на паралельних площинах).

Ці властивості дозволяють будувати прообрази полігонів на площині й поліедрів у просторі за скінченим набором точок – їх вершин.

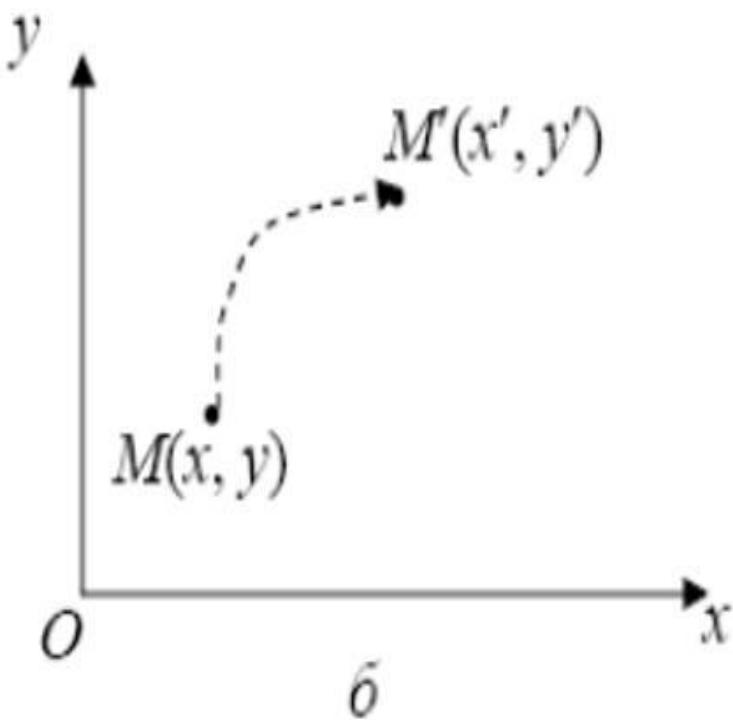
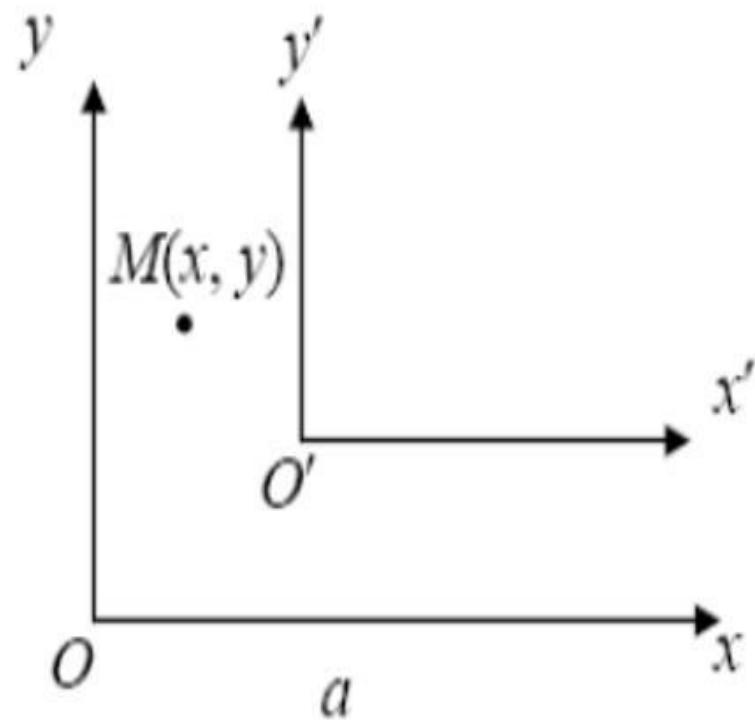
# Афінні перетворення З видів

є основою для рухомих  
зображень:

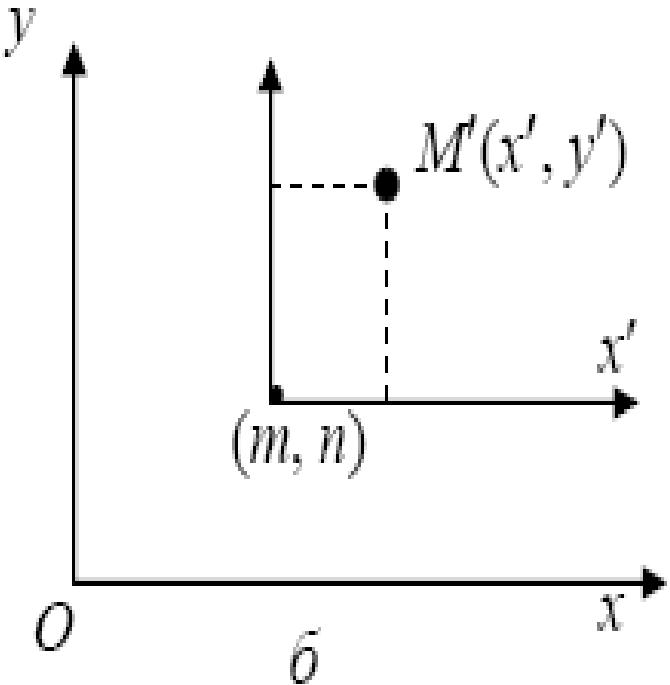
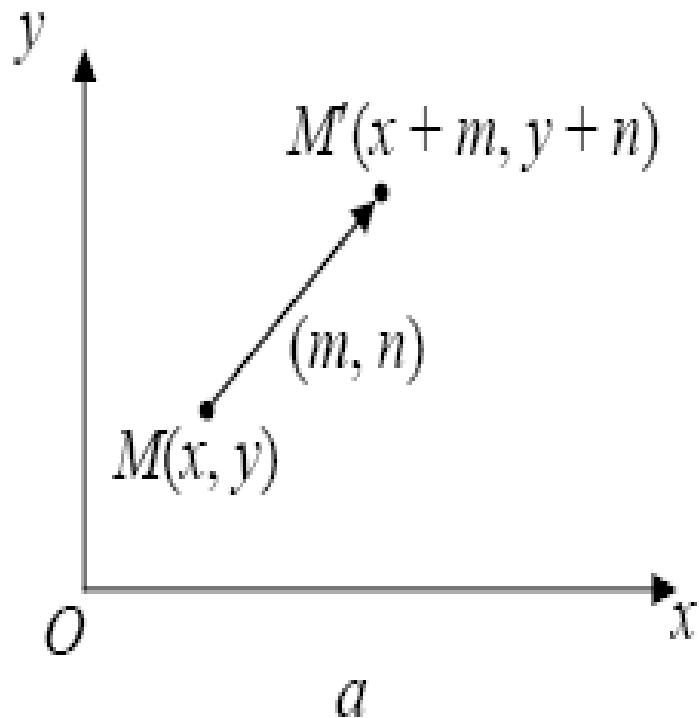
- ▶ переміщення/зсув;
- ▶ масштабування  
(збільшення/зменшення);
- ▶ поворот на кут

## 2 трактування перетворення координат

Ці перетворення можна проводити як відносно системи координат, так і відносно самого об'єкту зображення



# Перетворення Переміщення



а) паралельний зсув координат точки на вектор  $(m, n)$  в даній системі координат задається формулами :

$$\begin{aligned}x' &= x + m, \\y' &= y + n;\end{aligned}\quad (2)$$

б) зсув системи координат на вектор  $(m, n)$  задається формулами :

$$\begin{aligned}x' &= x - m, \\y' &= y - n\end{aligned}\quad (2')$$

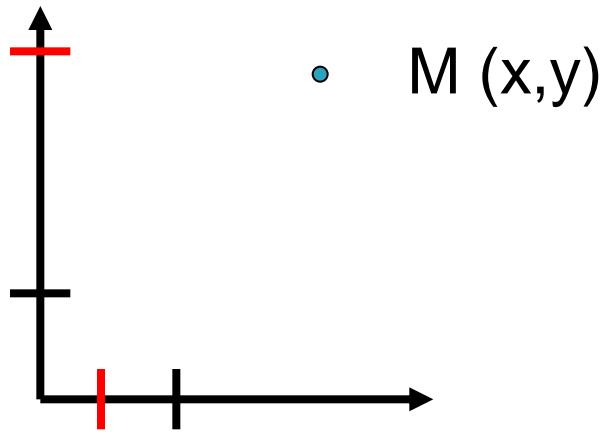
# Перетворення Масштабування

- розтяг/стиск вздовж координатних осей можна записати формулами

$$\begin{aligned}x' &= ax, \\y' &= dy.\end{aligned}\quad (3)$$

Якщо  $a = d$ , то маємо пропорційне масштабування, якщо  $a \neq d$ , то масштабування – непропорційне. При  $a = d > 1$  відбувається збільшення зображення, при  $a = d < 1$  – рівномірний стиск.

При  $a = 1, d = -1$  одержуємо дзеркальне відображення відносно осі  $x$ , при  $a = -1, d = 1$  – дзеркальне відображення відносно осі  $y$ .

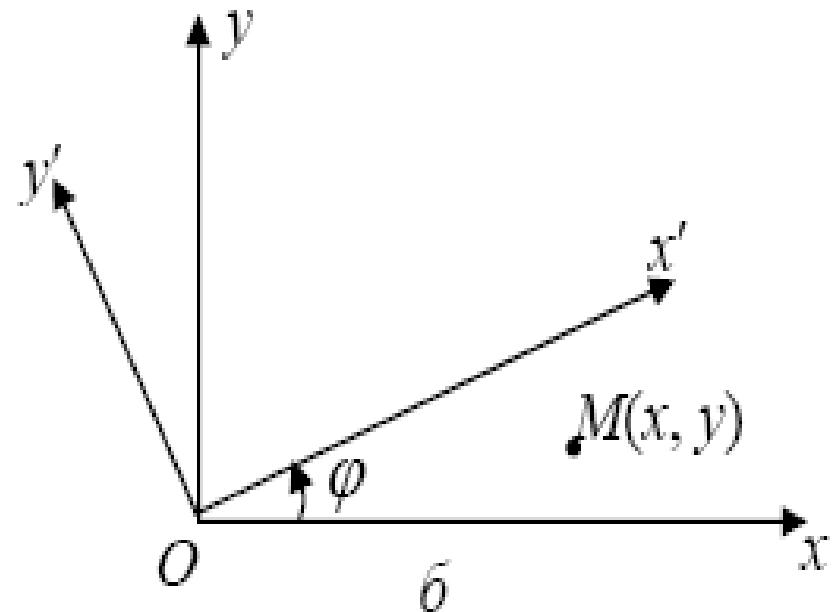
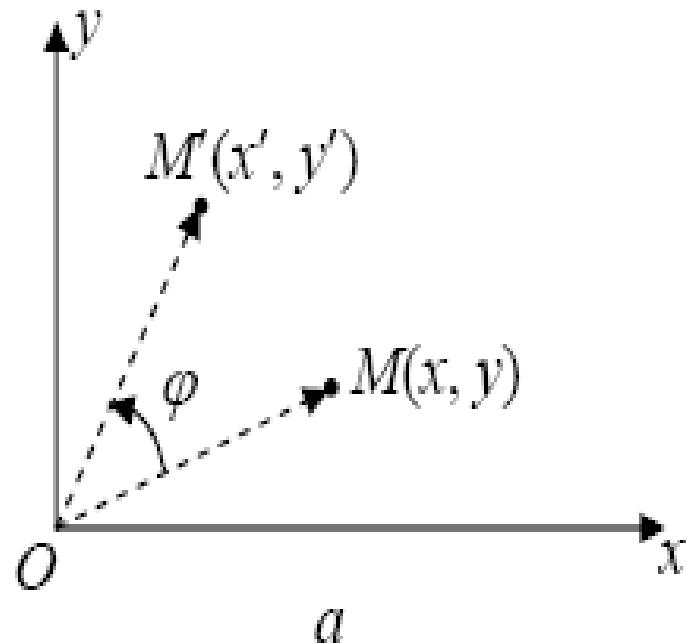


$$\begin{aligned}x' &= ax, \\y' &= dy. \quad (3')\end{aligned}$$

$a=1:i^` (1:1/2)$

$d=1:j (1:4)$

# Перетворення повороту



a) поворот точки відносно початку координат на кут  $\varphi$  проти годинникової стрілки задається формулами

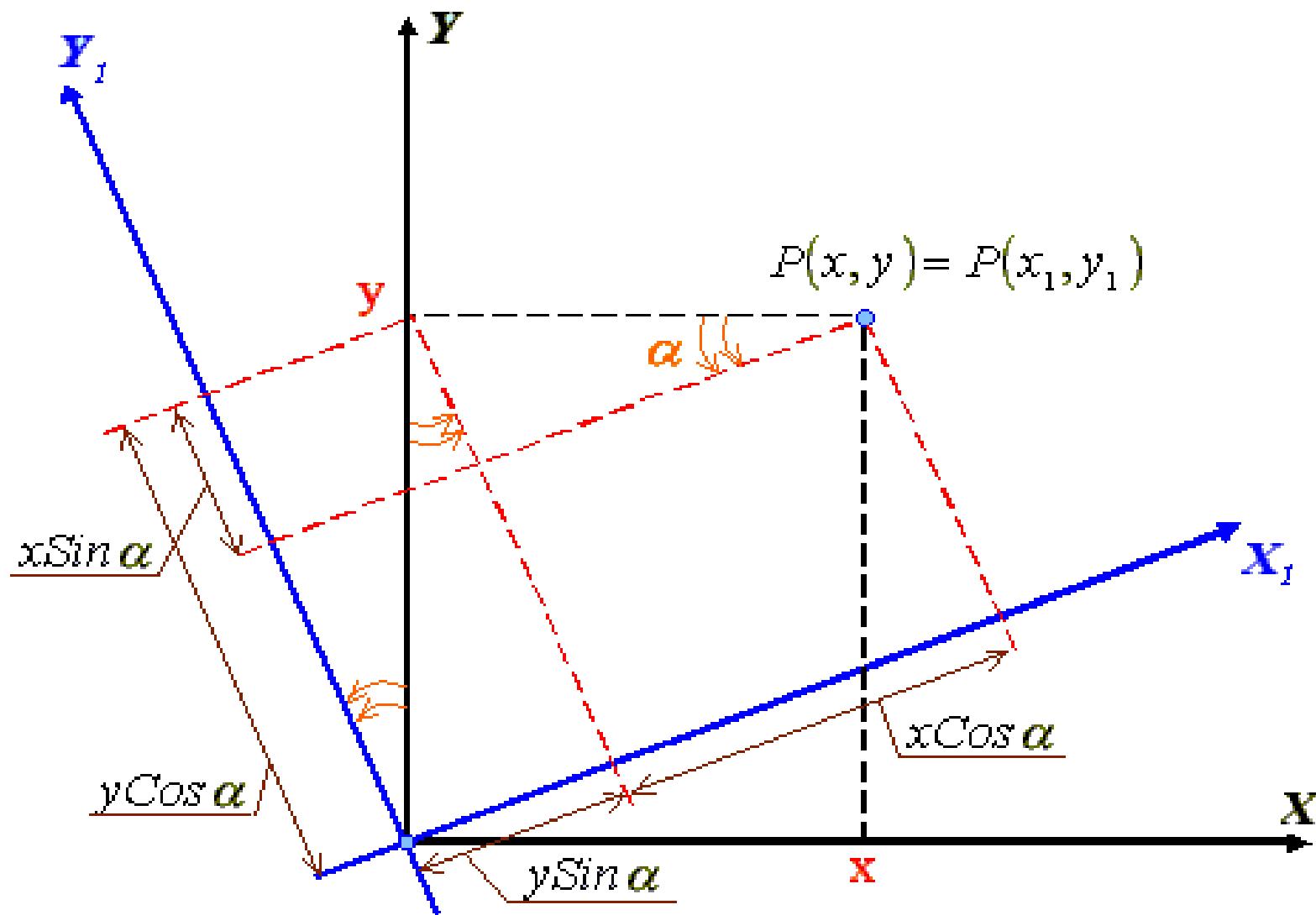
$$x' = x\cos\varphi - y\sin\varphi ,$$

$$y' = x\sin\varphi + y\cos\varphi ;$$

б) поворот системи координат на кут  $\varphi$  проти годинникової стрілки задається формулами

$$x' = x\cos\varphi + y\sin\varphi ,$$

$$y' = -x\sin\varphi + y\cos\varphi$$



Довільне афінне відображення (1) можна задати за допомогою композиції елементарних відображень (2) – (4) / (2') – (4').

Для ефективного використання цих формул у задачах КГ переходят до матричного запису

## Матричний запис масштабування (2)

$$\mathbf{A} = \mathbf{A}' \cdot S, \text{ де } S = \begin{pmatrix} a & 0 \\ 0 & d \end{pmatrix}, \quad a, d \neq 0.$$

# Матричний запис повороту (3)

$$\mathbf{A} = \mathbf{A}' \cdot R = (x \ y) \begin{pmatrix} \cos \varphi & \sin \varphi \\ -\sin \varphi & \cos \varphi \end{pmatrix}.$$

Матричний запис геометричних перетворень дає можливість здійснювати обернені перетворення, тобто точку (об'єкт)  $A'$  можна легко повернути в початковий стан  $A$ .

Наприклад, для повороту обернене перетворення  $A = A'R^{-1}$  задається матрицею

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

Обернена матриця водночас є транспонованою до заданої (такі матриці називаються ортогональними).

Цей факт є досить важливим для швидкодії алгоритмів.

Паралельне перенесення (2) не вдається задати перетворенням вигляду  $A' = A \cdot T$ ,

Не можна подати довільні сумарні афінні перетворення в матричній формі.

Щоб задати довільне перетворення (1) у матричному вигляді, необхідно перейти до однорідних координат.

# Однорідні координати

вводяться штучно:

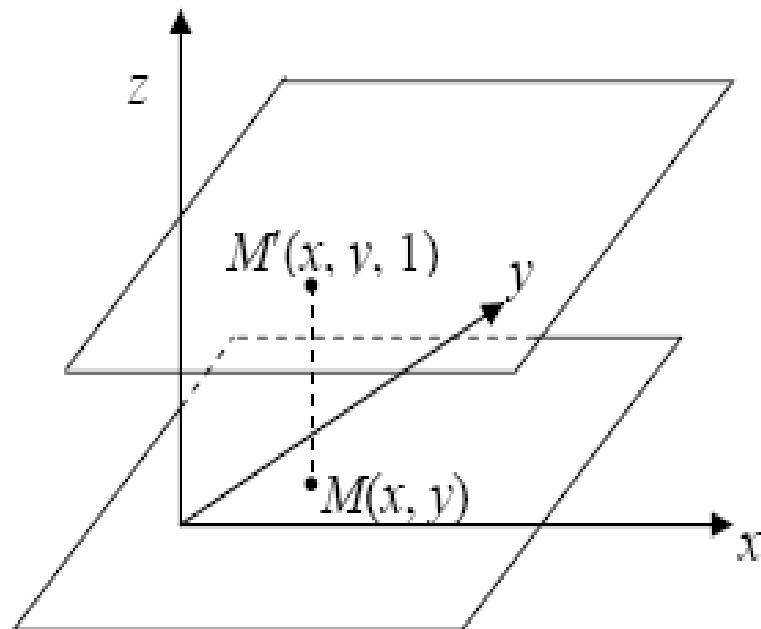
довільній точці  $M(x, y)$

площини ставиться у відповідність точка

$M'(x, y, 1)$  або  $M'(hx, hy, h)$ ,  $h \neq 0$  в просторі.

Фіктивна  $z$ -координата має значення скалярної константи, найчастіше  $h=1$ .

Вектор із координатами  $(hx, hy, h)$  є напрямним вектором прямої, що проходить через точки  $O(0, 0, 0)$  і  $M'(x, y, 1)$ . Ця пряма перетинає площину  $z = 1$  в точці  $(x, y, 1)$ , яка однозначно визначає точку  $(x, y)$  координатної площини  $Oxy$



# Практичне значення введення однорідних координат

- ▶ Над однорідними координатами за спеціальними формулами можна проводити операції, і тільки на останньому етапі перед виведенням зображення на екран необхідно перейти до декартових координат.
- ▶ Застосування однорідних координат дає багато зручностей при розв'язуванні графічних задач, для пристрій, що працюють з цілими координатами Наприклад, для точки  $M(0,5; 0,1)$  можна вибрати  $h = 10$  і працювати з однорідними цілими координатами  $M'(5; 1; 10)$ .

- ▶ однорідні координати дозволяють записувати безмежно віддалені точки простору, уникаючи переповнення розрядної сітки комп'ютера, завдяки нормалізації чисел. За допомогою однорідних координат можна оперувати точкою, що знаходитьться в нескінченності.
- ▶ зручність при заданні геометричних перетворень у матричній формі. За допомогою однорідних координат і матриць третього порядку можна описати довільне афінне перетворення.

# Матриці основних елементарних перетворень (для об'єктів)

Матриця перенесення (translation) точки на вектор  $(m, n)$ :

$$T=T(m, n) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{pmatrix};$$

Матриця розтягу (стиску) (dilation) відносно початку координат:

$$D = D(a, d) = \begin{pmatrix} a & 0 & 0 \\ 0 & d & 0 \\ 0 & 0 & 1 \end{pmatrix};$$

Матриця повороту (rotation) точки відносно початку координат у додатному напрямку:

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

# Перетворення системи координат

задається матрицями:

- ▶  $R(-\varphi)$  – поворот системи координат на кут  $\varphi$  у додатному напрямі;
- ▶  $T(-m, -n)$  – зсув системи координат на вектор  $(m, n)$ .

Елементи довільної матриці афінного перетворення не несуть в собі явно вираженого геометричного змісту. Для того, щоб реалізувати відображення, потрібно знайти елементи матриці, виходячи з геометричного опису перетворення. Як правило, побудову такої матриці в залежності від складності задачі розбивають на кілька етапів, що відповідають основним елементарним перетворенням.

# Матрична форма афінних перетворень

# *Складним (комбінованим)*

називається перетворення, яке містить ланцюжок базових перетворень (не менше двох).

Майже всі афінні перетворення залежать від порядку їх виконання перетворення повороту (не комутативні)

# Для визначення матриці перетворень

використовують рівняння:

$$M = K^{-1} K^*,$$

що випливає із рівняння афінних перетворень

$$K^* = K \cdot M$$

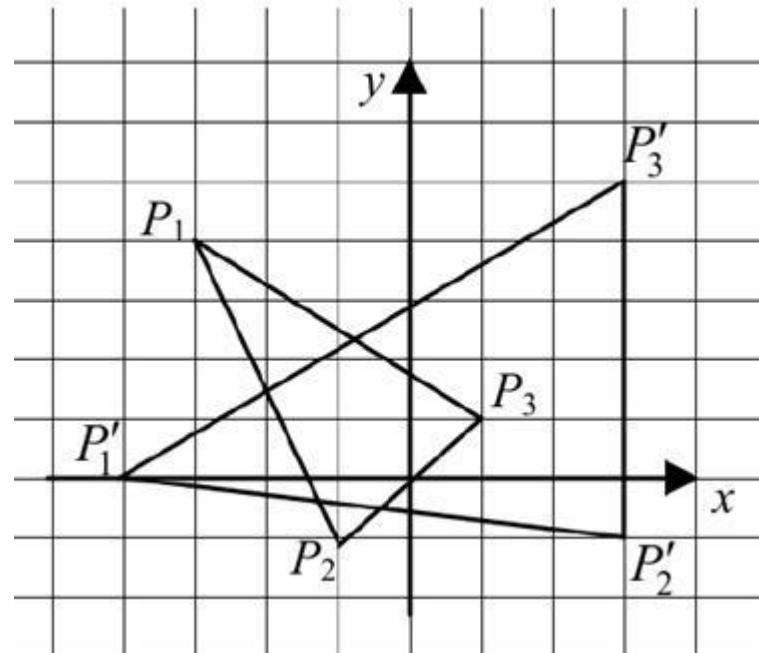
де  $K$  – матриця координат об'єкту до перетворення,

$K^*$  – після перетворення,

$M$  – матриця перетворення.

# Приклад 1.

Знайти матрицю перетворення  
трикутника  $P_1P_2P_3$  у трикутник  $P'_1P'_2P'_3$



Однорідні координати вершин трикутника  $P_1P_2P_3$  складають матрицю

$$K = \begin{bmatrix} -3 & 4 & 1 \\ -1 & -1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

а трикутника  $P_1'P_2'P_3'$

$$K^* = \begin{bmatrix} -4 & 0 & 1 \\ 3 & -1 & 1 \\ 3 & 5 & 1 \end{bmatrix}$$

Зі спiввiдношення  $M = K^{-1} K^*$

отримаємо

$$M = \begin{bmatrix} 1 & 2 & 0 \\ -1 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix}$$

# Правила виконання перетворень

1. Виявити елементарні перетворення у комбінації перетворень (зсув відносно  $(0,0)$ , поворот відносно  $(0,0)$  проти годинникової стрічки, масштабування відносно  $(0,0)$ ).

Необхідно здійснити такі перетворення систем координат (об'єкта), щоб оперувати з елементарними.

2. Вибрати нотацію **по стовпцю чи рядку**, якої необхідно дотримуватися у задачі.

Записуємо відповідно матрицю **однорідних координат** вершин об'єкта.

3. У залежності від нотації записати **послідовність** матриць елементарних перетворень.

Дотримайтесь правила «*Біля координат об'єкта розміщуємо матрицю перетворення, яке здійснюється спочатку, далі наступна матриця... Найдальше від координат буде розміщена матриця перетворення, яке виконується останнім*».

4. Перемножуємо матриці елементарних перетворень, щоб отримати **одну матрицю** для комбінації перетворень (зліва направо, у строгій матриці, не всі перетворення є комутативні!!!).

5. Формуємо оптимальний матричний вираз, де використано матриці однорідних координат початкового об'єкту та переміщеного, а також матрицю комбінації перетворень

6. Для обчислення координат переміщеного об'єкту виконуємо перемноження матриць.

# Матриці симетричних відображень

Відносно  $y=x$

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Відносно  $y=-x$

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Відносно  $y=0$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Відносно  $x=0$

$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Відносно  $(0,0)$

$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Це не елементарні відображення, проте матриці достатньо прості.

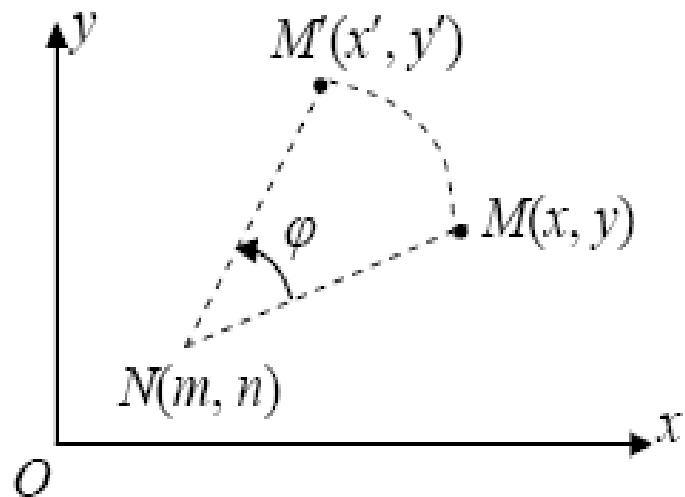
Наприклад, відносно прямої  $y=x$

- 1) Перенести систему координат так, щоб вісь  $X$  співпала з прямою  $y=x$ .
- 2) Дзеркально відобразити відносно  $X$ .
- 3) Повернути назад систему координат.

Дзеркальне відображення – це приклад масштабування. Усі матриці дзеркальних відображень відносно осей можна отримати як в Приклад 1.

# Приклад

Побудувати матрицю повороту точки  $M(x, y)$  відносно довільної точки  $N(m, n)$  на кут  $\varphi$  у додатному напрямку .



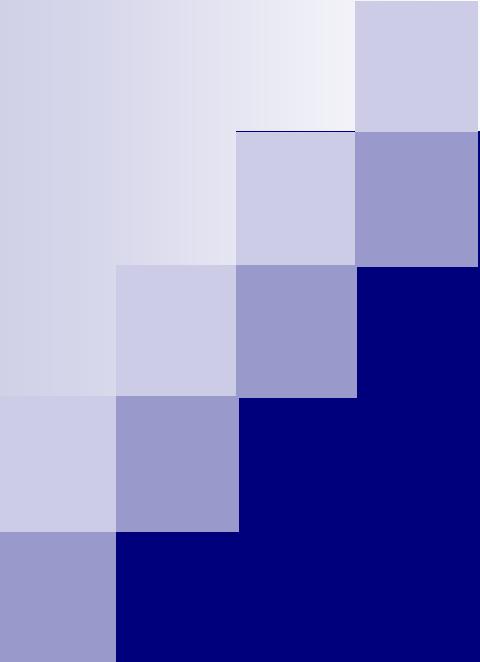
# Розв'язування

Матриця  $R$  задає поворот точки відносно початку координат. Однорідні координати дають можливість знайти матрицю повороту відносно довільної точки. У загальному випадку поворот відносно довільної точки може бути реалізований шляхом таких перетворень:

- 1) переміщення початку системи координат у  $M(m, n)$  на вектор  $(m, n)$  так, щоб точка повороту стала початком координат;
- 2)поворот точки на кут  $\phi$  у додатному напрямку відносно початку координат;
- 3) переміщення одержаного результату назад так (поворнення системи координат), щоб точка повороту співпала з точкою  $M$ .

Отже, для знаходження результуючого  
повороту точки  $M(x, y)$  відносно точки  
 $N(m, n)$  потрібно перемножити матриці  $T$ -,  
 $R$ ,  $T$  за вказаним порядком.

$$\begin{aligned}
 (x' \ y' \ 1) &= (x \ y \ 1) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -m & -n & 1 \end{pmatrix} \begin{pmatrix} \cos \varphi & \sin \varphi & 0 \\ -\sin \varphi & \cos \varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{pmatrix} = \\
 &= (x \ y \ 1) \begin{pmatrix} \cos \varphi & \sin \varphi & 0 \\ -\sin \varphi & \cos \varphi & 0 \\ -m \cos \varphi + n \sin \varphi + m & -m \sin \varphi - n \cos \varphi + n & 1 \end{pmatrix}.
 \end{aligned}$$



# Комп'ютерні моделі зображень

Векторна графіка. Частина 1

2020

# Допоміжні навчальні матеріали

- Методичні вказівки до лабораторної роботи №1
- Методичні вказівки «Використання кривої Без`є»
- Тексти лекцій



**Модель – опис за певними правилами,  
враховуючи деякі припущення**

У залежності від способу формування зображень КГ поділяють на:

- растрову;
- векторну;
- фрактальну; \*
- тривимірну. \*

Око працює як **растровий** пристрій,  
а мозок - як **векторний**.

# Векторні зображення

незамінні там, де принципове значення має  
**збереження чітких контурів**, а саме:

- ілюстрації (художня продукція);
- логотипи та емблеми;
- графічні зображення для Web;
- мультиплікація;
- складні креслення;
- схеми
- ....

# Прикладна векторна графіка

**Ілюстративну графіку** не можливо створити без дизайнера, художника, це первинна інформація.

**Ділова, інженерна графіка** створюється автоматизовано, є вторинною інформацією. Використовує початкові дані.

# Векторна графіка

Зображення описується з допомогою **базових примітивів.**

Базовий примітив – команда пристрою побудувати певну лінію (вектор).

Лінія може бути пряма, крива, замкнута...

Вектор – відрізок, коло/еліпс, прямокутник, багатокутник (полігон), літера...

Примітиви визначаються типом пристрою відображення.

# Основні параметри векторного графічного об'єкта:

- тип об'єкта (коло, крива, прямокутник тощо);
- параметри, що визначають розміри і розташування;
- тип, колір і товщина ліній контура об'єкта;
- стиль і колір заповнення внутрішньої області об'єкта.

# Інша назва ВКГ - ОБ'ЄКТНО-ОРІЄНТОВАНА ГРАФІКА

- всі операції користувач проводить не із зображенням у цілому і не з його найдрібнішими, атомарними частинками, а з об'єктами – семантично навантаженими елементами зображення.

Починаючи із стандартних об'єктів (кругів, прямокутників, текстів і т. д.), користувач може будувати складені об'єкти і маніпулювати з ними як з єдиним цілим.
- кожному стандартному класу об'єктів ставиться у відповідність **унікальна сукупність параметрів** (атрибутів класу).
- для кожного стандартного класу об'єктів визначений **перелік стандартних операцій.**

# Що це дає? Наприклад,

- зображення стає ієрархічною структурою, на самому верху якої є ілюстрація в цілому, а в самому низу - стандартні об'єкти.
- Прямоугутник висотою 200 мм і ширину 300 мм, залитий синім кольором, обведений жовтою лінією шириною 3 пункти, центр якого розташований на 150 мм по вертикалі і на 250 мм по горизонталі від лівого нижнього кута сторінки, а кут нахилу довшої сторони до горизонталі складає  $32^\circ$ . Це екземпляр класу - об'єкт, для якого зафіксовані значення параметрів.
- прямоугутник можна розвернути, масштабувати, закруглювати йому кути, перетворити його в об'єкт іншого класу – замкнену криву.

# Об'єктна орієнтація векторної графіки

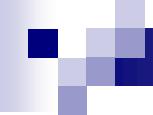
дає користувачеві майже необмежену гнучкість в роботі. Можна виділяти окремі об'єкти зображення і модифікувати їх на будь-якому етапі роботи, що неможливе ні при роботі з точковими зображеннями

- Лінія є елементарним об'єктом, якому притаманні певні особливі властивості: форма, товщина, колір, стиль тощо.
- Будь-який об'єкт (прямоокутник, еліпс, **текст** і навіть пряма лінія) сприймається як криві лінії.

# Основні складові векторного зображення

- *Шлях* – це маршрут, що з'єднує початкову та кінцеву точку.
- *Сегмент* - окрема частина шляху, може бути як прямою, так і кривою лінією.
- *Вузол* - початкова або кінцева точка сегмента.

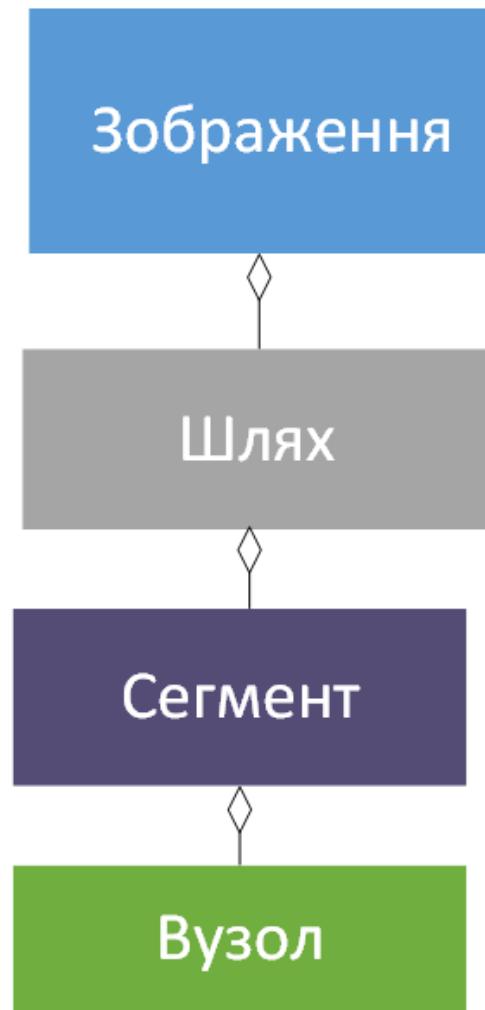
Кожен елемент векторної графіки містить ці три основні елементи і дозволяє їх редагування.



Векторні об'єкти завжди мають шлях, що визначає їх форму.

Всі шляхи містять дві компоненти: сегменти та вузли.

Якщо шлях є замкненим, тобто кінцева точка співпадає з початковою, об'єкт має внутрішню ділянку, яка може бути заповненою кольором або іншими об'єктами.



# Модель кривої у векторній КГ

- У основі моделі ліній лежать два поняття: вузол і сегмент.
- Вузлом називається точка на площині зображення, що фіксує положення одного з кінців сегменту.
- Сегментом називається частина лінії, що сполучає два суміжні вузли.

- Вузли і сегменти нерозривно зв'язані один з одним: у замкнuttій лінії вузлів стільки ж, скільки сегментів, а в незамкнuttій - на один більше.
- Будь-яка лінія в складається з вузлів і сегментів, і всі операції з лініями насправді є операціями саме з ними.

- Вузол повністю визначає характер попереднього йому сегменту, тому для незамкнutoї лінії важно знати, який з двох її крайніх вузлів є початковим, а для замкнutoї - напрям лінії (за годинниковою стрілкою або проти неї).
- За характером попередніх сегментів виділяють три типи вузлів: початковий вузол незамкнutoї кривої, прямолінійний (Line) і криволінійний (Curve).

# Заповнення можна розбити на 4 категорії:

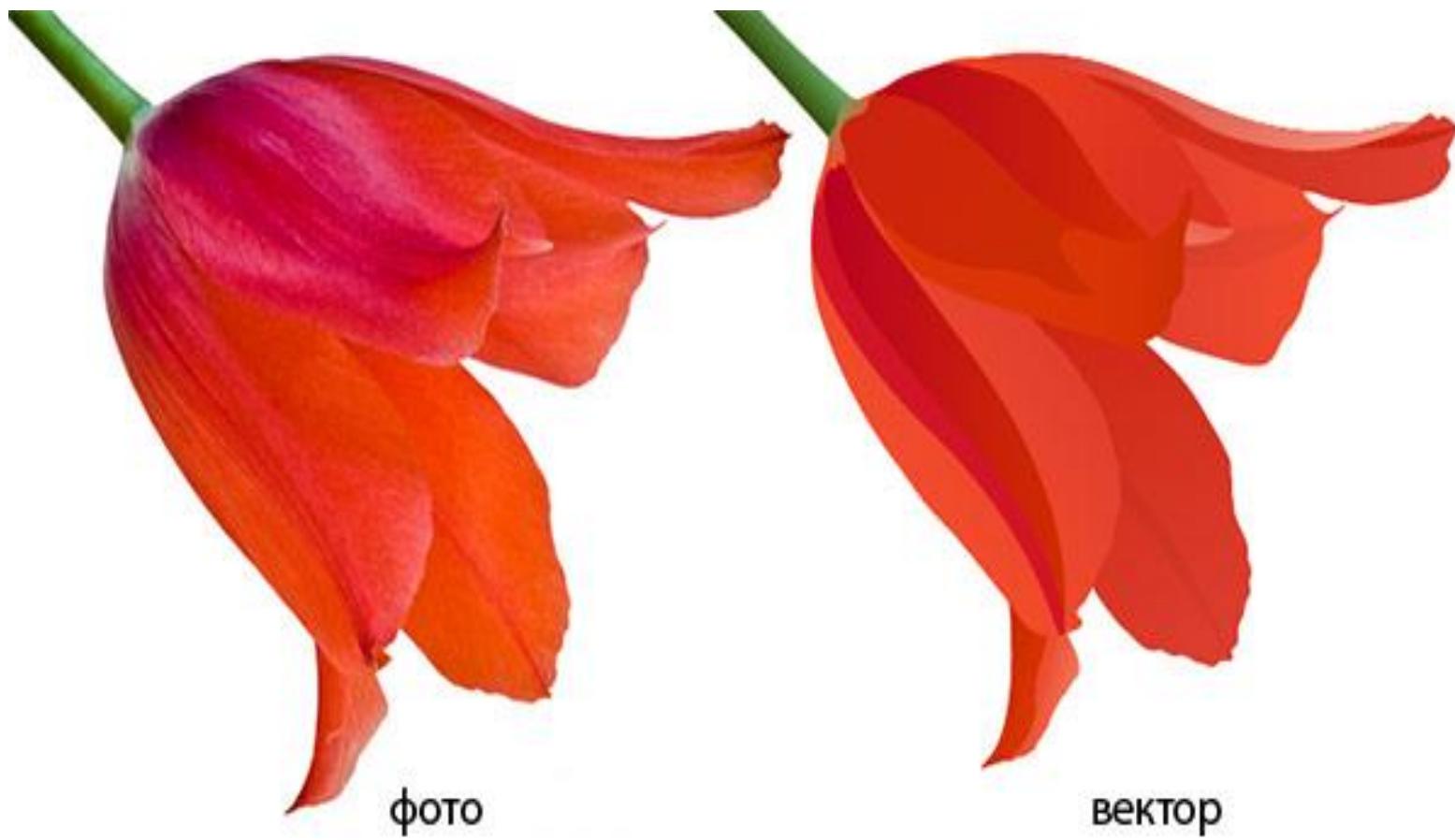
- однорідне заповнення одним кольором або штрихуванням;
- градієнтне, при якому кольори або тіні поступово змінюються (лінійна, радіальна, конічна, прямокутна закономірність тощо);
- візерункове, при якому об'єкт заповнюється повторювальними зображеннями (двоколірними або повноколірними);
- текстурне заповнення (художні зображення).

# Переваги векторної графіки

1. **невеликі за розміром** файли, оскільки зберігається не зображення, а лише його основні дані, використовуючи які, програма відновлює зображення;
2. об'єкти **легко трансформуються**, ними легко маніпулювати. Редагуючи векторний об'єкт, можна змінити властивості ліній, з яких складається зображення. Можна пересувати об'єкт, змінювати його розміри, форму та колір, не впливаючи на якість зображення;
3. векторна графіка не залежить від роздільності, тобто векторні об'єкти відтворюють на пристроях з різною роздільністю **без втрати якості зображення**;
4. **простий експорт** векторного малюнка в растровий.

# Недоліки векторної графіки

1. Низька фотoreалістичність
2. Обмежена область застосування
3. Низька швидкість промальовування на растрових пристроях
4. Практично повна неможливість експорту растрового малюнка у векторний.



# Прикладні програми векторної графіки

призначені для технічних креслень, зображень ділового призначення, рекламних ілюстрацій.

Найпопулярнішими прикладними програмами є продукти фірм

- Corel - CorelDraw,
- Adobe - Illustrator,
- Macromedia - FreeHand,

Для Linux - XFIG, ....

# Математичні основи векторної графіки

- Точка. Об'єкт на площині представляється двома числами  $(x, y)$  відносно початку координат.
- Пряма лінія. Її відповідає рівняння  $y=kx+b$ . Вказавши параметри  $k$  та  $b$  можна створити пряму лінію у відомій системі координат.
- Сегмент прямої. Для опису потрібно додатково вказати параметри  $x_1$  та  $x_2$ , відповідно початок та кінець відрізку.
- Крива лінія II порядку. До них належать еліпси, круги, параболи, гіперболи тощо. Пряма лінія є також випадком кривої II порядку. Крива II порядку описується рівнянням  $a_0x^2+a_1y^2+a_2xy+a_3x+a_4y+a_5=0$ . Для побудови відрізка кривої додатково потрібні ще два параметри початку та кінця відрізку.

- Крива лінія III порядку. Важлива наявність точки перегину, що дозволяє відобразити різноманітні об'єкти. Рівняння кривої III порядку  
$$a_0x^3 + a_1y^3 + a_2x_2y + a_3xy^2 + a_4x^2 + a_5y^2 + a_6xy + a_7x + a_8y + a_9 = 0.$$
 Для опису відрізка потрібні ще два параметри початку та кінця відрізку. Зауважимо, що пряма та криві II порядку є частковим випадком кривих III порядку.
- Криві Без'є. Використовується для спрощеного виду кривих III порядку. Метод побудови кривих Без'є заснований на використанні пари дотичних, що проведені до лінії в крайніх точках. На форму кривої лінії впливає кут нахилу дотичних та довжина її відрізка. Таким чином, дотичні відіграють роль віртуальних важелів, за допомогою яких керують формою кривої.

# Популярні формати векторної графіки

Серед векторних форматів, на відміну від растрових, ідея розумної стандартизації проявляється значно слабше.

Розробники практично всіх векторних графічних програм хочуть мати справу тільки зі своїми власними форматами, що пов'язано, швидше за все, зі специфікою алгоритмів формування векторного зображення.

Але, так як можливість перенесення файлів між різними додатками в векторній графіці не менш актуальна, ніж у растрової, то свого роду стандартом стали файлові формати найбільш популярного професійного графічного пакету

- **Adobe Illustrator** (AI – робочий формат, внутрішній)

EPS – використовується у видавничих системах, поєднує раstroву і векторну графіки. Надійний та універсальний формат, “рідна” програма – Adobe Illustrator.

PDF (Portable Document Format) – універсальний формат, створений і підтримуваний компанією Adobe Systems

# CorelDRAW

Досить суперечливим є формат CDR, основний робочий формат популярного пакета CorelDRAW, що є серед лідерів у класі векторних графічних редакторів на платформі PC.

Маючи порівняно невисоку стійкість і проблеми із сумісністю файлів різних версій формату, тим не менш формат CDR можна назвати професійним.



**SVG (Scalable Vector Graphics)** - формат файлів для двомірної векторної графіки, як статичної, так і анімованої та інтерактивної.

Чисто векторні формати AutoCAD DXF,  
Microsoft SYLK.

Містять список примітивів або набір  
інструкцій команд для побудови  
примітивів, також можливо включати  
растрові об'єкти.



