МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ „ЛЬВІВСЬКА ПОЛІТЕХНІКА”

ПОБУДОВА ДВОВИМІРНИХ ФІГУР ЗАСОБАМИ МОВИ ПРОГРАМУВАННЯ

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторної роботи №1
з дисципліни „Комп’ютерна графіка”
для здобувачів першого (бакалаврського) рівня вищої освіти
спеціальності 121, F2 «Інженерія програмного забезпечення»

Затверджено
на засіданні кафедри
програмного забезпечення

Львів – 2026

Побудова двовимірних зображень: Методичні вказівки до виконання лабораторної роботи №1 із дисципліни „Комп’ютерна графіка” для здобувачів першого (бакалаврського) рівня вищої освіти спеціальності F2, 121 «Інженерія програмного забезпечення»/ Укл.: Є.В. Левус – Львів: Національний університет „Львівська політехніка”, 2026. – 22 с.

Укладач Є.В. Левус, доцент кафедри ПЗ

Відповідальний за випуск Федасюк Д.В., д-р тех. наук, проф.

Тема. Програмування геометричних фігур у двовимірній декартовій системі координат засобами графічних примітивів.

Мета. Ознайомитися з використанням базових графічних примітивів у вибраній мові програмування, реалізувати побудову геометричних фігур у декартовій системі координат, навчитися працювати з математичними координатами та структурою даних для зберігання параметрів фігур.

Теоретичні відомості

1. Вступ до комп'ютерної графіки. Формування комп'ютерної графіки (надалі – КГ) як самостійного напрямку інформаційних технологій розпочалось в 60-х роках XX ст., коли Сазерленд створив спеціалізований програмний пакет машинної графіки. Уперше відображення інформаційних даних на екрані комп'ютера в графічному вигляді було продемонстровано на початку 50-х років спеціалістами Массачусетського технологічного інституту і згодом стало використовуватися в наукових і військових дослідженнях. За цей час комп'ютерна графіка пройшла шлях від окремих експериментів до одного з найважливіших інструментів сучасності. КГ розвивається з дуже великою швидкістю, адже у теперішніх умовах не можна уявити досягнення науки, техніки, індустрії розваг, освіти, торгівлі без використання КГ. Вона є багатофункціональною складовою усіх інформаційних технологій і важливою компонентою для взаємодії людини з комп'ютером.

На початку свого розвитку комп'ютерна графіка була пасивною. Незважаючи на такий суттєвий недолік, це вже забезпечувало наочну форму сприйняття інформації, а, як відомо, більшу частину інформації (до 75%) про навколишній світ людина сприймає візуально. Недарма говорять: «Краще раз побачити, ніж сто разів почути», «Один рисунок вартий тисячі слів». Зображення – не тільки місткий, але і доступний вид інформації, оскільки для сприйняття візуальної інформації від користувача вимагається менше зусиль. Інформація, що міститься у зображеннях, є найбільш концентрованою, найлегше сприймається, запам'ятовується та найшвидше обробляється.

Під пасивною графікою розуміється формування і вивід зображення на графічний пристрій під управлінням прикладних програм без втручання користувача. Пасивний контроль - це, коли система працює в пакетному режимі без діалогу з користувачем. Для модифікації вихідного зображення необхідно знову запустити прикладну програму. Тому з часом виникла необхідність у розвитку діалогових систем КГ, тобто систем інтерактивної комп'ютерної графіки, де користувач безпосередньо зміг би оперативно вносити зміни в процесі відтворення зображення (в реальному масштабі часу).

На сьогодні всі сучасні графічні програми є системами інтерактивної КГ і вони можуть відтворювати всі особливості реальних зображень. Серед найпопулярніших сфер застосування технологій комп'ютерної графіки – комп'ютерні ігри, системи віртуальної, доповненої реальності, спецефекти у кінематографії і телебаченні, обробка цифрових фотографій, візуалізація об'єктів проектування, графічний інтерфейс користувача.

Комп'ютерна графіка – це галузь знань, що вивчає та розробляє методи і засоби синтезу збереження й перетворення їх за допомогою комп'ютера та інших технічних пристроїв. Особливістю КГ є те, що, з одного боку, накопичено значний багаж знань, а з іншого боку, здійснюється постійний розвиток методів, алгоритмів та практичних застосувань, це складна і різноманітна дисципліна.

Окремо у межах КГ можна розглядати задачі трьох напрямків: візуалізацію, обробку і розпізнавання зображень.

Візуалізація – створення зображення на основі опису (моделі) деякого об'єкту. Існує велика кількість методів і алгоритмів візуалізації, які відрізняються між собою залежно від того, що і як має бути відображено: графік функції, діаграма, схема, карта або імітація тривимірної реальності – зображення сцен у комп'ютерних розвагах, художніх фільмах, тренажерах, в системах архітектурного проектування. Важливими і пов'язаними між собою факторами тут є: швидкість зміни кадрів, насиченість сцени об'єктами, якість зображення, врахування особливостей графічного пристрою .