Важко відокремлювати метафайли та  
векторні формати.

WMF - рідний векторний формат Windows.

Сприймається практично усіма програмами Windows, так чи інакше пов'язаними з векторною графікою. Однак, незважаючи на простоту і універсальність, користуватися форматом WMF варто тільки в крайніх випадках, оскільки він не може зберігати деякі параметри, які можуть бути присвоєні об'єктам в різних векторних редакторів, не сприймається Macintoshами, і, найголовніше, спотворює колірну схему зображення.

# Кросплатформені інструменти векторної графіки

- Gravit Designer
- Inkscape



# Растрова графіка є домінуючою

## Чому?

# Крива Без'є

– основний інструмент  
векторної графіки

- ▶ **Гнучкість, наглядність** з точки зору користувача-дизайнера
- ▶ **Простота** з точки зору математичного апарату
- ▶ **Універсальність** з точки програмування

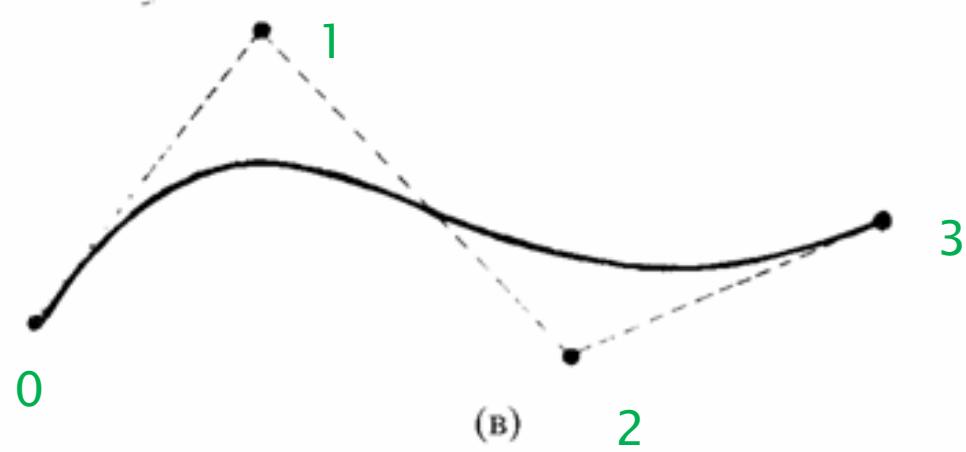
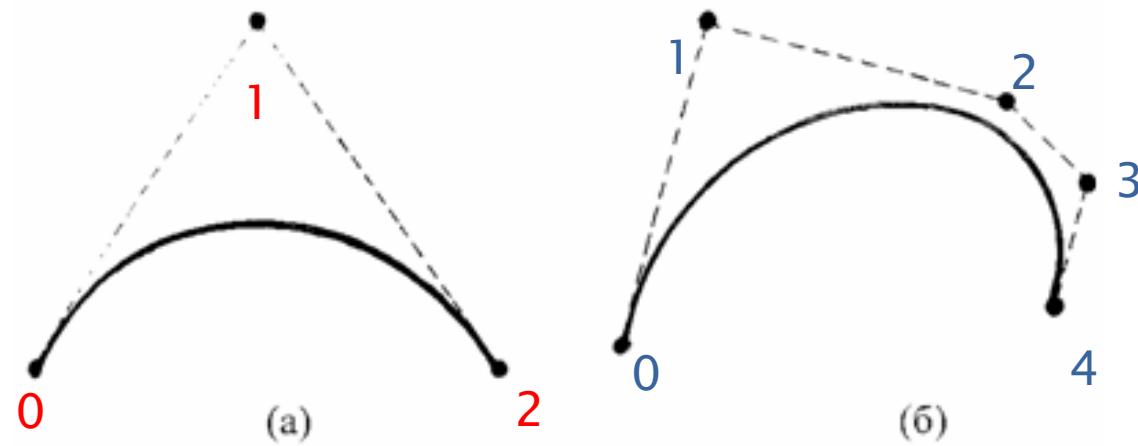
# Історія

- ▶ Криві та поверхні Без'є використовувалися в 60-х роках компанією "Рено" для комп'ютерного проектування форми кузовів автомобілів.
- ▶ Ще в 1959 р. криві Без'є використовувались Полем де Кастельє з компанії «Сітроен», але його дослідження не публікувались і приховувались компанією як комерційна таємниця до кінця 1960-х.
- ▶ Згодом це відкриття стало одним з найважливіших інструментів систем автоматизованого проектування та програм комп'ютерної графіки.

# Застосування кривої Без'є є досить широким:

- ▶ спосіб управління рухом,
- ▶ засіб створення анімаційних рисунків,
- ▶ метод опису художніх шрифтів,
- ▶ спосіб передачі виділених об'єктів, інтерактивний інструмент дизайну форми різноманітних об'єктів.

- ▶ П'єр Без'є розробив метод зображення кривої, який створює у користувача більш природне сприйняття.
- ▶ Крива Без'є визначається **вершинами багатокутника**, який задає форму кривої. Кривій належить перша та остання вершина, в той час як інші вершини характеризують похідні, порядок та вид кривої.
- ▶ Крива Без'є задається за допомогою відкритого багатокутника (характеристичної ламаної) сформованого заданими точками.



# Елементи керування

- ▶ Контрольні точки є опорні і керуючі
- ▶ Дотичні

У процесі інтерактивного конструювання **характеристична ламана** є засобом управління кривої Без'є

- ▶ **Характеристична ламана** – це **металева** конструкція, до якої притягується **крива Без'є** як **магнітна стрічка**.
- ▶ Сила притягування визначається кількістю контрольних точок

# Задача *апроксимації*

- ▶ (найдженоого представлення) виникає під час заміни кривої, що описується рівнянням функцій складної природи (наприклад з точки зору швидкодії обчислень її значень і похідних, інтегрування, диференціювання), іншою кривою, в деякому **найдженні** близької до заданої, рівняння якої більш просте.

# Сплайн –

- ▶ один зі способів побудови кривих, поверхонь складної форми, які були вперше використані для математичного опису форми нових автомобілів, літаків, космічних кораблів, побутових приладів під час їхнього проектування.
- ▶ це гладка крива, яка будується з використанням дуг і проходить через кілька контрольних точок, що керують формою сплайна.
- ▶ це функція, область визначення якої розбита на фрагменти (сегменти), на кожному з яких функція є деяким поліномом (многочленом).

# Для побудови кривої Без'є

в комп'ютерній графіці  
використовують три основні способи:

- ▶ *параметричний,*
- ▶ *рекурсивний,*
- ▶ *матричний.*

# Формула параметричного вигляду кривої Без'є

$$P(t) = \sum_{i=0}^n b_{in}(t) \cdot P_i, \quad i = \overline{0, n}$$

$$b_{in}(t) = \frac{n!}{i!(n-i)!} t^i \cdot (1-t)^{n-i} \quad - \text{коекоми}$$

Бернштейна

$P_i$  - вхідні точки

$t$  - бізичний параметр,  $t \in [0:1]$

А як буде виглядати формула  
кривої Без'є у скалярній формі?

$$\begin{cases} x = x(t) \\ y = y(t) \\ \vdots \end{cases}$$

# Приклади кривих 2- і 3-го порядків

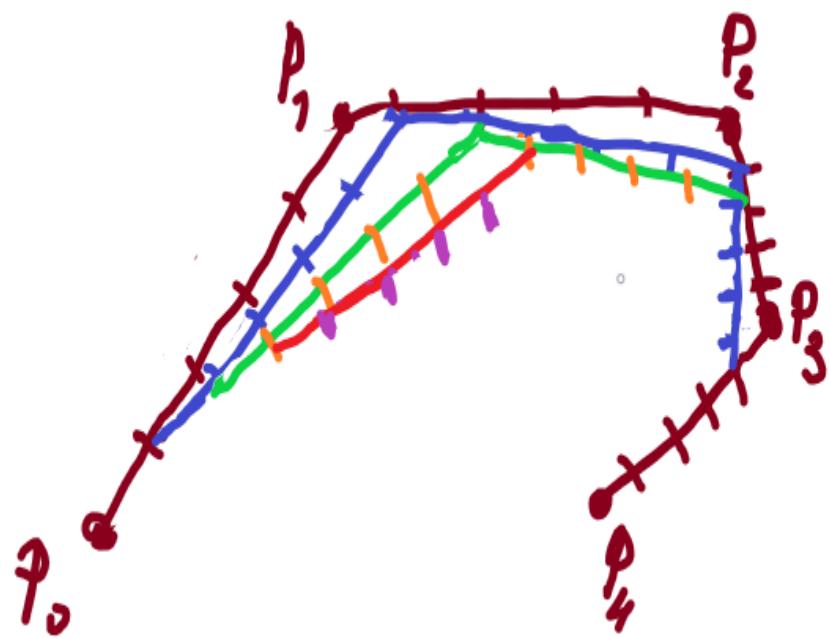
- ▶  $B(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2.$
- ▶  $B(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3.$

# Властивості поліномів Бернштейна

1. Многочлени Бернштейна набувають невід'ємних значень.
2. У сумі вони дають одиницю.
3. Многочлени не залежать від вершин масиву  $P$ , а залежать лише від кількості точок у масиві.

# Геометричний алгоритм побудови кривої Без'є

- ▶ 1. Кожна сторона опорної ламаної ділиться у відношенні  $t:(1-t)$ , де  $t$  - аргумент точки поділу.
- ▶ 2. Точки поділу з'єднуються відрізками прямих і утворюють нову ламану (багатокутник). Кількість вузлів нового контуру на одиницю менша, ніж кількість вузлів попереднього контуру.
- ▶ 3. Сторони нового контуру знову діляться пропорційно значенню  $t$  і т.д. Це продовжується до тих пір, поки не буде отримана єдина точка поділу - точка кривої Без'є.



$t = 0,2$

$\frac{1}{4}$

# Рекурсивна формула кривої Без'є



$$B_{R, R, \dots, R}(t) = (1-t)B_{R, R, \dots, R_{n-1}}(t) + tB_{R, R, \dots, R_n}(t).$$

Dato  $P_i, i = \overline{0, n}, t_0 \in (0; 1)$   
 $R(t_0) - ?$

$$\begin{cases} P_i^{(0)} = P_i, & i = \overline{0, n} \\ P_i^{(j)} = P_i^{(j-1)} (1 - t_0) + P_{i+1}^{(j-1)} \cdot t_0, & \begin{matrix} i = 0, \dots, n-j \\ j = \overline{1, n} \end{matrix} \end{cases}$$

$$R(t_0) = P_0^{(n)}$$

# Матричний вигляд кривої Без'є

$$B(t) = T \cdot N \cdot P$$

- ▶ Т – вектор параметру,  $T = [t^n \ t^{n-1} \ \dots \ t \ 1]$ ,
- ▶ Р – вектор вершин характеристичної ламаної,  $P^T = [P_0 \ P_1 \ \dots \ P_{n-1} \ P_n]$ .

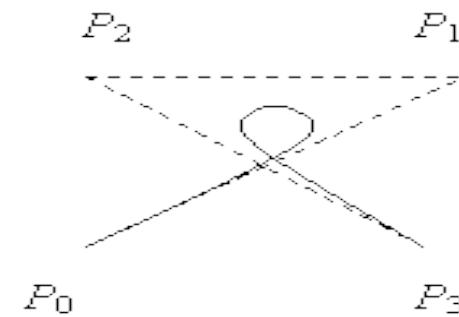
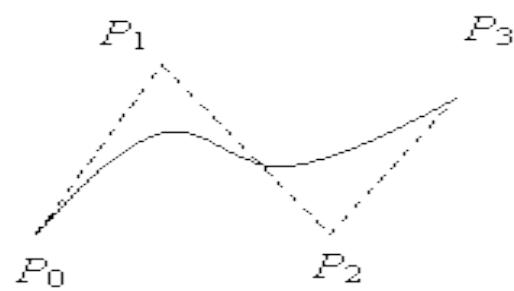
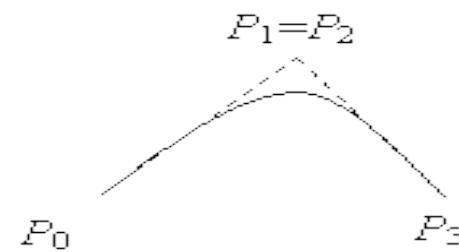
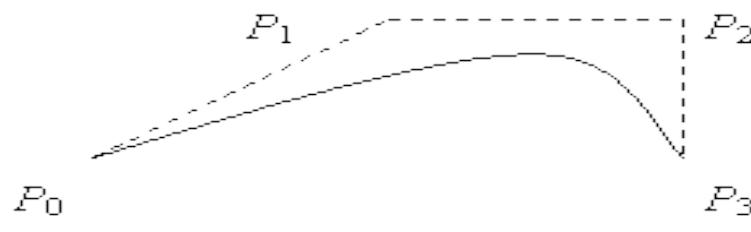
# Матриця коефіцієнтів

$$N = \begin{bmatrix} \binom{n}{0} \binom{n}{n} (-1)^n & \binom{n}{1} \binom{n-1}{n-1} (-1)^{n-1} & \dots & \binom{n}{n} \binom{n-n}{n-n} (-1)^0 \\ \binom{n}{0} \binom{n}{n-1} (-1)^{n-1} & \binom{n}{1} \binom{n-1}{n-2} (-1)^{n-2} & \dots & 0 \\ \dots & & & \\ \binom{n}{0} \binom{n}{1} (-1)^1 & \binom{n}{1} \binom{n-1}{0} (-1)^0 & \dots & 0 \\ \binom{n}{0} \binom{n}{0} (-1)^0 & 0 & \dots & 0 \end{bmatrix},$$

► У процесі інтерактивного  
конструювання ламана  
Без'є є засобом управління  
формою

# Властивості кривої Без'є

1. Форма кривої Без'є повторює хід ламаної  $P_0P_1\dots P_n$ . При зміні порядку точок повністю змінюється форма кривої.



2. Перша та остання точки кривої збігаються з відповідними точками масиву  $P$ , тобто  $B(0) = P_0$ ,  $B(1) = P_n$ . Усі інші вершини ламаної в загальному випадку знаходяться поза кривою.
3. Нахил дотичних векторів в крайніх точках кривої співпадає з нахилом, відповідно першої та останньої ланки ламаної Без'є.

4. Крива Без'є *інваріантна* відносно **афінних перетворень**. Тобто спочатку варто здійснити афінні перетворення над вершинами ламаної, а потім побудувати криву Без'є за новими вершинами.
5. Крива Без'є лежить в опуклій оболонці вершин масиву  $P$ .

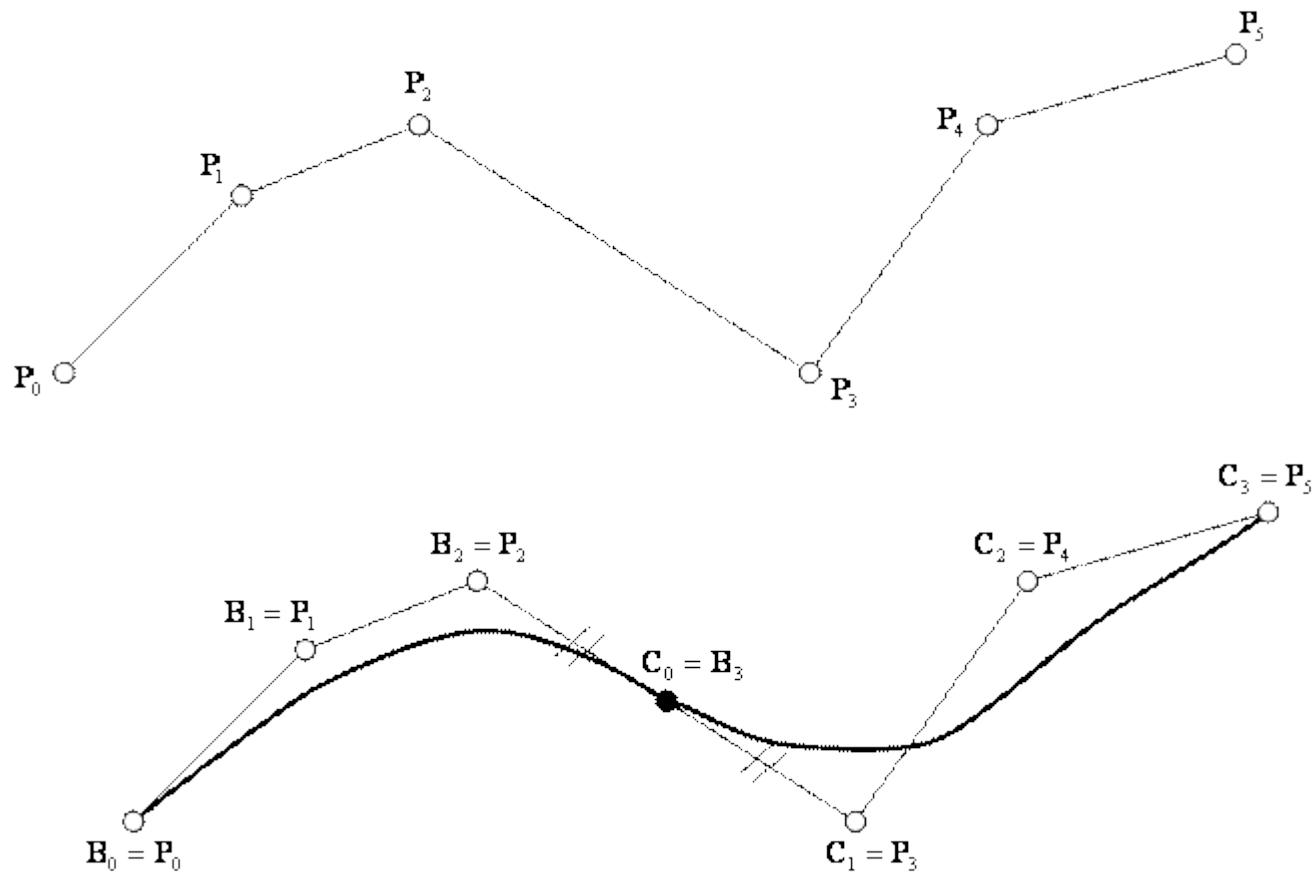
# Основний недолік

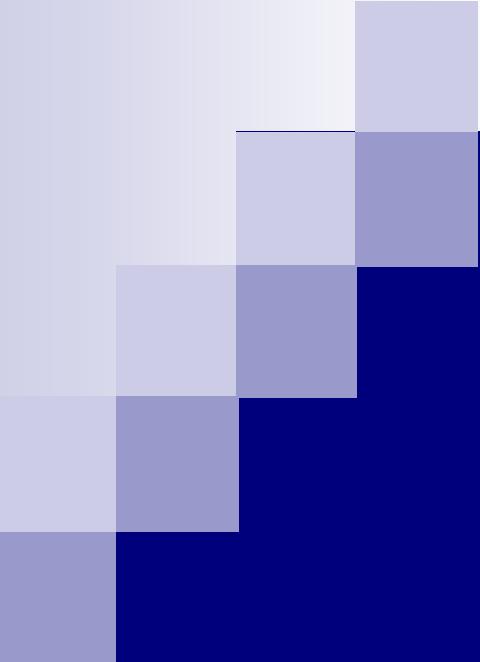
Зміна хоча б однієї точки в масиві  $P$  приводить до помітної зміни всієї кривої Без'є.

# Склейовання кривих Без'є

- ▶ Складена кубічна крива Без'є – це неперервна крива  $\gamma$ , що є об'єднанням елементарних кубічних кривих  $\gamma_0, \gamma_1, \dots, \gamma_l$ .
- ▶ Для побудови складеної кривої Без'є розбиваємо масив точок  $P_0, P_1, \dots, P_n$  ( $n$  має бути кратним 3) на  $l$  підмножин, кожна з яких містить чотири точки так, що остання точка попередньої підмножини – це перша точка наступної підмножини, тобто
- ▶  $\{P_0, P_1, P_2, P_3\}, \{P_3, P_4, P_5, P_6\}, \dots, \{P_{3l}, P_{3l+1}, P_{3l+2}, P_{3l+3}\}$ .
- ▶ На кожній такій підмножині можна побудувати елементарну криву Без'є (тобто окремий фрагмент складеної кривої Без'є).
- ▶ Об'єднання окремих фрагментів дає складену кубічну криву Без'є, яка належить до класу неперервних функцій.
- ▶ Щоб крива була гладкою, необхідно забезпечити виконання умови на краях двох сусідніх сегментів:

$$B'_i(l) = q B'_{i+1}(0)$$





# Комп'ютерні моделі зображень

## РАСТРОВА ГРАФІКА



Модель – опис за певними правилами,  
враховуючи деякі припущення

У залежності від способу формування зображень КГ поділяють на:

- растрову;
- векторну;
- фрактальну; \*
- тривимірну. \*

# Растрова графіка (Піксельна, точкова, дискретна)

Зображення представлено у вигляді комбінації точок (пікселів), яким притаманні **свій колір та яскравість** і які **певним чином розташовані** у координатній сітці.

# Pixel – Picture Element

## Растр – матриця пікселів

Відмінними особливостями піксела є його:

- однорідність (всі піксели за розміром одинакові) ;
- неподільність (всередині піксела не може бути ніяких більш дрібних елементів).

Такий підхід є ефективним у випадку, коли графічне зображення має багато напівтонів і інформація про колір важливіша за інформацію про форму (фотографії та поліграфічні зображення).

При редагуванні растрових об'єктів, користувач змінює колір точок, а не форми ліній.

# Сітка растру

Для опису розташування пікселів використовують систему координат.

Координати пікселів утворюють дискретний ряд значень, найчастіше використовують цілі числа для нумерації.

Піксел  $(0,0)$  – розміщено в лівому верхньому куті. Растроva графіка базується на відомих алгоритмах, у яких зручно використовуються координати точок.

Адреса пікселя (2 координати) лінеризується.  
ЯК?

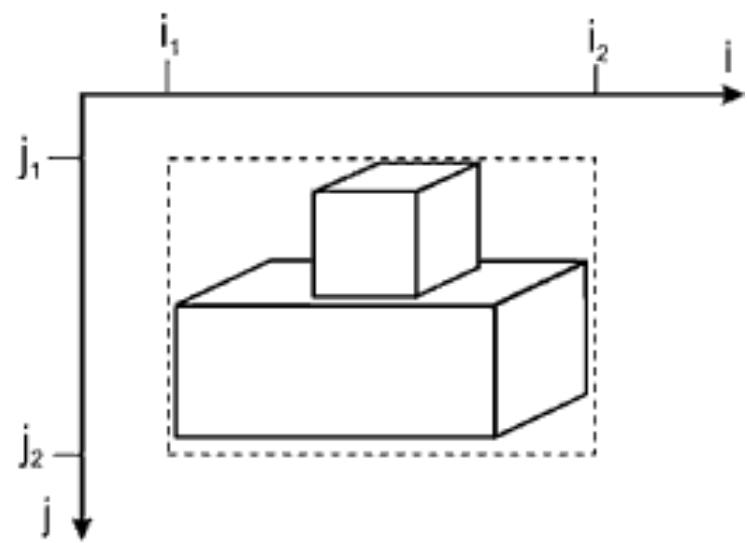
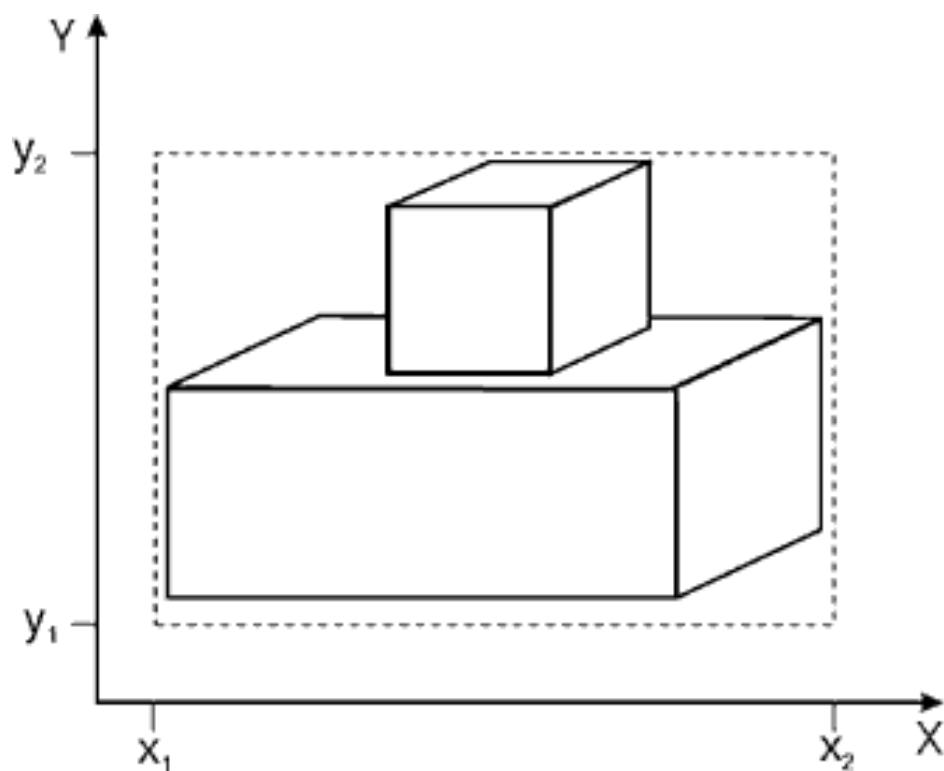
NxM –растр,  
(i,j) – адреса піксела

$$A(i, j) = N * (j - 1) + i$$

i – номер піксела по горизонталі (номер рядка/стовпця матриці?);

j - номер піксела по вертикалі (номер рядка/стовпця матриці?);

# Перехід між реальною системою координат й системою координат області відображення



- На екрані одиницею виміру є піксель
- При відображені рисунка на екран кожна точка вихідного прямокутника з координатами  $(x, y)$  перейде в деяку точку із координатами  $(i, j)$

$$\Delta x = x_2 - x_1, \quad \Delta y = y_2 - y_1, \quad \Delta i = i_2 - i_1, \quad \Delta j = j_2 - j_1,$$

$$S_x = \frac{\Delta i}{\Delta x}, \quad S_y = \frac{\Delta j}{\Delta y}$$

$$i = i_1 + S_x(x - x_1), \quad j = j_2 - S_y(y - y_1).$$

# Кількісна та якісна характеристики раству

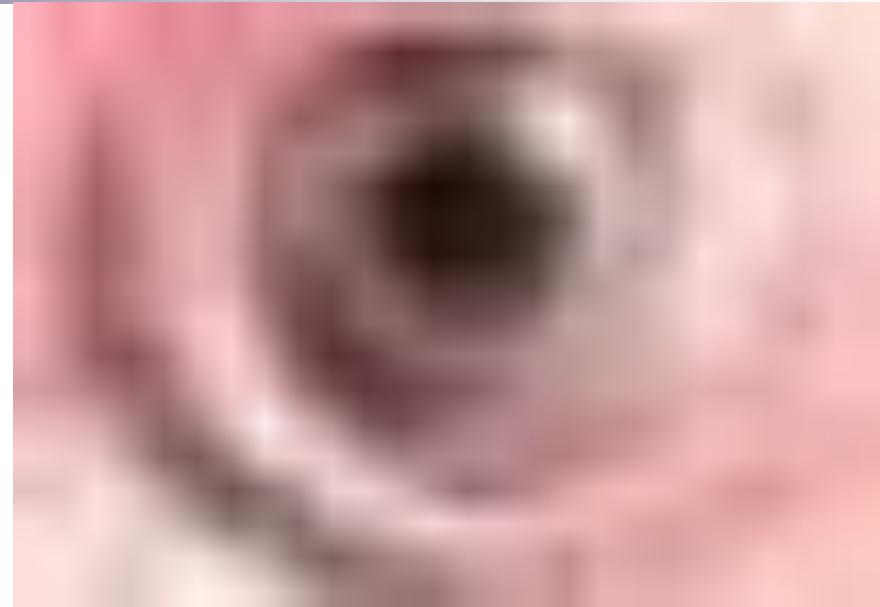
- Кількісна характеристика – це розмір растра, тобто кількість пікселів по вертикалі та горизонталі.
- Якісна характеристика - роздільна здатність, яка вказує кількість точок на одиницю довжини, вимірюється у точках на дюйм (dpi - dots per inch). Роздільна здатність залежить від вимог до якості зображення та розміру файлу, способу оцифрування або методу створення готового зображення, вираного формату файлу та інших параметрів.

Очевидно, чим вище вимоги до якості, тим більша має бути роздільність.

У залежності від пристосування, на якому виводиться зображення, можливе застосування таких одиниць вимірювання роздільної здатності (resolution):

- 1) spi (sample per inch) – елементів на дюйм,
- 2) dpi (dot per inch) – точок на дюйм,
- 3) ppi (pixel per inch) – пікセルів на дюйм,
- 4) lpi (line per inch) – ліній на дюйм.

Форма пікселів растра визначається особливостями пристрою графічного виводу. Піксели можуть мати прямокутну чи квадратну форми. Розмір піксела можуть дорівнювати кроку растра.



# Глибина кольору - важливий параметр растрових зображень

характеризує максимальну кількість кольорів, які використані у зображення, задається кількістю біт, необхідних для запам'ятовування кольору (**bpp**).

Існує декілька типів зображень із різною глибиною кольору:

- чорно-білі;
- у відтінках сірого;
- з індексованими кольорами;
- повноколірні;

- **Чорно-білі зображення.** На один піксел зображення відводиться 1 біт інформації - чорний та білий. Глибина кольору - **1 біт**.
- **Зображення у відтінках сірого.** Піксел сірого зображення кодується 8 бітами (**1 байт**). Глибина кольору - 8 біт, піксел може приймати 256 різних значень - від білого (255) до чорного (0).
- **Зображення з індексованими кольорами.** Перші кольорові монітори працювали з обмеженою колірною гамою (16, згодом 256 кольорів). Такі кольори називаються *індексованими* і кодуються **4 або 8 бітами** у вигляді колірних таблиць. У такій таблиці всі кольори вже визначені і можна використовувати лише їх.
- **Повноколірні зображення.** Глибина кольору не менше як **24 біти**, що дає можливість відтворити понад 16 мільйонів відтінків. Повноколірні зображення називаються *True Color (правдивий колір)*. Бітовий об'єм кожного піксела розподіляється по основних кольорах обраної колірної моделі, по 8 бітів на колір. Колірні складові організуються у вигляді каналів, спільне зображення каналів визначає колір зображення. Повноколірні зображення є багатоканальними і залежать від колірної моделі (RGB, CMY, CMYK, Lab, HBS), які різняться за глибиною кольорів і способом **математичного опису** кольорів.

# Опрацювання растрових зображень

- Задачі роботи з кольорами розв'язуються найефективніше.
- При збільшенні растрового зображення, можна спостерігати *пікселізацію*, тобто при масштабуванні збільшується розмір точок і стають помітними елементи раstra. Для усунення цього, потрібно заздалегідь оцифрувати оригінал із роздільністю, достатньої для якісного відтворення при масштабуванні. Або, при масштабуванні застосовують метод *інтерполяції*, коли при збільшенні зображення, додається необхідне число проміжних точок.
- При зменшенні растрового зображення, можна спостерігати спотворення, коли елементи раstra накладаються один на одного.

# Прикладні програми растрової графіки

більшість растрових графічних редакторів орієнтовані не стільки на створення зображень, скільки на їх обробку - оцифрованих фотографій, слайдів, відеокадрів, кадрів мультиплікаційних фільмів, створення книжкових та журналльних ілюстрацій,

# Найпопулярнішими прикладними програмами є продукти фірм

- Adobe - PhotoShop,
- Corel - PhotoPaint,
- Macromedia - FireWorks,
- стандартний додаток у Windows – Paint
- Linux – додаток  
редактор GIMP( розповсюджується  
безкоштовно).

# Популярні формати растрової графіки

- Найпростіший растровий формат **BMP** є рідним форматом системи Windows, він підтримується всіма графічними редакторами, які працюють під її управлінням. Цей формат застосовується для збереження растрових зображень, призначених для використання в Windows (і, по суті, більше ні на що не придатний).

# Найпопулярнішим форматом для Інтернет є

достатньо вже давній формат **GIF**, запропонований CompuServe в 1987 році. Його особливістю є використання режиму **індексованих кольорів** (не більш 256), що обмежує область застосування формату зображень, які мають різкі кольорові переходи. Цей формат не може застосовуватися для поліграфії

- Найпопулярнішим форматом для збереження фотографічних зображень є **JPEG** (JPG). JPEG може зберігати тільки 24-бітові повнокольорові зображення.

Хоча JPEG добре стискає фотографії (видалення надлишкової інформації від 10 до 60 відсотків від початкового об'єму файла), але цей стиск відбувається з втратами і псує якість, тим не менше, він може бути легко налаштований на мінімальні, практично непомітні для ока, втрати.

Проте не варто використовувати JPEG для збереження зображень, які будуть опрацьовуватися, так як при кожному збереженні в цьому форматі процес погіршення якості зображення носить *лавиноподібний характер*. Таким чином, можна зберегти якість зображення при мінімальному розмірі файла.

# Формат PSD (Photoshop Document)

- є внутрішнім для Photoshop, і дозволяє зберігати інформацію про контури, канали, шари, векторні надписи. Підтримуються всі кольорові моделі і будь-яка глибина кольору. Застосовується стиснення без втрат.

# Як універсальний формат для зберігання растрових зображень

- пропонується формат **TIFF**, який досить широко використовується у видавничих системах, що вимагають зображення найкращої якості. Можливість запису зображень у форматі TIFF є однією з ознак високого класу сучасних цифрових фотокамер. Завдяки своїй сумісності з більшістю професійного ПЗ для обробки зображень, формат TIFF дуже зручний при перенесенні зображень між комп'ютерами різних типів (наприклад, з РС на Mac і назад).

# Формат RAW

порівняно молодий формат.

Для відкриття такого файлу в графічному редакторі необхідно проінсталювати спеціальне додавання.

У цілому формат RAW має більше переваг, ніж TIFF, по-перше, тому, що дозволяє зберігати початкову кольорову інформацію знімка без застосування спеціальних алгоритмів,

по-друге, тому що файли формату RAW займають приблизно в два рази менший об'єм пам'яті, ніж TIFF.

# Переваги растрової графіки

- простота автоматизованого вводу (оцифрування) зображень, фотографій, слайдів, рисунків за допомогою сканерів, відеокамер, цифрових фотоапаратів;
- фотorealістичність. Можна отримувати різні ефекти, такі як туман, розмитість, тонко регулювати кольори, створювати глибину предметів.

# Недоліки растрової графіки

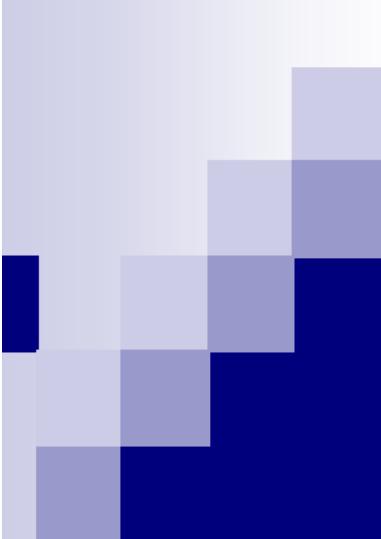
1. складність управління окремими фрагментами зображення. Потрібно самостійно виділяти ділянку, що є складним процесом;
2. ефекти спотворення при масштабуванні (“розмитість” при збільшенні, “ступінчастість” при зменшенні)
3. розмір файлу є пропорційним до площини зображення, роздільноті і типу зображення, і, переважно, при хорошій якості є великим.
4. растрове зображення має певну роздільність і глибину представлення кольорів. Ці параметри можна змінювати лише у визначених межах і, як правило, із втратою якості;

## Недолік (2)

може бути вирішений спеціальними методами, що базуються на методах інтерполяції

# Недолік (3)

може бути вирішений, якщо  
застосовувати спеціальні методи стиску  
зображень



# Комп'ютерні моделі кольорів. Частина 1

## Лекція 6

### Колір. Психофізичні аспекти

Колір - це властивість матеріальних об'єктів, яка сприймається як усвідомлене зорове відчуття та виникає в результаті дії на око потоків видимого **електронно-магнітного випромінювання** (з довжинами хвиль **від 380 до 760 нм**).

Той або інший колір «**привласнюється**» людиною об'єкту в процесі зорового сприйняття цього об'єкту.

Світло сприймається або безпосередньо **від джерела**, наприклад, від освітлювальних приладів, або як **відбиття** від поверхонь об'єктів або **заломлення** при проходженні крізь прозорі і напівпрозорі об'єкти. Амплітуда, що визначає енергію хвилі, відповідає за яскравість кольору.

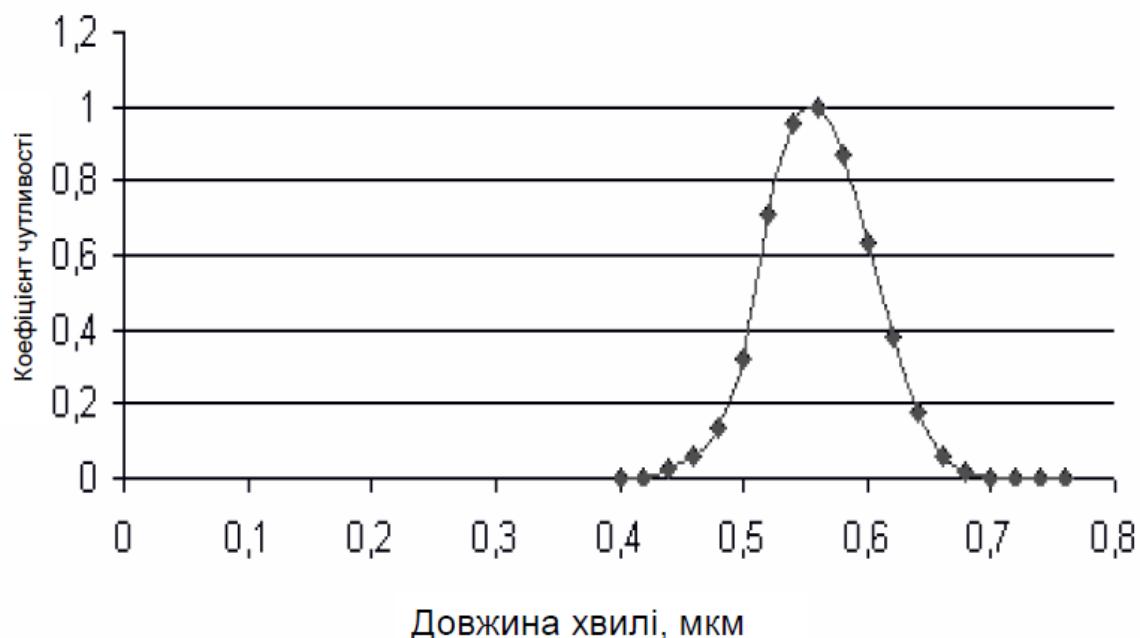
Саме поняття кольору є особливістю людського "бачення" навколишнього середовища. Перехід від одного кольору до іншого здійснюється безперервно, поступово. Кожному кольору співставляється не якось одна довжина хвилі світла, а довжини хвиль, що потрапляють в деякий інтервал значень.

	КОЛІР	ДОВЖИНА ХВИЛІ (мкм)
	Фіолетовий	0,38 - 0,45
	Синій	0,45 - 0,5
	Блакитний	0,5 - 0,53
	Зелений	0,53- 0,57
	Жовтий	0,57 - 0,59
	Оранжевий	0,59 - 0,62
	Червоний	0,62 - 0,76

Максимум спектральної чутливості ока знаходитьться в **жовто-зеленій** частині спектру (0,555 мкм).

Шартрез (жовто-зелений колір) знаходиться чітко посередині частот видимого спектру. Ваши очі мають рецептори для сприйняття синього, зеленого і червоного. Але мозок не отримує інформації про кольори, він отримує інформацію про різницю світлого і темного, і інформацію про різницю між кольорами. У результаті рецепторам мозку найлегше «побачити» саме колір шартрез. До речі, цей колір часто використовується психологами, екстрасенсами, художниками, як заспокійливий і одноточно найпомітніший для людини.

## Крива спектральної чутливості ока при денному світлі



- Сучасні моделі кольорів базуються на відомих теоріях фізиків.

До дослідів Ньютона (кінець 17 ст.) вважалося, що білий колір – найпростіший.

Ньютон це заперечив, білий колір – це суміш безлічі кольорів. Припущення – будь-який колір отримується змішуванням основних.

## Фізика світла

Природа світла носить двоїстий характер.

Світло – потік частинок (корпускулярна теорія).

Світло – потік електромагнітних хвиль різної довжини і амплітуди (теорія Гука і Гюйгенса).

- Будь-які три, по-різному зафарбовані, промені світла утворюють який завгодно колір, якщо змішати їх в потрібній пропорції.
- Колір виражається в тривимірному просторі, узагальнюючи, через три атрибути

- $color = k_1 \cdot \vec{C_1} + k_2 \cdot \vec{C_2} + k_3 \cdot \vec{C_3}$
- $\vec{C_1}, \vec{C_2}, \vec{C_3}$  - лінійно-незалежні вектори
- $k_1, k_2, k_3$  - коефіцієнти, кількості атрибутів

## Поняття моделі кольорів

Призначення колірної моделі - дати засоби опису кольору в межах деякого колірного діапазону, у тому числі і для виконання інтерполяції кольорів.

Колірні моделі враховують біологію, фізику і математику кольору.

Існують різні моделі, оскільки із зображенням виконуються різні дії: *відображення на екран, видрук на принтер, опрацювання кольорів, перетворення в cірі тони, корекція яскравості, інтенсивності і т.п.*

Кожна модель має своє призначення, тобто ефективна для виконання окремих операцій.

# Класифікація моделей

Розглянемо

**апаратно-орієнтовані** триколірні моделі

RGB (aRGB, sRGB), CMY (CMYK)

**апаратно-незалежні** триатрибутні моделі,

які поділяються на

**перцепційні** HSV, HSL...

та

**модифіковані** XYZ, Lab...

## Адитивна модель RGB

- це аппаратно-орієнтована модель, в якій кольори описуються за допомогою змішування трьох базових кольорів – **червоного** (R),  
**зеленого** (G),  
**синього** (B) – в різних пропорціях.

Тому модель RGB наз. *адитивною*  
(від англ. «add» складати, додавати).

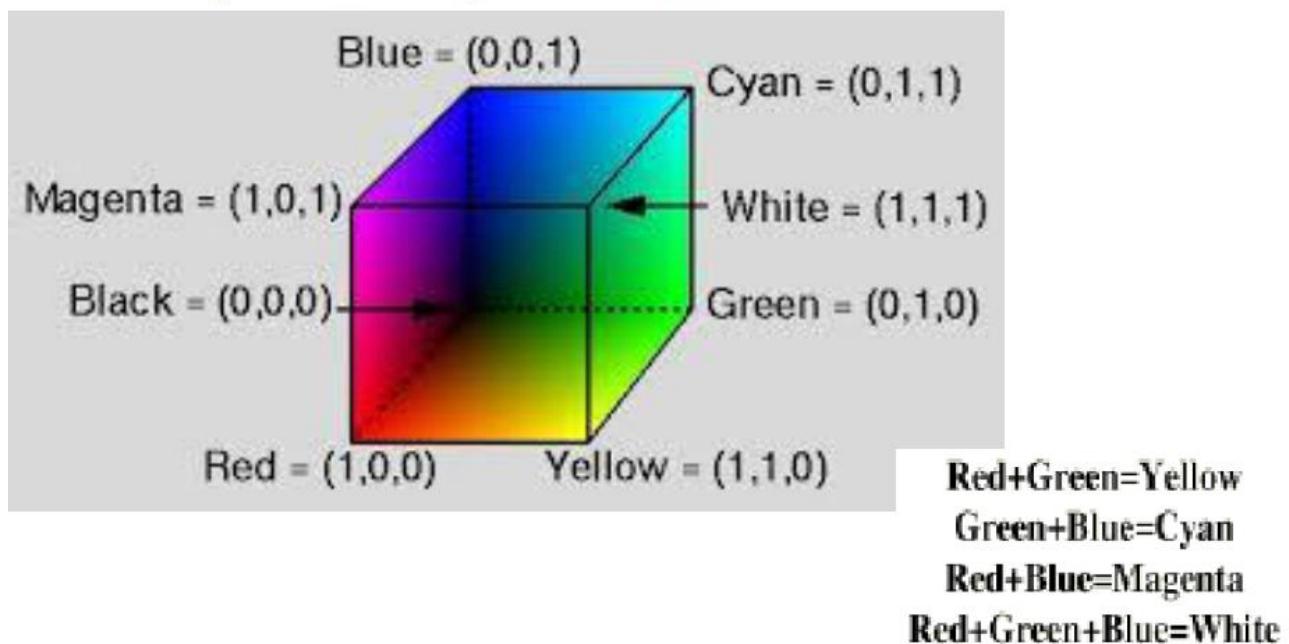
Схема адитивних кольорів працює на основі принципу випромінювання світла.