Обробка зображень – це перетворення зображень (тобто вхідними даними є зображення і результат – теж зображення). Прикладами обробки зображень можуть бути підвищення контрасту, чіткості, корекція кольорів, редукція кольорів, згладжування, зменшення шумів і т.д. Матеріалами обробки можуть бути космічні знімки, скановані зображення, радіолокаційні, інфрачервоні зображення і т.д. Основними завданнями обробки зображень є покращення зображення в залежності від певного критерію (реставрація, відновлення) та спеціальне перетворення, що кардинально змінює зображення. В останньому випадку обробка зображень може бути проміжним етапом для подальшого його розпізнавання.

Наприклад, перед розпізнаванням часто необхідно виділяти контури, створювати бінарне зображення, розділяти вихідне зображення за кольорами. Методи обробки зображення можуть істотно відрізняються в залежності від того, яким чином воно отримане: синтезовано системою КГ, отримано в результаті оцифровки чорно-білої або кольорової фотографії.

Розпізнавання зображень полягає в отриманні опису зображених об'єктів. Методи та алгоритми розпізнавання розроблялися, перш за все, для

забезпечення зору роботів і для систем спеціального призначення. Але останнім часом комп'ютерні системи розпізнавання зображень все частіше з'являються в повсякденній практиці, наприклад, офісні системи розпізнавання текстів чи програми векторизації. Для отримання ефективного і точного результату задачі розпізнавання зображень застосовують методи штучного інтелекту.

2. Види комп'ютерної графіки. За різними критеріями КГ поділяють на різні види (за сферою застосування, способом поданням кольору, методом відображення, вимірністю простору відображення, принципом формування зображення тощо).

За принципом формування (побудова і кодування) зображення КГ поділяють на такі основні види:

- векторну,
- растрову,
- фрактальну,
- тривимірну графіку.

У векторній графіці базовим елементом є лінія (при цьому неважливо, пряма це лінія чи крива, замкнена чи ні). Векторний рисунок (Рис.1) – це сукупність ліній (векторів).

Рис. 1. Приклад векторного зображення

Зрозуміло, що в альтернативній графіці – растровій (точковій) теж існують лінії, але там вони розглядаються як комбінації точок. Чим довша растрова лінія, тим більше пам'яті вона займає. У векторній графіці об'єм пам'яті, що займає лінія, не залежить від розмірів лінії, оскільки вона представляється у вигляді команди з кількома параметрами. Що б не робити з цією лінією, змінюються тільки її параметри, які зберігаються в комірках пам'яті. Кількість же комірок залишається незмінною для будь-якої лінії. Найпростіші об'єкти у векторній КГ об'єднуються в більш складні (наприклад, об'єкт чотирикутник можна розглядати як чотири пов'язані лінії, а об'єкт куб ще більш складний: його можна розглядати або як 12 пов'язаних ліній, або як 6 пов'язаних

чотирикутників). Через такий підхід векторну графіку часто називають об'єктно-орієнтованою. Програмні засоби для роботи з векторною графікою призначені, у першу чергу, для створення ілюстрацій та їх редагування, а точніше - перетворень геометрії зображень. Такі засоби широко використовують у рекламних агентствах, дизайнерських бюро, для технічної документації, інженерних креслень. Існують і приклади високохудожніх творів, створених засобами векторної графіки, але вони скоріше виключення, ніж правило, оскільки художня підготовка ілюстрацій засобами векторної графіки надзвичайно складна.

Основним елементом растрового зображення є точка, піксель (picture element). Растрове зображення – це матриця пікселів (Рис.2). Кожен піксель має свій колір, відповідно, растрові зображення призначені для опрацювання кольорових характеристик зображень.