RGB використовується в дисплеях для формування кольорів.

У цій схемі відсутністю всіх кольорів є чорний колір, а присутністю всіх кольорів - білий.

Будь-який колір представляється точкою з трьома координатами  $(r,g,b)$ , де  $r,g,b \in [0;1]$ .

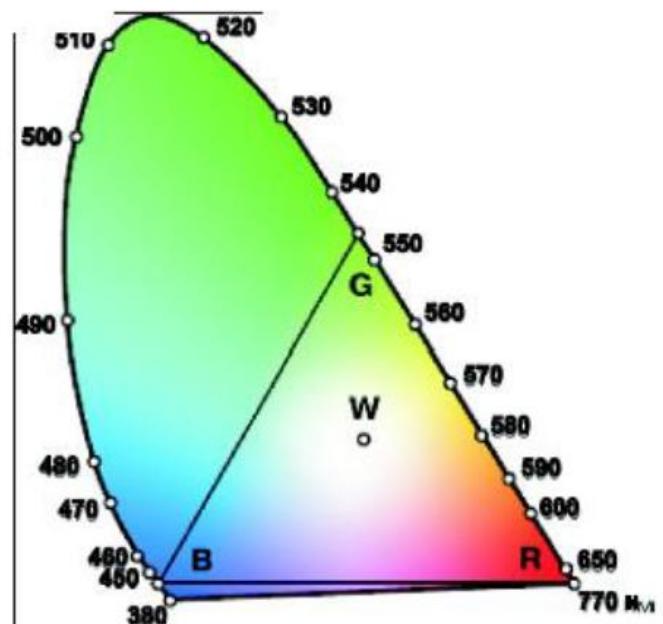
## Колірний куб моделі RGB



# Недолік RGB

- обмеженість у застосуванні, лише на пристроях, які працюють за принципом випромінювання,
- неможливість відображення деяких кольорів, а саме насычених зелено-синіх.

## Трикутник Максвела



# Субтрактивна модель CMY

- апаратно-орієнтована модель, яка використовується для формування кольорів на основі принципу **віднімання** від падаючого світлового потоку частини, яка формується шаром фарби з трьох компонентів **блакитний/Cyan**, **пурпурний/Magenta**, **жовтий/Yellow**.

Тому модель CMY називають *субтрактивною* (від англ. «sub» віднімати).

Колір в CMY утворюється при відніманні інших кольорів від загального променя світла ( $r+g+b$ ).

У цій схемі білий колір утворюється в результаті відсутності всіх кольорів, тоді як їх присутність дає чорний колір.

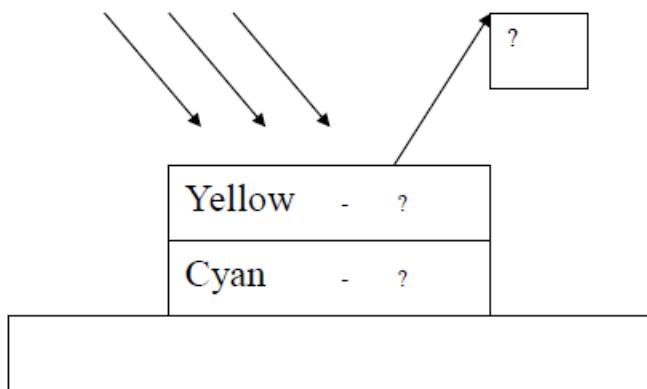
Схема субтрактивних кольорів працює на основі принципу відбиття та поглинання світла.

Дана система була широко відома задовго до того, як комп'ютери почали використовуватися для створення графічних зображень.

Її область застосування – поліграфія.

# Доповнюючі кольори

- У просторі RGB навпроти базових кольорів розміщено базові кольори CMY. Кажуть, що вони доповнюють один одного, поглинають.



Кольори моделі CMY є додатковими до кольорів моделі RGB, тобто доповнюючими їх до білого.

Таким чином система координат CMY - той же куб, що і для RGB, але з початком відліку в точці з RGB-координатами  $(1,1,1)$ , відповідною білому кольору.

# Звідки четверта компонента?

Насправді користуються системою CMYK (а не CMY).

Компонента K (від “Black”) введена додатково, оскільки це не економно змішувати 3 “непрості” фарби для отримання “простого” чорного кольору, тим більше ідеального чорного кольору змішуванням трьох базових не отримується.

## Зв’язок адитивної та субтрактивної моделей

$$\begin{pmatrix} r \\ g \\ b \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} c \\ m \\ y \end{pmatrix}$$

Перетворення кольорів з системи RGB в систему CMYK стикається з рядом проблем. Основна складність полягає в тому, що в різних системах кольори можуть змінюватися.

У даний час більшість графічних редакторів дозволяють працювати безпосередньо в кольорах CMYK.

**Тема.** Створення ілюстративних зображень (на прикладі використання Adobe Illustrator).

**Мета.** Вивчити методи та інструменти побудови векторної графіки на прикладі редактора Adobe Illustrator

## Теоретичні матеріали

*Adobe Illustrator* — професійний графічний редактор для створення та редагування векторної графіки від компанії Adobe. Програма відповідає всім галузевим стандартам, дозволяє створювати логотипи, піктограми, креслення, типографіку й ілюстрації для друку, веб-сайтів, відео та вмісту для мобільних пристройів.

### 1. Робоче середовище та інтерфейс користувача Adobe Illustrator

Створення документів та файлів і керування ними здійснюється за допомогою різноманітних елементів, наприклад панелей, смуг та вікон. Будь-яке розміщення цих елементів називається робочим середовищем. Програму Illustrator можна налаштувати за своїм бажанням, вибравши одне з попередньо встановлених робочих середовищ або створивши власне.

Робоче середовище «Початок роботи» відображається під час запуску Illustrator або коли немає відкритих документів.

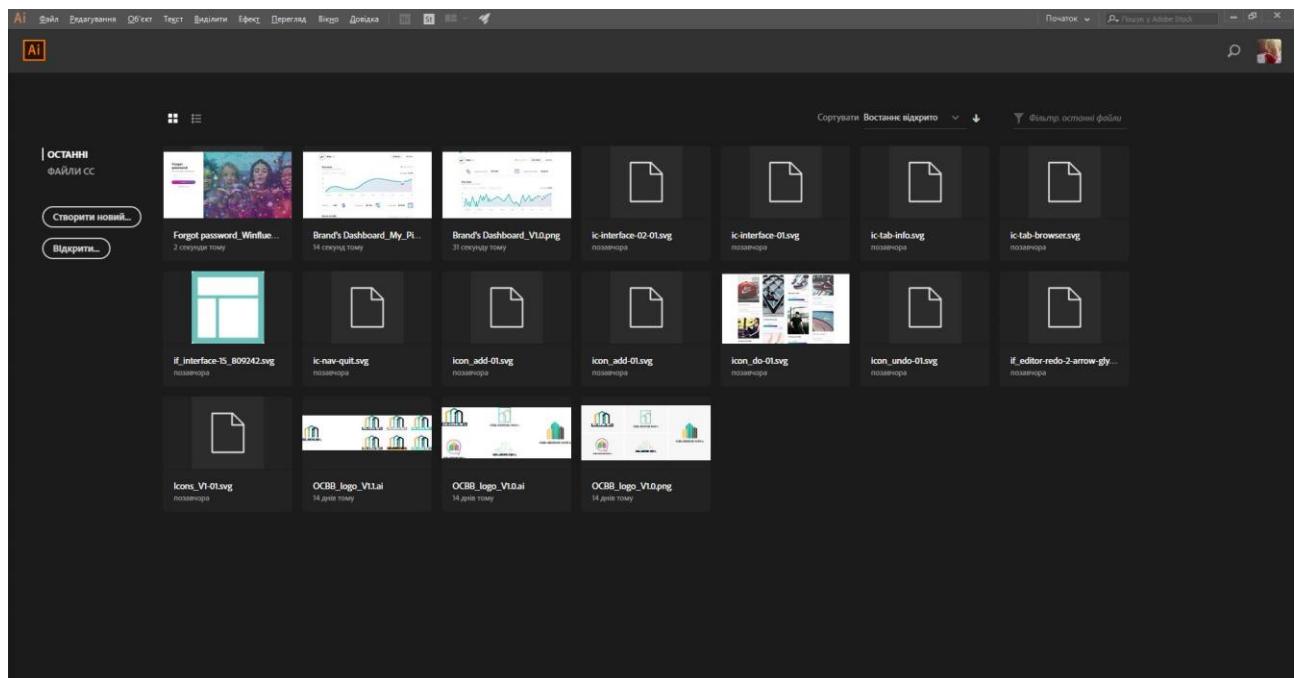


Рис. 1. Робоче середовище «Початок роботи»

У робочому середовищі «Початок роботи» можна здійснювати перехід між переглядом сегментів та списків. Це зручно коли ми можемо орієнтуватись на

візуальне відображення, та в той же момент переглядати властивості файла, такі як: останнє відкриття, розмір, тип.

Після відкриття чи створення файла - користувач потрапляє у робоче середовище програми.

Важливо зазначити що після створення, наш новий файл має 2 важливі площини. Перша це робоча область (сіре тло) та монтажна область /арт борд (біла область). Створювати та малювати об'єкти можна у двох областях, але різниця полягає в тому при збереженні файла можна увімкнути чекбокс, який або виводить на збереження усі елементи, або лише ті що розміщені у монтажних областях, також кожну монтажну область можна експортувати в окремий файл.

На Рис.2 відображене загальне розміщення панелей у робочому середовищі.

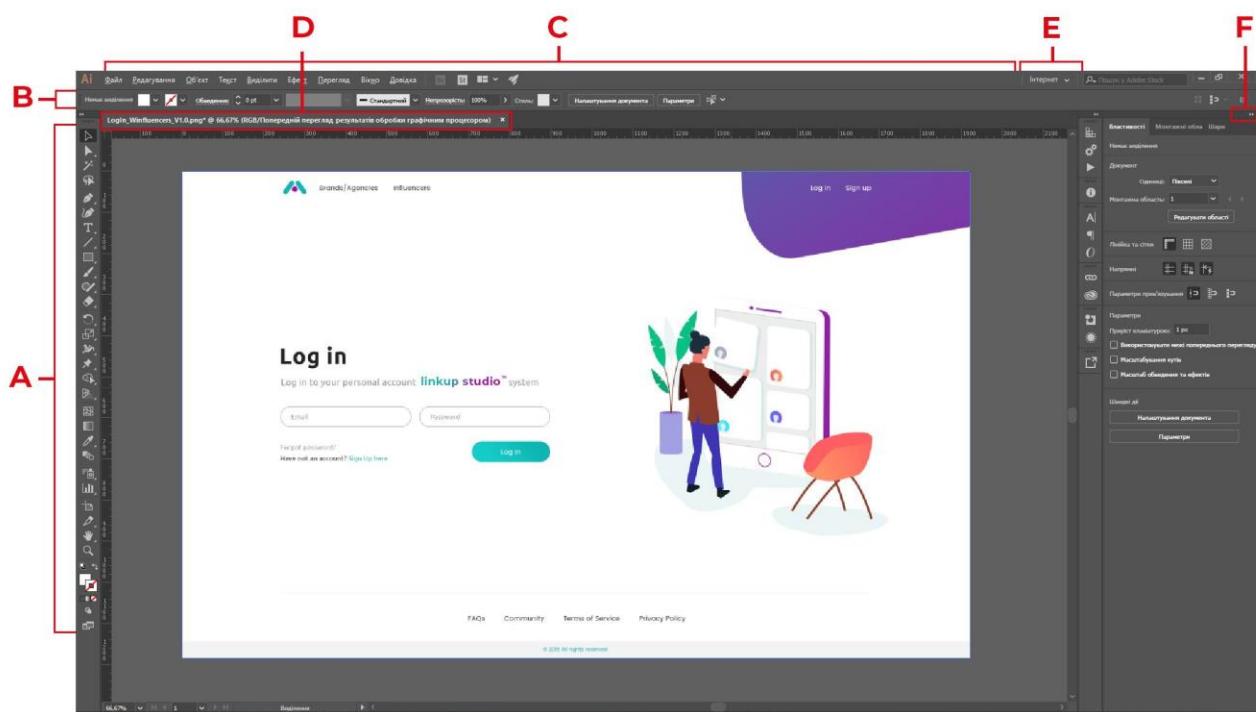


Рис.2. Основне робоче середовище. **А.** Панель інструментів; **В.** Панель керування; **С.** Панель програми; **Д.** Вкладки вікон документів; **Е.** Перемикач робочих середовищ; **Ф.** Кнопка «Згорнути до значків»

- У вікні *програми* - всі елементи групуються в одне інтегроване вікно, яке дозволяє працювати з програмою як з окремим елементом. Під час переміщення або зміни розміру вікна програми або будь-яких його елементів усі елементи реагують на цю дію, що попереджає накладання.
- Панель *програми* у верхній частині містить перемикач робочого середовища, меню (лише для Windows) та інші елементи керування програми. На комп'ютері Mac панель програми доступна лише тоді, коли вікно програми вимкнено.
- Панель *інструментів* містить інструменти для створення та редагування зображень, графіки, елементів сторінок і так далі. Пов'язані інструменти згруповано.

- *Панель керування* відображає параметри для вибраного об'єкта.
- У *вікні документа* відображається назва файлу, з яким зараз виконується робота. Вікна документа можна групувати, накладати одне на інше або закріплювати.

Якщо потрібно змінити робоче середовище перейдіть у розділ **Вікно > Робоче середовище**, для додавання додаткових панелей **Вікно** та відмітьте чекбоксами які панелі вам потрібні.

У головній панелі (Рис. 2. С.) зібрані всі команди програми:

- *Файл* - робота з документами: створення, відкриття, збереження, експорт, імпорт зображень, друк і т. д.;
- *Редагування* - редагування, робота зі стилями документу, налаштування;
- *Об'єкт* - редагування та зміна об'єктів, робота з групами, контурами, трастування і т. д.;
- *Текст* - робота з властивостями шрифтів, та символами;
- *Виділити* - можливість виділяти певні об'єкти / монтажні області / точки та інше.;
- *Ефект* - накладення та налаштування ефектів.
- *Перегляд* - параметри перегляду документа, відображення сітки, фрагментів, параметрів кольорів, зумування та інше.;
- *Вікно* - керування вікнами документів та їхній виклик;
- *Довідка* - система допомоги, керування обліковим записом і корисні посилання.

У панелі інструментів (Рис. 2. А.) розміщені інструменти для вибору та маніпулювання об'єктами. Деякі інструменти мають параметри, які з'являються за подвійного клапання на інструменті. Це стосується, зокрема, інструментів, що дозволяють вам виділяти, використовувати текст, розфарбовувати, малювати, брати зразки, редагувати й переміщати зображення. Для перегляду назву інструмента утримуйте вказівник над ним.

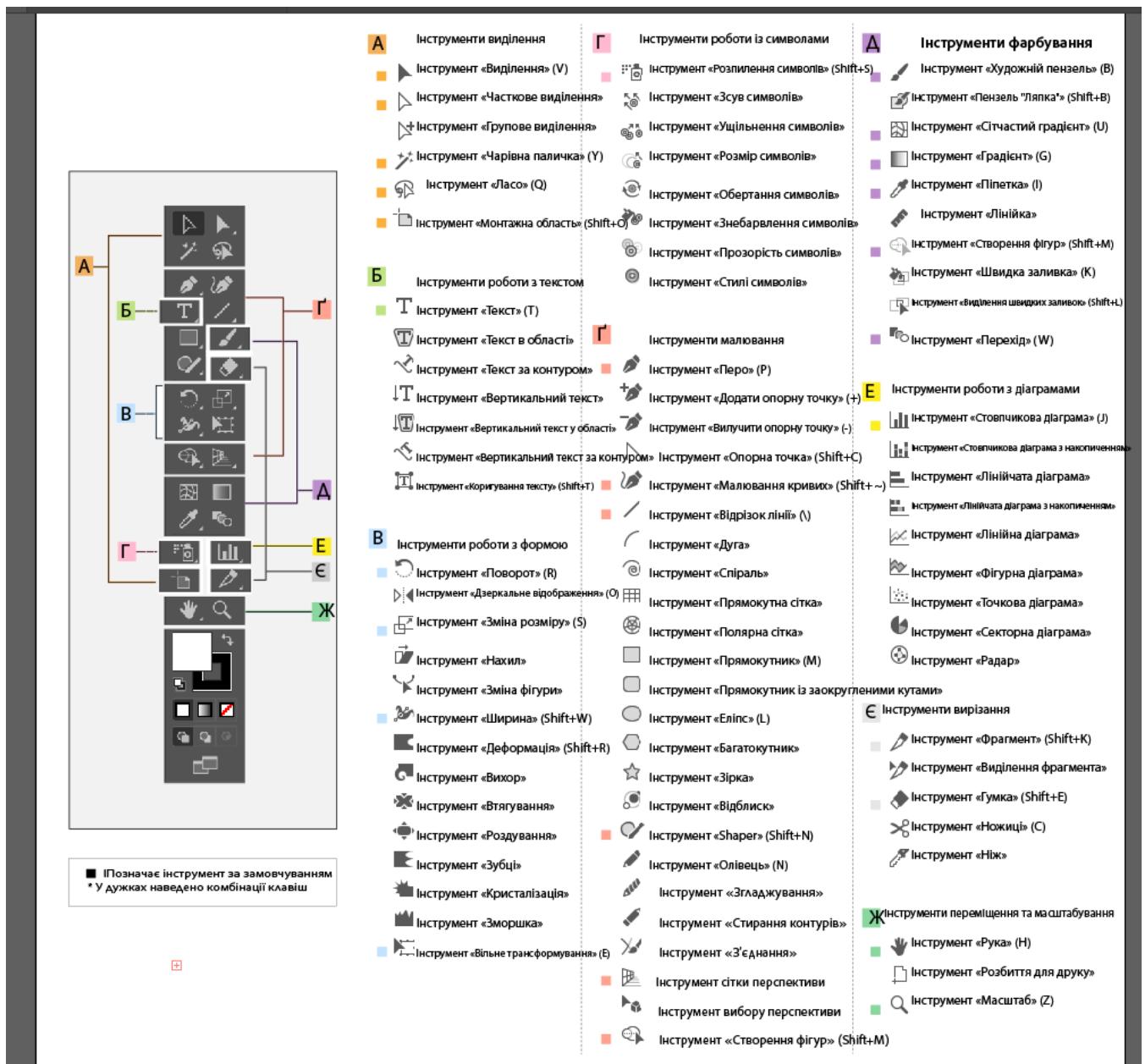


Рис. 3. Огляд: панель інструментів. **А.** Інструменти виділення; **Б.** Інструменти роботи з текстом; **В.** Інструменти роботи з формою; **Г.** Інструменти роботи з символами; **Г.** Інструменти малювання; **Д.** Інструменти фарбування; **Е.** Інструменти роботи з діаграмами; **Е.** Інструменти вирізання; **Ж.** Інструменти переміщення та масштабування;

У нових версіях програми Illustrator з'явилася панель «Властивості» у програмі можна переглядати налаштування та елементи керування, що залежать від контексту поточного завдання або процесу. Ця панель має полегшити роботу в програмі й забезпечити зручний доступ до елементів керування в потрібний момент.

За замовчуванням панель властивостей доступна в робочому середовищі «Основне». Її також можна викликати в меню Вікно > Властивості.

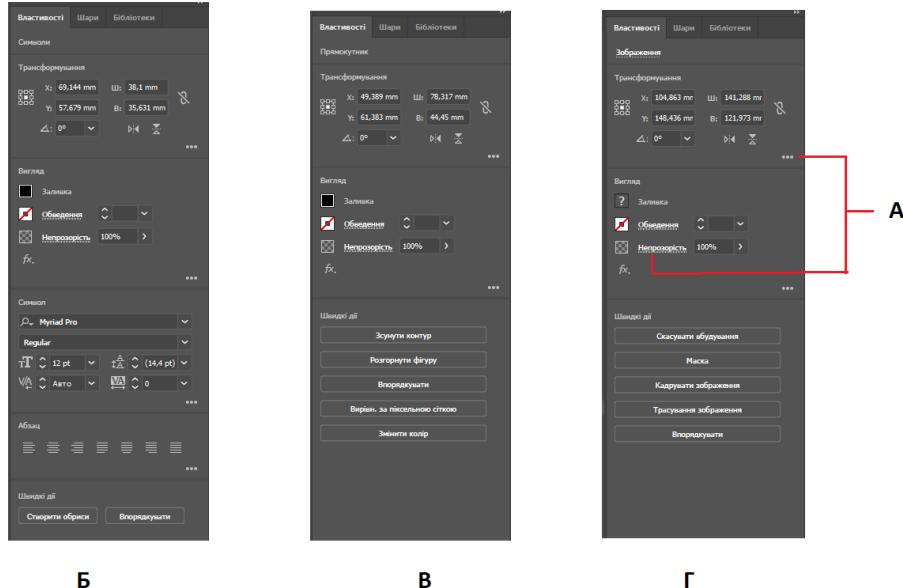


Рис.4. Контекстні опції на панелі «Властивості». **А.** Відображення додаткових опцій **В.** Елементи керування текстовим об'єктом **С.** Елементи керування контуром **Д.** Елементи керування прив'язаним зображенням

Незалежно від вираного вмісту, на панелі властивостей з'являється два набори елементів керування:

- Для перетворення й зміни вигляду: ширина, висота, заповнення, обведення, непрозорість тощо.
- Динамічні функції: спеціальні елементи керування, що залежать від вмісту вираної ділянки. Це може бути, напаштування атрибутів символів й абзаців для текстових об'єктів. Для графічних об'єктів на цій панелі відображаються інструменти кадрування, накладання масок, вбудовування та його скасування, а також трастування зображення

Незалежно від вираного вмісту, на панелі властивостей з'являється два набори елементів керування:

## 2. Виділення об'єктів

Перед внесенням змін до об'єкту, потрібно виділити його серед інших об'єктів, які його оточують. Illustrator надає такі способи та інструменти виділення:

- *Режим ізоляції* - дає змогу швидко ізолювати шар, підшар, контур або групу об'єктів від усіх інших елементів у документі. Під час роботи у режимі ізоляції всі неізольовані об'єкти у документі відображаються затіненими та не можуть виділятися або редагуватися.
- *Панель «Шари»* - дозволяє швидко та точно виділяти поодинокі або декілька об'єктів. Можна виділяти один об'єкт (навіть якщо він знаходитьться у групі), всі об'єкти, що містяться у шарі, та цілі групи.
- *Інструмент «Виділення»*  - дає можливість виділяти об'єкти та групи, натискаючи на них кнопкою миші або перетягуючи їх. Ви також можете виділяти групи, які входять до складу інших груп, та об'єкти, які входять до груп.
- *Інструмент «Часткове виділення»*  - дає змогу виділяти окремі опорні точки або відрізки контура шляхом натискання на них або виділення всього контура чи групи, виділяючи будь-яку плашку на елементі. Ви також можете виділяти один або декілька об'єктів у групі об'єктів.
- *Інструмент «Групове виділення»*  - дає змогу виділяти об'єкт у групі, окрему групу у складі декількох груп або набір груп у межах ілюстрації. Кожне додаткове натискання додає об'єкти з наступної групи в ієрархії.
- *Інструмент вибору перспективи*  - дозволяє застосовувати для об'єктів і тексту перспективу, змінювати активні площини, переміщати об'єкти в перспективі та переміщати їх перпендикулярно до робочої площини.
- *Інструмент «Ласо»*  - дає змогу виділяти об'єкти, опорні точки або сегменти контура шляхом перетягування всіх об'єктів або їх частини.
- *Інструмент «Чарівна паличка»*  - дозволяє виділяти об'єкти однакового кольору, товщину лінії, колір обведення, непрозорість або режим змішування шляхом натискання на об'єкт.
- *Інструмент «Виділення швидких заливок»*  - дозволяє виділяти фацети (площі, обведені контурами) та краї (частини контурів між перетинами) груп зі швидкою заливкою.

Команди виділення що знаходяться у меню «Вибір» Дозволяє швидко виділяти або знімати прaporець з усіх об'єктів та виділяти об'єкти на основі їх позиції по відношенню до інших об'єктів. Ви можете виділяти всі об'єкти, які відносяться до якогось конкретного виду

або мають певні спільні атрибути, та зберігати або завантажувати виділенні об'єкти. Також можна виділяти всі об'єкти в активній монтажній області.

### 3. Складові елементи зображення

#### 3.1. Прямоуглини

Виберіть інструмент «Прямоуглиник»  або «Прямоуглиник з округленими кутами» . Щоб намалювати прямоуглиник, протягніть вказівник по діагоналі, поки прямоуглиник не досягне бажаного розміру. Щоб намалювати квадрат, утримуйте клавішу Shift і протягніть вказівник по діагоналі, поки квадрат не досягне бажаного розміру. Або щоб створити квадрат чи прямоуглиник із використанням значень, клацніть у точці, де повинен знаходитися верхній лівий кут. Задайте ширину й висоту (а для прямоуглиника з округленими кутами – ще й радіус кута), і натисніть «OK». (Рис. 5)

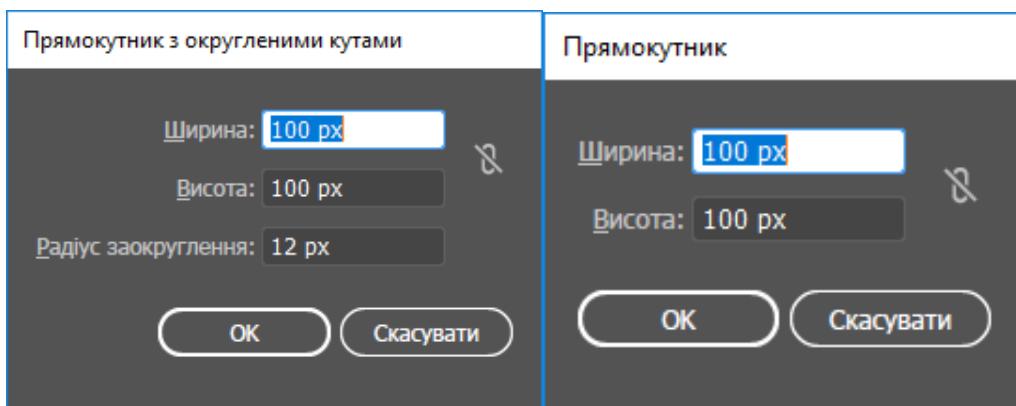


Рис.5. Параметри інструмента «Прямоуглиник»

Намалювавши прямоуглиник із заокругленими кутами, ви вже не зможете змінити радіус заокруглення. Якщо знадобиться змінити радіус заокруглення, намалюйте звичайний прямоуглиник і виберіть меню «Ефект» > «Перетворити на фігуру» > «Прямоуглиник з округленими кутами», а потім укажіть параметри прямоуглиника з округленими кутами. Щоб змінити радіус заокруглення або інші параметри, змініть параметри ефектів на панелі «Вигляд».

Щоб швидко округлити кути будь якої фігури оберіть на панелі інструментів інструмент «Часткове виділення» , натисніть на об'єкт кутам якого потрібно надати округлої форми, і біля гострих кутів з'являться точки надання радіусі, ви можете активувати одні натиснувши на неї два рази, або всі одночасно – натиснувши один раз, ра протягуючи мишкою змінювати радіус заокруглення (Рис.6).



Рис.6. Надання Радіусу за допомогою інструмента “Часткове виділення”

### 3.2. Еліпси

Клас об'єктів "Еліпс" включає об'єкти, які утворюються з еліпса засобами, аналогічними закругленню кутів прямокутника, а саме сектори і дуги еліпсів. У геометрії розміри еліпса визначаються розмірами його півосей, у Illustrator – розмірами габаритного прямокутника (що співпадає з рамкою виділення, описаний навколо еліпса).

Виділіть інструмент «Еліпс» (гаряча клавіша L), або натисніть на «Прямокутник» та утримуйте поки не з'явиться вікно з додатковими фігурами. Перетягніть вказівник по діагоналі, поки не одержите еліпс бажаного розміру. Або клацніть мишею в точці, де має бути верхній лівий кут описаного прямокутника еліпса. Введіть висоту й ширину еліпса та натисніть кнопку «OK». Щоб створити коло утримуйте клавішу Shift.

### 3.3. Дуги, спіралі

Для того щоб намалювати дугу натисніть й утримуйте інструмент «Відрізок лінії» ( ). Виділіть інструмент «Дуга» . Розташуйте вказівник у бажаній початковій точці дуги та протягніть його до бажаної кінцевої точки. Або у діалоговому вікні клацніть квадрат на локаторі контрольної точки , щоб визначити точку, з якої починяється малювання дуги. Потім задайте один із представлених варіантів і натисніть кнопку «OK».

*Довжина, вісь X:* задає ширину дуги.

*Довжина, вісь Y:* задає висоту дуги.

*Тип:* визначає, чи повинен об'єкт мати відкритий або замкнутий контур.

*База вздовж:* задає напрям дуги. Виберіть вісь X чи Y, залежно від того, має основа дуги розміщуватися вздовж горизонтальної (x) чи вертикальної (y) осі.

**Нахил:** вказує напрям нахилу дуги. Для одержанняувігнутого (всередину) нахилу введіть від'ємне значення. Для одержання вигнутого (всередину) нахилу введіть додатне значення. При нульовому нахилі утворюється пряма лінія.

**Залити дугу:** заливає дугу поточним кольором заливки.

Для того щоб намалювати спіраль натисніть й утримуйте інструмент «Відрізок лінії» (⌖). Виберіть інструмент «Спіраль» (spiral) далі протягніть вказівник, поки не одержите спіраль бажаного розміру. Протягніть вказівник по колу, щоб обернути спіраль. Або Клацніть мишею в точці, де має починатися спіраль. У діалоговому вікні задайте один із наступних варіантів і натисніть кнопку «OK».



Рис.7. Спіралі з різною кількістю сегментів

**Радіус:** задає відстань від центру до зовнішньої точки спіралі.

**Зменшення:** задає, наскільки меншим є наступний виток відносно попереднього.

**Сегменти:** задає, скільки сегментів має спіраль. Кожний повний виток складається із чотирьох сегментів.

**Стиль:** вказує напрям спіралі.[1]

### 3.4. Зірки, відблиски

Для побудови зірки Натисніть і утримуйте інструмент «Прямокутник» (rectangle) на панелі інструментів. Виберіть інструмент «Зірка» (star). Перетягніть вказівник по діагоналі, поки не одержите зірку бажаного розміру. Протягніть вказівник по колу, щоб обернути зірку. Натисніть клавіші «Стрілка вгору» чи «Стрілка вниз», щоб додати чи видалити вершини зірки.



Рис.8. Зірки із різною кількістю вершин

Або клацніть мишею в точці, де має бути розміщений центр зірки. У діалоговому вікні (Рис.9) задайте - «Радіус 1» радіус від центру зірки до її внутрішніх вершин. «Радіус 2» задайте радіус від центру зірки до її зовнішніх вершин. «Кількість вершин» задайте, скільки вершин повинна мати зірка. Потім натисніть «ОК». Під час малювання зірки використовуйте стрілки вгору й вниз для збільшення або зменшення кількості променів зірки.

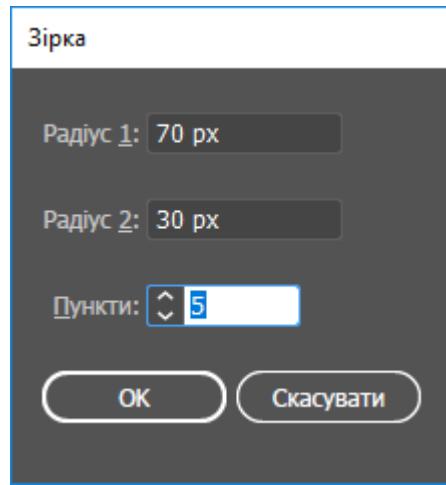


Рис.9. Вікно параметрів інструменту “Зірка”

Інструмент «Відблиск» створює об'єкти відблисків зі світлим центром, променями та кільцями. Даний інструмент потрібен для створення ефекту, подібного до відблиску об'єктива фотоапарата. Об'єкт відблиску складається з таких параметрів (Рис.10): **A**. Центральна ручка **B**. Кінцева ручка **C**. Промені (для чіткості показані чорним кольором) **D**. Гало **E**. Кільце

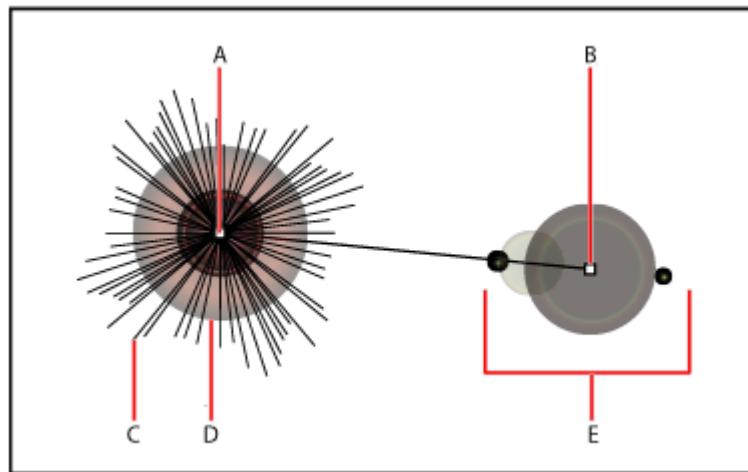


Рис.10. Компоненти відблиску

Відблиски містять центральну ручку та кінцеву ручку. Ручки служать для розташування відблиску та його кілець. Центральна ручка розташована в яскравому центрі відблиску – із цієї точки починається контур відблиску.

Для створення стандартного відблиску натисніть та утримуйте інструмент «Прямоугольник» (□). Виберіть інструмент «Відблиск» (○). Натисніть клавішу Alt і клацніть мишею в точці, де має бути розташована центральна ручка відблиску.

Для створення власного відблиску оберіть інструмент “Відблиск” натисніть для має розташуватись центр відблиску потім перетягніть, щоб задати розмір центру, розмір гало та кут повороту променів. Перш ніж відпустити кнопку миші, натисніть клавішу **Shift**, щоб обмежити розташування променів під кутом. Щоб додати чи видалити промені, натисніть клавішу «Стрілка вгору» чи «Стрілка вниз». Щоб утримати на місці центр відблиску, натисніть клавішу **Ctrl**.

Або створити відблиск можна за допомогою діалогового вікна (Рис.11). У діалоговому вікні «Параметри інструмента “Відблиск” задайте:

- Загальний діаметр, непрозорість та яскравість центра відблиску.
- «Ріст» гало в процентах від загального розміру, а також розкид гало (0 – чітке, а 100 – розкидане).
- Якщо відблиск має містити промені, виділіть «Промені» та вкажіть кількість променів, найдовший промінь (в процентах від середнього) та розкид променів (0 – чіткі, а 100 – розкидані).
- Якщо відблиск повинен містити кільця, виділіть «Кільця» та вкажіть відстань контура між середньою точкою гало (центральною ручкою) та центральною точкою найвіддаленішого кільця (кінцевою ручкою), кількість кілець, найбільше кільце (в процентах від середнього кільця) та напрям кута кілець.

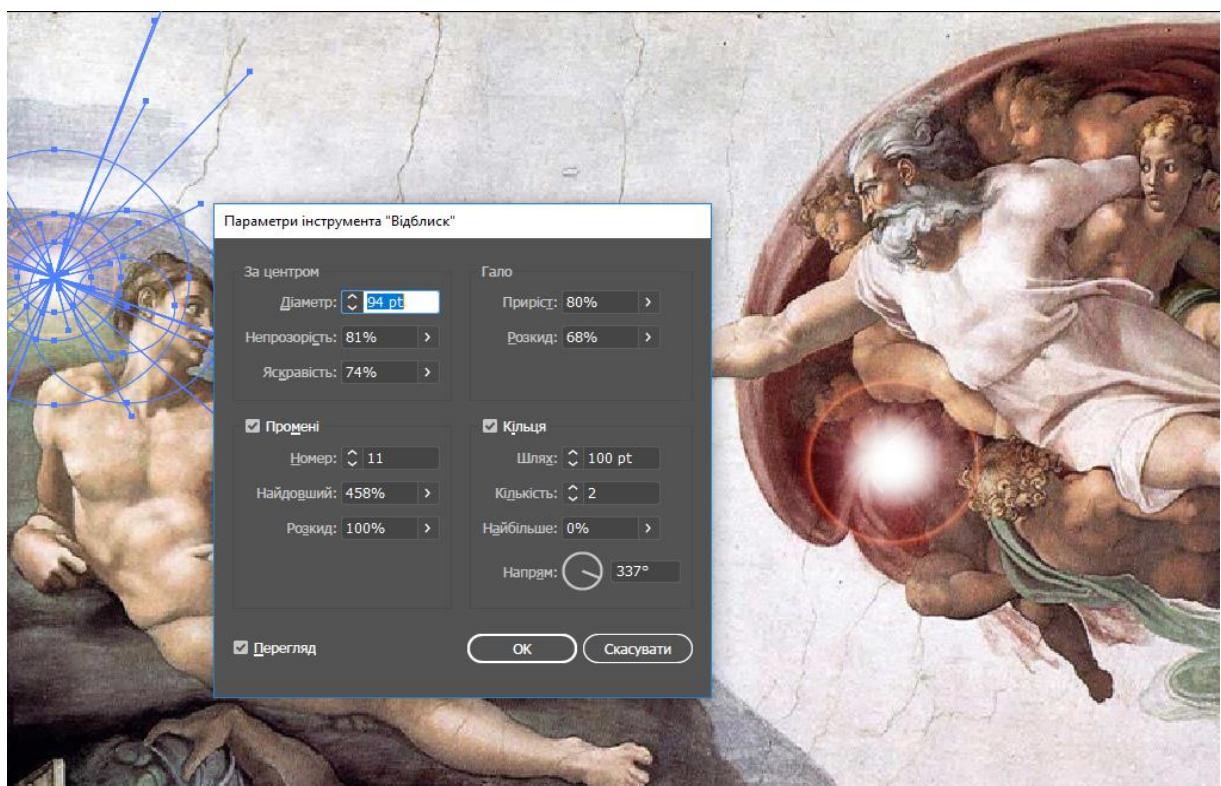


Рис.11. Параметри інструмента «Відблиск»

Для того щоб об'єднати контури з використанням багатьох можливостей взаємодії слід скористатися панеллю “Обробка контурів” («Вікно» > «Обробка

контурів») . Ефекти обробки контурів дозволяють створювати нові фігури з об'єктів, які накладаються.

## 4. Побудова ліній. Робота з кривими

Для представлення різних класів ліній передбачено декілька класів об'єктів. Об'єкти об'єднуються в один клас по ознаках загальної структури і поведінки, тобто реакції на дії з ними.

### 4.1. Модель кривої

*Крива Безье* - це параметрично задана крива яка застосовується в комп'ютерній графіці для побудови плавних вигинів та гладких ліній. Крива цілком лежить в опуклій оболонці своїх опорних точок. Це властивість кривих Безье з одного боку значно полегшує завдання знаходження точок перетину кривих (якщо не перетинаються опуклі оболонки опорних точок, то не перетинаються і самі криві), а з іншого боку дозволяє здійснювати інтуїтивно зрозуміле управління параметрами кривої в графічному інтерфейсі за допомогою її опорних точок. Крім того афінні перетворення кривої (перенесення, масштабування, обертання та ін.) також можуть бути здійснені шляхом застосування відповідних трансформацій до опорних точок.

Для того щоб отримати величезну різноманітність форм, із яких можна скласти об'єкт будь-якої складності потрібно змінити форму канонічної кривої Безье. У програмах векторної графіки існує єдиний спосіб – це інтерактивне переміщення опорних і керуючих точок. Якщо переміщуються початкова чи кінцева точки, то крива стане відповідним чином змінюватися (витягуватися чи стискатися). Переміщення керуючих точок змінює кривизну відповідної частини кривої Безье. Таким чином, за допомогою переміщення цих чотирьох точок отримують необмежену кількість форм кривих Безье, яка може бути одним окремим сегментом складного векторного контуру. У кожному сегменті можна добавляти опорні точки, які теж дозволяють змінювати форму кривої. Додавання нових опорних точок у межах одного сегмента кривої не протирічить тій умові, що окремі криві з'єднуються в ланцюг. Просто крива Безье добавляється не до кінця контуру, а розміщується всередині вже існуючого контуру.

У основі прийнятої в *Illustrator* моделі ліній лежать два поняття: вузол і сегмент. Вузлом називається точка на площині зображення, що фіксує положення одного з кінців сегменту. Сегментом називається частина лінії, що сполучає два суміжні вузли. Вузли і сегменти нерозривно зв'язані один з одним: у замкнuttій лінії вузлів стільки ж, скільки сегментів, а в незамкнuttій - на один більше. Будь-яка лінія складається з вузлів і сегментів, і всі операції з лініями насправді є операціями саме з ними. Вузол повністю визначає характер попереднього йому сегменту, тому для розімкненої лінії важливо знати, який з двох її крайніх вузлів є початковим, а для замкнutoї - напрям лінії (за годинниковою стрілкою або проти неї). За

характером попередніх сегментів виділяють три типи вузлів: початковий вузол розімкненої кривої, прямолінійний (Line) і криволінійний (Curve). [2]

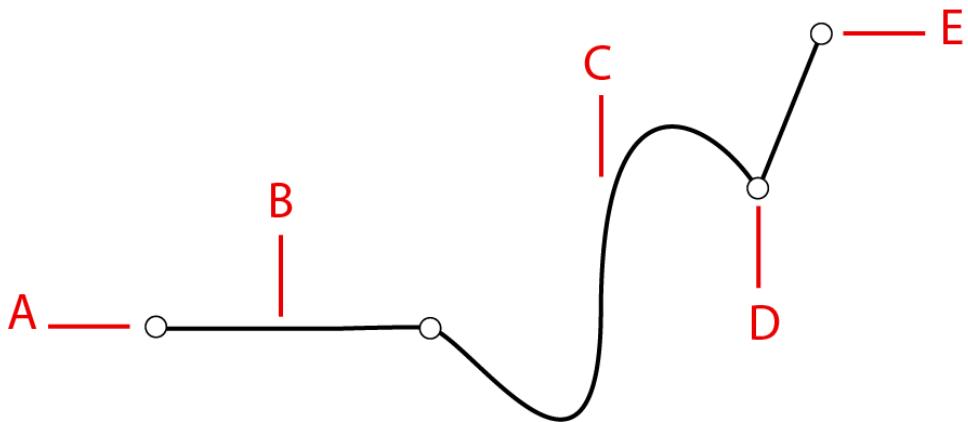


Рис.12. Сегменти,вузли, лінії. **А.** Початковий вузол; **В.** Прямолінійний сегмент лінії; **С.** Криволінійний сегмент лінії; **Д.** Проміжний вузол; **Е.** Кінцевий вузол;

З боку примикання до виділеного вузла криволінійного сегменту відображається так звана точка напрямної. Чим далі точка напрямної від вузла, тим повільніше криволінійний сегмент відхиляється від своєї дотичної. При виділенні вузла, що розділяє два криволінійні сегменти, на екрані відображаються чотири направляючі точки – з обох кінців кожного сегменту.

#### 4.2. Лінії замкнені, розімкнені і сполучені

Існує ще одна класифікація ліній в залежності від кількості і стану крайніх вузлів лінії. Крайнім вузлом називається вузол лінії, суміжний тільки з одним її сегментом. Вузол лінії, що не має попереднього сегменту, називається початковим. Лінія, що має початковий вузол, називається розімкненою (Open curve). Лінія, в якій крайні вузли відсутні, називається замкнutoю (Closed curve). Сполучені лінії складаються з декількох гілок (subpath), кожна з яких є замкненою або розімкненою лінією. Сполучені об'єкти виникають, зокрема, при виконанні операції з'єднання об'єктів і при перетворенні в криві інших об'єктів (наприклад, текстів).

Практично будь-який графічний об'єкт Illustrator може бути перетворений у криву. І навпаки, багато складних об'єктів будуються на базі однієї або декількох ліній. Тому розуміння моделі лінії і прийомів роботи з лініями відіграють дуже важливу роль в побудові зображення.

#### 4.3. Малювання інструментом “Перо”

Для роботи з кривими Безье в Adobe Illustrator застосовують інструмент “Перо” та інструмент “Малювання кривих”.

Найпростіший контур, який можна нарисувати інструментом «Перо», – це пряма лінія. Для цього слід клацнути інструментом «Перо», щоб створити дві

опорні точки. Продовжуючи клацати, ви створюєте контур із відрізків прямих ліній, що з'єднуються кутовими точками. Перший нарисований відрізок не буде видимий, поки ви не клацнете, щоб створити другу опорну точку. Щоб замкнути контур, поставте інструмент «Перо» на першу (порожню) опорну точку. Маленьке коло з'являється поряд із вказівником інструмента «Перо» , якщо його розташовано правильно. Клацніть або перетягніть вказівник, щоб замкнути контур.

Щоб залишити контур відкритим, клацніть, утримуючи **Ctrl** (Windows) або **Command** (macOS), будь-де за межами об'єктів. Щоб залишити контур відкритим, можна також взяти інший інструмент або вибрати «**Виділити** > «**Зняти виділення**». Також можна просто натиснути клавішу **Enter** або **Return**, щоб залишити контур відкритим.

Створити криву можна додаючи опорну точку, де крива змінює напрямок, або перетягуючи вказівник за лінією напряму, що утворює форму кривої. Довжина й нахил лінії напряму визначає форму кривої.

Криві легше редагувати, і ваша система може показувати та друкувати їх швидше, якщо ви рисуєте їх з меншою кількістю опорних точок. Якщо використовувати забагато точок, це може спричинити зайні вигини кривої. Натомість рисуйте, розставляючи опорні точки далеко одна від одної, і навчіться утворювати форму кривої, регулюючи довжину й кут ліній напряму.

1. Виберіть інструмент «Перо».
2. Поставте інструмент «Перо» на місце, де ви бажаєте розпочати криву, і утримуйте натиснутою кнопку миші. З'являється перша опорна точка, і вказівник інструмента «Перо» відображається як стрілка.
3. Перетягніть, щоб встановити крутість створюваного відрізка кривої, потім відпустіть кнопку миші.
4. Загалом подовжуйте лінію напряму приблизно на третину відстані до наступної опорної точки, що її плануєте поставити. (Ви можете відрегулювати лінію напряму пізніше з одного чи двох боків).
5. Утримуйте клавішу **Shift**, щоб обмежити значення повороту кутами, кратними  $45^\circ$ .
6. Поставте інструмент «Перо» там, де ви бажаєте закінчити відрізок кривої.

Щоб створити криву S-форми, перетягуйте в тому ж напрямку, що й напрямок попередньої лінії напряму. Потім відпустіть кнопку миші. (Рис.13)

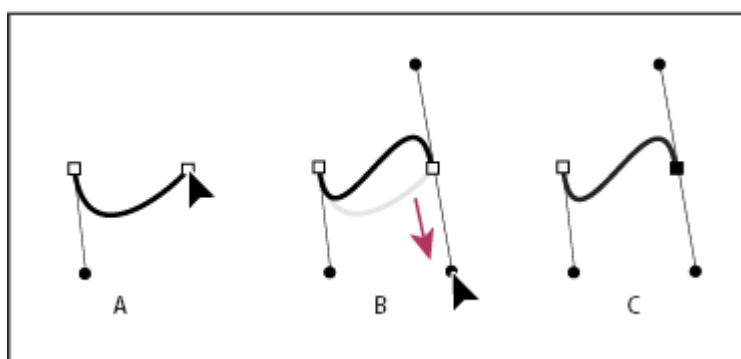


Рис.13. Малювання кривої S-форми

Для зміни місця опорних точок у ході малювання клацніть мишею, щоб створити опорну точку. Після цього, не відпускаючи кнопку, натисніть пробіл і перетягніть вказівник, щоб переставити опорну точку.

Для намалювання двох криволінійних сегментів, поєднаних під кутом - інструментом «Перо» перетягніть вказівник, щоб створити першу точку згладжування викривленого відрізка. Переставте «Перо» й перетягніть вказівник, щоб створити криву з другою точкою згладжування, потім, натиснувши й утримуючи Alt перетягніть лінію напрямку до її протилежного кінця, щоб задати нахил наступної кривої. Відпустіть клавішу та кнопку миші. Ця процедура перетворює точку згладжування на кутову, розділяючи лінії напряму. Переставте «Перо» на те місце, де бажаєте закінчити другий кривий відрізок, і перетягніть вказівник на нову точку згладжування, щоб закінчити другий кривий відрізок. (Рис. 14)

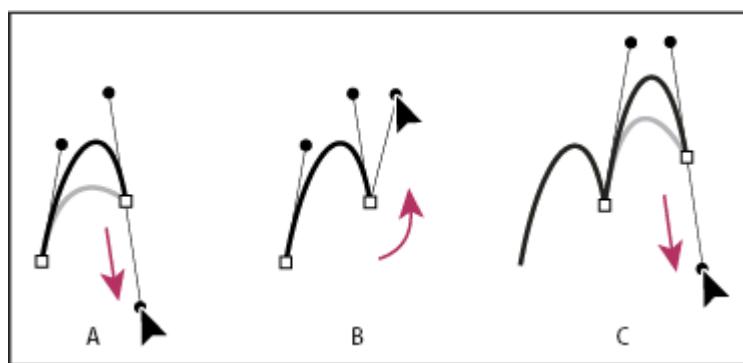


Рис.14. Малювання двох кривих

#### 4.4. Робота з інструментом малювання кривих

Інструмент малювання кривих спрощує процес створення контурів і робить малювання зручнішим і простішим. Цей інструмент дає змогу створювати, перемикати, редагувати, додавати та видаляти точки згладжування або кутові точки. Для швидкої та точної роботи з контурами більше не потрібно перемікатися між різними інструментами.

Виберіть інструмент малювання кривих (��). Розмістіть дві точки на монтажній області, а потім завдяки функції попереднього перегляду еластичного контуру спостерігайте, як змінюється форма контуру, що отримується в результаті, залежно від руху вказівника миші.

*Примітка: параметр еластичності контуру ввімкнено за замовчуванням. Щоб вимкнути його, виберіть команди меню «Параметри» > «Відображення виділення та прив'язки» > «Увімкнути параметр еластичності контуру».*

Розмістити точку згладжування можна, клацнувши кнопкою миші або торкнувшись потрібної ділянки на екрані. Щоб розмістити кутову точку, двічі клацніть кнопкою миші або, утримуючи клавішу **Alt**, клацніть один раз кнопкою миші чи торкніться екрана. (Рис.15)

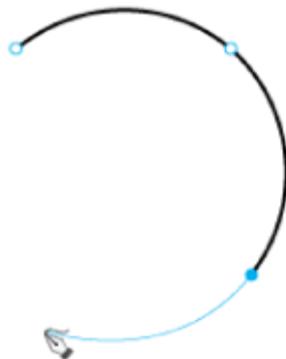


Рис.15. Точки згладжування для кривих

Для попереднього перегляду еластичного контура:

1. За допомогою «Пера» або інструмента малювання кривих один раз клацніть на монтажній області, щоб намалювати точку згладжування, а потім потягніть мишу, щоб створити держаки так, як необхідно.
2. Відпустіть кнопку миші. Коли вказівник миші переміщається по монтажній області, відображається контур, який показує, що буде намальовано, якщо встановити опорну точку в тому місці, де наразі розміщено вказівник миші.

Якщо відображений контур попереднього перегляду збігається з тим, що потрібно намалювати, клацніть у цьому місці мишею, і **Illustrator** намалює відповідний контур.

#### 4.5. Малювання інструментом “Олівець”

Інструмент «Перо» дає змогу малювати відкриті та закриті контури так, ніби ви малюєте олівцем на папері. Він є найзручнішим для швидкого ескізу або щоб створити вигляд, ніби намальовано від руки. Намалювавши контур, ви можете одразу змінити його, якщо треба.

Опорні точки встановлюються, коли ви малюєте інструментом «Олівець»; ви не визначаєте, де їх ставити. Проте ви можете налаштувати їх після того, як накреслите контур. Число поставленіх опорних точок визначається довжиною та складністю контуру, а також параметрами допуску в діалоговому вікні «Уподобання інструмента "Олівець"». Ці параметри регулюють, наскільки чутливим є інструмент «Олівець» до руху миші або стилуса графічного планшета.

Для малювання контурів натисніть і утримуйте інструмент Shaper (  ). Виберіть інструмент «Олівець» (Гаряча клавіша N). Поставте інструмент там, де має починатися контур, і потягніть, щоб його намалювати. Інструмент «Олівець» (  \* ) відображає маленький хрестик (x), щоб позначити малювання контуру довільної форми. Коли ви перетягуете вказівник, за ним слідує пунктирна лінія. Опорні точки з'являються на обох кінцях контуру та в різних місцях на ньому. Контур приймає поточні атрибути обведення та заливки, і залишається виділеним за замовчуванням.

Інструмент «Олівець» можна використовувати для малювання сегментів прямого контуру з обмеженням чи без.

- *Сегменти прямого контуру з обмеженням:* натисніть і утримуйте клавішу **Shift**, після чого почніть малювати сегменти прямого контуру за допомогою інструмента «Олівець» з обмеженням 0, 45 або 90 градусів. Під час малювання сегмента прямого контуру відображається курсор відповідної форми (  ).
- *Сегменти прямого контуру без обмеження:* натисніть і утримуйте клавішу **Alt**, щоб почати малювати сегменти прямого контуру без обмежень. Однак, для малювання ламаного контуру слід зробити таке:
  - Намалюйте сегмент лінії.
  - Виконайте одну з таких дій:
  - Утримуючи клавішу **Shift**, помістіть вказівник на кінцеву точку лінії. Після появи курсору продовження контуру (  ) клацніть кнопкою миші й намалюйте ще одну лінію.
  - Утримуючи натисненою кнопку миші, відпустіть і натисніть клавішу **Shift** або **Option/Alt** і намалюйте наступний сегмент.

За допомогою інструмента «Олівець» легко змінювати контури для цього виберіть контур, який бажаєте змінити. Поставте інструмент «Олівець» на контур або поряд із ним, щоб перемалювати. Ви побачите, що знаходитесь достатньо близько до контуру, коли маленький хрестик біля кінчика інструмента зникне. Перетягуйте інструмент, доки контур не набуде бажаної форми. (Рис. 16). Зверніть увагу щоб випадково не зробити з закритого контура відкритий чи навпаки.

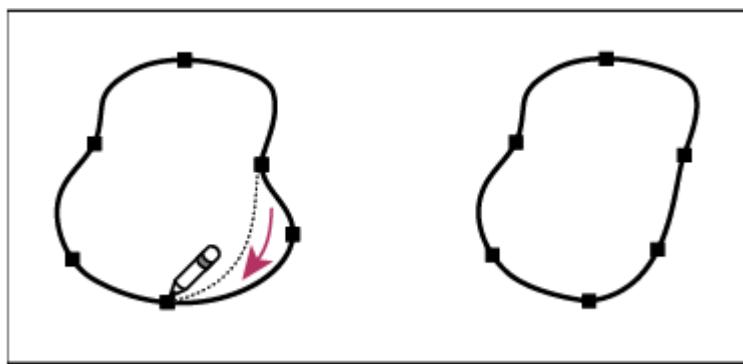


Рис.16. Зміна закритого контура за допомогою інструмента “Олівець”

Для налаштувань параметра інструмента «Олівець» (kritzel), двічі клацніть на нього. У діалоговому вікні ви зможете змінити наступні параметри:

**Точність** - Регулює, наскільки ви маєте пересунути мишу або стилус до того, як нова опорна точка буде додана до контуру. Для повзунка параметра «Точність» передбачено п’ять позначок, що дозволяють налаштувати точність малювання контуру. Крайня позначка зліва («Чіткий контур») забезпечує малювання якомога чіткішого контуру. Крайня позначка справа («Згладжений контур») дозволяє намалювати дуже м’який контур. Виберіть потрібну позначку.

**Залити нові штрихи олівця** - Застосовує заливку до штрихів олівця, намальованих після вибору цього параметра, але не до існуючих штрихів олівця. Не забудьте вибрати заливку перед тим, як наносити штрихи олівцем.

**Параметр «Для переходу до інструмента "Згладжування"»** - Якщо цей параметр активовано, натиснення клавіші Alt (Windows) або Option (macOS) під час використання інструмента «Олівець» або «Пензель» дозволяє перейти до інструмента «Згладжування».

**Замкнути контури, якщо краї перебувають у межах: пікс.** - Курсор замикання контуру відображається, коли кінцеві точки намальованого контуру розташовано на близькій відстані та в межах установленого діапазону пікселів одна від одної. Після відпускання кнопки миші такий контур замикається автоматично. За допомогою цього параметра ви можете встановити певну кількість пікселів.

**Редагувати виділені контури** - Визначає можливість змінювати або сполучати вибраний контур, знаходячись на певній відстані від нього (визначається наступним параметром).

**Не більше: пікселів** - Визначає, як близько ваша миша чи стилус повинні бути до наявного контуру для того, щоб редагувати його інструментом «Олівець».

Цей параметр доступний лише за вибраного параметра «Редагувати виділені контури».

## 5. Текс

### 5.1. Шрифти та їх класифікація

В наш час дуже важливим для візуального сприйняття текстової інформації є типографіка. Типографіка - графічне оформлення печатного тексту за допомогою набору і верстки з використанням норм і правил, специфічних для цієї мови. Основний аспект типографіки - робота зі шрифтами.

Шрифт - це сукупність букв, цифр та знаків певного стилю, розміру (кегля). Шрифтовим файлом є фізичне втілення або опис гарнітури шрифту: в комп'ютерних кодах, на фотографічній плівці або металі.

Гарнітура - це набір із одного або декількох шрифтів в одному або кількох розмірах та накресленнях, що імітують стильову єдність малюнка і складаються з певного набору друкарських знаків. Гарнітура зазвичай містить алфавітно-цифрові і пунктуаційні знаки та спеціальні символи. Також існують гарнітури, що повністю складаються з неалфавітних символів - наприклад, що містять математичні або картографічні знаки.

Термін «гарнітура» часто змішують з терміном «шрифт», значення цих слів були більш помітні до появи настільних видавничих систем. Різниця між термінами полягає в тому, що шрифт визначає властивості конкретного члена сімейства шрифтів, наприклад, напівжирний або курсивне накреслення, в той час як гарнітура визначає узгоджений стиль сімейства шрифтів. Відповідність змісту - один з головних принципів типографіки. Вона повинна повністю підкорятися тексту, його змістом і призначенням, полегшувати, а не ускладнювати сприйняття.

Слід пам'ятайте: не текст для типографіки, а типографіка для тексту! Тому важливо брати до уваги психологічне сприйняття шрифтів, знати їх основні класифікації, закони використання, поєднання з іншими шрифтами, зображеннями, врешті-решт, з типом паперу, якщо ми робимо друковане видання.

З точки зору конструкції розрізняють три групи шрифтів :

1. Serif - Антикли (шрифти з засічками)
2. Sans Serif - Гротески (рублені, без засічок)
3. Slab - Брускові

На Рис.17 показана найпростіша анатомія шрифту. В рядках звичайного тексту всі букви розташовуються на невидимій лінії – лінії шрифту.

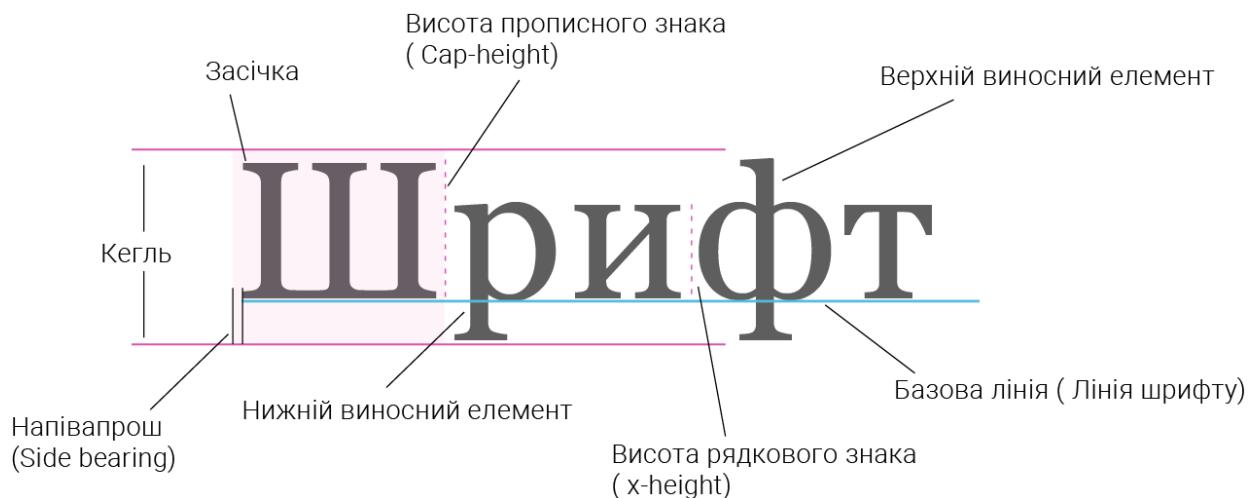


Рис.17. Анатомія шрифту

До **верхніх виносних елементів** відносяться штрихи рядкових літер, що виступають за середню лінію. **Нижні виносні елементи** – це елементи літер, які опускаються нижче лінії шрифту. Розмір виносних елементів звичайно стримується розміром майданчика кегля, на якому вони розміщаються. Основну читабельність шрифту задають саме ці елементи. [3]

**Кегельний майданчик** - це площа, яку займають усі елементи літер у сукупності. Кожному символу виділяється ділянка, пропорційна його ширині. Ширина символу у різних шрифтах одного розміру може істотно змінюватись. Відповідно до ширини літер розрізняють **пропорційні** і **моноширинні** шрифти.

**Інтерлінняж** - відстань між базовими лініями сусідніх рядків, яка вимірюється у пунктах і складається із кегля шрифту і відстані між рядками.

**Напівапрош** - відстань від крайньої точки контуру букви до кордону кегельного майданчика. Для кожної літери знаходять такі напівапроши, щоб вийшов рівний, ритмічний набір. Але навіть точно підібрані напівапроши не дають позитивного результату в усіх випадках. Деякі пари букв вимагають вирівнювання -



кернінга. В цьому випадку створюються **кернігові пари** (Рис. 18).

Рис.18. Кернінг

По суті, *кернінг* - це вибоче виправлення напівапрошів, які не впоралися зі своїм завданням. Необхідність обов'язкових поправок - щоб відстань між деякими парами не виглядала дивною.

Для того щоб гармонійно підібрати пару зі шрифтів, потрібно провести аналіз шрифту:

- Будова
- Контраст
- Динаміка
- Вага
- Ширина
- Апертура
- Висота рядових
- Баланс прописних
- Конструкція деталей

У більшості випадків для підбору хорошої пари спираються на перших п'ять пунктів, і якщо три з них сходяться, то ця пара має право на життя =)

*Контраст* - це різниця товщини основних і додаткових штрихів, контраст відповідає за читабельність шрифту, для порівняння контрасту можна опиратися на вигляд літери "О" (для прикладу **О** - з контрастом, ширина лінії з боку є іншою ніж в горі) .

*Динаміка* - малюнок букв шрифту може бути статичним або динамічним. Чим більше горизонтальних і вертикальних ліній в рядку, тим більше статичним є шрифт. А лінії і дуги, що не збігаються з горизонталлю рядки, надають шрифту динамічності. На динаміку впливають: нахил осі напливу у букв *o, e, c, p, b, d*, нахил поперечини *e*, закінчення штрихів букв *s, c, a* і форма зарубок. Всі ці елементи можуть бути прямими, діагональними або вигнутими.

*Вага (насиченість)* - це жирність (Weight) шрифту яку ми помічаємо в першу чергу і в будь-яких умовах. Основні типи насиченості: світлий, нормальній і жирний. Нормальні накреслення (Regular) емоційно нейтральні, на них найменше впливають оптичні спотворення або погана різкість.

*Ширина* - (Width) як і насиченість, різиться швидко і сильно впливає на читабельності і помітність написи. Ширину шрифту можна порівнювати за літерою "П" як кількістю простору між основними штрихами. [4,5]

## 5.2. Робота з текстом в Adobe Illustrator

Текст від точки – це горизонтальний або вертикальний рядок тексту, що починається там, де ви клацнули мишею, і продовжується в міру введення символів. Кожен рядок тексту є незалежним – він розширюється або звужується,

коли ви редагуєте його, але не переноситься на новий рядок. Введення тексту в такий спосіб є зручним для додавання кількох слів до ілюстрації.

Для додавання оберіть інструмент «Текст»  або інструмент «Вертикальний текст»  . Вказівник набуде вигляду «I» з пунктирним квадратиком. Невеличка горизонтальна риска внизу І-вказівника позначає позицію **базової лінії**, на яку лягатиме текст. (За потребою) Встановіть параметри форматування тексту на панелі керування, панелі «Символ» або «Абзац». Клацніть, де бажаєте почати рядок тексту. Введіть текст. Натисніть Enter або Return, щоб розпочати новий рядок тексту в межах того самого текстового об'єкта. Коли ви закінчите вводити текст, клацніть на інструменті «Виділення»  , аби виділити текстовий об'єкт. Можна просто клацнути на тексті, утримуючи Ctrl .

Adobe Illustrator надає можливість розміщувати текст в області. Текст в області (також – *тип абзацу*) використовує межі об'єкта для керування потоком символів, як по горизонталі, так і по вертикалі. Коли текст досягає межі, він автоматично переноситься, щоб уміститися на визначеній ділянці. Вводити текст в такий спосіб зручно, якщо вам потрібно створити більше одного абзацу, наприклад для брошури.

Для розміщення тексту у область визначте обмежувальну ділянку (Рис.19), візьміть інструмент «Текст»  або «Вертикальний текст»  і перетягніть по діагоналі, щоб визначити прямокутну обмежувальну ділянку. Намалюйте об'єкт, який ви бажаєте застосовувати як обмежувальну ділянку. (Неважливо, чи має цей об'єкт атрибути обведення або заливки, бо Illustrator автоматично вилучає їх). Візьміть інструмент «Текст»  , «Вертикальний текст»  , «Текст в області»  або «Вертикальний текст в області»  і клацніть будь-де на контурі об'єкта.

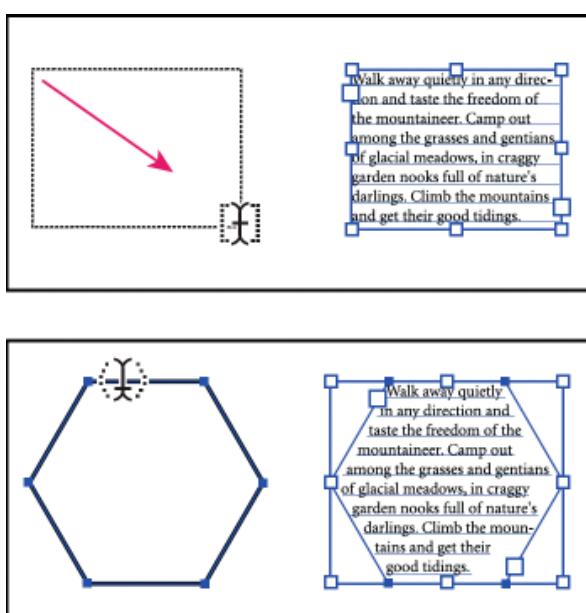


Рис.19. Створення текстової ділянки шляхом перетягування

За потребою) Встановіть параметри форматування тексту на панелі керування, панелі «Символ» або «Абзац». Ви можете змінити розмір області тексту або розширити контур, щоб нормально відобразити весь текст. Також ви можете перенести частину тексту до іншого об'єкта.

Щоб надати зміни для форматування тексту, чи окремих символів у документі - слід скористатись панеллю **«Символ»** (Вікно > Текст > Символ). Завдяки цим параметрам можна налаштувати текст до найменших дрібниць (Рис. 20).

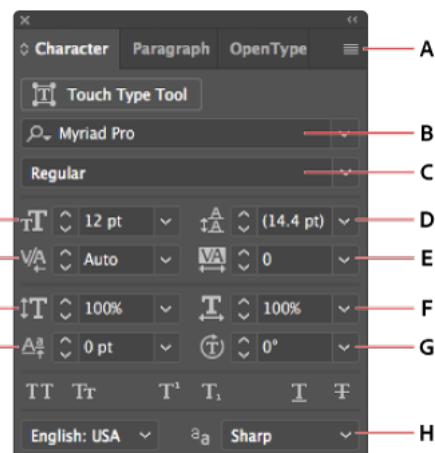


Рис.20. Панель символ.

Де **A.** Меню панелі - можна відкоригувати панель під свої потреби, назначити текст як індекс чи підіндекс, та ін.; **B.** Шрифт - вибрати шрифт; **C.** Накреслення - обрати тип шрифта (Regular, Bold, Black...); **D.** Інтерлінняж - встановити міжрядкову відстань; **E.** Трекінг - відстань між символами; **F.** Горизонтальне масштабування; **G.** Поворот символів; **H.** Метод згладжування; **I.** Розмір шрифту; **J.** Кернінг; **K.** Вертикальне масштабування; **L.** Зсув від опорної лінії.

## 6. Імітація об'єму. Покрокова інструкція

Adobe Illustrator надає багато можливостей для імітації об'єму та 3d ефектів. Розглянемо один із найпростіших, з використанням додаткових властивостей (ефектів) об'єкта.

Як основу малюнка я створюю літери, за допомогою інструмента текст, та перетворюю їх в обрис (це можна зробити натиснувши правою клавішею на текстовий об'єкт та в контекстному меню обрати “Створити обрис” ), це допоможе редагувати літери за допомогою кривих. Надаємо обведення для літер. Заливку залишаємо порожньою. (Рис. 21)

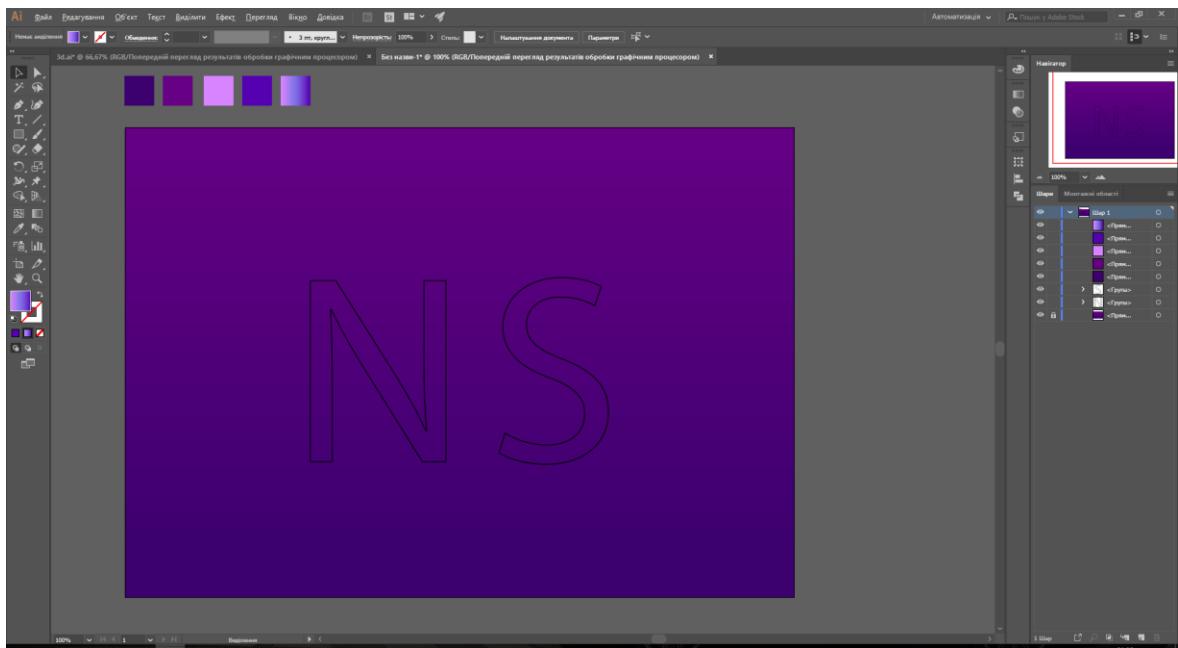


Рис.21. Текст в кривих з обведенням

За допомогою інструменту “Ножиці” розділяємо контури літер, так щоб залишилось лише по одній описовій кривій. Та інструментом “Перо” додаємо мінімальні поправки отриманим контурам (Рис. 22).

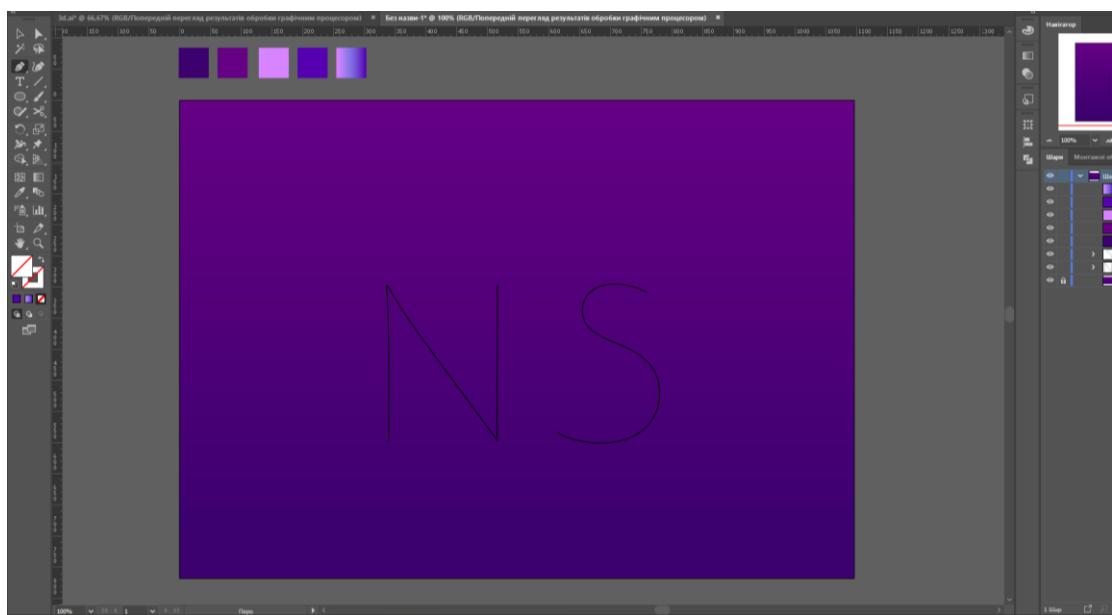


Рис.22. Контури літер після використання інструмента “Ножиці”

Збоку створюємо коло з використанням інструмента “Еліпс” (Коло не повинен бути надто великим оскільки він буде обтікати кругом літер), та надаємо йому градієнт. Створюємо копію еліпса нижче під ним (Рис. 23).

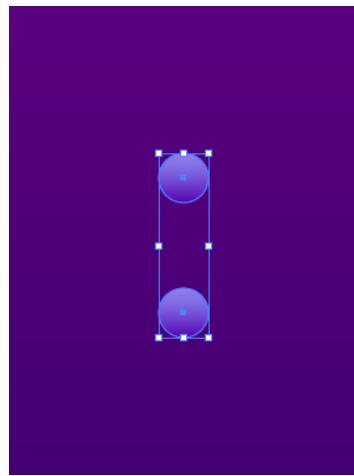


Рис.23. Створені еліпси

На панелі програми обираємо меню **Об'єкт** (кола залишаються виділеними), в меню **Об'єкт** знаходимо пункт **Перехід > Зробити** (або Alt+Ctrl+B). Якщо все виконано правильно то ми отримуємо ще декілька кіл між нашими двома.

Наступним кроком (залишаючи всі кола виділеними ) обираємо інструмент “Перехід” (подвійним кліком на іконку на панелі Інструментів), та в діалоговому вікні вказуємо параметр Задана відстань (в спадному меню), та кількість пікселів яка нам потрібна (Рис. 24).

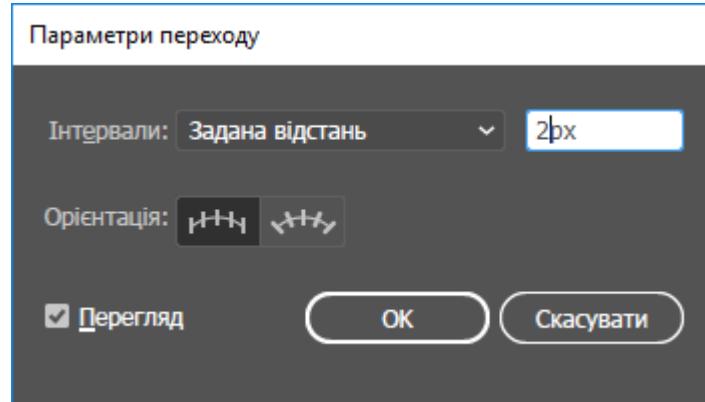


Рис.24. Діалогове вікно параметрів переходу

Отже, кола утворюють складну фігуру з плавним перетіканням градієнту. Тепер нам потрібно надати даного ефекту нашим літерам.

Виділяємо фігуру яку ми отримали після всіх маніпуляцій з колом, та контур літери. На панелі програми обираємо меню **Об'єкт > Перехід > Замінити напрямну**. Якщо з певних причин даний пункт у меню є недоступним, то потрібно повернутися до кривої, яку ми отримали з лінії та перевірити чи немає вона згрупованих об'єктів та складених контурів (Це можна побачити натиснувши на криву правою клавішою, та у контекстному меню буде вказаний один із пунктів **Розгрупувати / Звільнити складений контур**).

Якщо все пройшло вдало ви отримаєте об'єкт з ефектом об'єму (Рис. 25).

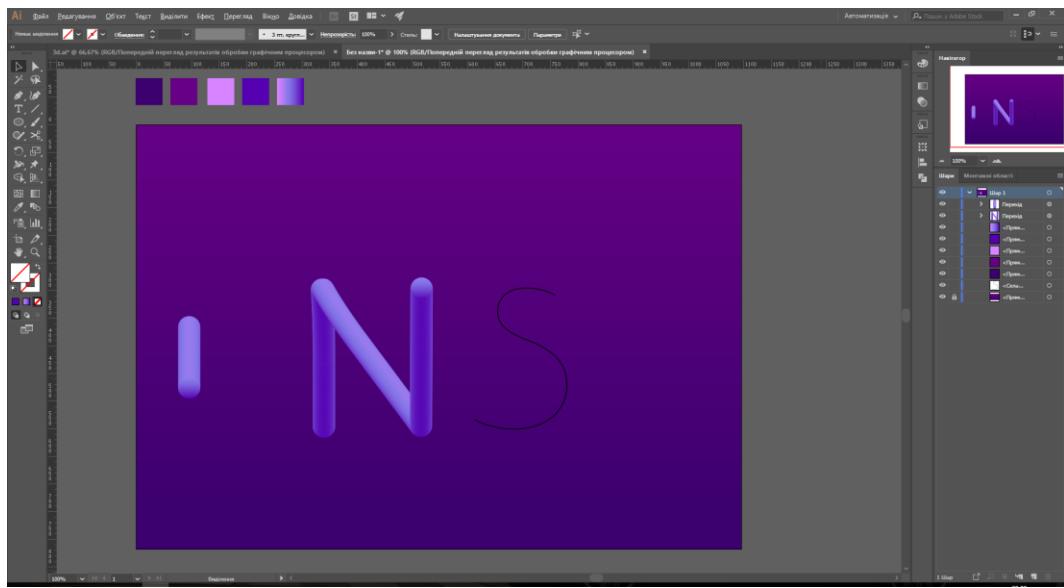


Рис.25. Об'єкт з ефектом 3d об'єму

Для добавлення наступних ефектів об'єкту вам буде потрібно панель Вигляд (Вікно > Вигляд). Виділіть об'єкт який ви отримали до цього та на панелі Вигляд натисніть на іконку “Додати новий ефект” (Рис. 26).

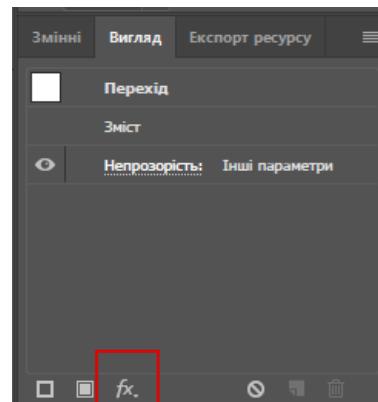


Рис.26. Панель Вигляд, меню додавання нового ефекту

Для добавлення наступних ефектів об'єкту вам буде потрібно панель Вигляд (Вікно > Вигляд). Виділіть об'єкт який ми отримали раніше, натисніть Вікно > Вигляд > Додати новий ефект > Деформувати та Трансформувати > Огрубіння та налаштуйте параметри як вам до вподоби.

Тепер ви можете більше проекспериментувати з іншими фігурами, та добавити нових елементів в композицію.

Отриманий результат показаний на Рис. 27.

Кафедра програмного забезпечення

**МЕТОДИЧНІ ВКАЗІВКИ ДО ВИКОНАННЯ  
ЛАБОРАТОРНОЇ РОБОТИ №2  
з дисципліни «Комп’ютерна графіка»**

**для студентів першого рівня вищої освіти спеціальності  
121 «Інженерія програмного забезпечення»**

Укладачі  
асистент Самбір Н.Б.  
доцент Левус Є.В.

Львів-2020

**ТЕМА.** Створення Wireframes простих систем.

**МЕТА.** Вивчити методи та інструменти створення структури та каркасу дизайну інтерфейсу.

## ТЕОРЕТИЧНІ ВІДОМОСТІ

У галузі UX wireframes – це широко прийнята методика, яка дозволяє дизайнеру та клієнту спільно працювати при визначені змісту та функціональності екранів. Wireflow робить UX ще ефективнішим, поєднуючи wireframes у блок-схему, яка представляє екрани, що беруть участь у конкретному сценарії.

Місце wireframes у різних процесах:

*Sketch > Wireframe > UI > Code*

*Sketch > Wireframe > Hi-Def > UI > Code*

*Sketch > Code*

*Wireframe > Interactive Prototype > UI > Code*

Wireframe (каркас) – це простий наочний посібник, який представляє скелетну структуру веб-сайту, системи чи іншого цифрового продукту. Wireframes в деталях показують, яка інформація, контент і елементи управління повинні виводитися на кожній сторінці інтерфейсу будь-якої системи, створюють правильні взаємодії з інтерфейсом та сприяють покращенню користувачького досвіду.

Якщо користувач не може зрозуміти, що робити на чорно-білому wireframe, то колір теж навряд чи допоможе. Кнопка повинна бути помітна навіть без яскравого забарвлення.

Оскільки каркасні конструкції настільки прості, люди можуть легше зосередитись на функціональності та користувачькому досвіді, а не на тому, щоб зациклюватися на кольорах та інших естетичних елементах.

Wireframes повинні ідентифікувати:

- *Основні групи вмісту. Що?*
- *Інформаційну структуру. Де?*

- *Опис взаємодії користувача з інтерфейсом і його приблизну візуалізацію.*

## **Як?**

Тобто Wireframes, іншими словами, забезпечує розробку таких складових:

- *Інформаційна архітектура*: упорядковує вміст та візуальні компоненти, щоб забезпечити логічний та приемний досвід користувача.
- *Структура та навігація*: показує послідовну структуру та зв'язки елементів навігації, щоб користувачі могли вільно пересуватися по системі.
- *Макет*: включає базові візуальні елементи інтерфейсу і цим полегшує початок роботи з візуальним дизайном.

## **Що ми отримуємо від Wireframes**

- 1) Фокус на цілях. Не забирають увагу на другорядні (на цьому етапі) речі: кольори, шрифти тощо.
- 2) Попередження помилок у дизайн-процесі та девелопменті.
- 3) Швидкість створення — не до порівняння з розробкою повноцінного UI.
- 4) Задання надійної основи для дизайну.

До обговорення та розробки можуть бути залучені всі: стейкхолдери, розробники, бізнес-аналітики, дизайнери та ін.

## **Типи Wireframes**

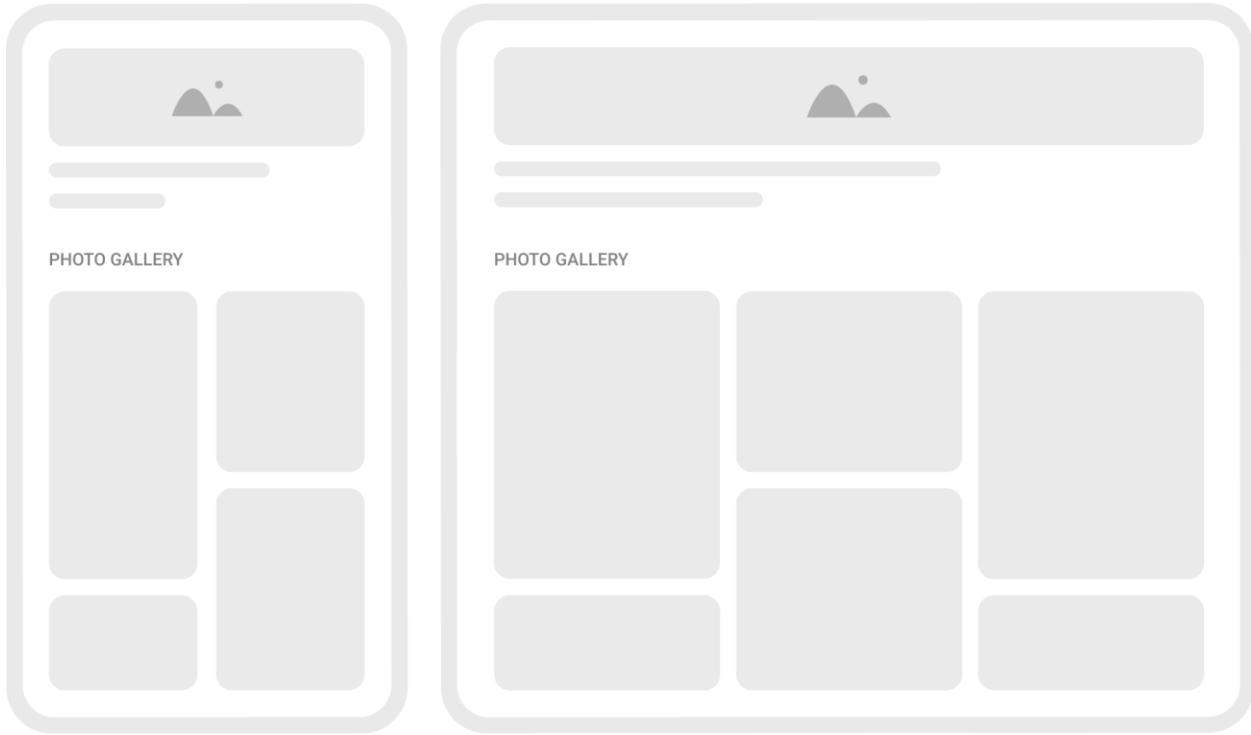
У традиційному процесі проектування до цифрових Wireframes ми б мали прийти після мальованих ескізів і безпосередньо перед мокапами чи прототипами високої точності. Але інколи, через низький бюджет, тісні часові рамки чи простоту проекту процес може змінюватися.

Існує три рівні каркасного зв'язку: низький (**low fidelity**), середній (**medium fidelity**), та високий (**high fidelity**).

### **- Low fidelity wireframes**

Низька точність - це базовий тип каркасів, який є настільки спрощеним, що навіть паперу та ручки вистачить для представлення вашої ідеї. Wireframes низької точності виконуються у відтінках сірого з акцентом на макет та взаємодію

високого рівня. Елементи та вміст інтерфейсу представлені базовими фігурами, такими як квадрати, трикутники, кола та лінії.



*Рис.1. Приклад Low fidelity wireframe*

#### **- Medium fidelity wireframes**

Wireframes середньої точності - є більш вдосконаленим і наповненим відображенням низькорівневих каркасів. Вони включають візуальні маркери та текстовий контент, підписи для кнопок та імпутів, виглядають практично як проміжний продукт у відтінках сірого, але не задають стиль. На цьому кроці дизайнери залишають собі невеликий «простір для маневру», де в разі потреби можна буде перебудувати контент в межах певного блоку/картки, щоб він виглядав органічніше в фінальному інтерфейсі (Рис. 2).

#### **- High fidelity wireframes**

Wireframes високої точності - фіксують зовнішній вигляд продукту на останніх стадіях процесу проектування. Вони включають фактичний вміст, шрифти, кольори, розміри зображення та елементи брендингу.