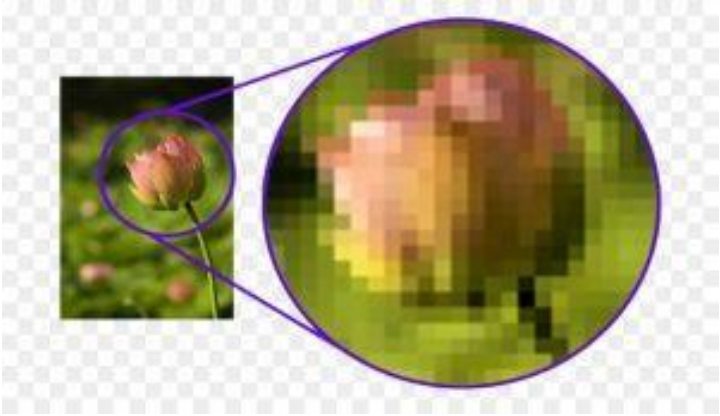

Рис. 2. Приклад растрового зображення

Із растровим зображенням безпосередньо пов'язана його роздільна здатність. Цей параметр вимірюється в кількості точок на дюйм (dots per inch – dpi). dpi вказує на відстань між пікселями, тобто наскільки якісним (чітким, виразним) буде зображення. Ілюстрації, що належать до растрової графіки, рідко створюють вручну за допомогою комп'ютерних програм. Частіше для цієї мети використовують цифрові фотографії, сканують ілюстрації, підготовлені художником на папері. Відповідно, графічні редактори растрової графіки призначені для опрацювання зображень, а саме, в основному – для роботи з кольорами, а не геометрією об'єктів.

У фрактальній графіці основним елементом є фрактал – структура, яка складається з частин, що в деякому розумінні подібні цілому.

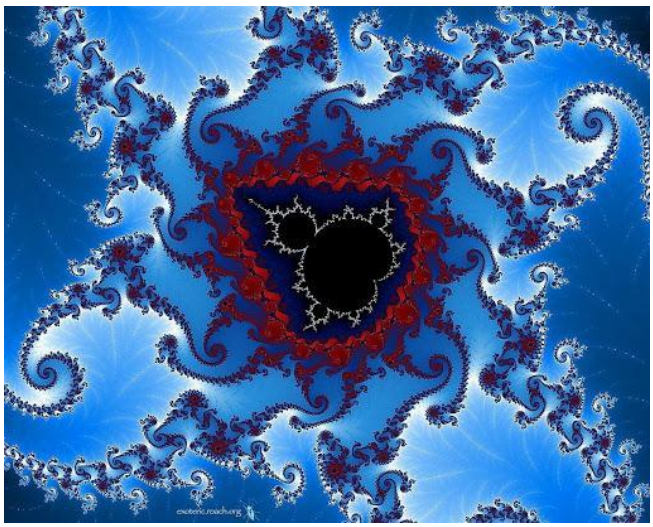

Рис. 3. Приклад фракталу

Поняття фрактала, фрактальної геометрії з’явилося в кінці 70-х років і по сьогодні має широке застосування. Програмні засоби для роботи з фрактальною графікою призначені для автоматичної генерації зображень шляхом математичних розрахунків. Створення фрактальної художньої композиції полягає не в малюванні або оформленні, а в програмуванні. Фрактальна графіка, як і векторна, використовує обчислення, але відрізняється від неї тим, що ніякі об'єкти в пам'яті комп'ютера не зберігаються. Зображення будується на основі рівнянь (або через систему рівнянь), тому нічого, крім формул, зберігати не треба. Фрактальну графіку рідко застосовують для створення друкованих та електронних документів, але її часто використовують у розважальних програмах.

Тривимірну графіку використовують як в «чистому» вигляді, так і можливе трактування як «псевдо 3d-графіка». Все залежить від пристрою відображення. Якщо мова йде про відображення тривимірного об’єкту на площину, то це графіка, яка використовує спеціальні моделі та складні алгоритми для імітації об’єму на площині і тому вважається тривимірною (Рис.4).

Рис.4. Приклад двовимірного і тривимірного (псевдо3d) зображення

Якщо є відображення тривимірного об’єкту у тривимірний простір, то це тривимірна графіка у точному сенсі цього терміну.

Рис.5. Приклад тривимірного відображення

Тривимірна графіка використовується в системах віртуальної (VR), доповненої (AR), змішаної (MR) реальності. Сферами її застосування є тренажери, освітні, наукові програми, і звичайно – індустрія розваг і комерція.

Рис.6. Приклад використання доповненої реальності в торгівлі

КГ одна з найпопулярніших комп’ютерних галузей. ЇЇ розвиток розпочався з вузькопрофільного використання науковцями у військових, технічних задачах, сьогодні КГ широко застосовується в повсякденному житті звичайної людини – як то графічний інтерфейс програмного застосунку чи редактор фотографій.

Виділяють такі основні сфери застосування комп'ютерної графіки:

- ділова графіка (схеми, графіки, діаграми);
- візуалізація процесів і явищ у наукових дослідженнях (комп'ютерне графічне моделювання);

- САПР (системи автоматизованого проектування);
- медицина (комп'ютерна томографія, УЗД і т.д.);
- геодезія і картографія (ГІС);
- комп'ютерні ігри;
- поліграфія (схеми, плакати, ілюстрації);
- сфера масової інформації (графіка в Інтернеті, ілюстрації, фото);
- кінематографія (спецефекти, комп'ютерна мультиплікація);
- системи з VR/AR/MR для освіти;
- системи з VR/AR/MR для торгівлі;
- людино-машинна взаємодія.