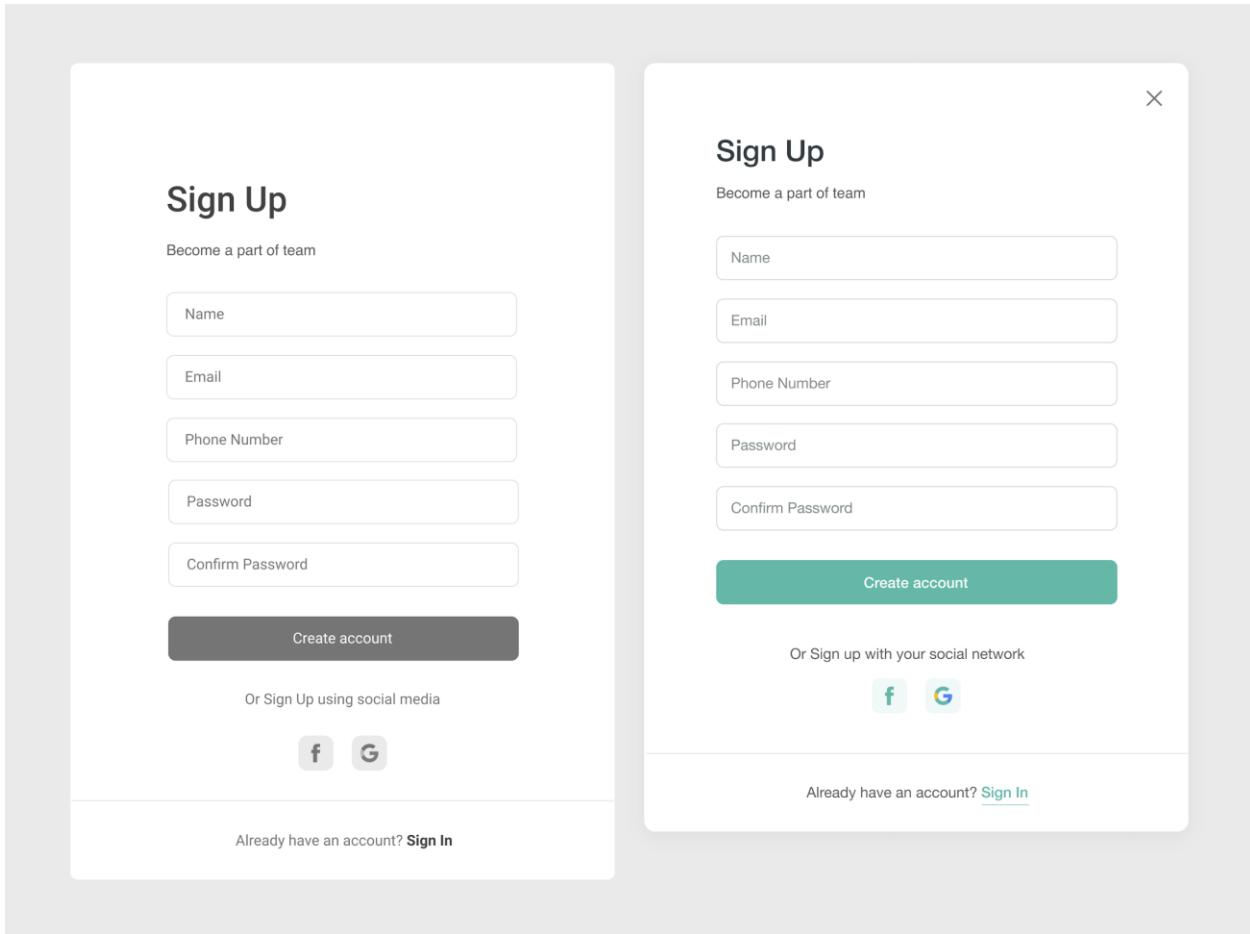


Рис.2. Приклад Medium fidelity та High fidelity Wireframes

## Як пов'язати Wireframes

Wireflow - це представлення потоку екрана шляхом складання набору пов'язаних wireframes, які послідовно відображаються в потоці. Використання рішення (форми) в wireflow дає можливість представити декілька навігаційних шляхів в одному потоці.

Wireflows використовуються в основному для додатків, де існують складні робочі процеси та є різноманіття взаємодії користувачів. Дизайнери UX створюють wireflows, щоб продемонструвати кроки, які користувач здійснить для виконання певного завдання. Використання wireflows допомагає повідомляти про функціональність зацікавленим особам та членам команди та заохочує генерувати нові дизайнерські ідеї.

Для кращого розуміння, як буде працювати система та визначення слабких сторін треба перейти до етапу розробки User Flow. User Flow – перехід користувачів від одного сценарію взаємодії з інтерфейсом до іншого, очікуваний

алгоритм дій користувача. User Flow – це інструмент, який UX дизайнери використовують при розробці інтерфейсів. У центрі уваги користувач. Проектування сценаріїв взаємодії користувача (User Flow) допомагають на ранніх стадіях визначити, як користувач буде взаємодіяти з інтерфейсом сайту чи додатку, виявити слабкі сторони і вчасно виправити їх.

User Flow показує, які дії здійснює користувач користуючись системою.

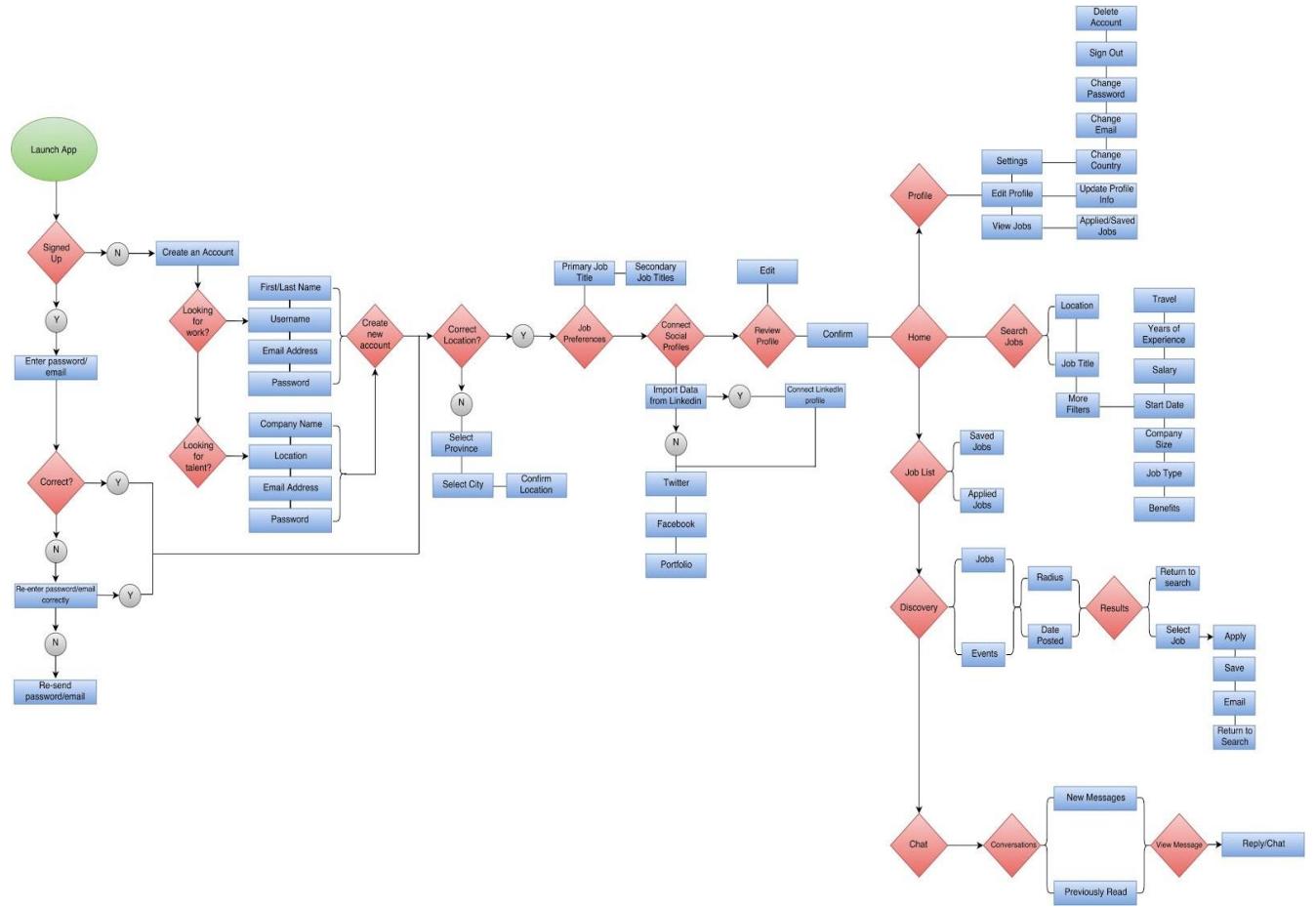


Рис 3. Приклад User Flow.

## Інструментарій

- Олівець, папір
- Moqups, Balsamiq, Axure, Illustrator, Figma, Sketch, Notepad, інші — не має великого значення.

Під час вибору інструментів для побудови wireframe, опирайтесь на швидкість, маштабованість та легкість переходу з wireframe до ui.

Дизайн - це живий процес. Тому різні дизайнери по-різному ставляться до wireframe і до того, як їх потім перетворювати в макети або код. Ви повинні знайти те, що підкреслить ваші сильні сторони і буде більш зручним для вас.

## **ЕТАПИ СТВОРЕННЯ MEDIUM FIDELITY WIREFRAMES**

### **Етап 1. Дослідження**

Зробіть ретельне дослідження. Не починайте процес каркасного проектування до того, як не окреслили цілі проекту, визначили цільову аудиторію та зрозуміли проблеми, які вирішуються продуктом. У таких випадках рекомендується проводити евристичні оцінки, перш ніж вступати в процес каркасного проектування.

### **Етап 2. Скетч**

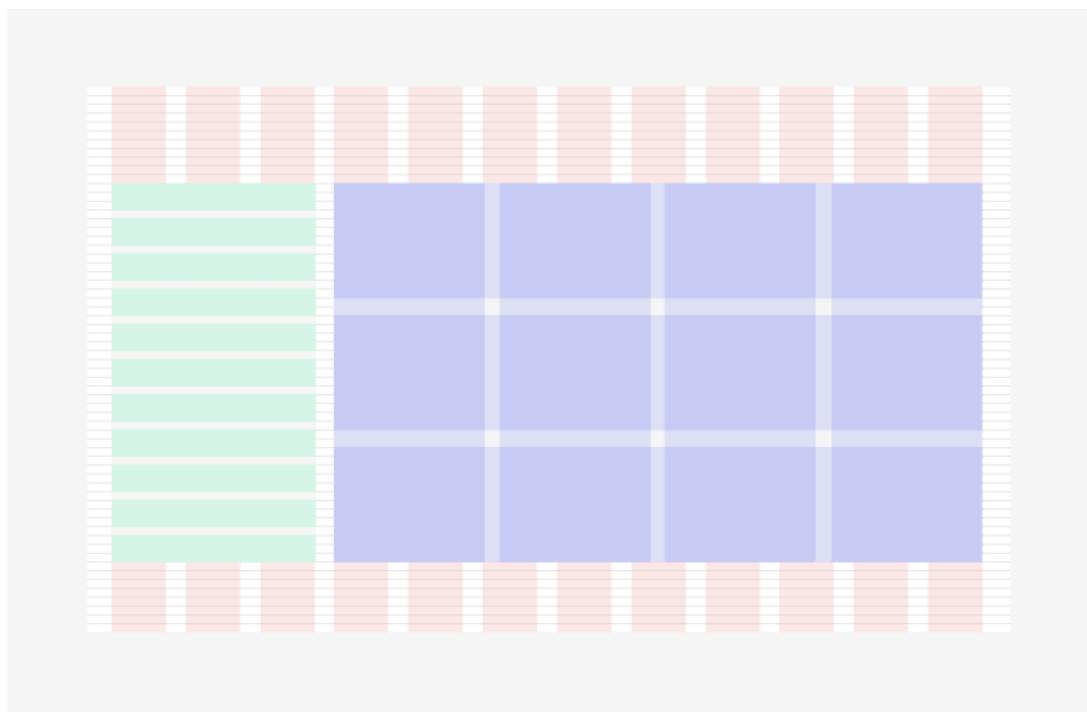
Мета ескізу - створити структуру дизайну сторінки. Прибережіть тонкі деталі на потім, на даному етапі обдумайте - який функціонал і тип контенту ви будете розміщувати на сторінці.

### **Етап 3. Сітка**

Перед налаштуванням сітки, задайте розміри сторінки (артборду/фрейму). Відштовхуючись від того чи ви працюєте з десктопом чи мобільним інтерфейсом. Далі перейдіть до налаштування сітки та контейнерів.

Сітка - це структуроване і просте розташування елементів.

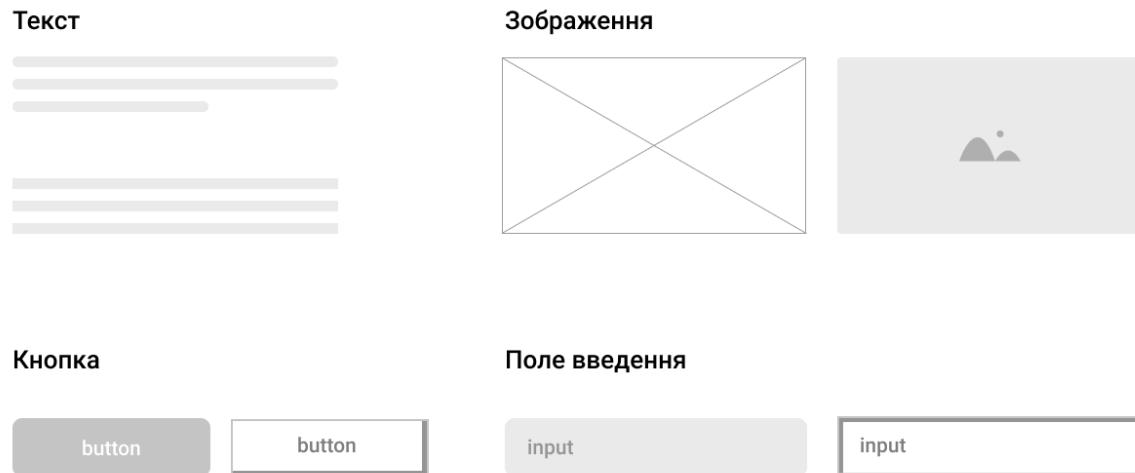
Існує багато теорій про систему сіток, але на даному етапі вивчення ви можете розглянути Bootstrap сітки, які гарно адаптуються під різні розширення або створити власну модульну сітку, яка дозволить зробити інтерфейс більш гнучким.



*Рис.4. Сітки*

#### **Етап 4. Планування контенту**

На даному етапі - проходимо ітерацію низькорівневих Wireframes та створюємо розмітку компоновки (layout) прямокутниками.



*Рис.5. Подання елементів на Wireframe*

## Етап 5. Побудова інформаційної ієрархії

Коли вас влаштовує розташування блоків, приступайте до поступового наповнення їх текстом, щоб отримати уявлення, чи добре структурована інформація.

Використовуйте різний розмір шрифту, щоб почати виділяти різні види інформації.

На даному етапі починаємо працювати з функціоналом, текстом, кнопками, інпутами та ін.

Іноді, додавши більше деталей, ви можете побачити, що дане розташування елементів не зовсім підходить. В цьому і полягає мета wireframe: зробити якомога більше ітерацій, щоб знайти найкращі способи подання інформації, яку необхідно донести.

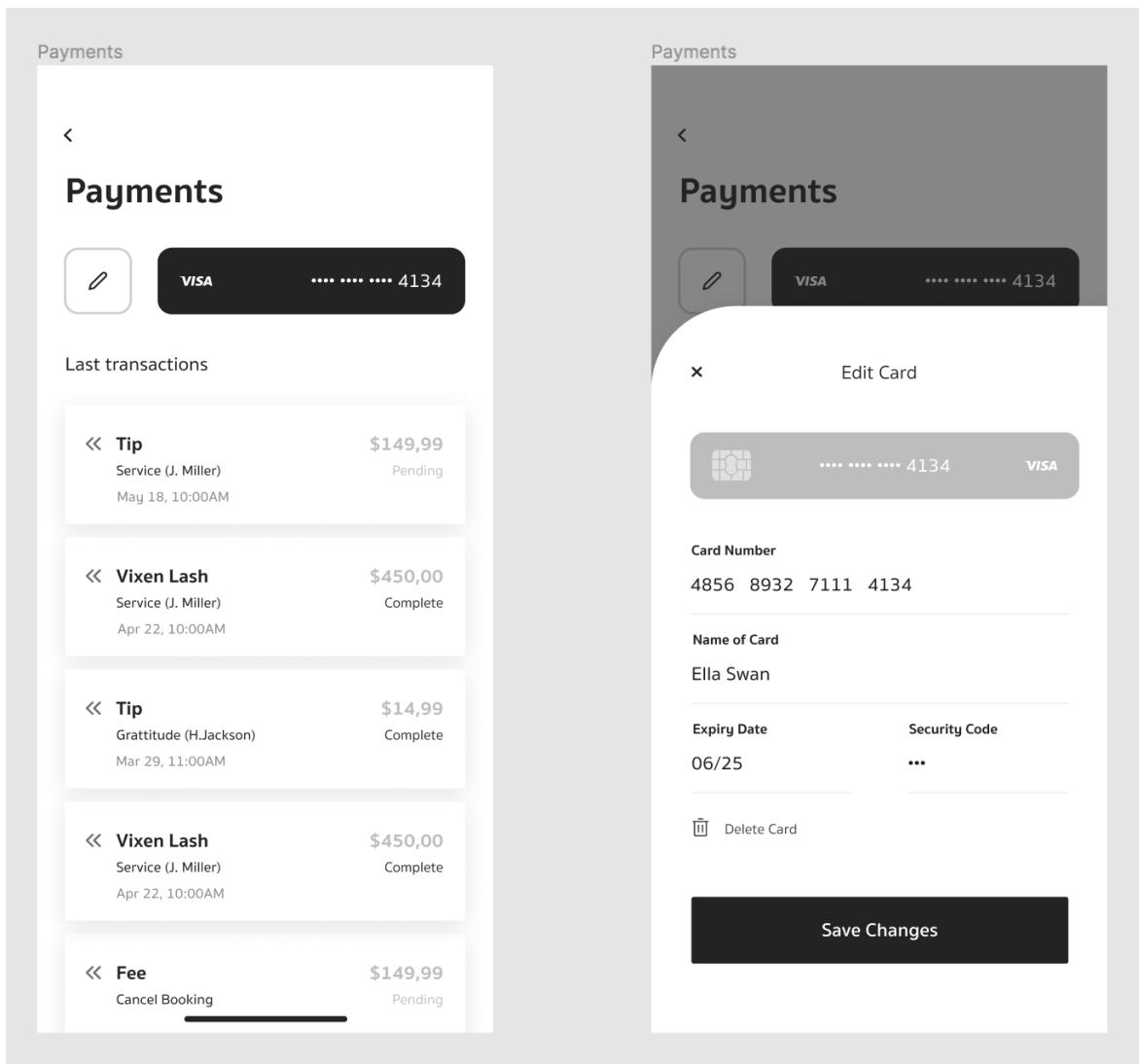


Рис.6. Приклад Wireframes

## Етап 6. Wireflow\*

Wireflow - це поєднання user flow та wireframes. Він потрібен для того щоб розуміти загальну архітектуру проекту, послідовність та розгалуження сторінок.

Впорядкуйте свої Wireframes - та вкажіть стрілками переходи між сторінками. Старайтесь, щоб стрілки не перехрещувалися, бо тоді їх важче читати.

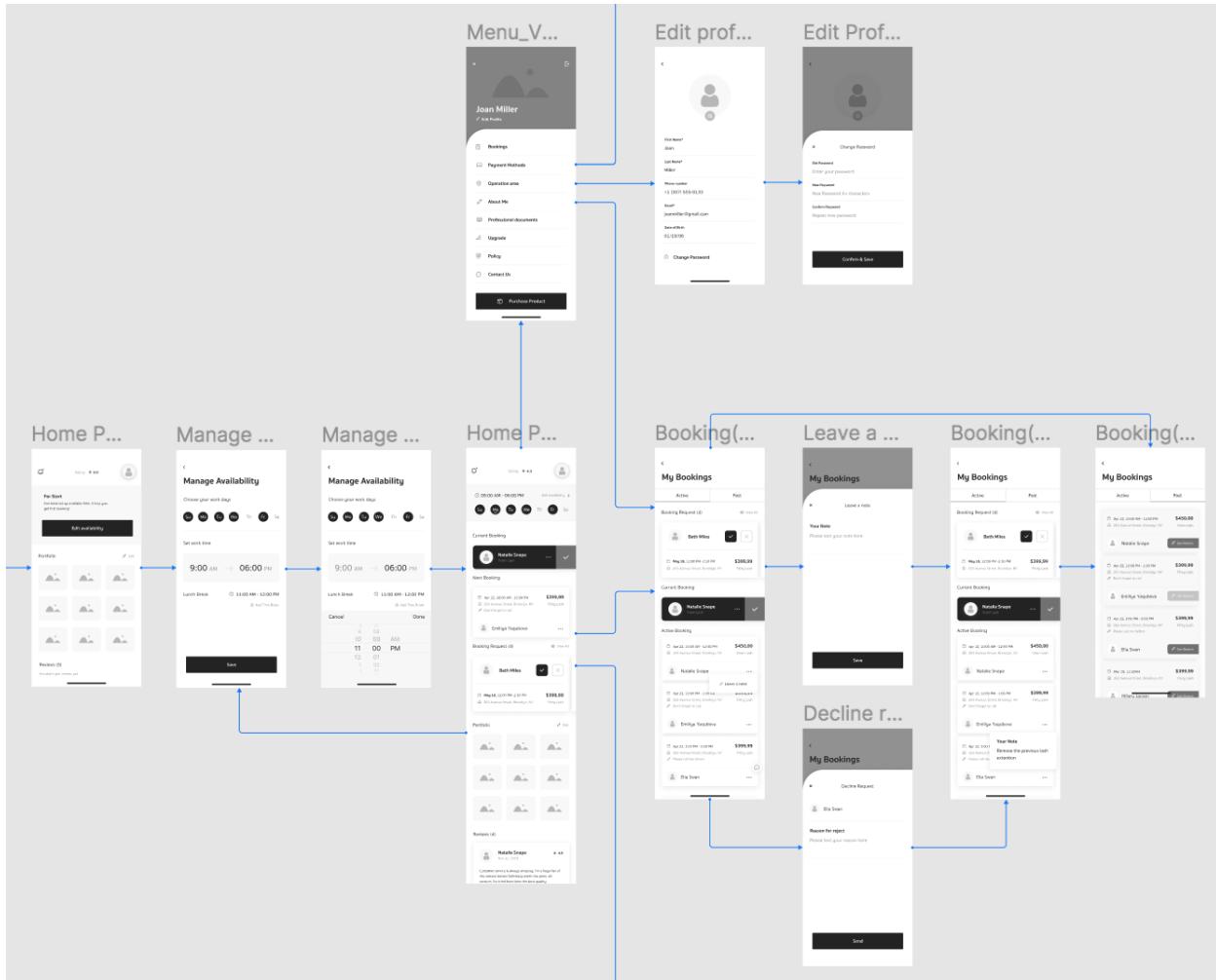


Рис.7. Приклад Wireflow

\* етап, можливо, буде пропущено в реальних проектах. Але не в цій лабораторній роботі)..

## Висновок:

Wireframes - є вагомою ланкою у проектуванні якісного інтерфейсу. Побудова відбувається у відтінках сірого, тому слід уникати всього що може бути розцінено, як UI-дизайн.

Слід пам'ятати що продукт розробляється для людей, конкретних користувачів. Тому постійно згадуйте про свого кінцевого користувача. Він повинен бути задоволеним.

Використовуйте однакові елементи інтерфейсу. Один і той же процес не повинен здійснюватися різними шляхами. Пам'ятайте про евристики.

Постійно перевіряйте цілісність результату.

**Завдання 1** (одноосібне виконання).

Ознайомтесь з основними вимогами до програми згідно варіанту.

Виконайте послідовно описані вище етапи 1-6 для побудови Wireframes і Wireflow для всіх основних екранів (сторінок) програми.

**Завдання 2** (командне виконання).

Ознайомтесь з основними вимогами до програми згідно варіанту.

Виконайте послідовно описані вище етапи 1-6 для побудови Wireframes і Wireflow для всіх основних та проміжних екранів (сторінок) програми. Проміжними екранами (сторінками) є, зокрема, коли користувач потрапляє на сторінку і інформацію потрібно ввести, коли інформація вже додана, також відображення інформації у випадаючих списках, підказках, модальних вікнах та ін.

## **Список питань для контролю знань**

1. Що таке Wireframe?
2. Що таке Wireflow?
3. Що таке Userflow?
4. Які є типи Wireframes? Чим вони визначаються?
5. Опишіть перший тип Wireframes.
6. Чим відрізняється перший тип Wireframes від другого?
7. Які відмінності між другим та третім типами Wireframes?
8. Назвіть етапи створення medium fidelity wireframes.
9. У чому суть першого етапу створення medium fidelity wireframes? Як ви його виконали?
10. У чому суть другого і третього етапів створення medium fidelity wireframes?
11. Які елементи використовуються на етапі планування контенту?
12. Хто може залучатися до обговорення Wireframes?
13. Які переваги від використання Wireframes як елементів UI?
14. У чому полягає етап побудови інформаційної ієрархії?
15. Як використовуються Wireframes у Wireflow?
16. Які евристики ви передбачили при виконанні завдання?
17. Яку евристику, на вашу думку, найскладніше забезпечити?
18. Які евристики стосуються роботи з помилками?
19. Що ідентифікує Wireframe?
20. Які інструменти використовують для побудови Wireframes? Який інструмент було обрано вами? Обґрунтуйте свій вибір.

## **Інформаційні ресурси**

<https://designmodo.com/wireframing-prototyping-mockuping/>

<https://help.figma.com/article/116-getting-started>

<https://www.youtube.com/channel/UCQsVmhsa4X-G3lHIUtejzLA>

<https://www.sketch.com/docs/>

<https://www.toptal.com/designers/ui/ui-styleguide-better-ux>

## Додаток

*Список питань, які рекомендується розглянути на першому етапі створення medium fidelity wireframes:*

1. Як користуватися навчальними матеріалами у програмі?
2. Які є повідомлення про помилку, попередження?
3. Як здійснюється доступ до блоків : фрактали, колірні моделі, рух?
4. Як вводяться вхідні дані фігури, для якої реалізується рух?
5. Що вводиться для побудови фракталів?
6. Як відображаються значення кольору для пікселів у різних просторах (моделях) згідно варіанту?
7. Що задається користувачем для відображення координатної площини?
8. Як виглядатиме форма з побудованим фракталом?
9. Як ініціюється рух? Чи є потреба вводити параметри для програмування афінних перетворень?
10. Як зчитуються файли зображень і як здійснюється збереження згенерованих зображень?

**Тема.** Створення UI простих систем.

**Мета.** Вивчити базові методи та інструменти створення дизайну інтерфейсу.

## Теоретичні матеріали

Робота з User interface (UI) design є об'ємною і клопіткою, оскільки включає в себе знання з багатьох галузей. Адже це не тільки про зовнішній вигляд, а й про інтерактивність та інтуїтивність системи. Нерідко помилково поняття «дизайн користувацького інтерфейсу» підмінюють графічним дизайном чи брендингом. User Interface - це складна структура, яка відповідає за створення зрозумілого та прийнятного досвіду користування, базуючись на дослідженнях та процесах, пророблених до цього зі сторони бізнес-аналізу, сервіс-аналізу, их та частково гіпотезах майбутньої маркетологічної стратегії .

Це мистецтво, яке фокусується на всіх елементах продукту, що роблять його привабливим: колір, стиль кнопок, графіка, анімація, типографія, інфографіка, віджети, взаємодія елементів і так далі. Вдале поєднання цих елементів допоможе користувачеві без додаткових зусиль орієнтуватися в готовому додатку.

Дизайн інтерфейсу стосується як графічних користувальницьких інтерфейсів так і інших форм, наприклад, голосово-керованих інтерфейсів.

Користувач повинен відчувати себе комфортно, використовуючи продукт. Це залежить не тільки від інтуїтивно зрозумілого інтерфейсу, але і від того, як цей інтерфейс виглядає і наскільки він професійно побудований. 85% користувачів припиняють використовувати додатки через поганий візуальний дизайн.

Точки доступу, де користувачі взаємодіють з дизайном інтерфейсу бувають трьох форматів:

**Графічні інтерфейси користувача (GUI)** - користувачі взаємодіють з візуальними зображеннями на цифрових панелях управління. Робочий стіл комп'ютера - це графічний інтерфейс.

**Голосові інтерфейси (VUI)** - користувачі взаємодіють з ними за допомогою голосу. Більшість розумних помічників - наприклад, Siri на iPhone та Alexa на пристроях Amazon - це VUI.

**Інтерфейси на основі жестів** - користувачі взаємодіють із просторами 3D-дизайну за допомогою фізичних рухів: наприклад, у іграх з віртуальною реальністю (VR).

Працюючи з дизайном, пам'ятайте:

- Враження про дизайн є швидкоплинними, вони вражають, є нейтральними або відштовхують користувача.
- Приємний і зручний (у сенсі простий) дизайн матиме більшу віддачу, ніж мега стильний дизайн з складними взаємодіями. Користувачі дбають про те, щоб виконувати свої завдання легко і з мінімальними зусиллями.
- Інтерфейси користувача повинні передавати цінності бренду та зміцнювати довіру користувачів.
- Розуміння потоків завдань ваших користувачів допомагають точно налаштовувати найбільш інтуїтивні інтерфейси, що забезпечують бездоганний досвід. Дуже важливо розуміти, як людський мозок реагує на певні візуальні сигнали. Наприклад, показати користувачеві що картинка, на яку вони дивляться, це ще і кнопка, на яку можна натиснути і отримати додаткову інформацію.
- Анімації, ілюстрації, та різні графічні елементи і плавні переходи створюють відчуття особливості і унікальності.
- Спілкування з своїм користувачем є важливим. Коли ваш дизайн передбачає потреби користувачів, вони можуть насолоджуватися більш персоналізованими та захоплюючими враженнями. Радуйте їх, і вони будуть до вас повернатися.

- UI прототипування, анімація та адаптивність - це ті аспекти, які забезпечують максимальний комфорт від використання продукту на будь-яких пристроях.

**Хороший дизайн - це емоційний дизайн.** Користувачі пов'язують добре почуття з компаніями та брендами, які розмовляють з ними на всіх рівнях і зберігають відчуття приємних, цілісних переживань.

**Як створити правильний UI?**

- 1) Зробіть кнопки та інші поширені елементи інтерфейсу передбачуваними (включаючи адаптивність та масштабування), щоб користувачі могли їх несвідомо використовувати скрізь. Форма повинна відповідати функції.
- 2) Робіть правильні акценти та ієрархію. Слід чітко розуміти на чому повинен зосередитись користувач, який основний фокус його уваги і куди зміститься даний фокус при певній дії.
- 3) Всі елементи повинні бути вирівняні, пам'ятайте про сітку.
- 4) Зверніть увагу на ключові функції, використовуючи колір, яскравість і контраст. Уникайте використання широкого спектру кольорів, пропрацюйте свою колірну гаму, адже колір тільки посилює дизайн а не формує його.
- 5) Створіть ієрархію шрифтів, які ви будете використовувати. Текст подається через розмір шрифту, його вагу, динаміку, розмір літер та відстань між ними. Користувачі повинні знаходити потрібну інформацію просто скануючи, без зайвих зусиль.
- 6) Мінімізуйте кількість дій для виконання завдань, зосередьтеся головній функції на сторінці. Направляйте користувачів, вказуючи бажані дії. Полегшуйте складні завдання, використовуючи прогресивне розкриття інформації.

- 7) Розташуйте елементи керування поблизу об'єктів, якими користувачі хочуть керувати. Наприклад, кнопка для подання форми повинна знаходитись поруч із формою.
- 8) Тримайте діалог між користувачем та системою. Інформуйте користувачів про відповіді чи дії системи, підтримуйте зворотній зв'язок.
- 9) Допомагайте користувачу, якщо є така можливість. Наприклад, користувачу потрібно заповнити форму, ви можете додати автозаповнення вже наявною інформацією.
- 10) Завжди надавайте наступні кроки, які користувачі можуть зробити природним чином, незалежно від їхнього контексту.

Далі розглянемо базові аспекти, які допоможуть тримати дизайн у «чистоті».

## 1. Композиція (складання та зв'язування)

Це побудова цілісного витвору, всі складові якого гармонійно узгоджуються.

Вона допомагає:

- *Контрлювати увагу користувачів.* У цьому випадку сприйняття користувача є свого роду шлях, по якому він проходить через інтерфейс і вивчає його візуальні компоненти. Користувач буде слідувати за маршрутом, який йому заздалегідь зададуть.
- *Концентрувати увагу користувача.* Люди легше сприймають структуровану інформацію, тому що докладають менше зусиль на її засвоєння.

У цифровому світі розглядають різні шаблони сприйняття інформації, найбільш поширені - це F і Z шаблони сприйняття контенту користувачами. Вони пояснюють як наші очі натреновані на пошук інформації. Пошук (у F шаблоні) починається з верхнього лівого кута, і йде по горизонталі, потім спускається до наступного рядка і робить те ж саме, поки ми не знаходимо щось цікаве .

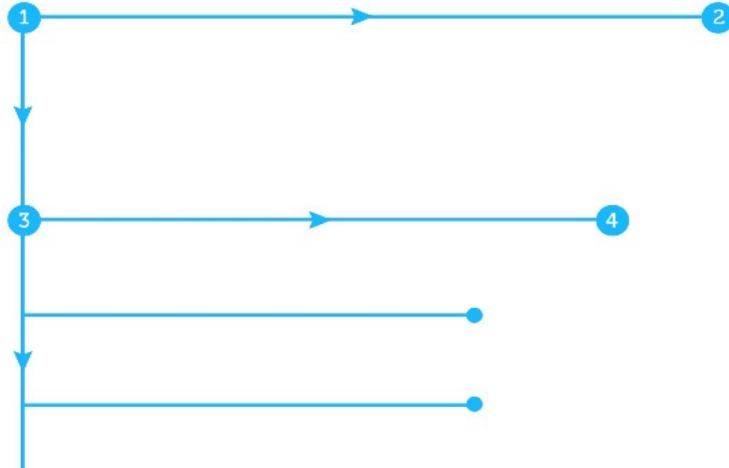


Рис.1 Схема F шаблону

Шаблон Z гарно працює для цільових сторінок, сайтів з невеликою кількістю тексту: по верхній горизонталі розташовані назви розділів сайту (меню), далі погляд ковзає по діагоналі вниз, (перетинаючи центральне поле із зображенням) і переходить до блоку інформації в нижній частині.

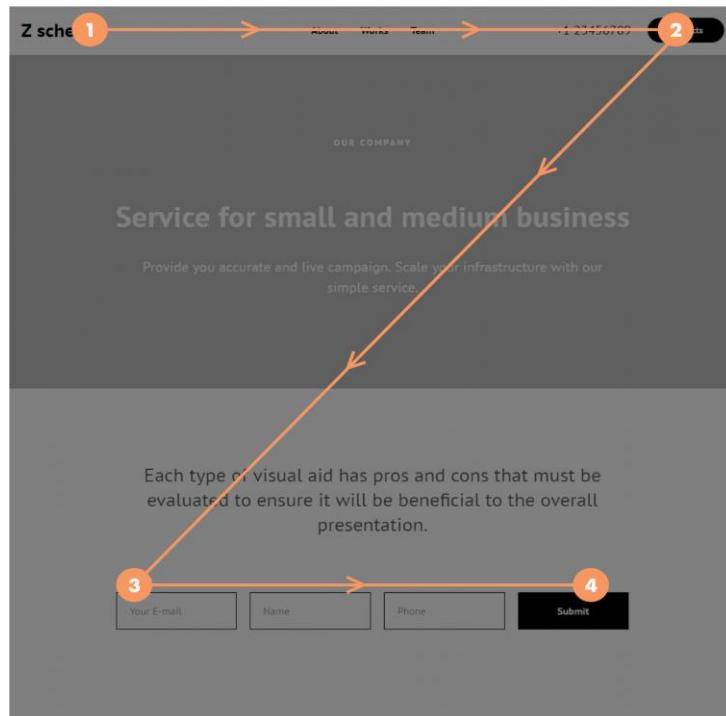


Рис.2 Схема Z шаблону

У композиції виділити елемент і за допомогою цього додати вагу до поданої ним інформації можна через:

- *Розмір*;
- *Колір*;
- *Форму*;
- *Негативний простір*.

Але не варто намагатися виділити всі елементи дизайну одночасно, щоб не створити хаос.

*Розмір*. Чим більший елемент, тим більше уваги йде до нього. Ідея ієрархії за допомогою розміру полягає в намірі дати фокус. Заголовок первого рівня більше за заголовка другого рівня і т. д.

Погляд чіпляється за великі заголовки, але також необхідно пам'ятати, що важливі елементи дизайну не повинні бути занадто великими, тому що таким чином може виникнути небажаний дисбаланс.

*Колір*. Це відмінний спосіб виділити об'єкти. У UI найяскравіший колір часто використовується для елементів, що взаємодіють з користувачем.

Існує 3 способи створення ієрархії за допомогою кольору:

- 1) Контрастність - деякі кольори здатні контрастувати з іншими, що покращує сприйняття, але не правильно заданий тон також може створити кілька типів конфліктів для людського зору, наприклад, червоний на зеленому.
- 2) Насиченість - насичені кольори більш помітні і виділяються на тлі сірих.
- 3) Яскравість - яскраві кольори виділяються поверх темних і навпаки. Гра з яскравими елементами на темному тлі створює пряму ієрархію. Це також може бути застосовано, коли у нас білий фон і діапазон темних елементів.

*Форма*. Чим складніше форма елемента інтерфейсу, тим більше його візуальний вага в порівнянні з об'єктами правильної форми. За конфігурацією ви

також можете вгадати, який елемент перед вами: поле введення, кнопка або випадаючий список.

**Якірні об'єкти.** Досягти балансу в композиції можна і за допомогою якірних об'єктів. Якірні об'єкти - це найпомітніші елементи на сторінці. Правило говорить, що будь-який якірний об'єкт повинен бути вписаний у візуальний прямокутник, і примикати до одного із кутів або у візуальний центр прямокутника. Також такий об'єкт може бути прив'язаний до однієї зі сторін прямокутника.

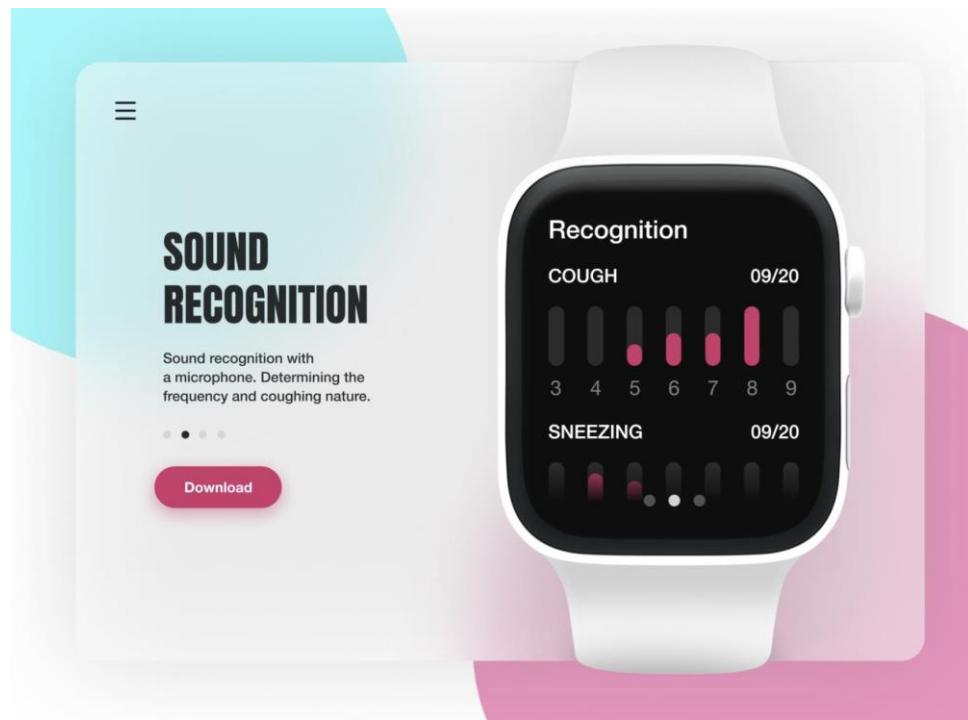


Рис.3 Приклад якірного об'єкту прив'язаного до правої сторони прямокутника



Рис.4 Приклад якірного об'єкту розміщеного в візуальному центрі прямокутника

*Негативний простір (white space).* Негативний простір або мінус-простір – це порожній простір на інтерфейсі, це простір між елементами вашої композиції.

Цей простір залишається не використаним, що робить макет більш читабельним і виразним. Користувач може зосередити увагу на краще представлених елементах.

Однак він не повинен бути буквально порожнім. Це фон, який не має когнітивної ваги для користувачів, щоб вони могли зосередити увагу на основних елементах.

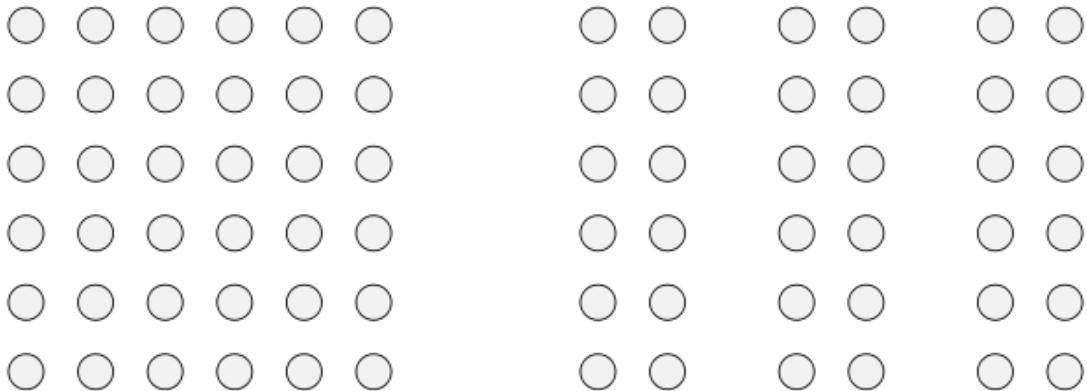
Коли справа доходить до графічного дизайну, побудова візуальної ієрархії між різними елементами інтерфейсу є ключовою. Це допомагає вказувати користувача на стратегічні точки взаємодії.

Крім того, білий простір допомагає зменшити інформаційне забруднення на сторінці, надаючи контенту більшого простору для “дихання”.

При роботі з простором між елементами треба розуміти **Принцип близькості (Гештальт)**.

Елементи дизайну, розташовані близько один до одного, сприймаються як пов'язані. Людський мозок має тенденцію класифікувати спостережувані об'єкти, тому створення таких груп зазвичай спрощує сприйняття контенту користувачем.

Якщо об'єкти розташовані далеко один від одного, це має означати, що вони не можуть бути пов'язані. Близькість будує відносини і дає інформацію, а також її впорядкованість.



*Рис. 5 Приклад близькості*

Внутрішні відстані елемента повинні бути менше зовнішніх.

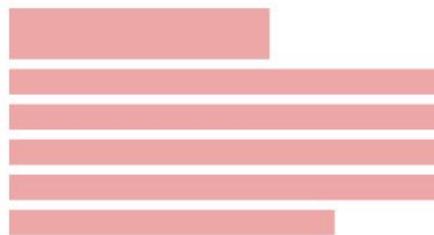
Закон близькості працює на всіх рівнях: у буквах, словах, пропозиціях, абзацах і модульних макетах. Якщо цього принципу не дотримуватися, то елементи можуть утворювати випадкові небажані зв'язки.

Розглянемо текстовий блок. Відстань між словами є внутрішнім, а відстань між рядками (інтерліньяж) - зовнішнім. Тому, згідно з правилом, збільшимо інтерліньяж:

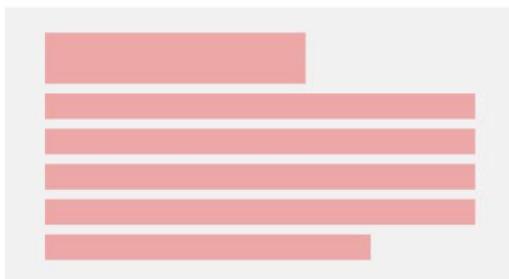


Справа інтерліньяж збільшений. Блок виглядає акуратніше і легше.

Якщо додати до тексту заголовок, то інтерліньяж заголовка - зовнішнє, повинен бути більше, ніж інтерліньяж тексту - внутрішнє. Пам'ятайте, зовнішнє завжди більше внутрішнього:



Якщо цей блок покласти на сіру плашку, то її поля гратимуть роль зовнішнього, тому вони повинні бути більші інтерліньяжу заголовка:



## 2. Колір

Колір - найскладніша область візуального дизайну. Кожен колір має візуальну вагу, і використовується, щоб встановити ієархію контенту.

Використовуючи різні відтінки кольору, можна призначати різні рівні важливості елементів.

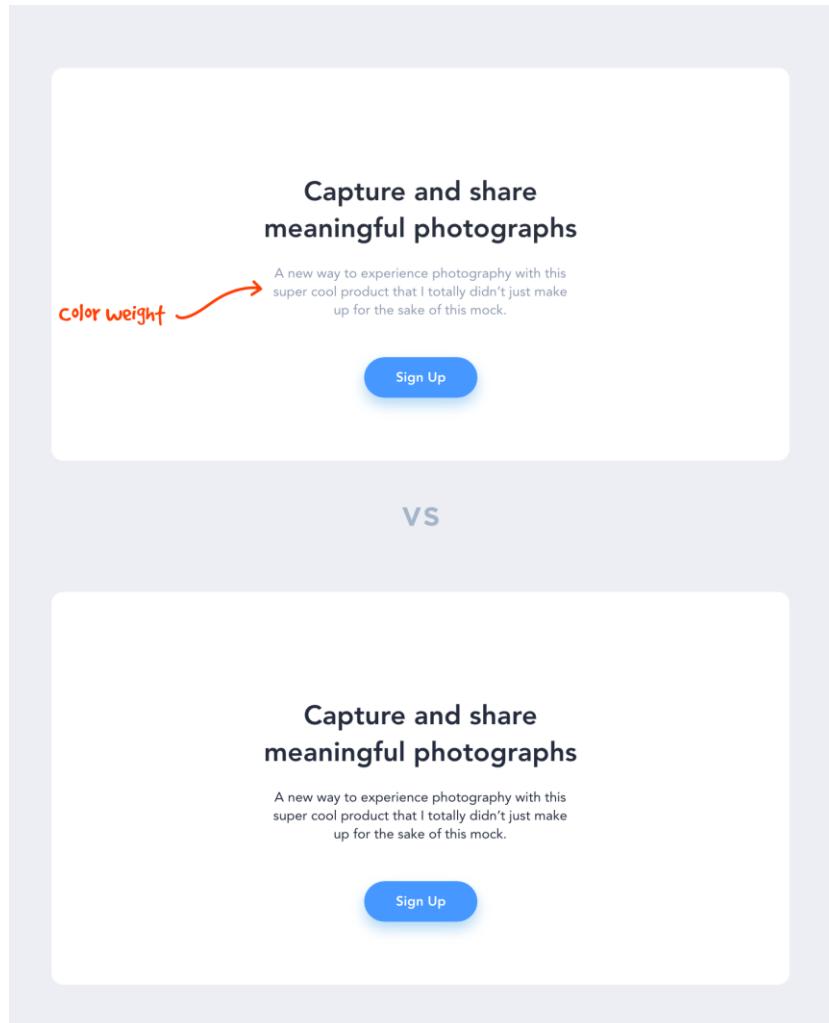


Рис. 6 Приклад ваги кольору

Правило полягає в тому, що, якщо один елемент є важливішим за інший, то він повинен мати більшу візуальну вагу. Це дозволяє користувачеві швидко переглядати сторінки і розрізняти важливу і другорядну інформацію.

Спочатку користувачі звертають увагу на інформацію, яка більшою і яскравішою, а потім вони переходят до допоміжної інформації під нею.

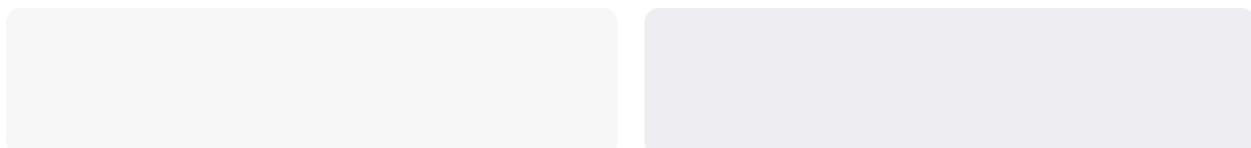
Щоб швидко скласти колірну схему можна скористатися **правилом 60-30-10**.

Пропорція  $60\% + 30\% + 10\%$  відображає баланс кольорів (домінанта, допоміжний, акценти). Ця формула працює тому, що створює відчуття балансу і дозволяє очам комфортно пересуватися з однієї точки до іншої. Крім того, цей прийом дуже простий у використанні.

Але спочатку градації сірого. Правильно структуровані сірі вайрфрейми допоможуть створити макети з правильними пропорціями і простором, і заощадити час на вибір стилю. Коли вайрфрейми вже мають кінцевий вигляд, перейдіть до вибору кольору. Протестуйте різні відтінки, якщо хочете домогтися приемного зовнішнього вигляду.

У фінальному дизайні уникайте сірих кольорів без насиченості. У реальному житті чистих сірих відтінків не існує. Те ж стосується і чорних.

Пам'ятайте, завжди потрібно додавати трохи насиченості кольору. Підсвідомо це буде сприйматися більш натурально.



*Рис. 7 Чистий сірий колір та сірий з блакитною компонентою*

*Зберігайте контраст.* Деякі кольори добре поєднуються один з одним, а деякі - зовсім ні. Є правила, що визначають взаємодії відтінків, найкраще ці правила демонструє колірний круг.

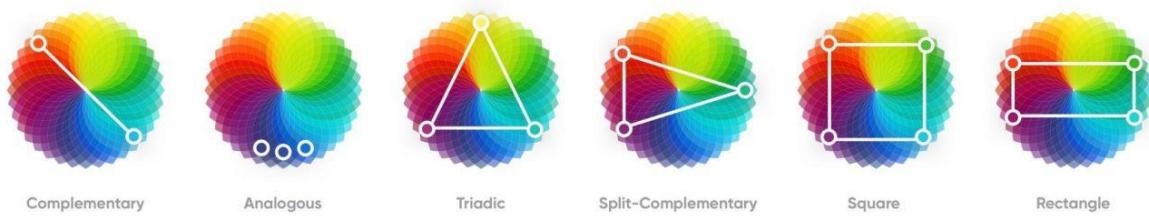


Рис. 8 Шість основних правил сполучень кольорів

*Надихайтесь.* Коли потрібно знайти приклади UI, **dribbble** - одне з кращих місць для цього. У ньому є інструмент для пошуку за кольорами. Так що, якщо потрібно провести візуальний ресерч на тему використання конкретного кольору, йдіть на [dribbble.com/colors](https://dribbble.com/colors)

Також можете скористатися одним із інструментів для вибору кольорової палітри.

Colors.co - ви можете просто вибрати один колір і згенерувати палітру, натиснувши пробіл. Colors дає можливість завантажити зображення і створити палітру з нього. Ви не обмежені одним результатом - селектор дозволяє змінювати вихідний орієнтир.

Kuler (color.adobe.com) - інструмент від Adobe, який має браузерну та десктопну версії. Дозволяє експортувати колірну схему в Photoshop.

Paletton (paletton.com) - схожий на Kuler, відмінність в тому, що ви не обмежені 5 відтінками. Дуже добре підходить, якщо у вас є основні кольори, і хочеться погратися з додатковими відтінками.

Accessible Color Generator (<https://learnui.design/tools/accessible-color-generator.html>) - доступний генератор кольорів. Зручний інструмент для того, щоб взяти один колір (колір теми, колір логотипу тощо) і перетворити його на цілу доступну кольорову палітру, яка підійде також людям з вадами зору.

Також ви можете підібрати кольорову палітру власноруч. В Інтернеті загалом говориться про колір як RGB-код. RGB не є доброю основою для створення дизайну. Набагато кориснішим є HSB (який є синонімом HSV і подібний до HSL). HSB кращий за RGB, оскільки він відповідає тому, як ми природно думаємо про колір, і ви можете передбачити, як зміни значень HSB вплинуть на колір.

Змінюючи насиченість і яскравість одного відтінку, ви можете згенерувати кілька кольорів - темні, світлові, фонові, акценти, привертаючі увагу.

Використання кількох кольорів з одного або двох базових відтінків - це найнадійніший спосіб підкреслити та нейтралізувати елементи, не роблячи дизайн безладним.



Рис 9. Приклад інтерфейсу by Kerem Suer

### 3. Типографія

Ви створюєте правильну структуру, встановлюючи візуальну ієархію, яка організовує всі візуальні елементи, щоб користувачі могли легко сприймати контент. Шрифтові ієархії використовуються для друкованого вмісту шляхом

поділу його на різні типи стилів: заголовки, підзаголовки, текст, т. д. Використання чіткої ієрархії робить текст розбірливим і легким для читання. Крім того, він дозволяє виділяти ключові частини, щоб привернути увагу користувача. Гарне емпіричне правило полягає в збільшенні або зменшенні розміру в два рази при зміні розмірів тексту. Наприклад, якщо ви використовуєте 32px для заголовка, то використовуйте 16px для основного тексту, щоб створити контраст.

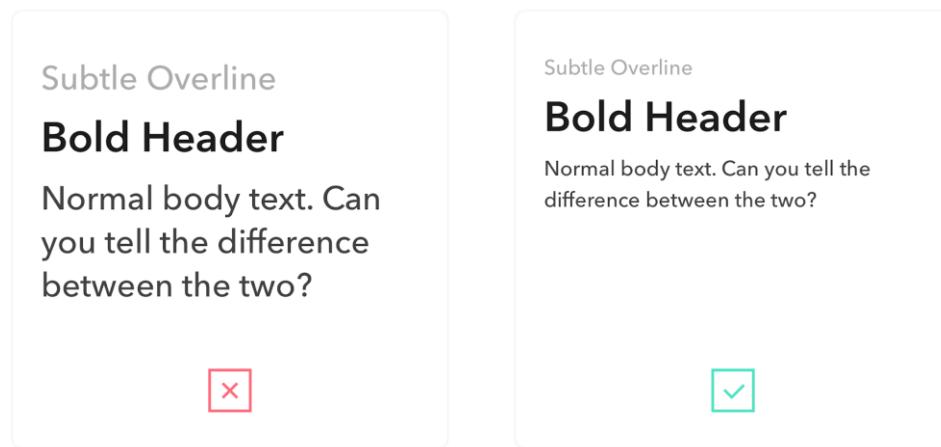


Рис 10. Приклад текстового контрасту

В останні роки розміри заголовків збільшилися, через зміну модних тенденцій. Зазвичай для заголовків H1 використовують розмір шрифту більший 38px. Для H2, H3 - поділ в межах 32-20px. Для загального тексту 14-20px.

Старайтесь не використовувати шрифти менші ніж 11px для десктопів та 10px для мобайлу, оскільки вони будуть складними для читання.

Нехай ваш текст дихає. Міжбуквенний інтервал, висота і довжина рядка - це інструменти коректування прогалин в шрифтах. Відсутність пробілу може привести до поганої розбірливості. З іншого боку, занадто багато порожнього простору може привести до того, що користувач загубиться під час читання.

Довжина рядка - це ширина текстового поля. Краще мати більш короткі рядки, щоб поліпшити читабельність тексту. Рекомендована довжина рядка для екранного тексту знаходиться в діапазоні від 60 до 80 символів, що засноване на розмірі екрану 1440px і розмірі шрифту 16px. Для мобільних екранів замість цього ви повинні налаштувати діапазон від 35 до 45 символів в рядку.

Коли справа доходить до вибору шрифту, важливо враховувати контекст тексту і потенційну аудиторію. Існує безліч різних шрифтів, кожен з яких приносить в інтерфейс свій настрій і стиль.

Намагайтесь не використовувати більше двох шрифтів. Загальноприйнятою практикою проектування є обмеження кількості шрифтів, використовуваних в інтерфейсі. Як правило, двох різних гарнітур має бути достатньо. Це не означає, що ви не можете використовувати більше, але, якщо у вас немає вагомих причин, краще цього не робити.

Виходом може бути використання сімейств шрифтів. Використовуючи сімейство шрифтів, ми можемо використовувати один і той же шрифт з різними варіаціями. Шрифти з однієї сім'ї прекрасно поєднуються, тому що вони гнучкі і послідовні.

При виборі шрифту знайдіть сімейства, які мають різні ваги (light, regular, medium, bold, extra bold, а також такі стилі, як стиснений, розширений і курсив). Це дасть вам більше можливостей для вивчення різних стилів без додавання додаткових шрифтів.

Ultralight  
Thin  
Light  
Regular  
Medium  
**Semibold**  
**Bold**  
**Heavy**  
**Black**

Рис 11. Приклад шрифтової сім'ї

Вирівнювання тексту для кращої читабельності. У більшості частин світу користувачі читають зверху вниз, зліва направо. Вирівнявши текст зліва, ви можете поліпшити його читабельність. Послідовний лівий край забезпечує місце, куди око може повернутися після закінчення рядка, що значно полегшує читання. Також важливо уникати одного слова на початку нового рядка.

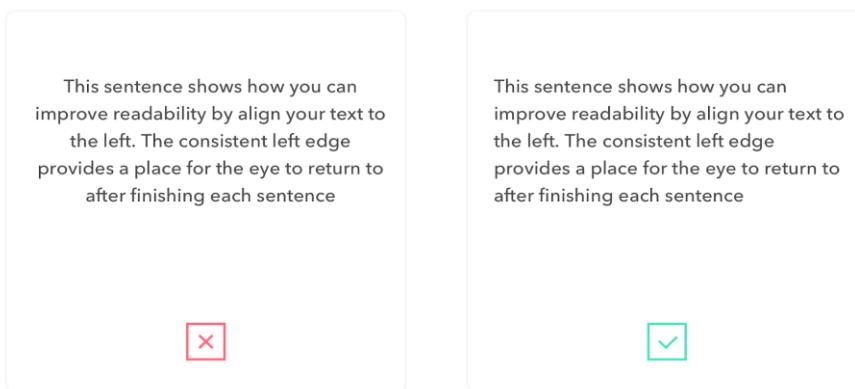


Рис 12. Вирівнювання тексту

По правому краю текст вирівнюють в рідкісних випадках, наприклад, цифри в таблицях. Це допомагає візуально відокремити значення від значень без додаткових візуальних інструментів.

Вирівнювання по центру використовують для одностовпчикових сайтів зі спокійною композицією і рівною побудовою.

Вирівнювання по ширині - це газетний стиль. Так верстають і книги, спеціалісти працюють над забезпеченням якісного і гармонійного вигляду, без розривів між символами.

#### 4. Взаємодія з елементами

*Враховуємо закон Фіттса.*

Його можна охарактеризувати як: “Час, необхідний для переходу до цілі є функцією розміру цілі та відстані до цілі”. Ми можемо застосувати це до дизайну, дивлячись на зони прийняття сигналу наших об’єктів.

Це означає, що чим більшою ми можемо зробити інтерактивну зону ключових лінків і навігаційних елементів, тим легше вони будуть натискатися користувачами.



Рис 13. Порівняння клікабельної області (Зображення: [www.usertesting.com](http://www.usertesting.com))

Якщо говорити про дизайн загалом, то, елементи, якими користувач має користуватися – повинні бути великими і розташовуватися поблизу. І, якщо ви

хочете, щоб користувач легко натиснув на call-to-action кнопку – зробіть її великою близько до центру екрану. Цей закон актуальний не лише для UI, а є одним із основних для розробки ергономіки систем керування.

### *Враховуємо Закон Хіка*

Закон стверджує, що час реакції при виборі з якогось числа альтернативних сигналів залежить від їх кількості. Вперше цю закономірність встановив у 1885 р. німецький психолог В. Меркель, а в 1952 р. її експериментально підтвердив В. Е. Хік.

Ще грецький філософ Аристотель висловив думку, що людина, зіткнувшись з проблемою вибору між рівноцінними речами або подіями, довго вагається і не знає, чому віддати перевагу.

Вільям Едмунд Хік на пару з Реем Хайманом надали цим закономірностям вигляд логарифмічної функції:

$$BP = a * \log(n + 1),$$

де  $BP$  – середнє значення часу реакції на всі альтернативні сигнали;  $n$  – число альтернативних сигналів;  $a$  – коефіцієнт пропорційності. Одниця введена в формулу для врахування ще однієї альтернативи – у вигляді пропуску сигналу.

Простіше кажучи, чим більше ми маємо варіантів рішення, тим довше будемо приймати рішення. Багато хто стверджує, що хотіли б мати більше варіантів у сценарії прийняття рішень, але їхня поведінка показує зворотне. Складність прийняття рішення насправді збільшується з додатковими опціями і в крайніх випадках це стає настільки складним, що ми взагалі можемо не прийняти ніякого.

Щодо дизайну то принцип Хіка спонукає нас робити мінімалістичні дизайни, щоб у них було поменше варіантів вибору. Це стосується всього: утримання на сторінці, елементів навігації, зображення, акцентів і т.д. Видалення всього не

суттєвого, зменшення кількості непотрібних опцій – все це у дизайні саме через закон Хіка

### *Кнопки призову до дії (CTA / C2A, Call to action)*

Кнопка заклику до дії, в залежності від ситуації, зазвичай спонукає користувачів зареєструватися / купити зараз / залогінитися і т.д. Такі кнопки потрібно використовувати там, де система хоче запропонувати дію, яку слід зробити користувачу.

За важливістю кнопки можна поділити на *main button*, *secondary button*, *tertiary button*.

*Main buttons* (первинні кнопки) - повинні бути сильним візуальним індикатором, який допоможе користувачеві здійснити головну дію. Такі кнопки слід використовувати в ситуаціях, коли користувач хоче «Завершити», «Почати», «Далі» і так далі.

*Secondary buttons* (вторинні кнопки) - це кнопка «Назад» біля первинної кнопки «Далі». Або кнопка «Скасувати» біля кнопки «Підтвердити». Вторинні кнопки - це альтернатива первинного дії, яке ми надаємо користувачам.



*Рис 14. Два варіанти вторинних дій поруч з первинними.*

*Tertiary buttons* (третинні кнопки) - зазвичай використовуються для різних дій. Наприклад, коли дія важлива, але може не відповідати тому, що користувач хоче зробити в даний момент. Це може бути кнопка «Додати друга», «Змінити», «Додати нове» або «Видалити», за умови, що ці дії непервинні.

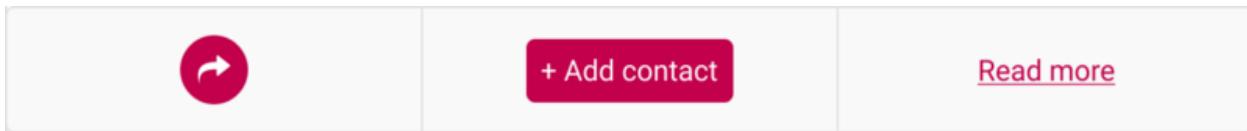


Рис 15. Третинні кнопки в різних формах

Розмір кнопки грає велику роль у доступності інтерфейсу. Не зовсім правильним буде твердження - “кнопки повинні мати висоту 36 пікселів”. Завжди потрібно враховувати висоту рядка використованого вами шрифту і додавати до нього певні одиниці вимірювання. Наприклад, напис кнопки має висоту рядка 20 пікселів, а відступ по вертикалі 8 пікселів.

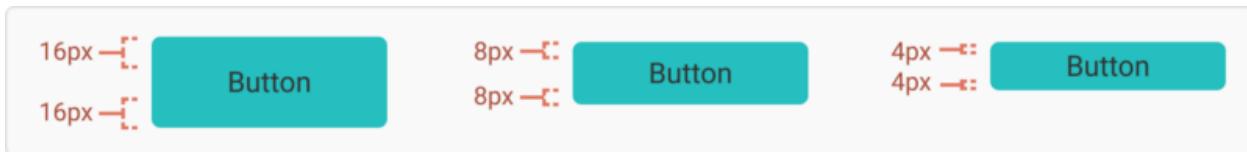


Рис 16. Приклад кнопок з різними висотами

## UI Kit\*

Дизайн-система - це набір правил створення інтерфейсів і дизайну продукту. Вона не тільки описує компоненти, стилі і патерни, а й формує візуальну мову бренду. Дизайн система полегшує роботу дизайнерам на складних системах, коли над розробкою працює декілька людей.

Вона поєднує бібліотеку компонентів, яка супроводжується обґрунтуваннями UX, технічною документацією та найпоширенішими перевіреними рішеннями.

UI-kit - це частина дизайн-системи, бібліотека компонентів у коді. Ці елементи мають дизайнер, а фронтенд-розробник регулює їх стан у статиці, при наведенні, в русі. Бібліотека може містити елементи фірмового стилю: типографіку, кольори, інтервали, анімацію, іконки й ілюстрації, а може тільки елементи інтерфейсу: кнопки, підказки, меню, чекбокси.

Для створення UI-kit по вашому проекту, потрібно на окремій сторінці зібрати всі елементи системи та показати їх у різних станах; описати розміри та типи шрифтів та де вони використовуються; описати подання кольору.

Немає чітких правил подачі чи розміщення елементів в ui kit, але намагайтесь дотримуватися чистоти в поданні. Також ви можете подосліджувати material ui (<https://material-ui.com>), IOS Design Guidelines (<https://developer.apple.com/design/human-interface-guidelines/>), чи інші, для збагачення знань у роботі з дизайном системами та ui кітами.

Нижче наведені приклади простої подачі компонентів у UI-kit:

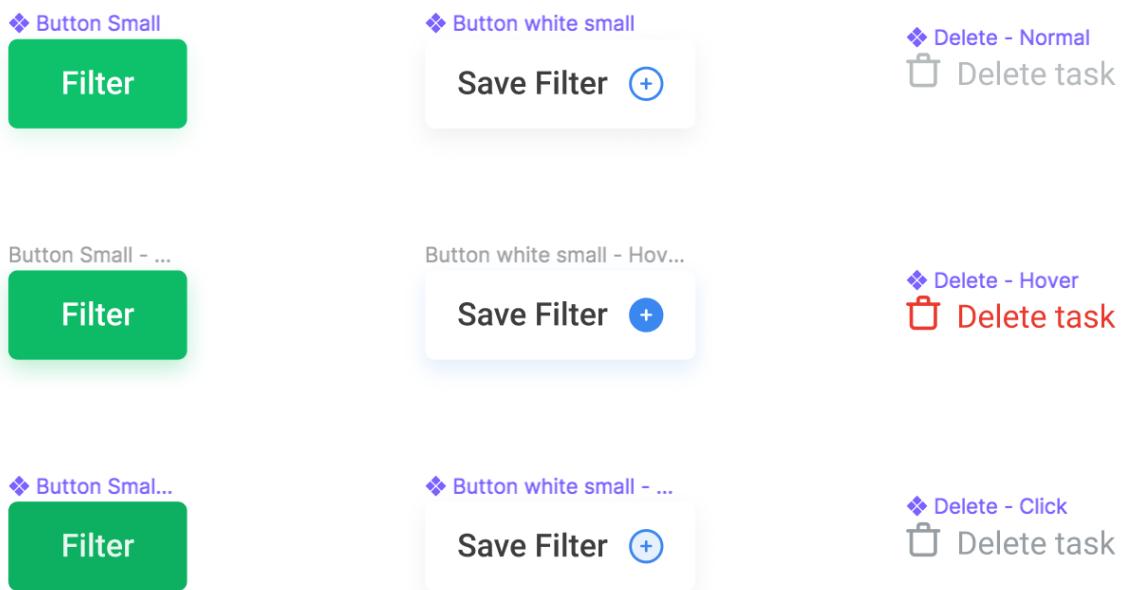


Рис 17. Приклад кнопок у різних станах



Рис 18. Приклад типографіки в ui kit

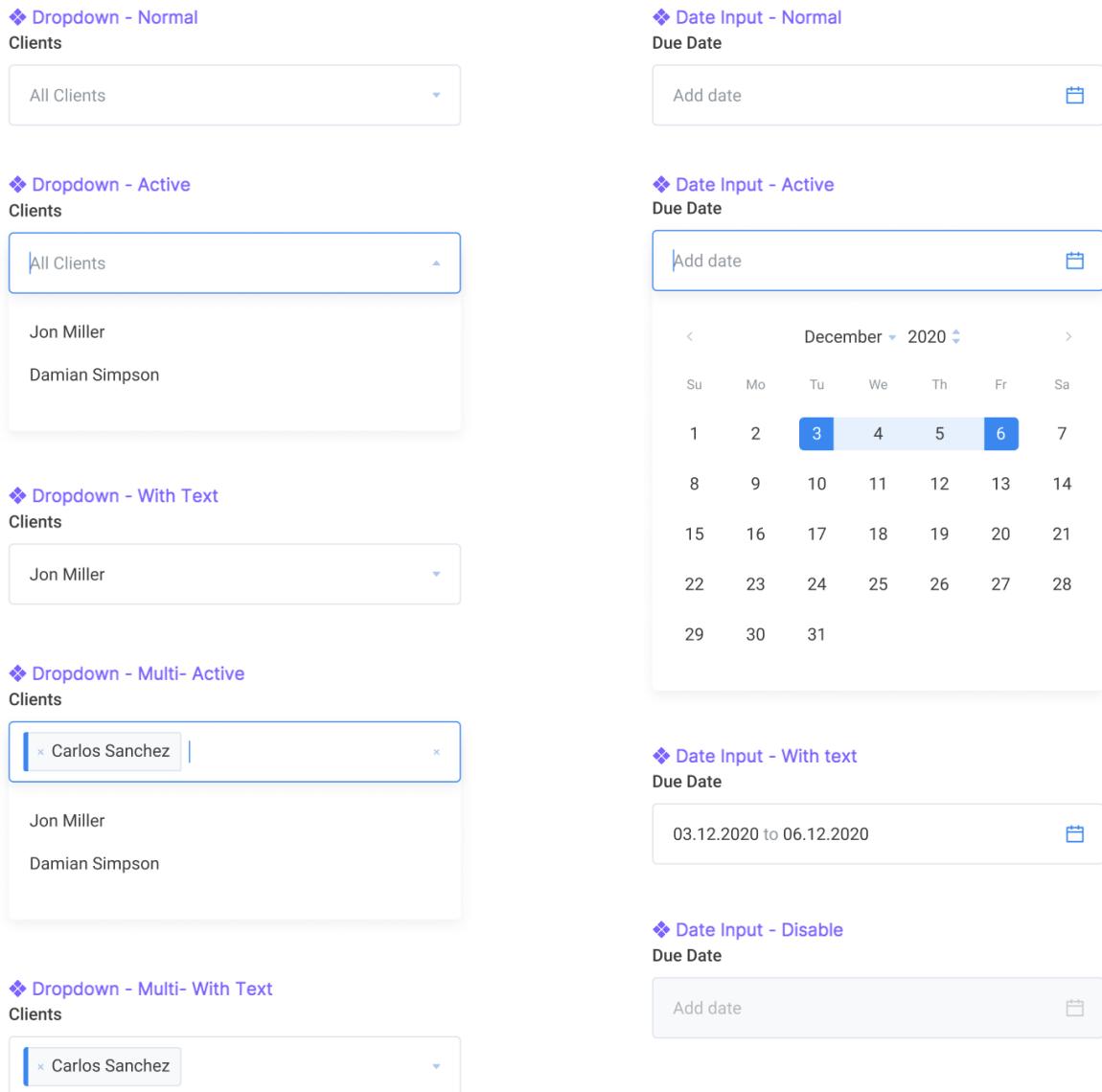


Рис 19. Приклад полів для введення в ui kit

## Висновок

У загальному UX - це те, як користувач взаємодіє з продуктом, а UI - це те, як користувач сприймає цей продукт.

User interface design - це процес візуального управління користувачем через інтерфейс продукту за допомогою інтерактивних елементів, що забезпечує обмін інформацією між людиною і програмними компонентами. Також він відповідає за

перенесення візуальних аспектів бренду на інтерфейс продукту з метою його поліпшення.

***Завдання 1 (одноосібне виконання).***

На базі створених Wireframe у лабораторній роботі №2 опрацюйте та створіть UI-дизайн, застосовуючи вище описані рекомендації.

Пролінкуйте сторінки у Figma для створення інтерактивного прототипу.

***Завдання 2 (командне виконання).***

На базі створених Wireframe у лабораторній роботі №2 опрацюйте та створіть UI-дизайн, застосовуючи вище описані рекомендації.

Пролінкуйте сторінки у Figma для створення інтерактивного прототипу.

Створіть та опишіть UI kit до вашого дизайну.

**Рецензенти**

Журавчак Л.М., докт. тех. наук, проф.

Денисюк П.Ю., канд. техн. наук, доц.

**Мета роботи:** Ознайомитись на практиці із основними поняттями теорії фракталів, навчитись будувати різні фрактальні зображення та використовувати IFS, як простий засіб отримання фрактальних структур.

## Теоретичні відомості

Поняття **фрактал** і **фрактальна геометрія**, що з'явилися в кінці 70-х, із середини 80-х міцно увійшли до ужитку математиків і програмістів. Слово **фрактал** утворено від латинського "fractus", що в перекладі означає "той, що складається з фрагментів". Воно було запропоноване Бенуа Мандельбротом у 1975 році для позначення нерегулярних, але самоподібних структур, якими він займався. Народження фрактальної геометрії прийнято пов'язувати з виходом у 1977 році книги Мандельброта "The Fractal Geometry of Nature". У його роботах використані наукові результати інших учених, що працювали в період 1875–1925 років у тій же області (Пуанкарє, Фату, Жюліа, Кантор, Хаусдорф). Але тільки у наш час вдалося об'єднати їхні роботи в єдину систему.

Роль фракталів в комп'ютерній графіці сьогодні велика. Вони приходять на допомогу, наприклад, коли потрібно за допомогою декількох коефіцієнтів задати ліній і поверхні дуже складної форми. З погляду комп'ютерної графіки, фрактальна геометрія незамінна під час генерації штучних хмар, гір, поверхні морів.

Фактично знайдено спосіб легкого представлення складних об'єктів, які схожі на природні.

Однією з основних властивостей фракталів є самоподібність. У найпростішому випадку невелика частина фрактала містить інформацію про весь фрактал.

Визначення фрактала, дане Мандельбротом, звучить так: "Фракталом називається структура, що складається з частин, які в якомусь сенсі подібні до цілого".

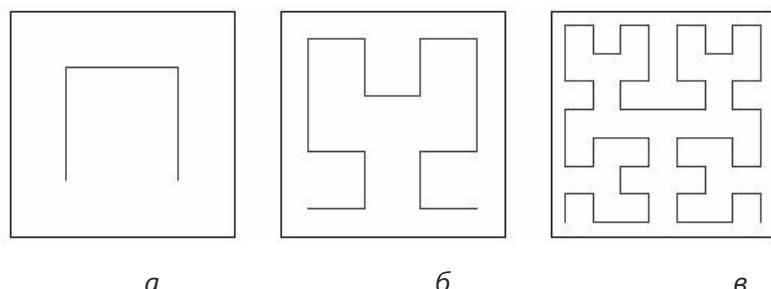
## 1. Геометричні фракталі

Фрактали цього класу найнаочніші. Цей тип фракталів утворюється шляхом простих геометричних побудов. Наприклад, у двомірному випадку їх отримують за допомогою деякої ламаної (або поверхні в тривимірному випадку), званої **генератором**. За один крок алгоритму кожен з відрізків (складових ламаної) замінюється на ламану-генератор, у відповідному масштабі. У результаті нескінченного повторення цієї процедури, виходить геометричний фрактал.

Перші ідеї фрактальної геометрії виникли в XIX ст. Кантор за допомогою простої рекурсивної процедури перетворив лінію на набір незв'язаних крапок (так званий **Пил Кантора**). Він брав лінію і видаляв центральну третину, після цього повторював те ж саме з відрізками.

Пеано ж намалював особливий вид **лінії Пеано** (рис. 1). Для її малювання італійський математик взяв квадрат і видавив у ньому нижню сторону. Утворилася крива Пеано 1-го порядку (рис. 1, а). Далі вчений зменшив квадрат рівно вдвічі, і зробив його 4 копії. Дві з них поставив паралельно одна одній, а інші дві ще повернув на чверть обороту в протилежні сторони та з'єднав кінці ліній квадратів трьома одинаковими відрізками, довжиною, що дорівнює стороні нового зменшеного квадрата. Утворилася крива Пеано 2-го порядку (рис. 2, б). Процедура повторюється знову: зменшується крива 2-го порядку вдвічі, робиться чотири її копії, дві з яких повертаються, і знову з'єднуються відрізками, які теж зменшенні вдвічі (рис. 1, в–е).

Повторювати даний алгоритм можна до нескінченності.



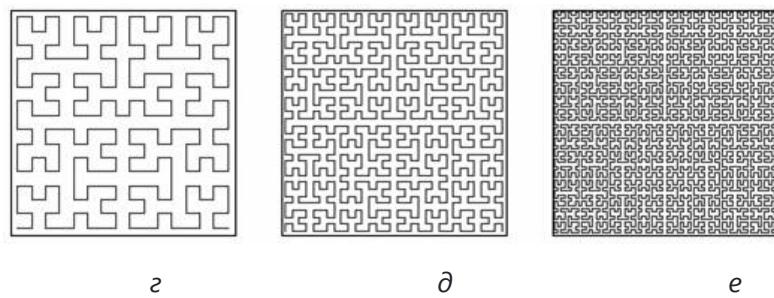


Рис.1. Крива Гильберта-Пеано

Розглянемо фрактальний об'єкт – **тріадну криву Коха**. Побудова кривої починається з відрізка одиничної довжини (рис. 2, а) – це 0-е покоління кривої Коха. Далі кожна ланка (у нульовому поколінні один відрізок) замінюється на утворюючий елемент, позначений на рис. 2, б. У результаті такої заміни виходить наступне покоління кривої Коха. У 1-му поколінні – це крива з чотирьох прямолінійних ланок, кожна завдовжки 1/3. Для отримання 2-го покоління проробляються ті ж дії – кожна ланка замінюється на зменшений утворюючий елемент. Отже, для отримання кожного подальшого покоління, всі ланки попереднього покоління необхідно замінити зменшеним утворюючим елементом. Крива  $n$ -го покоління при будь-якому кінцевому  $n$  називається *передфракталом*. При  $n$ , прямуочому до нескінченності, крива Коха стає фрактальним об'єктом.

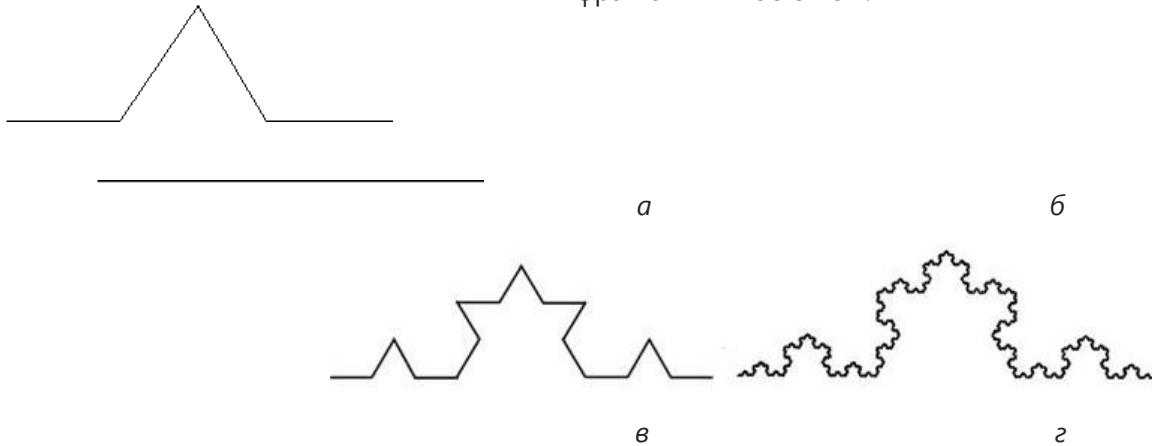


Рис. 2. Побудова тріадної кривої Коха

Дуже цікавим і відомим фракталом є **сніжинка Коха**. Будується вона на основі рівностороннього трикутника, кожна лінія якого замінюється на 4 лінії, довжини кожної дорівнюють  $1/3$  від початкової. І якщо ми зробимо нескінченне число ітерацій – отримаємо фрактал – сніжинку Коха нескінченної довжини.

Виходить, що нескінчена крива покриває обмежену площину (рис. 3).

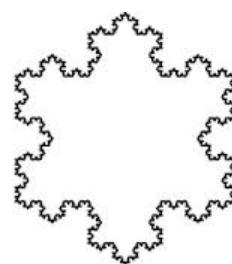


Рис. 3. Сніжинка Коха

Розглянемо інший фрактал – **“дракон” Хартера-Хейтуея** (рис. 4). Вважається, що таку назву фрактал отримав за схожість із традиційними китайськими драконами. Принаймні, так здалося вченим, які вперше його досліджували. Кожна ламана – “дракон” є лише наближенням до фракталу – “дракона” та складається з відрізків. Ламана з номером  $n$  складатиметься з  $2^n$  відрізків. Довжина кожного дорівнює  $\frac{1}{2^n}d$ , де  $d$  – довжина вихідного відрізка. Якщо відрізки пронумерувати числами 0, 1, 2, ... і йти по ламані, то після кожного відрізка потрібно здійснювати поворот. Напрямок повороту визначається номером  $k$  поточного відрізка:

- ✓ повернути праворуч, якщо  $k$  дає залишок 1 від ділення на 4;

- ✓ повернути ліворуч, якщо  $k$  дає залишок 3 від ділення на 4;
- ✓ повертати так, як після відрізка з номером  $k/2$ , якщо  $k$  парне.

Можна переформулювати ці правила, щоб отримати рекурсивну процедуру побудови ламаних-“драконів”. На кожному кроці потрібно замінити кожний із відрізків, що складають дану ламану, на куточки – сторони рівнобедреного прямокутного трикутника, у якого цей відрізок є основою. При цьому потрібно по черзі відкладати ці трикутники то вліво, то вправо за ходом руху від одного кінця ламаної до іншого.

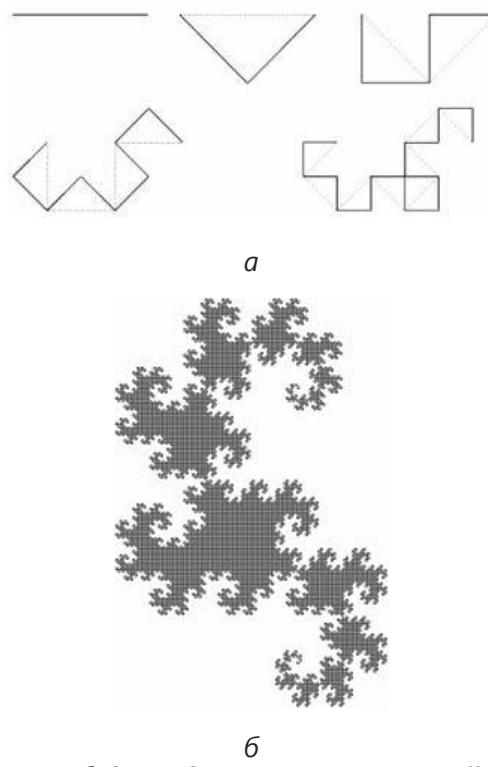


Рис. 4. Побудова “дракона” Хартера-Хейтуея

**H-фрактал.** Для побудови цього фракталу будують фігуру у вигляді букви Н (рис. 5), у якої вертикальні і горизонтальні відрізки рівні. Потім до кожної з 4 вершин фігури присвоюється її копія, зменшена в два рази. Знову до кожного кінця (іх вже 16) необхідно присвоювати копії літери Н, зменшені вже в 4 рази. І так далі. Якщо кількість кроків спрямувати в нескінченність, то вийде фрактал, який візуально майже заповнює деякий квадрат. Н-фрактал всюди щільний у ньому. Тобто в будь-якому околі будь-якої точки квадрата знайдуться точки фрактала.

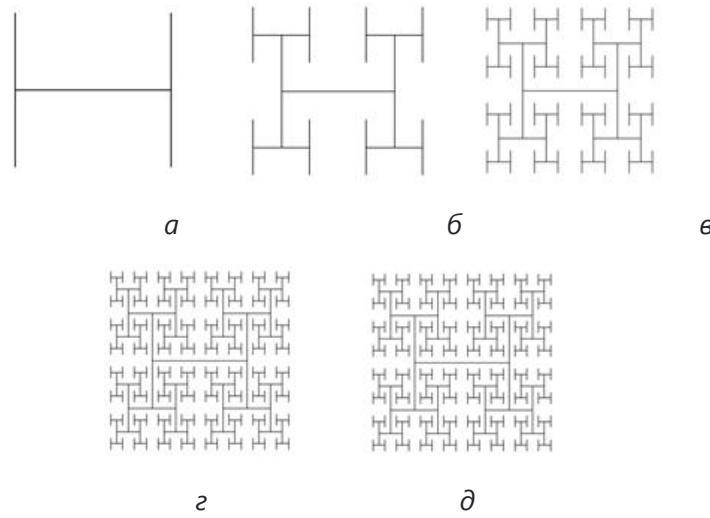


Рис. 5. Н-фрактал

**Крива Мінковського** – класичний геометричний фрактал. Ініціатором є відрізок (рис. 6, а), а генератором – ламана з восьми ланок (две рівні ланки продовжують одна одну) (рис. 6, б).

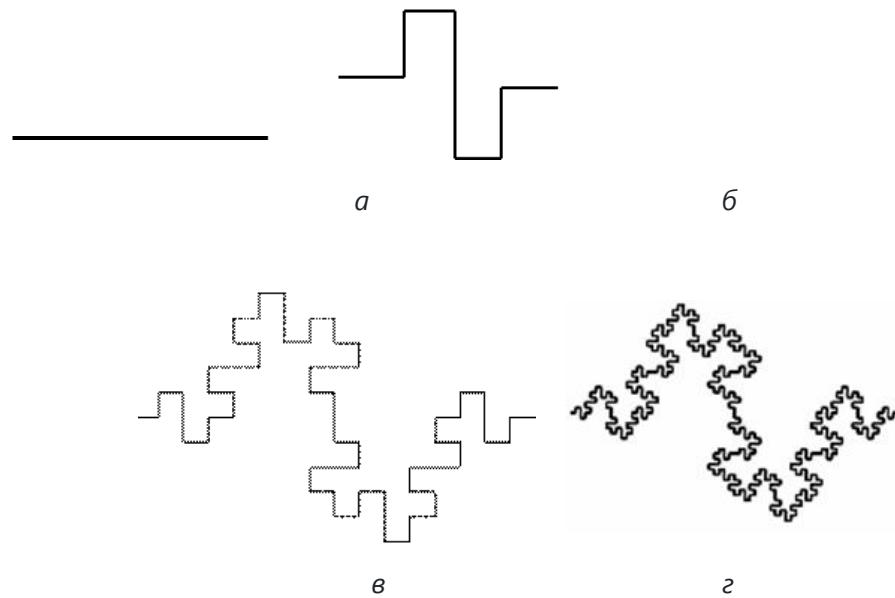


Рис. 6. Побудова кривої Мінковського

**Крива Леві** – фрактал, запропонований французьким математиком П.Леві (рис. 7). Отимається, якщо взяти половину квадрата виду з рис.7, а, а потім кожну сторону замінити таким же фрагментом, і, повторюючи цю операцію, ми отимаємо криву Леві (рис. 7, б-г).

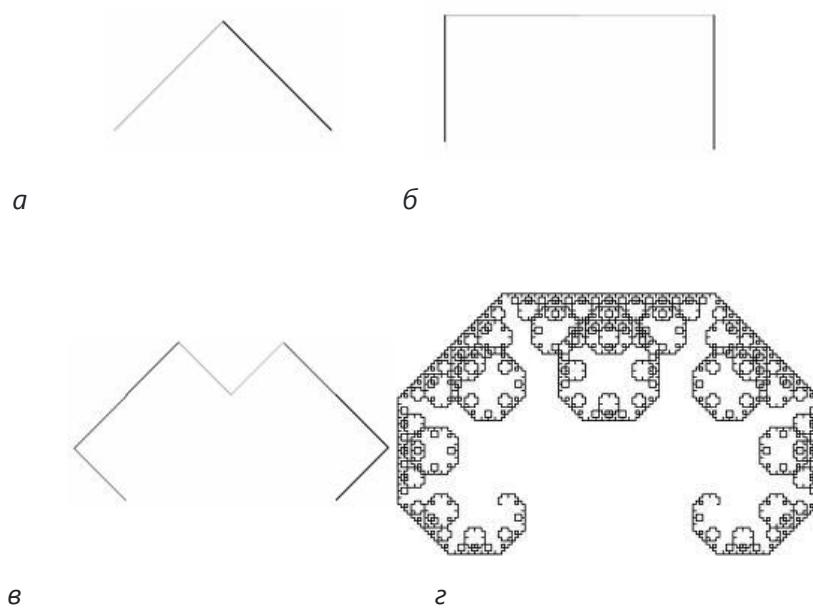
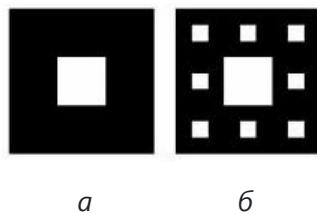


Рис. 7. Побудова кривої Леві

Польський математик Врацлав Серпінський запропонував фрактал – **килим Серпінського** (рис. 8). Для побудови береться суцільний квадрат, розрізається на 9 рівних квадратів і видаляється середина центрального квадрата. На другому кроці видаляється 8 центральних квадратів із решти 8 квадратів і т.д. Після безконечного повторення цієї процедури, від суцільного квадрата залишається замкнута підмножина – килим Серпінського.



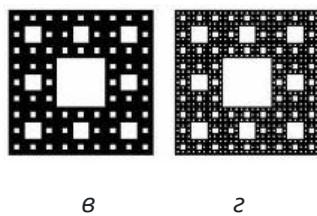


Рис. 8. Килим Серпінського

У 1915 році Врацлав Серпінський розглянув ще один фрактал – **трикутник Серпінського** (рис. 9). Цей фрактал відомий також як "серветка" або "решітка" Серпінського. Щоб побудувати даний фрактал необхідно взяти рівносторонній трикутник (рис. 9, а). На першому кроці видаляється трикутник з вершини в середині сторін початкового трикутника (рис. 9, б). На другому кроці видаляються аналогічні трикутники із трьох менших трикутників, що залишилися після першого кроку, і т.д. Після нескінченного повторення цієї процедури від суцільного трикутника залишається підмножина – трикутник Серпінського.

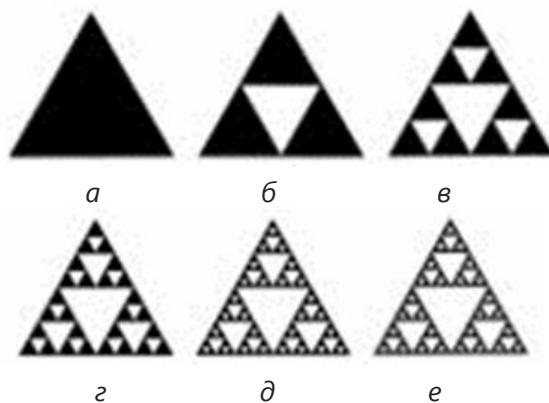


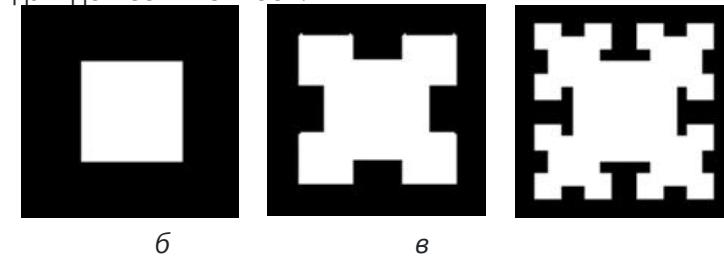
Рис. 9. Трикутник Серпінського

Т-фрактал. Ймовірно, цей фрактал отримав таку назву за схожість з рейсшиною (рис. 10) з причіпленою перпендикулярно планкою у вигляді букви Т. По-англійськи цей інструмент так і називається – T-square.



Рис. 10. Рейсшина

Побудова Т-фракталу розпочинається із одиничного квадрата (рис. 11, а). На першому кроці необхідно зафарбувати в центрі білим кольором квадрат зі стороною  $1/2$ . Потім потрібно подумки розділити квадрат на 4 одинакових квадрати і в центрі кожного з них зафарбувати квадрат зі стороною  $1/4$  (рис. 11, б). Далі кожен з цих 4 квадратів знову ділиться на 4 частини, всього вийде 16 квадратиків, і з кожним з них потрібно повторити процедуру (рис. 11, в). І так далі до нескінченності.



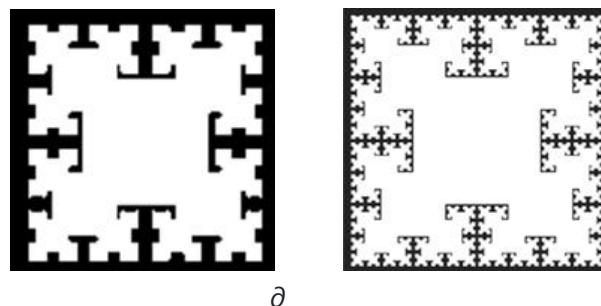


Рис. 11. Побудова Т-фракталу

Дерево Піфагора. Даний фрактал (рис. 12) називається так тому, що кожна трійка попарно дотичних квадратів обмежує прямокутний трикутник і виходить картинка, якій часто ілюструють теорему Піфагора: «Піфагорові штані на всі сторони рівні».

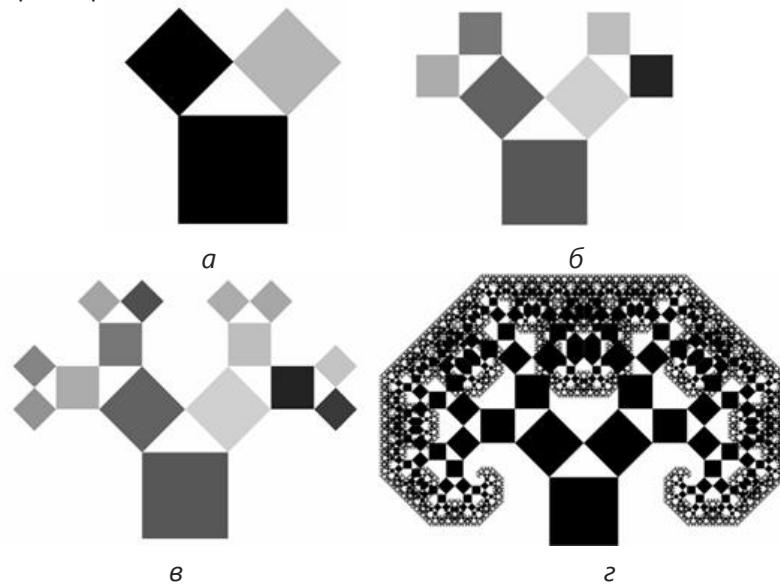


Рис. 12. Кроки побудови дерева Піфагора

Добре видно, що все дерево обмежене. Якщо найбільший квадрат одиничний, то дерево поміститься в прямокутнику  $6 \times 4$ . Отже, його площа не перевищує 24. Але з іншого боку, кожен раз додається в два рази більше трійок квадратиків, ніж у попередньому кроці, а їх лінійні розміри в  $\sqrt{2}$  разів менші. Тому на кожному кроці додається одна і та ж площа, яка дорівнює площі початковій конфігурації, тобто 2. Здавалося б, тоді площа дерева повинна бути нескінченна! Але насправді суперечності тут немає, тому що досить швидко квадратики починають перекриватися, і площа збільшується не так швидко. Вона таки скінчена, але, досі точне значення невідоме, і це відкрита проблема.

У комп'ютерній графіці використання геометричних фракталів необхідне при отриманні зображень дерев, кущів, берегової лінії. Двомірні геометричні фракталі використовуються для створення об'ємних текстур (малюнка на поверхні об'єкта).

## 2. Алгебраїчні фракталі

Це найбільша група фракталів. Отримують їх за допомогою нелінійних процесів в  $n$ -мірних просторах. Найбільше вивчені двомірні процеси. Інтерпретуючи нелінійний ітераційний процес, як дискретну динамічну систему, можна користуватися термінологією теорії цих систем: *фазовий портрет*, *сталий процес*, *аттрактор* і т.д.

Відомо, що нелінійні динамічні системи володіють декількома стійкими станами. Той стан, в якому опинилася динамічна система після деякого числа ітерацій, залежить від її початкового стану. Тому кожен стійкий стан (або як говорять – *аттрактор*) володіє деякою областю початкових станів, з яких система обов'язково потрапить в дані кінцеві стани. Таким чином, фазовий простір системи розбивається на області тяжіння аттракторів. Якщо фазовим є двомірний простір, то забарвлюючи області тяжіння різними кольорами, можна отримати *колірний фазовий портрет* цієї системи (ітераційного процесу). Міняючи алгоритм вибору кольору, можна отримати складні фрактальні картини з химерними багатоколірними узорами. Несподіванкою для математиків стала можливість за допомогою примітивних алгоритмів породжувати дуже складні нетривіальні структури.

Як приклад, розглянемо **множину Мандельброта** (рис. 13, 14). Алгоритм його побудови достатньо простий і заснований на простому ітеративному виразі:

$$Z[i+1] = Z[i]^* Z[i] + C, \quad (1)$$

де  $Z[i]$  і  $C$  – комплексні змінні. Ітерації виконуються для кожної стартової точки  $C$  прямокутної або квадратної області – підмножини комплексної площини. Ітераційний процес продовжується до тих пір, поки  $Z[i]$  не вийде за межі кола радіусу 2, центр якої лежить в точці  $(0,0)$  (це означає, що аттрактор динамічної системи знаходитьться в нескінченості) або після достатньо великого числа ітерацій (наприклад, 200–500)  $Z[i]$  зійдеться до якої-небудь точки кола. Залежно від кількості ітерацій, в перебігу яких  $Z[i]$  залишалася усередині кола, можна встановити колір точки  $C$  (якщо  $Z[i]$  залишається усередині кола протягом достатньо великої кількості ітерацій, ітераційний процес припиняється і ця точка раствується в чорний колір).

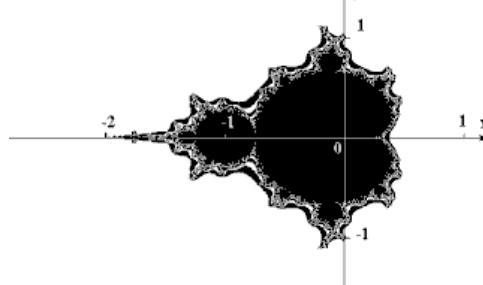


Рис. 13. Множина Мандельброта



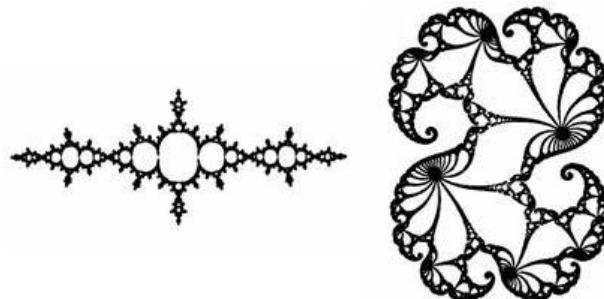
Рис. 14. Ділянка межі множини Мандельброта, збільшена в 200 разів

Вищеписаний алгоритм дає наближення до так званої множини Мандельброта. Множині Мандельброта належать точки, які протягом нескінченного числа ітерацій не йдуть в нескінченність (точки мають чорний колір). Точки, що належать межі множини (саме там виникають складні структури) йдуть в нескінченність за кінцеве число ітерацій, а точки за межами множини, йдуть в нескінченність через декілька ітерацій (білий фон).

Розглянемо ще множину Жюліа, що утворюється за тією ж самою формулою (1), що й множина Мандельброта. Множину Жюліа було винайдено французьким математиком Гастоном Жюліа. Досить дивно, але існують різні типи цієї множини. При малюванні фрактала з використанням різних початкових точок (щоб почати процес ітерацій), генеруються різні зображення. Це може бути застосоване тільки до множини Жюліа.

Вигляд множини Жюліа залежить від значення параметра  $C$ . На рис. 15, а зображене множину Жюліа для  $C=0,27334+0,00742i$ , а на рис. 15, б для  $C=-1,25$ .

Якщо  $C = -0,74543 + 0,11301i$ , то множина Жюліа перетворюється в цікаву ламану лінію (рис. 15, в). Казковим килимом стає множина Жюліа для  $C=i$  (рис. 15, г).



а

б

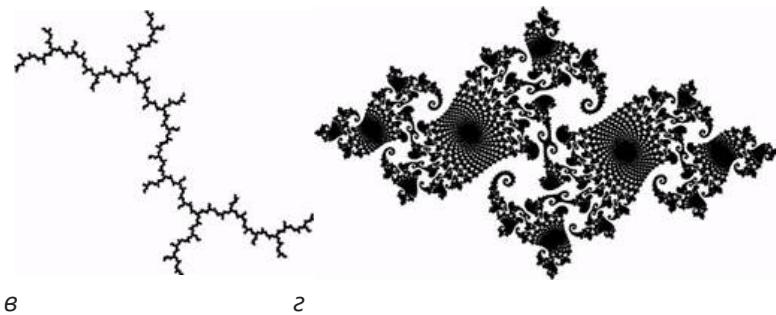


Рис. 15. Множина Жюліа з різними параметрами С

### 3. Стохастичні фрактали

Ще одним відомим класом фракталів є стохастичні фрактали, які виходять в тому випадку, коли в ітераційному процесі випадковим чином міняти які-небудь його параметри. При цьому утворюються об'єкти дуже схожі на природні – несиметричні дерева, порізані берегові лінії і так далі. Двовимірні стохастичні фрактали використовуються при моделюванні рельєфу місцевості і поверхні моря. У зв'язку з цим двовимірні стохастичні фрактали дуже часто використовуються під час моделювання різних природних об'єктів: рельєфу місцевості, поверхні моря тощо (рис. 16).

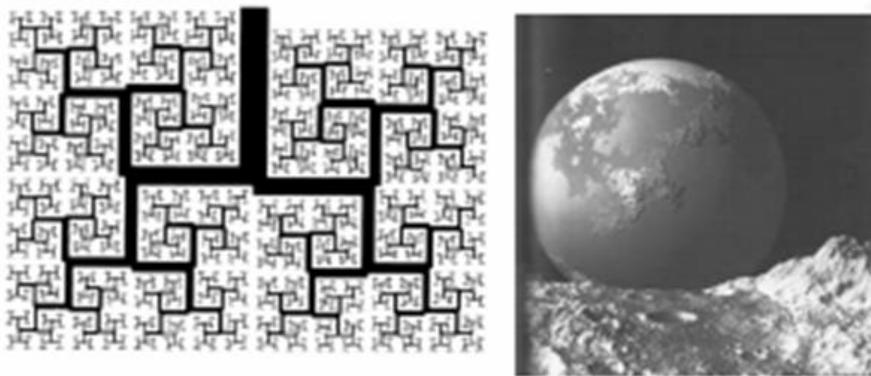


Рис. 16. Використання стохастичних фракталів для моделювання природних об'єктів

**Плазма** – найвідоміший стохастичний фрактал (рис. 17). Покроковий алгоритм побудови:

1. Нехай потрібно заповнити плазмою квадрат  $X \times X$  точок.
  2. Задається яким-небудь чином, наприклад, випадково, кольори кутів квадрата.
  3. Визначається колір середини кожної сторони як середнє між кольорами інцидентних їй вершин плюс / мінус деяка випадкова величина.
  4. Визначається колір центру квадрата як середнє між кутами плюс / мінус деяка випадкова величина.
  5. Виходить 4 квадрати із заданими вершинами – для кожного з них повторюється алгоритм з третього кроку.
- Розмах випадкової величини має залежати від розміру квадрата – чим менший квадрат, тим менше відхилення.

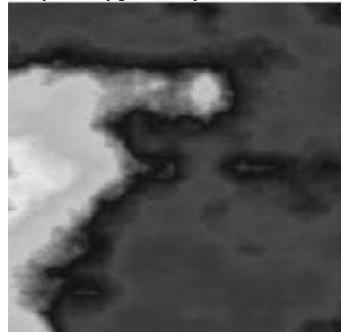


Рис. 17. Плазма

Існують і інші класифікації фракталів, наприклад поділ фракталів на детерміновані (алгебраїчні і геометричні) і недетермінованих (стохастичні).

### 4. Iterated Functions System (система ітераційних функцій)

Метод Iterated Functions System (IFS) з'явився в середині 80-х років як простий засіб отримання фрактальних структур.

IFS є системою деякого фіксованого класу функцій, що відображають одну багатовимірну множину в іншу. Найбільш проста IFS складається з афінних перетворень:

$$X' = AX + BY + C; \quad (2)$$

$$Y' = DX + EY + F.$$

У 1988 році відомі американські фахівці Барнслі і Слоан запропонували деякі ідеї, засновані на міркуваннях теорії динамічних систем, для стискання і зберігання графічної інформації. Вони назвали свій метод "методом фрактального стискання інформації". Походження назви пов'язане з тим, що геометричні фігури, що виникають у цьому методі, зазвичай мають фрактальну природу в сенсі Мандельброта.

На підставі цих ідеї Барнслі і Слоан створили алгоритм, який, за їх твердженням, дозволить стискувати інформацію в 500–1000 разів. Коротко метод можна описати таким чином. Зображення кодується декількома простими перетвореннями (у нашому випадку афінними), тобто коефіцієнтами цих перетворень (у нашому випадку A,B,C,D,E,F).

Наприклад, закодувавши якесь зображення двома афінними перетвореннями, ми однозначно визначаємо його за допомогою 12-ти коефіцієнтів. Якщо тепер задати яку-небудь початкову точку (наприклад,  $X=0 Y=0$ ) і запустити ітераційний процес, то після першої ітерації отримаємо дві точки, після другої – чотири, після третьої – вісім і так далі. Через декілька десятків ітерацій сукупність отриманих точок описуватиме закодоване зображення. Але проблема полягає в тому, що дуже важко знайти коефіцієнти IFS, які кодували б довільне зображення.

Для побудови IFS застосовують окрім афінних і інші класи простих геометричних перетворень, які задаються невеликим числом параметрів. Наприклад, проектні:

$$X' = (A_1X + B_1Y + C_1)/(D_1X + E_1Y + F_1); \quad (3)$$

$$Y' = (A_2X + B_2Y + C_2)/(D_2X + E_2Y + F_2);$$

або квадратичні:

$$X' = A_1X^2 + B_1XY + C_1Y^2 + D_1X + E_1Y + F_1; \quad (4)$$

$$Y' = A_2X^2 + B_2XY + C_2Y^2 + D_2X + E_2Y + F_2.$$

Як приклад використання IFS для побудови фрактальних структур, розглянемо "дракон" Хартера-Хейтуея (рис.18, 19) і криву Коха. Виділимо в цих структурах подібні частини і для кожної з них обчислимо коефіцієнти афінного перетворення. У афінний колаж буде включено стільки афінних перетворень, скільки існує частин подібних цілому зображеню.

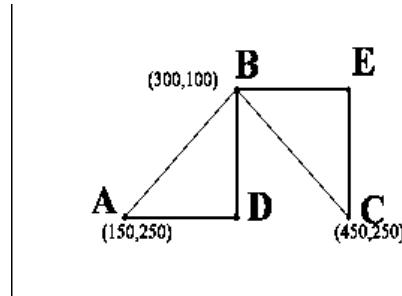


Рис. 18. Заготовка для побудови IFS "дракона" Хартера-Хейтуея

Побудуємо IFS для "дракона" Хартера-Хейтуея. Для цього розташуємо перше покоління цього фрактала на сітці координат дисплея 640x350 (рис. 18). Позначимо точки ламаної, що отримали, як A, B, C. За правилами побудови у цього фрактала дві частини подібні цілому – на рис.18 це ламані ADB і BEC. Знаючи координати кінців цих відрізків, можна обчислити коефіцієнти двох афінних перетворень, що переводять ламану ABC в ADB і BEC:

$$\begin{aligned} X' &= -0.5X - 0.5Y + 490; \\ Y' &= 0.5X - 0.5Y + 120; \\ X' &= 0.5X - 0.5Y + 340; \\ Y' &= 0.5X + 0.5Y - 110. \end{aligned} \quad (5)$$

Задавши початкову стартову точку (наприклад,  $X=0, Y=0$ ), й, ітераційно діючи на неї IFS, після десятої ітерації на екрані отримаємо фрактальну структуру, зображену на рис.19, яка є "драконом" Хартера-Хейтуея. Його кодом (стислим описом) є набір коефіцієнтів двох афінних перетворень.

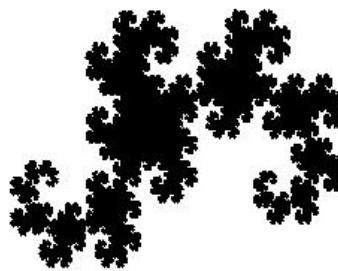


Рис. 19. "Дракон" Хартера-Хейтуея, побудований за допомогою IFS в прямокутнику  $640 \times 350$

Аналогічно можна побудувати IFS для кривої Коха. Неважко бачити, що ця крива має чотири частини, подібні до цілої кривої (рис. 20). Для знаходження IFS знову розташуємо перше покоління цього фрактала на сітці координат дисплея  $640 \times 350$ .

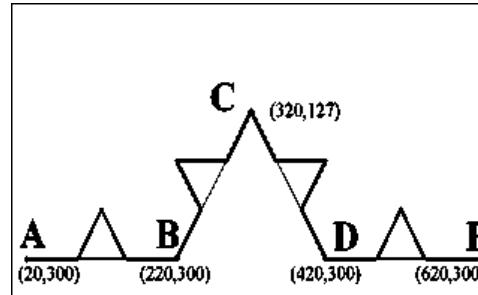


Рис. 20. Заготовка для побудови IFS кривої Коха

Для її побудови потрібний набір афінних перетворень, що складається з чотирьох перетворень:

$$\begin{aligned}
 X' &= 0.333X + 13.333; \\
 Y' &= 0.333Y + 200; \\
 X' &= 0.333X + 413.333; \\
 Y' &= 0.333Y + 200; \\
 X' &= 0.167X + 0.289Y + 130; \\
 Y' &= -0.289X + 0.167Y + 256; \\
 X' &= 0.167X - 0.289Y + 403; \\
 Y' &= 0.289X + 0.167Y + 71.
 \end{aligned} \tag{6}$$

Результат застосування цього афінного колажа після десятої ітерації можна побачити на рис. 21.

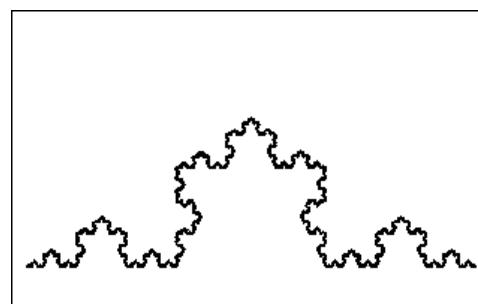


Рис. 21. Крива Коха, побудована за допомогою IFS у прямокутнику  $640 \times 350$

Використання IFS для стиску звичайних зображень (наприклад, фотографій) засноване на виявленні локальної самоподібності, на відміну від фракталів, де спостерігається глобальна самоподібність. За алгоритмом Барнслі відбувається виділення в зображенні пар областей, менша з яких подібна більшій, і збереження декількох коефіцієнтів, що кодують перетворення та переводять велику область в меншу. Потрібно, щоб безліч "менших" областей покривали все зображення. При цьому у файл, що кодує зображення будуть записані не тільки коефіцієнти, що характеризують знайдені перетворення, але і місцеположення та лінійні розміри "великих" областей, які разом з коефіцієнтами описуватимуть локальну самоподібність кодованого зображення. Алгоритм відновлення у цьому випадку повинен застосовувати кожне перетворення не до всієї множини крапок, що вийшли на попередньому кроці алгоритму, а до деякої підмножини, що належить області, відповідній перетворенню.

## Висновки

Виявилося, що в природі фрактальні структури спостерігаються дуже часто: контури хмар, блискавка, дим, дерева, берегова лінія і русла річок, тріщини в матеріалах, бронхи легенів, пористі губки, поверхні порошків, артерії і багато інших структур, які не мають, на перший погляд закономірностей у своїй будові. Але відсутність порядку в них – це ілюзія, яка виникає при першому ознайомленні.

Введення понять фрактала і фрактальної геометрії дозволяє виділити раніше приховані закономірності в будові і властивостях природних об'єктів, класифікувати та досліджувати їх особливості. Зараз вважається, що фрактальний світ добре відображає реальний, оскільки властивості фракталів демонструють багато природних об'єктів. Тому часто говорять, що книга природи написана мовою фракталів. Така дивна схожість реального і фрактального світів обумовлена, перш за все, тим, що властивості фізичного світу змінюються повільно із зміною масштабів. У піску на березі багато властивостей спільних із властивостями гальки, гори мають багато спільного із каменями, що їх складають, маленький струмок багато в чому схожий на велику річку. Саме така незмінність відносно масштабу – характерна риса фракталів.

Цікаво відзначити, що фрактальна математика сьогодні дуже широко застосовується, вона може бути використана для аналізу змін цін і заробітної платні, статистики помилок на телефонних станціях і навіть опису Всесвіту у цілому.

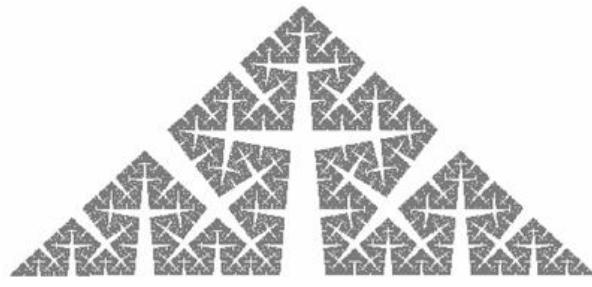
### Контрольні питання

1. Дайте визначення, що таке «фрактал».
2. Яка основна властивість фракталів?
3. Назвіть типи фракталів.
4. Що таке генератор-ламана?
5. Охарактеризуйте термін «аттрактор» .
6. Наведіть приклад передфракталу.
7. Що таке система ітераційних функцій?
8. Які класи геометричних перетворень використовуються для побудови IFS?
9. Як відбувається стиснення зображення за допомогою IFS?
10. Який алгоритм побудови трикутника Серпінського?
11. Чим відрізняється крива Коха від сніжинки Коха?
12. Опишіть побудову множини Мандельброта.
13. Де використовуються фрактали?
14. Наведіть приклад стохастичного фракталу, які кроки його побудови?
15. Поясніть властивість рекурсивності фракталів.

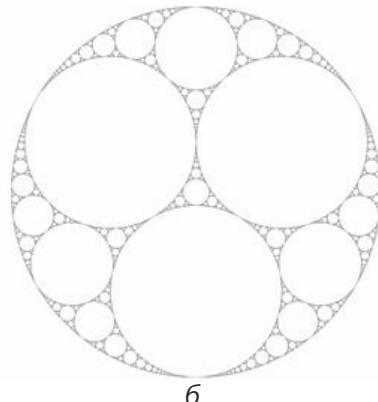
### Варіанти завдань

1. Побудувати тріадну криву Коха.
2. Побудувати криву Міньковського.
3. Побудувати криву Леві.
4. Побудувати "дракон" Хартера-Хейтуея.
5. \*Побудувати множину Мандельброта.
6. Побудувати килим Серпінського.
7. Побудувати трикутник Серпінського.
8. Побудувати сніжинку Коха.
9. \*Побудувати множину Жюліа.
10. Побудувати криву Пеано.
11. Побудувати Н-фрактал.
12. Побудувати Т-фрактал.
13. Побудувати дерево Піфагора.
14. \*Побудувати фрактал, який наведено на рис.22, а.
15. \*Побудувати фрактал, який наведено на рис.22, б.

16. \*Побудувати фрактал, який наведено на рис.22, в.
17. Побудувати фрактал, який задається послідовністю з рис.23, а.
18. Побудувати фрактал, який задається послідовністю з рис.23, б.
19. \*Побудувати фрактал, який задається послідовністю з рис.23, в.
20. \*Побудувати фрактал, який задається послідовністю з рис.23, г.



а



б



в

Рис. 22. Варіанти фракталів:

- а – фрактал Цесаро – варіант кривої Коха з кутом 85°;
- б – мережа Аполона: починаючи з 3 дотичних кіл, вписувати нові кола у вузли, що утворилися;
- в – лист папороті.



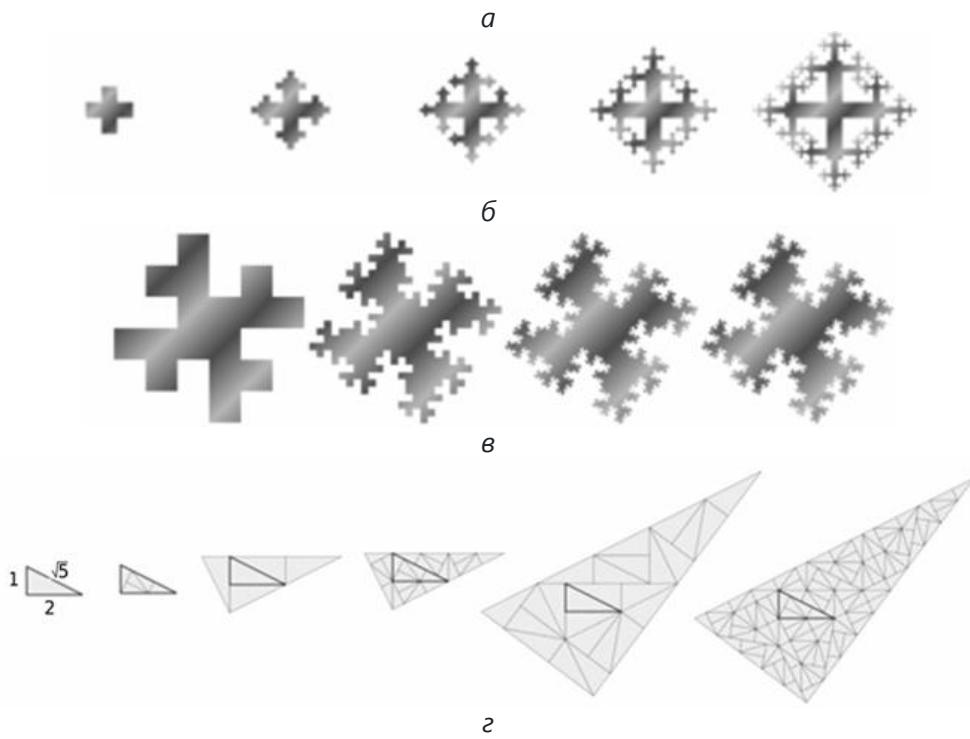


Рис. 23. Варіанти фракталів:  
 а – квадратична пушинка, б – квадратичний хрест,  
 в – 8-сегментний квадратичний фрактал, г – фрактальна вертушка

### Складові звіту

1. Тутильний аркуш.
2. Тема.
3. Мета.
4. Теоретичні відомості.
  - 4.1.1. Опис фракталу, який необхідно побудувати.
  - 4.1.2. Опис алгоритму побудови фракталу:
    - 4.1.2.1. Назва алгоритму.
    - 4.1.2.2. Номер кроку. Назва кроку. Реалізація кроку.
5. Формульовання завдання.
6. Текст програми з коментарями.
7. Результати роботи програми.
8. Висновки.

### Вимоги до програми

Програма має передбачати такі можливості:

1. Автоматична побудова фракталу:
  - 1.1. Починати побудову із центру Canvas.
  - 1.2. Кількість ітерацій – 10.
2. Ввід вхідних даних вручну:
  - 2.1. Задати центр побудови фракталу.
  - 2.2. Задати кількість ітерацій.
  - 2.3. Можливість додавання декількох ітерацій з відповідним відображенням змін на Canvas.
  - 2.4. Можливість відкидання декількох останніх ітерацій з відповідним відображенням змін на Canvas.
3. Передбачити можливість некоректного введення даних.
4. Передбачити можливість покрокового відображення побудови фракталу.
5. Передбачити розрахунок розмірів фігури у відповідності до розмірів Canvas, щоб межі фігури не вийшли за межі Canvas.
6. Підпис кроку побудови фракталу та назви фракталу.

### СПИСОК ЛІТЕРАТУРИ

електромагнітних хвиль, який після взаємодії з оточуючим середовищем попадає в око, де в результаті фізичної і хімічної реакції виробляються електроімпульси, що сприймаються мозком людини. Різна довжина хвилі сприймається людьми як різний колір (колір – це один із факторів світлового випромінювання). Видиме світло з найбільшою довжиною хвилі буде червоним (780 нанометрів), із найменшою – синім (380 нанометрів). При зміні довжини хвилі кольори плавно переходят один в один (рис.1). Чисті кольори існують лише при певних довжинах хвилі (наприклад, чистий фіолетовий – при довжині 400 нм). Найбільш чутливе око людини до зелених кольорів (520 нм), потім до червоного і синього. У видимому спектрі людське око розрізняє 120 кольорів. За допомогою хвильової теорії, висунутої Гюйгенсом у 1678 р., було пояснено багато властивостей світла, зокрема закони відбиття та заломлення.

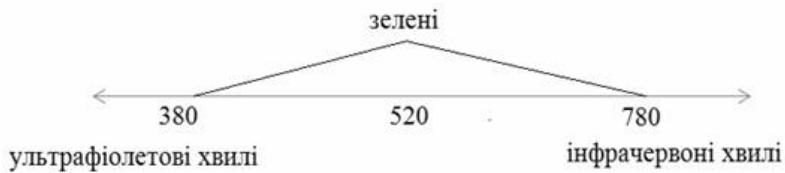


Рис.1. Чутливість ока до довжини хвилі

Коли предмети випромінюють світло (такими є монітори), вони мають той колір, який ми бачимо. Коли деякі предмети, наприклад, папір, відбивають світло, то їхній колір визначається кольором світла, що падає на предмет, і кольором, який ці предмети відбивають.

Розглянемо, як відбувається сприйняття світла людським оком. Людське око дуже складна система. Коли очі дивляться на світ – світло попадає в око через рогівку, далі за допомогою кришталика проектується на сітківку ока, де фоторецептори перетворюють світлову інформацію в імпульси у нервових волокнах. Сітківка ока містить два принципово різні типи світлочутливих рецепторів: палички, які володіють широкою спектральною кривою чутливості, внаслідок чого вони не розрізняють довжини хвиль, а отже, і кольору та колбочки, які характеризуються вузькими спектральними кривими і тому володіють чутливістю до кольору. У кожному оці знаходиться біля 6 млн. колбочок і 120 млн. паличок (приблизно 250 млн. рецепторів на два ока).

В основі трикомпонентної теорії світла лежить той факт, що в центральній частині сітківки знаходяться три типи чутливих до кольору колбочок, які відповідають за чутливість до довгих, середніх і коротких хвиль, тобто один тип колбочок реагує на зелений колір, другий – на червоний, а третій – на синій колір. Ці три кольори називаються основними (базовими).

Якщо на всі три види колбочок діє одинаковий рівень енергетичної яскравості, то світло буде білим. При низькому освітленні колбочки втрачають свою чутливість, зате зростає чутливість паличок, що забезпечує нашу здатність бачити при освітленні низького рівня, тому колбочки працюють вдень, а палички – вночі.

Отже, колір – це характеристика дії випромінювання на око людини.

Слід зауважити, що колір має і психофізичну природу, тобто сприйняття кольору носить суб'єктивний характер і залежить не тільки від фізичних властивостей світла, а й від інтерпретації світла зоровою системою людини.

Кольори, що існують в природі, діляться на ахроматичні і хроматичні. До групи ахроматичних відносяться всі білі, чорні і проміжні між ними сірі кольори. Група хроматичних кольорів включає всі кольори спектру, а також кольори, яких немає в спектрі, але є результатом змішування спектральних кольорів – золотисті, тютюнові, теракотові, пурпурні і т.д. Промені світла, потрапляючи на сітківку ока, дають відчуття кольору.

Наука, що вивчає колір і його вимірювання, називається *колориметрією*. Вона описує загальні закономірності сприйняття кольору людиною. Основними законами колориметрії є закони змішування кольорів. Ці закони в найбільш повному вигляді були сформульовані в 1853 р. німецьким математиком Германом Грасманом.

1. *Закон тривимірності*. Колір виражається в тривимірному просторі. Це означає, що для його опису потрібні три компоненти. Довільні чотири кольори знаходяться в лінійній залежності. Іншими словами, для будь-якого кольору  $C$  можна записати рівняння:

$$C = k_1 C_1 + k_2 C_2 + k_3 C_3,$$

де  $C_1$ ,  $C_2$ ,  $C_3$  – базисні, лінійно незалежні кольори,  $k_1$ ,  $k_2$ ,  $k_3$  – коефіцієнти, що вказують на кількість змішуваних кольорів.

2. *Закон неперервності.* Якщо в суміші трьох кольорів один з них змінюється неперервно, а інші залишаються сталими, то колір суміші теж змінюється неперервно.
3. *Закон адитивності.* Колір суміші залежить тільки від кольорів компонент і не залежить від їх спектральних складових, тобто змішувана компонента в свою чергу може бути отримана змішуванням інших компонент.

Колір може бути отриманий у процесі випромінювання та в процесі відбивання, тому існують два протилежних методи опису кольору: система адитивних кольорів і система субтрактивних кольорів. Для математичного опису кольору використовується поняття моделі кольору. Кольори в природі рідко бувають простими. Більшість кольорових відтінків утворюється змішуванням основних кольорів. Спосіб розкладання кольору на складові компоненти називається *моделлю кольору*. Вся безліч кольорів, які можуть бути створені в колірній моделі, називається колірним діапазоном.

Колірні моделі потрібні для математичного опису спектру кольорів, видимих на екрані монітора, або на скануючих та друкуючих пристроях. Кольори представляються моделлю як результат зміщення декількох базових (основних) кольорів, з яких вони складаються. Кожен базовий колір має свій діапазон інтенсивності. При складанні всіх базових кольорів з різною інтенсивністю утворюються кольори, доступні для даної моделі. Колірні діапазони моделей можуть розрізнятися.

Розглянемо основні колірні моделі, найчастіше використовувані в комп'ютерній графіці.

## 1. Колірні моделі

### 1.1. Адитивна колірна модель RGB

Адитивна модель кольору найпростіша для розуміння. Вона є досить штучним прийомом, оскільки продиктована технологією виготовлення електронно-променевих трубок. Це апаратно-орієнтована модель, в якій кольори описуються за допомогою складання трьох базових кольорів – червоного, зеленого, синього – в різних пропорціях. Тому модель RGB називається адитивною (від англ. «add» складати, додавати). Кольори також називають колірними каналами моделі RGB.

Модель названа за першими буквами англійських слів:

*R (RED) – червоний;*

*G (GREEN) – зелений;*

*B (BLUE) – синій.*

Кожен з базових кольорів може приймати інтенсивність (насиченість) у діапазоні від 0 до 255. Повна кількість кольорів, які представляються цією моделлю, дорівнює  $256 \cdot 256 \cdot 256 = 16\ 777\ 216$ . За допомогою моделі RGB описуються кольори, що отримуються змішуванням світлових променів. Дану модель використовують монітори, телевізори, сканери, слайд-проектори, кольорові лампи реклами і інші пристрої, в яких колір виходить шляхом змішування світлових пучків. Вона також використовується для опису кольорів на сторінках Інтернету в спеціальному шістнадцятковому вигляді (#RRGGBB).

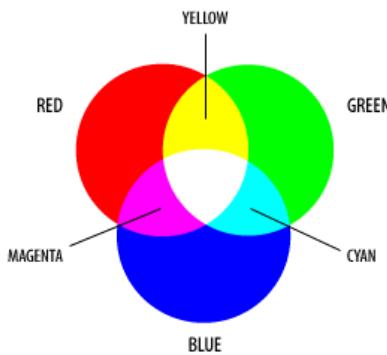


Рис.2 Комбінації базових кольорів моделі RGB

Змінюючи інтенсивність свічення кольорових крапок, можна створити велике різноманіття відтінків. Якщо інтенсивність кожного з них максимальна (255), то виходить білий колір. Відсутність всіх трьох кольорів дає чорний колір. Якщо змішуються всі кольори з однаковою інтенсивністю (але не максимальною і не мінімальною), отримуємо сірий колір.

Для зображення адитивної моделі найчастіше застосовують одиничний куб з розподілом кольорів уздовж одиничних векторів (рис. 3). Початок відліку  $(0,0,0)$  відповідає чорному кольору. Максимальне значення RGB  $(1,1,1)$  відповідає білому кольору,  $(1,0,0)$  – червоному,  $(0,1,0)$  – зеленому,  $(0,0,1)$  – синьому.

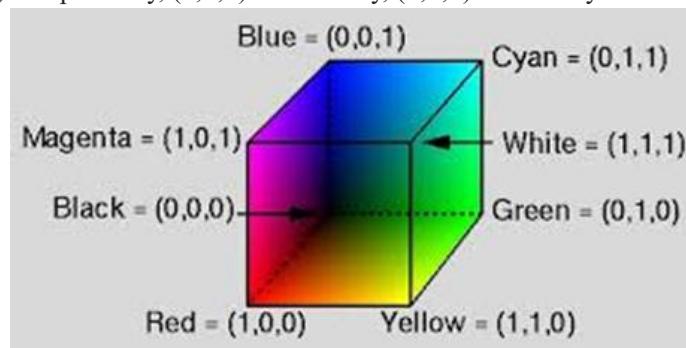


Рис. 3. Колірний куб моделі RGB

На сьогодні система RGB є офіційним стандартом. Рішенням Міжнародної Комісії з освітлення (МКО) в 1931 р. були стандартизовані основні кольори. Комісія рекомендувала використовувати як  $R$ ,  $G$ ,  $B$  такі монохроматичні кольори: випромінювання хвиль довжиною для  $R$  – 700 нм, для  $G$  – 546,1 нм, для  $B$  – 435,8 нм.

Недолік моделі RGB полягає в тому, що не всі кольори, утворені в ній, можна вивести на друк. Проте більше 16 млн кольорів, що представляються в RGB, виявляються цілком достатніми для практичних потреб. Іншими словами, кольори на екрані вашого монітора можуть виглядати інакше при їх виведенні на друк, причому ця відзнака може виявитися принциповою, а не обумовленою низькою якістю принтера або монітора.

### 1.2. Субтрактивна колірна модель CMY (CMYK)

Субтрактивна модель використовується для підготовки не екранних, а друкованих зображень, тобто для пристрій, які реалізують принцип поглинання (віднімання) кольорів. Друковані зображення відрізняються від екранних зображень тим, що їх бачать не у світлі, що проходить, а у відбитому світлі, оскільки аркуш паперу не випромінює світло.

Базовими кольорами моделі CMY є кольори, які виходять у результаті віднімання основних кольорів RGB від білого. Звідси назва моделі субтрактивна (від англ. «to subtract» – віднімати). Базові кольори моделі CMY:

$C$  (CYAN) – блакитний = білий - червоний = зелений + синій;

$M$  (MAGENTA) – пурпурний = білий - зелений = червоний + синій;

$Y$  (YELLOW) – жовтий = білий - синій = червоний + зелений.

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. 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.

На рис. 4 показано, як різні комбінації блакитного, жовтого і пурпурного кольорів, що дають червоний, синій і зелений кольори. Таким чином, система координат CMY – той же куб, що і для RGB, але з початком відліку в точці, що відповідає білому кольору. Колірний куб моделі CMY наведено на рис.5.

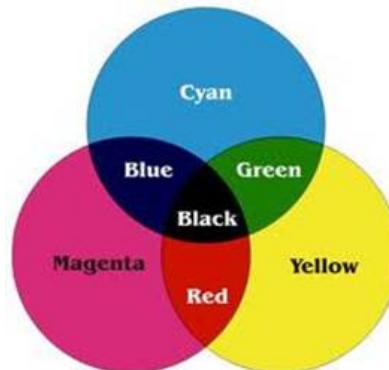


Рис.4 Комбінації базових кольорів моделі CMYK

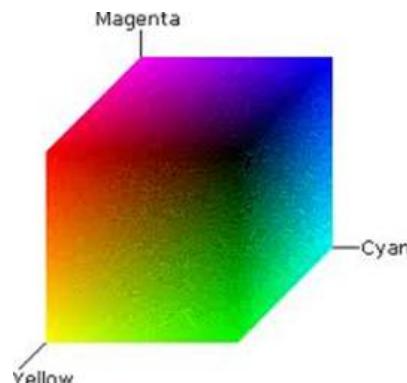


Рис. 5. Колірний куб моделі CMYK

Істотною проблемою в поліграфії є чорний колір. Теоретично його можна отримати змішуванням трьох доповнювальних 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 дорогої чорнила та високу вологість паперу на струменевих принтерах, а також 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 діапазоном.

### 1.3. Модель HSB

Системи кольорів RGB і CMYK базуються на обмеженнях, які накладаються апаратним забезпеченням (моніторами комп'ютерів у разі використання RGB і друкарських фарб у разі CMYK). Більш інтуїтивним способом опису кольору є представлення його у вигляді тону, насиченості і яскравості – система HSB (Hue – тон або відтінок, Saturation – насиченість, Brightness – яскравість) (рис. 6).

HSB не строга математична модель, але вона дуже зручна для підбору відтінків і кольорів. Ця модель заснована на моделі RGB, але має циліндричну систему координат. Будь-який колір в моделі HSB визначається своїм колірним тоном (власне кольором), насиченістю (тобто відсотком доданої до кольору білої фарби) і яскравістю (відсотком доданої чорної фарби).

Перевага HSB перед іншими моделями полягає в тому, що вона більше відповідає природі кольору і добре узгоджується з моделлю сприйняття кольорів людиною. Тон є еквівалентом довжини хвилі світла, насиченість – інтенсивності хвилі, а яскравість – загальної кількості світла. Модель HSB відповідає поняттю кольору, яке використовують професійні художники. У них зазвичай є декілька основних фарб, а всі інші виходять додаванням до них білої і чорної. Таким чином, потрібні кольори – це деяка модифікація основних фарб: освітлених або затемнених. Хоча художники і змішують фарби, але це вже виходить за рамки моделі HSB.

Тон – це основний колір, який можна виділити в кольорі (довжина хвилі, яка переважає при випромінюванні).

Насиченість кольору характеризує його «чистоту»: чим вона більша, тим колір «чистіший» (тобто більше до тонової хвилі). Нульова насиченість відповідає сірому кольору, а максимальна насиченість – найбільш яскравому варіанту даного кольору. Можна вважати, що зменшення насиченості відповідає додаванню білої фарби. У білому кольорі насиченість дорівнює 0, оскільки неможливо виділити його колірний тон.

Під яскравістю розуміється ступінь освітленості. При нульовій яскравості колір стає чорним. Максимальна яскравість при максимальній насиченості дають найбільш виразний варіант даного кольору. Можна вважати, що яскравість показує величину чорного відтінку доданого до кольору. Яскравість чорного кольору – 0, а білого – 1.

Графічно модель HSB можна представити у вигляді кільця, уздовж якого розташовуються відтінки кольорів. Кожному відтінку відповідає свій градус, тобто всього налічується 360 варіантів (червоний – 0, жовтий – 60, зелений – 120 градусів і так далі). На зовнішньому краю круга знаходяться чисті спектральні кольори або колірні тони (параметр  $H$  вимірюється в кутових градусах, від 0 до 360). Чим більше кількість кольорів, тим менше його насиченість, тим він більш блідий, пастельний (параметр  $S$  вимірюється у відсотках). Яскравість (освітленість) відображується на лінійці, перпендикулярній площині колірного круга (параметр  $B$  вимірюється у відсотках). Всі кольори на зовнішньому кружку мають максимальну яскравість.

Модель HSB не є орієнтованою ні на який технічний пристрій відтворення кольорів, тому її називають апаратно незалежною.

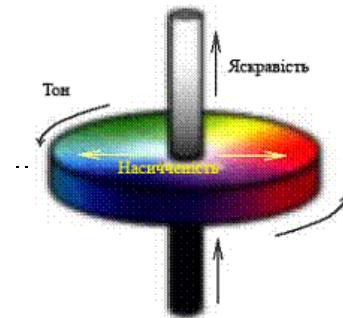
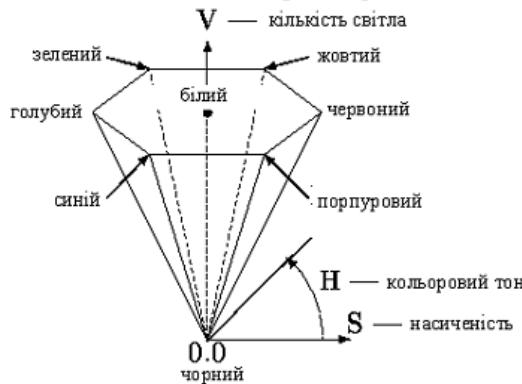


Рис. 6. Графічне представлення моделі HSB

Недолік HSB полягає в тому, що для роботи на моніторах комп’ютерів її необхідно перетворювати на систему RGB, а для чотирьох кольорового друку – в систему СМУК.

#### 1.4. Модель HSV

Модель HSV створена Елві Смітом к 1978 році. Її зручно представляти у вигляді світлової шестигранної піраміди. При цьому по вертикальній осі відкладається значення  $V$ , а відстань від осі до бічної грані в горизонтальному перетині відповідає параметру  $S$  (за діапазон зміни цих величин приймається інтервал від нуля до одиниці) (рис. 7). Шестикутник, що лежить в основі піраміди, є проекцією колірного куба в напрямку його головної діагоналі. Тон кольору  $H$  задається кутом, відкладеним навколо вертикальної осі, починаючи від червоного. Точки на самій окружності відповідають чистим (максимально насиченим) кольорам. Точка в центрі відповідає нейтральному кольору мінімальної насиченості (білий, сірий, чорний – це залежить від яскравості). Тобто можна сказати, що кут нахилу вектора визначає відтінок, довжина вектора – насиченість кольору. Величина  $S$  змінюється від нуля на осі конуса, до одиниці на його гранях. Значення  $V=0$  відповідає вершина піраміди (чорний колір), значення  $V=1$  – основа піраміди; кольори при цьому найбільш інтенсивні. Точка з координатами  $V=1$ ,  $S=0$  – центр основи піраміди (відповідає білому кольору). Проміжні значення координат  $V$  при  $S=0$  (тобто на осі піраміди) відповідають сірим кольорами, якщо  $S=0$ , то значення відтінку  $H$  вважається невизначенним,  $S=1$ , якщо точка лежить на бічній грани піраміди.



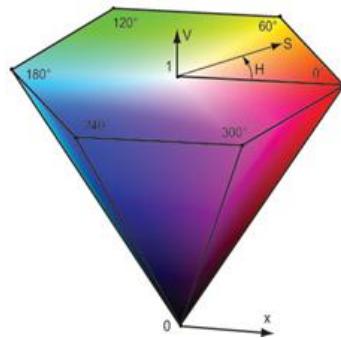


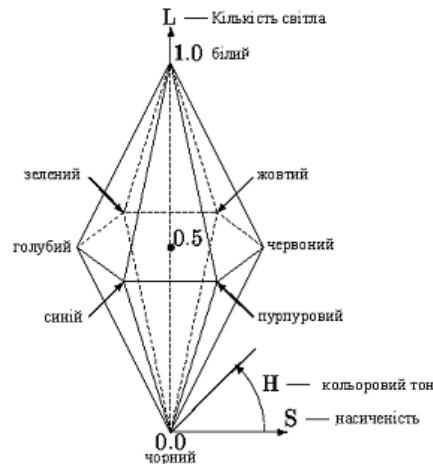
Рис. 7. Графічне представлення моделі HSV

Колірна модель HSV (тон, насиченість, значення кольору) була розроблена з метою забезпечення художника засобами інтуїтивного вибору кольору.

## 1.5. Модель HSL

Розширенням колірної моделі HSV/HSB є HLS. Параметрами якої є Hue, Lightness, Saturation, відповідно: кольоровий тон, кількість світла(освітленість), насиченість (рис. 8). Різниця між моделями HSV/HSB і HLS полягає в заміні нелінійного компоненту «яскравість» на лінійний компонент «освітлення».

В основі колірної моделі HLS лежить система Освальда. Ця модель утворює простір у формі подвійного конуса. Колірний тон задається кутом повороту навколо вертикальної осі конусів. За початок відліку прийнятий синій колір. Кольори йдуть у спектральному порядку і замикаються пурпуровим. Отже, по вертикальній осі відкладається  $L$  (освітленість), а інші два параметри задаються як і в HSV/HSB. Ця модель утворює підпростір, що представляє собою подвійний конус, у якому чорний колір задається вершиною нижнього конуса і відповідає значенню  $L=0$ , білий колір максимальної інтенсивності задається вершиною верхнього конуса і відповідає значенню  $L=1$ . Максимально інтенсивні кольорові тони відповідають основі конуса з  $L=0,5$ , що не зовсім зручно. Тон кольору  $H$ , аналогічно системі HSV/HSB, задається кутом повороту. Насиченість  $S$  змінюється в межах від 0 до 1 і задається відстанню від вертикальної осі  $L$  до бічної поверхні конуса. Тобто максимально насичені кольори розташовуються при  $L=0,5$  і  $S=1$ .



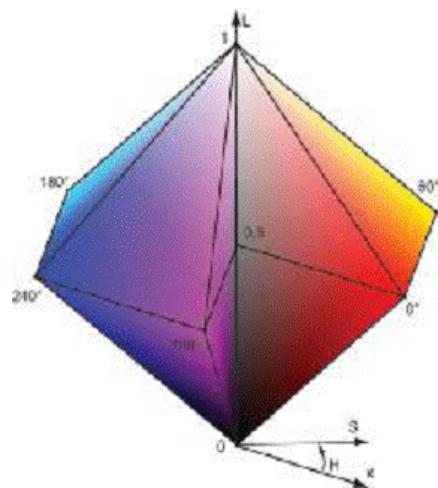


Рис. 8. Графічне представлення моделі HSL

HSL добре пристосована для опису кольорів таким чином, як це властиво людині. Дивлячись на пофарбований об'єкт, людині простіше його описати за допомогою кольору, освітленості і насыщеності, що і роблять дані колірні моделі. Безперечною перевагою даних моделей при побудові алгоритмів обробки зображень є простота розуміння, оскільки в їхній основі лежить природній і інтуїтивно близький людині опис кольору, адже саме людина в кінцевому рахунку і є розробником та користувачем цих алгоритмів.

## 1.6. Модель LAB

Модель LAB базується на людському сприйнятті кольору. При однаковій інтенсивностіоко людини сприймає промені зеленого кольору найбільш яскравими, дещо менш яскравими – червоного кольору, і ще менш яскравими – синього. Яскравість при цьому є характеристикою сприйняття, а не характеристикою самого кольору. При розробці моделі LAB переслідувалася мета математичного коректування нелінійності сприйняття кольору людиною.

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

Будь-який колір у колірній моделі LAB обумовлюється параметрами  $L$  – яскравість (Lightness) і хроматичними складовими – двома декартовими координатами:  $a$ , (змінюється від зеленого до червоного, через сірий і  $b$  (змінюється від синього до яскраво жовтого через сірий) (рис. 9).  $L$  здійснює контроль за яскравістю кольорів, утворених  $a$  і  $b$ . Білий колір співставляється з максимальною інтенсивністю.  $L$  лежать в межах 0-100,  $a$  і  $b$  в межах [-200;200]. Якщо  $a$  і  $b$  рівні 0, змінюючи  $L$ , отримуємо зображення, що містить градації сірого (grayscale).

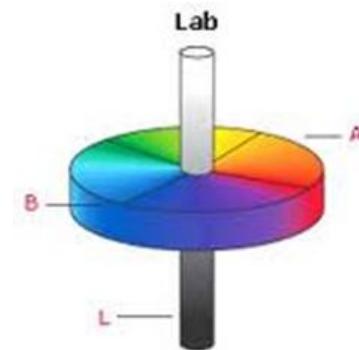


Рис. 9. Графічне представлення моделі LAB

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

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

Враховуючи позитивні характеристики, а саме величезний колірний обхват, модель LAB знайшла широке застосування в програмному забезпеченні для обробки зображень, через яку відбувається конвертація даних між іншими кольоровими просторами під час їх підготовки (наприклад, з RGB сканера в CMYK друкованого пристрою).

### 1.7. Модель XYZ

Протягом першої третини ХХ століття Міжнародна комісія з освітлення проводила дослідження фізіології людського зору, на основі яких у 1931 році була запропонована перцептивна колірна модель, що одержала назву XYZ.

Колірний простір моделі кольору XYZ являє собою криволінійний конус з вершиною в початку колірних координат. У міру віддалення від вершини освітлення кольорів відповідних точок, що лежать усередині цього конуса, зростає. Наведена на рис. 10 видима частина колірного трикутника в моделі кольору XYZ має форму сегмента неправильної параболи. На її криволінійній межі розташовуються спектрально чисті кольори, на прямолінійній хорді – кольори, одержані змішуванням червоного і пурпурного. При видаленні від межі фігури насиченість кольору зменшується і в центрі розташовується ахроматична точка.

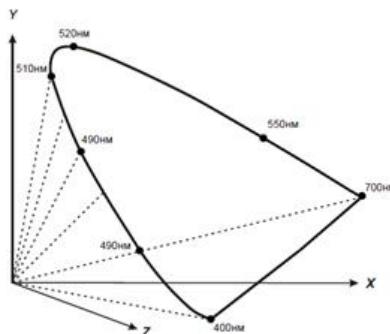


Рис. 10. Графічне представлення моделі XYZ

Оскільки працювати з об'ємним поданням колірного простору у вигляді неправильного конуса не дуже зручно, на практиці частіше користуються нормованим колірним простором, який отримав назву xyY.

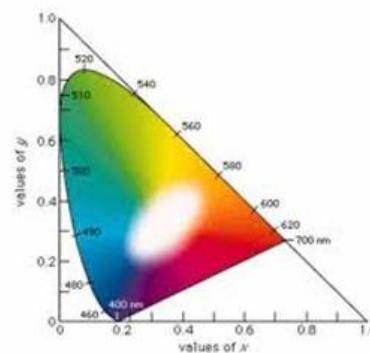


Рис. 11. Графічне представлення моделі xyY

Ця назва пояснюється тим, що в нормованому варіанті координати  $x$  і  $y$  зберігаються, а координата  $z$  зникає, оскільки цей варіант колірного простору двомірний (колірна діаграма або локус). Вона будується шляхом проектування трикутника кольору на площину  $xy$  за формулами:

$$x = \frac{X}{X + Y + Z}; y = \frac{Y}{X + Y + Z}; z = \frac{X + Y + Z}{Z}; x + y + z = 1. \quad (1)$$

Яскравість можна отримати координатою  $Y$ , а  $X$ ,  $Z$  визначити із формул:

$$X = \frac{Y}{y}x, Z = \frac{Y}{y}(1 - x - y). \quad (2)$$

Звичайно, на кольоровій діаграмі представлений не всі кольори простору XYZ, але для порівняння колірних обхватів і перетворення колірних просторів, заради яких і створювалася ця колірна модель, фактор освітленості відтінків можна не розглядати.

## 2. Перетворення моделей

Потреба перетворення зображення з однієї колірної моделі в іншу виникає досить часто, особливо якщо зображення готується для друку. До прикладу, на деяких етапах створення документа застосування колірної моделі RGB є неминучим (коли колірна модель RGB використовується сканером, цифровою камерою), але у документа, призначеного для друку, кінцевою колірною моделлю повинна бути модель CMYK. Звідси виникає необхідність конвертації моделей.

Переходи між колірними моделями завжди пов'язані з якимись втратами кольорів через невідповідність колірного простору моделей. У моделі CMYK неможливо відобразити дуже яскраві кольори моделі RGB, моделі RGB, в свою чергу, не здатна передати темні відтінки моделі CMYK, оскільки природа кольору різна. На жаль, втрати, спричинені перетворенням моделей, непоправні.

### 2.1. Перетворення моделі RGB

Конвертування моделі RGB в будь-яку іншу колірну модель виконується після нормалізації значень її червоної, зеленої та синьої складових. Для цього значення яскравості по кожній складовій переводяться з діапазону [0 .. 255] в діапазон [0 .. 1].

#### 2.1.1 Перетворення RGB в HSB (HSV)

У ході перетворення значення яскравостей по червоній, зеленій і синій складовій, які задані в діапазоні [0 .. 1], конвертуються в модель HSB (HSV). Отримують значення в наступних діапазонах:

$H$  – колірний тон (0-360°);

$S$  – насиченість (0-1);

$B (V)$  – яскравість (0-1).

$Max$  – функція визначення максимуму серед трьох складових  $R$ ,  $G$ , і  $B$ .

$Min$  – функція визначення мінімуму серед трьох складових  $R$ ,  $G$ , і  $B$ .

Алгоритм перетворення RGB в HSB (HSV) такий:

$$H = \begin{cases} \text{не визначено, якщо } Max = Min; \\ 60^\circ \times \frac{G - B}{Max - Min} + 0^\circ, \text{ якщо } Max = R \text{ і } G \geq B; \\ 60^\circ \times \frac{G - B}{Max - Min} + 360^\circ, \text{ якщо } Max = R \text{ і } G < B; \\ 60^\circ \times \frac{B - R}{Max - Min} + 120^\circ, \text{ якщо } Max = G; \\ 60^\circ \times \frac{R - G}{Max - Min} + 240^\circ, \text{ якщо } Max = B. \end{cases} \quad (3)$$

$$S = \begin{cases} 0, \text{ якщо } Max = 0; \\ 1 - \frac{Min}{Max}, \text{ в інших випадках} \end{cases}$$

$$V = Max. \quad (4)$$

#### 2.1.2 Перетворення RGB в HSL

Початкові умови, діапазони змін і позначення аналогічні попередньому пункту. Алгоритм перетворення наведено нижче.

$$H = \begin{cases} \text{не визначено, якщо } Max = Min; \\ 60^\circ \times \frac{G - B}{Max - Min} + 0^\circ, \text{ якщо } Max = R \text{ і } G \geq B; \\ 60^\circ \times \frac{G - B}{Max - Min} + 360^\circ, \text{ якщо } Max = R \text{ і } G < B; \\ 60^\circ \times \frac{B - R}{Max - Min} + 120^\circ, \text{ якщо } Max = G; \\ 60^\circ \times \frac{R - G}{Max - Min} + 240^\circ, \text{ якщо } Max = B. \end{cases} \quad (5)$$

$$S = \begin{cases} 0, \text{ якщо } L = 0 \text{ або } Max = Min; \\ \frac{Max - Min}{Max + Min} = \frac{Max - Min}{2L}, \text{ якщо } 0 < L < \frac{1}{2}; \\ \frac{Max - Min}{2 - (Max + Min)} = \frac{Max - Min}{2 - 2L}, \text{ якщо } L > \frac{1}{2}. \end{cases}$$

$$L = \frac{1}{2}(Max + Min). \quad (6)$$

### 2.1.3 Перетворення RGB в CMYK

Перед конвертацією значення яскравостей за червоною, зеленою та синьою складовою нормалізуються. Основний принцип перетворення моделей полягає в наступному:

$$CMY = \{1 - R, 1 - G, 1 - B\};$$

$$K = \min\{C, M, Y\};$$

$$CMYK = \{0, 0, 0, 1\}, \text{ якщо } K = 1;$$

$$CMYK = \{(C - K)/(1 - K), (M - K)/(1 - K), (Y - K)/(1 - K), K\}.$$

### 2.1.4 Перетворення RGB в XYZ

Як і в попередньому підпункті, перед конвертацією значення яскравостей по червоній, зеленій та синій компонентах нормалізуються. Основний принцип перетворення полягає в наступному ( $a = 0.055$ ):

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{1}{0.1769} \begin{bmatrix} 0.4900 & 0.3100 & 0.2000 \\ 0.1769 & 0.8124 & 0.0106 \\ 0.0000 & 0.0100 & 0.9900 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \quad (7)$$

### 2.2. Перетворення моделі HSB в RGB

Діапазони зміни величин є такими:

- величина  $H$  може приймати значення в діапазоні  $[0, 360]$ ;
- величини  $S, V, R, G, B$  - в діапазоні  $[0, 1]$ .

Конвертація колірних моделей виконується згідно з такими правилами:

$$H_i = [H / 60] \bmod 6;$$

$$f = (H / 60) - H_i;$$

$$p = V(1 - S);$$

$$q = V(1 - fS);$$

$$t = V(1 - (1 - f)S);$$

$$\text{якщо } H_i = 0 \Rightarrow R = V, G = t, B = p;$$

$$\text{якщо } H_i = 1 \Rightarrow R = q, G = V, B = p;$$

якщо  $H_i = 2 \Rightarrow R = p, G = V, B = t;$

якщо  $H_i = 3 \Rightarrow R = p, G = q, B = V;$

якщо  $H_i = 4 \Rightarrow R = t, G = p, B = V;$

якщо  $H_i = 5 \Rightarrow R = V, G = p, B = q.$

### 2.3. Перетворення моделі HSL в RGB

Діапазони зміни величин є такими:

- величина  $H$  може приймати значення в діапазоні  $[0, 360];$
- величини  $S, L, R, G, B$  – у діапазоні  $[0, 1].$

Конвертація колірних моделей виконується згідно із такими правилами:

$$C = (1 - |2L - 1|) \times S_{\text{HSL}}; H' = \frac{H}{60}; X = C(1 - |H' \bmod 2 - 1|).$$

$$(R_1, G_1, B_1) = \begin{cases} (0, 0, 0), \text{ якщо } H \text{ не визначене}; \\ (C, X, 0), \text{ якщо } 0 \leq H' < 1; \\ (X, C, 0), \text{ якщо } 1 \leq H' < 2; \\ (0, C, X), \text{ якщо } 2 \leq H' < 3; \\ (0, X, C), \text{ якщо } 3 \leq H' < 4; \\ (X, 0, C), \text{ якщо } 4 \leq H' < 5; \\ (C, 0, X), \text{ якщо } 5 \leq H' < 6; \end{cases}$$

(8)

$$m = L - \frac{1}{2}C; (R, G, B) = (R_1 + m, G_1 + m, B_1 + m).$$

(9)

### Контрольні питання

1. Що таке колір?
2. Яка різниця між хроматичними та ахроматичними кольорами?
3. Розкрийте поняття колориметрія.
4. Що таке колірна модель та колірний діапазон?
5. Чому колірна модель RGB є адитивною? Які недоліки моделі RGB?
6. Яка різниця між адитивною та субтрактивною колірними моделями?
7. Як утворюється модель CMYK із моделі RGB?
8. Що таке яскравість, насиченість і тон у моделі HSB?
9. Які недоліки та переваги моделі HSB?
10. В чому різниця моделі HSB від HSV?
11. Яка система лежить в основі моделі HSL? У чому її суть?
12. У чому переваги моделі LAB?
13. Яке графічне представлення колірної моделі XYZ?

### Варіанти завдань

Написати програму переведення однієї колірної системи в іншу. Забезпечити зміну певного компоненту моделі в залежності від варіанту.

1. Перетворити RGB в HSL. Змінити освітленість певного кольору.
2. Перетворити RGB в HSV. Змінити насиченість зображення.
3. Перетворити RGB в CMYK і навпаки. Змінити яскравість по жовтій компоненті кольору.
4. Перетворити RGB в XYZ. Змінити яскравість зображення.

**Укладачі** Є.В. Левус, доцент кафедри ПЗ

О.О. Нитребич, асист. кафедри ПЗ

**Відповідальний за випуск** Федасюк Д.В., д-р тех. наук, проф.

**Рецензенти** Левченко О.М, канд. тех. наук, асистент.

Марусенкова Т.О, канд. техн. наук, асистент.

**Тема.** Використання афінних перетворень на площині для генерування рухомих зображень

**Мета.** Навчитись програмувати основні перетворення на площині – зсув, масштаб та поворот об'єктів для генерування рухомих зображень.

Теоретичні відомості

Виконання різноманітних дій з геометричними об'єктами є центральною задачею в комп'ютерній графіці. Тому вибір математичних методів і алгоритмів для її реалізації суттєво впливає на ефективність цілої графічної системи. У сучасній комп'ютерній графіці досить широко використовується метод координат, оскільки графічне зображення складається з пікселів, які задаються координатами. Крім цього, координати використовуються для опису розміщення об'єктів та для створення зображень шляхом перетворень з однієї системи координат в іншу.

## 1. Найпростіші двовимірні афінні перетворення

Афінним називається перетворення, що має такі властивості:

- будь-яке афінне перетворення може бути представлене як послідовність операцій з числа найпростіших: зсув, розтягнення/стиснення, поворот;
- зберігаються прямі лінії, паралельність прямих, відношення довжин відрізків, що лежать на одній прямій, і відношення площ фігур.

**Афінні перетворення координат на площині:**

$(X, Y)$  – двовимірна система координат,

$(x, y)$  – координати старої системи в новій системі координат.

$$\begin{cases} X = Ax + By + C \\ Y = Dx + Ey + F \end{cases}.$$

Загальний вигляд афінного перетворення:

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

де  $A, B, C, D, E, F$  – константи.

Обернене перетворення теж є афінним:

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

*Найпростіші афінні перетворення об'єкта на площині.*

1. Здиг об'єкта:

$$\begin{cases} X = x + dx \\ Y = y + dy \end{cases}$$

або матрицею:  $\begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix}$ .

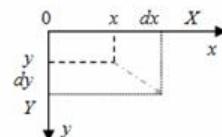


Рис. 1 Зсув об'єкта

Обернене перетворення задається формулами :

$$\begin{cases} x = X - dx \\ y = Y - dy \end{cases}$$

або відповідною матрицею:

$$\begin{bmatrix} 1 & 0 & -dx \\ 0 & 1 & -dy \\ 0 & 0 & 1 \end{bmatrix}.$$

Примітка! У літературі часто використовують іншу матрицю зсуву координат об'єкта:

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -dx & -dy & 1 \end{bmatrix}.$$

З використанням даної матриці, нові координати об'єкту утворюються внаслідок множення старих координат на дану матрицю  $[X, Y, 1] = [x, y, 1] \cdot M$ .

2. Масштабування об'єкту:

$$\begin{cases} X = xk_x \\ Y = yk_y \end{cases},$$

а в матричній формі:

$$\begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

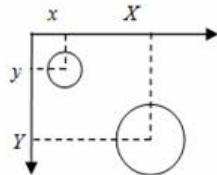


Рис. 2 Масштабування об'єкта

Обернене перетворення задане:

$$\begin{cases} x = X/k_x \\ y = Y/k_y \end{cases},$$

та відповідно матрицею

$$\begin{bmatrix} 1/k_x & 0 & 0 \\ 0 & 1/k_y & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Коефіцієнти можуть бути від'ємними. Наприклад, при  $k_x = -1$  отримуємо дзеркальне відображення відносно осі  $Y$ .

Також варто зауважити, що у вище згаданих формулах йдеться про пропорційне зміщення всіх сторін об'єкту.

Якщо взяти загальну матричну перетворення  $\begin{bmatrix} k_{x_1} & k_{y_1} & 0 \\ k_{x_2} & k_{y_2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$ , то при  $k_{x_1} = k_{y_1}$ , а  $k_{x_2} = k_{y_2} = 0$  виконується пропорційне масштабування, при  $k_{x_1} \neq k_{y_1}$ , а  $k_{x_2} = k_{y_2} = 0$  – непропорційне.

*Примітка!* У даних формулах йде мова про масштабування об'єкту та його переміщення. Якщо потрібно зробити чисте масштабування без ефекту переміщення, тоді варто об'єкт помістити в центр координат.

3. Поворот об'єкта відносно центру координат на кут  $\alpha$  відповідає системі рівнянь:

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

або задається матрицею

$$\begin{bmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

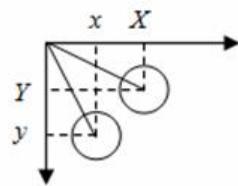


Рис. 3 Поворот об'єкта

Обернене перетворення – поворот на кут  $(-\alpha)$  задане системою рівнянь:

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

та відповідною матрицею

$$\begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

**Увага!** У всіх афінних перетвореннях, описаних вище, використовуються однорідні координати – це координати неоднорідного вектора  $[x, y]$ , що є трійкою  $[x', y', h']$ , де  $x = x' / h'$ ,  $y = y' / h'$ ,  $h'$  – деяке дійсне число. Зазвичай використовують вектор  $[x, y, 1]$ . При бажанні можна користуватись звичайними координатами, але формули матимуть наступний матричний вигляд:

$$\text{зсув} - K^* = K + \begin{bmatrix} dx & 0 \\ 0 & dy \end{bmatrix},$$

$$\text{масштаб: } K^* = K \times \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix},$$

$$\text{поворот: } K^* = K \times \begin{bmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{bmatrix},$$

де  $K$  – матриця координат об'єкту до перетворення,  $K^*$  – після перетворення.

## 2. Тривимірні перетворення

Подібно до двовимірних афінних перетворень, під час роботи з якими з використанням однорідних координат, здійснюють дії над матрицями 3x3, тривимірні перетворення можуть бути представлені матрицею 4x4.

Отже, аналогічно двовимірному випадку матриця зсуву матиме вигляд:

$$M = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

а нові координати обчислюватимуться за формулою  $[X, Y, Z, 1] = [x, y, z, 1] \cdot M$ .

Матриця масштабування –

$$S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

тоді нові координати обчислюватимуться за формулою  $[X, Y, Z, 1] = [x, y, z, 1] \cdot S$ .

Стосовно матриці повороту, то в залежності від вибраної осі матимемо такі матриці:

- поворот відносно осі OZ:

$$P = \begin{bmatrix} \cos \alpha & \sin \alpha & 0 & 0 \\ -\sin \alpha & \cos \alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

- поворот відносно осі OX:

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha & \sin \alpha & 0 \\ 0 & -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

- поворот відносно осі OZ:

$$P = \begin{bmatrix} \cos \alpha & 0 & -\sin \alpha & 0 \\ 0 & 1 & 0 & 0 \\ \sin \alpha & 0 & \cos \alpha & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Для обчислення нових координат застосовується формула  $[X, Y, Z, 1] = [x, y, z, 1] \cdot P$ .

### 3. Визначення матриці перетворень

Складним (комбінованим) називається перетворення, яке містить ланцюжок базових перетворень (не менше двох). Зауважимо, що майже всі афінні перетворення залежать від порядку їх виконання. Наприклад, перетворення повороту не комутативні ( $x \cdot y \neq y \cdot x$ ).

Найчастіше на практиці для визначення матриці перетворень використовують рівняння:

$$M = K^{-1} K^*,$$

що випливає із рівняння афінних перетворень  $K^* = K \cdot M$ , де  $K$  – матриця координат об'єкту до перетворення,  $K^*$  – після перетворення,  $M$  – матриця перетворення.

**Приклад 1.** Знайти матрицю перетворення трикутника  $P_1P_2P_3$  у трикутник  $P'_1P'_2P'_3$  (рис. 4)

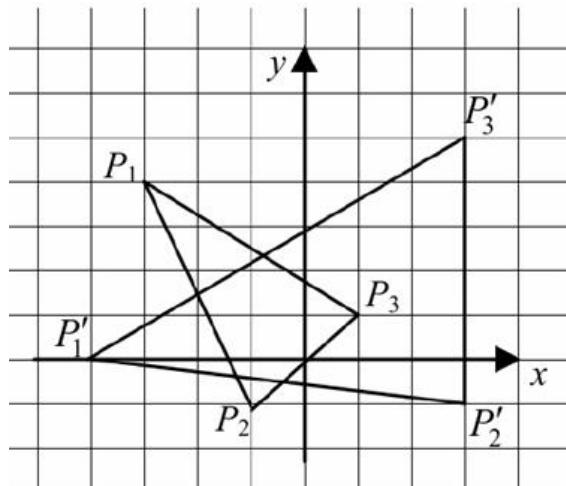


Рис. 4 Перетворення трикутників

Однорідні координати вершин трикутника  $P_1P_2P_3$  складають матрицю

$$K = \begin{bmatrix} -3 & 4 & 1 \\ -1 & -1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \text{ а трикутника } P_1'P_2'P_3' - K^* = \begin{bmatrix} -4 & 0 & 1 \\ 3 & -1 & 1 \\ 3 & 5 & 1 \end{bmatrix}.$$

Зі співвідношення  $M = K^{-1}K^*$  отримаємо:  $M = \begin{bmatrix} 1 & 2 & 0 \\ -1 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix}.$

#### 4. Деякі правила виконання перетворень

Для представлення даних та виконання різноманітних перетворень використовуються певні узгодження. Найбільшу увагу потрібно приділяти формуллюванню завдання та інтерпретації результатів. Наприклад, перед виконанням повороту необхідно отримати відповіді на наступні питання:

- У правосторонній чи лівосторонній системі координат (у залежності від взаємного розміщення напрямків координатних осей, розрізняють правосторонню та лівосторонню координатні системи, рис. 4) визначаються координатні вектори?
- Обертається об'єкт або система координат?
- Поворот здійснюється за чи проти годинникової стрілки?
- Координати записуються у вигляді рядка або стовпця матриці?
- Навколо якої лінії або осі здійснюється поворот?

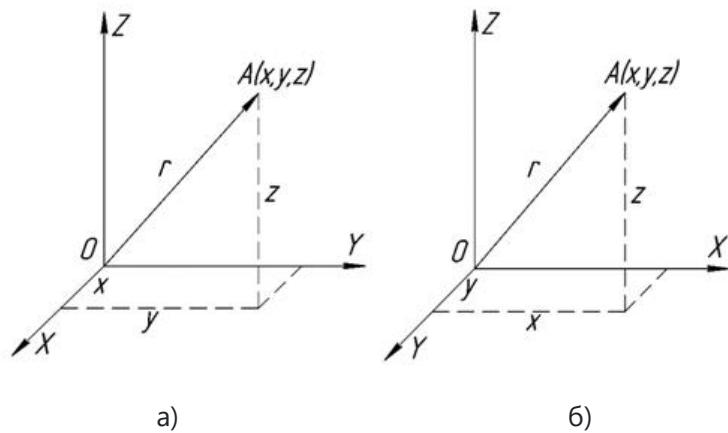


Рис.5. Типи систем координат а) правостороння; б) лівостороння.

Нехай маємо правосторонню систему координат, об'єкт обертається в нерухомій координатній системі, поворот визначається правилом правої руки (тобто поворот за годинниковою стрілкою) і координатні вектори представляються у вигляді рядка матриці.

Так як вектор задається рядком матриці, то матрицю перетворення слід розмістити після даних (матриці координатних векторів). Це перетворення задається шляхом множення справа. У разі однорідних координат для повороту об'єкта за годинникової стрілкою на кут навколо початку координат (осі  $OZ$ ) використання множення праворуч приводить до наступного результату:

$$[X, Y, 1] = [x, y, 1] \begin{bmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Якщо ми виразимо координатні вектори, задані в однорідних координатах у вигляді стовпця матриці, то поворот можна виконати наступним чином:

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

Для того, щоб повернути систему координат і залишити незмінними координатні вектори, необхідно кут  $\alpha$  замінити на  $-\alpha$ .

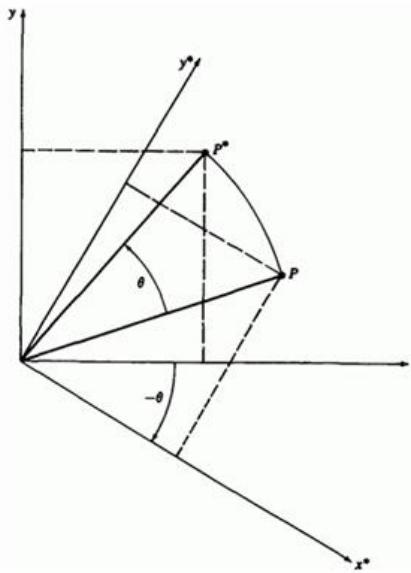


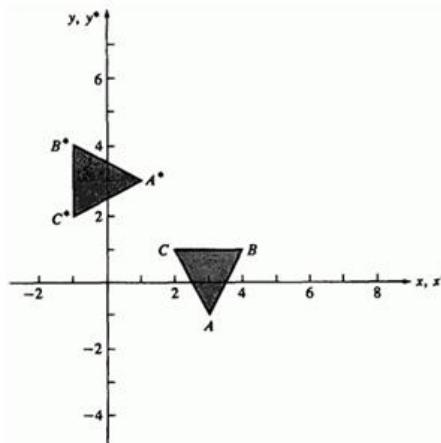
Рис. 6. Перетворення координатних векторів та системи координат

## 5. Приклади афінних перетворень

**Приклад 1.** Повернути трикутник ABC з координатами (3;-1), (4;1), (2;1) на  $90^\circ$  проти годинникової стрілки відносно початку координат.

Використовуючи матрицю координат  $3 \times 2$  та формули повороту отримаємо:

$$\begin{bmatrix} 3 & -1 \\ 4 & 1 \\ 2 & 1 \end{bmatrix} \times \begin{bmatrix} \cos 90^\circ & \sin 90^\circ \\ -\sin 90^\circ & \cos 90^\circ \end{bmatrix} = \begin{bmatrix} 3 & -1 \\ 4 & 1 \\ 2 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ -1 & 4 \\ -1 & 2 \end{bmatrix}.$$

Рис. 7. Поворот трикутника ABC в  $A^*B^*C^*$ 

**Приклад 2.** Відобразити трикутник DEF з координатами (8;1), (7;3), (6;2) спочатку відносно осі  $Y=0$  у трикутник  $D^*E^*F^*$ , а потім відносно прямої  $x=y$  у трикутник  $D^+E^+F^+$ .

Відображення відносно осі  $Y=0$ :

$$\begin{bmatrix} 8 & 1 \\ 7 & 3 \\ 6 & 2 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 8 & -1 \\ 7 & -3 \\ 6 & -2 \end{bmatrix}.$$

Відображення відносно прямої  $x=y$ :

$$\begin{bmatrix} 8 & 1 \\ 7 & 3 \\ 6 & 2 \end{bmatrix} \times \begin{bmatrix} 1 & 8 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 8 \\ 3 & 7 \\ 2 & 6 \end{bmatrix}.$$

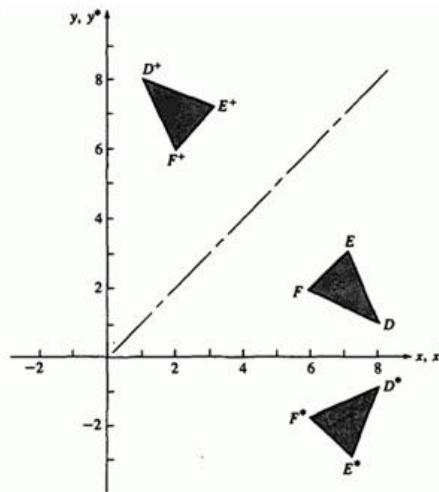


Рис. 8. Відображення трикутника DEF

### Приклад 3.

Нехай задано трикутник ABC з координатами (2;2), (4;2), (4;4). Знайти координати нового трикутника, повернутого на  $90^\circ$  відносно початку координат та відображеного відносно прямої  $y=-x$ .

Перша матриця повороту має вигляд:

$$P = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Матриця відображення відносно  $y = -x$  відповідно рівна:

$$V = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

Результатом повороту та відображення координат  $K$  трикутника  $ABC$  будуть координати  $K^*$ :

$$K^* = V \times P \times K$$

$$\begin{bmatrix} 2 & 2 \\ 4 & 2 \\ 4 & 4 \end{bmatrix} \times \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \times \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} -2 & 2 \\ -4 & 2 \\ -4 & 4 \end{bmatrix}.$$

Якщо провести перетворення в оберненому порядку (спочатку відображення, а потім поворот), то отримаємо трикутник  $D^*E^*F^*$ :

$$\begin{bmatrix} 2 & 2 \\ 4 & 2 \\ 4 & 4 \end{bmatrix} \times \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \times \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & -2 \\ 4 & -2 \\ 4 & -4 \end{bmatrix}.$$

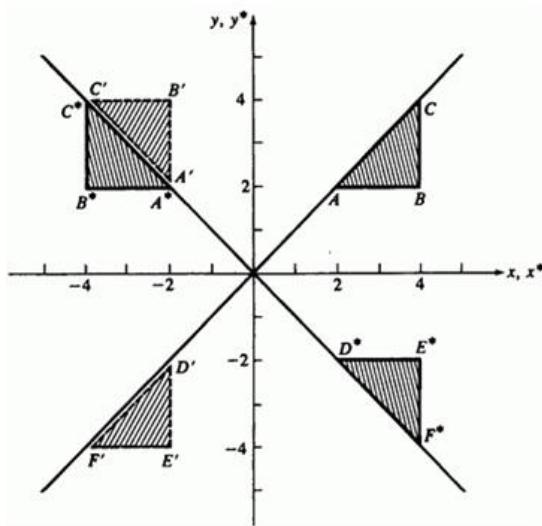


Рис. 9. Комбінований поворот та відображення трикутника  $ABC$  ( $A^+B^+C^+$  – проміжний трикутник прямої задачі,  $D^+E^+F^+$  – проміжний трикутник оберненої задачі)

**Приклад 4.** Побудувати матрицю повороту точки  $M(x, y)$  відносно довільної точки  $N(m, n)$  на кут  $\phi$  у додатному напрямку.

Однорідні координати дають можливість знайти матрицю повороту відносно довільної точки. У загальному випадку поворот відносно довільної точки може бути реалізований шляхом таких перетворень:

- 1) переміщення точки  $N(m, n)$  на вектор  $(-m, -n)$  так, щоб центр повороту сумістився з початком координат. Матриця цього перетворення має вигляд:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -m & -n & 1 \end{bmatrix}$$

2) поворот точки на кут  $\varphi$  у додатному напрямку відносно початку координат. Матриця цього перетворення визначається формулою:

$$\begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Отже, для знаходження результуючого повороту точки  $M(x, y)$  відносно точки  $N(m, n)$  потрібно перемножити задані матриці за вказаним порядком:

$$[X, Y, 1] = [x, y, 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -m & -n & 1 \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{bmatrix}.$$

**Приклад 5.** Нехай задано рівняння прямої  $L: y = \frac{1}{2}(x+4)$  та трикутник ABC з координатами вершин (2,4,1), (4,6,1), (2,6,1). Дзеркально відобразити трикутник відносно даної прямої.

Пряма  $L$  пройде через початок координат під час зсуву її на 2 одиниці по осі  $OY$  (матриця зсуву матиме вигляд  $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix}$ ). В результаті повороту навколо початку координат на  $-tg^{-1}(\frac{1}{2}) = -26.57^\circ$  пряма співпаде з

віссю  $OX$  (матриця повороту матиме вигляд  $\begin{bmatrix} 2/\sqrt{5} & -1/\sqrt{5} & 0 \\ 1/\sqrt{5} & 2/\sqrt{5} & 0 \\ 0 & 0 & 1 \end{bmatrix}$ ). Далі необхідно дзеркально відобразити

об'єкт за допомогою матриці дзеркального відображення  $\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$  і повернувшись в початкову орієнтацію.

Комбінація перетворень матиме вигляд:

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix} \begin{bmatrix} 2/\sqrt{5} & -1/\sqrt{5} & 0 \\ 1/\sqrt{5} & 2/\sqrt{5} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2/\sqrt{5} & 1/\sqrt{5} & 0 \\ -1/\sqrt{5} & 2/\sqrt{5} & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 3/5 & 4/5 & 0 \\ 4/5 & -3/5 & 0 \\ -8/5 & 16/5 & 1 \end{bmatrix}.$$

Отже, координати нового трикутника матимуть будуть такими:

$$\begin{bmatrix} 2 & 4 & 1 \\ 4 & 6 & 1 \\ 2 & 6 & 1 \end{bmatrix} \begin{bmatrix} 3/5 & 4/5 & 0 \\ 4/5 & -3/5 & 0 \\ -8/5 & 16/5 & 1 \end{bmatrix} = \begin{bmatrix} 14/5 & 12/5 & 1 \\ 28/5 & 14/5 & 1 \\ 22/5 & 6/5 & 1 \end{bmatrix}.$$

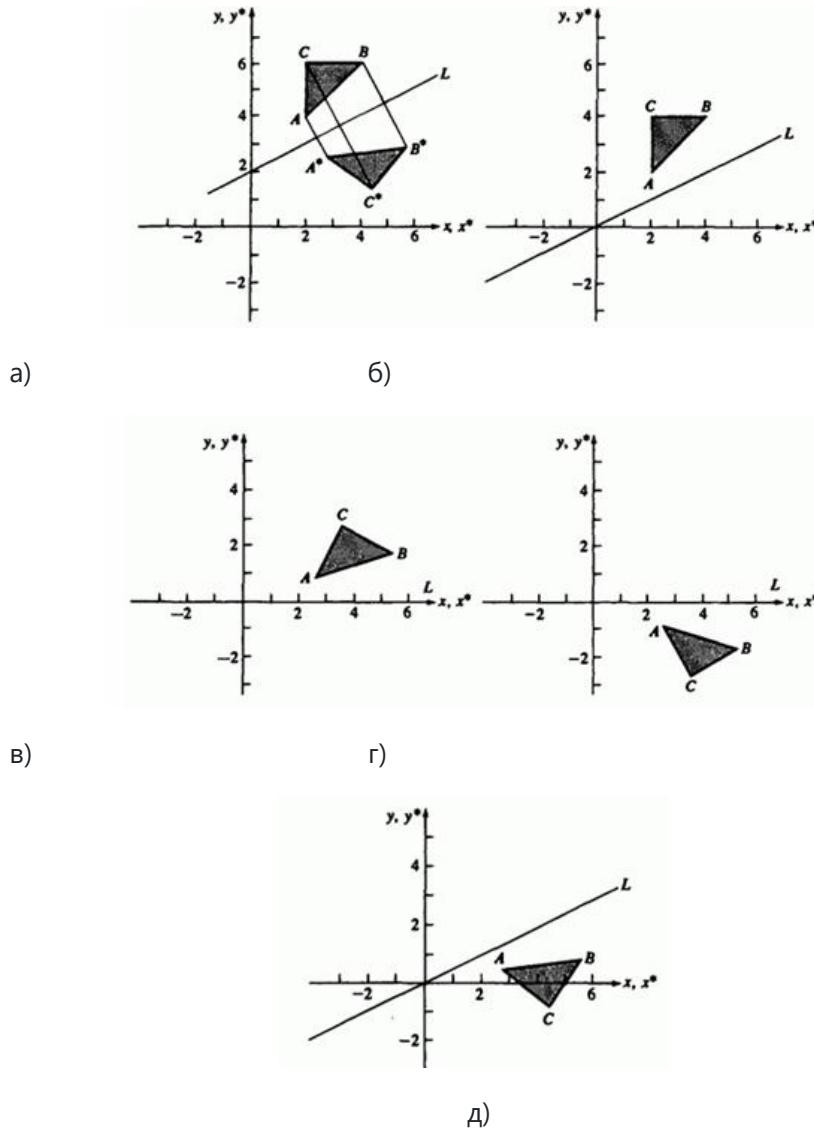


Рис 10. Відображення відносно будь-якої кривої

- початкове та кінцеве положення трикутника;
- зсув прямої в початок координат;
- поворот прямої та її спів падіння з віссю  $OX$ ;
- відображення відносно осі  $OX$ ; д) обернений поворот.

### Висновки

Ідея опису точки вектором виникла з геометричних уявлень. Теореми геометрії розвивалися для афінної геометрії з часом. У них важливими є поняття паралельності і співвідношення між паралельними прямими.

Афінне перетворення є комбінацією лінійних перетворень, супроводжуваних переносом зображень. Афінні перетворення формують зручну підсистему білінійних перетворень, тому що добуток двох афінних перетворень також є афінним. Це дозволяє представити узагальнену орієнтацію системи точок стосовно довільної координатної системи при збереженні одиничного значення однорідної координати  $h$ . Отже, афінні перетворення найбільш часто використовуються в комп'ютерній графіці. І як було показано, вони значно спрощують масштабування, поворот і зсув зображень.