Потреби науки, техніки, освіти, медицини, індустрії розваг і, на жаль, військової справи ставлять нові задачі і зумовлюють розвиток КГ. Актуальними задачами у КГ є розробка алгоритмів для якісної візуалізації чи розпізнаванні складних зображень з мінімальними витратами обчислювальних ресурсів.

3. Система КГ. Базовими елементами системи КГ з точки зору програмування є область виведення зображення (полотно, канва) та графічні примітиви (команди для формування та перетворення графічних об’єктів). У КГ об'єкти моделювання та графічного введення-виведення описуються в різноманітних системах координат.

Система координат – це сукупність правил, які ставлять у відповідність кожному об'єкту (точці) визначений набір чисел (координат). Координати – це значення деяких характеристик об'єктів, які однозначно визначають положення об'єкта в просторі. У геометричному моделюванні та комп'ютерній графіці застосовуються класичні системи координат: афінна, декартова, полярна, циліндрична, сферична, система однорідних координат тощо.

Окрім класичних, в комп'ютерній графіці визначаються ще дві основні системи координат: світова (реальна) система та система пристрою виведення.

Світова система координат – тривимірна або двовимірна прямокутна декартова система координат, в якій описуються об'єкти певного світу (реальні або абстрактні об'єкти та процеси), що моделюються на комп'ютерах. У геометричному моделюванні всі об'єкти мають світові координати. Значення світових координат належать множині дійсних чисел.

Система координат пристрою виведення – двовимірна (можливо, тривимірна) прямокутна декартова система координат, в якій формується і виводиться зображення об'єктів на екран дисплею (плотер) або в яку

перетворюються зображення з графічних пристроїв введення даних в комп'ютер (сканерів, планшетів тощо).

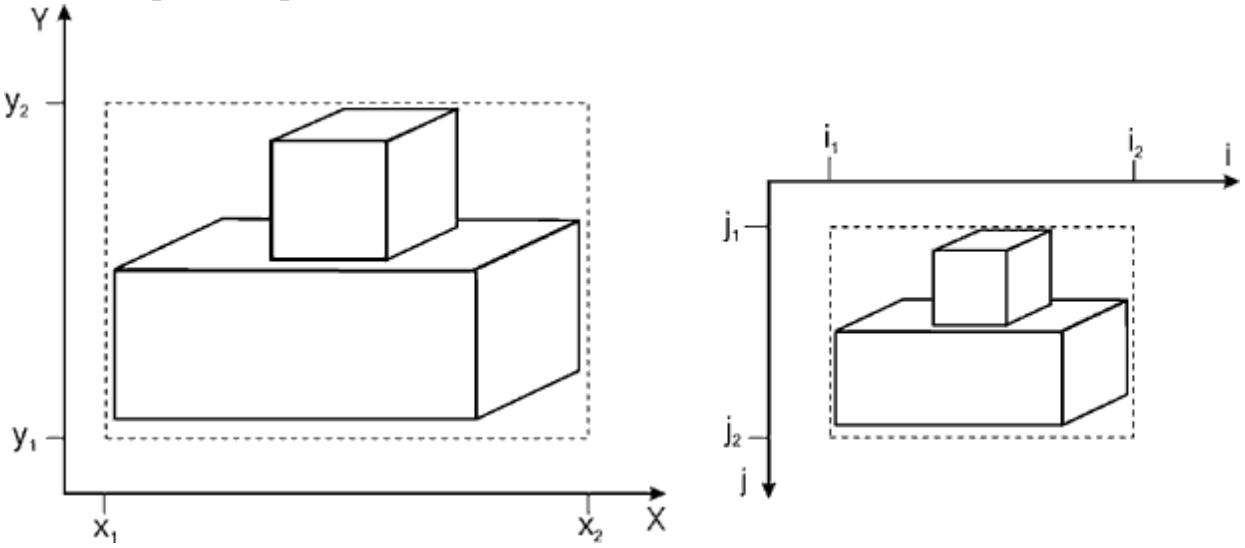

Рис. 7. Системи координат: реальна та пристрою виведення

Моделювання об'єктів та формування зображення виконується в реальній системі координат, а виведення зображення на екран супроводжується перетворенням координат із світової в систему області виведення.

Координати точки *(i, j)* на пристрої виведення обчислюються за формулами перетворення систем координат

$$i = i_1 + S_x(x - x_1), \quad j = j_2 - S_y(y - y_1).$$

де

$$\Delta x = x_2 - x_1, \quad \Delta y = y_2 - y_1, \quad \Delta i = i_2 - i_1, \quad \Delta j = j_2 - j_1,$$

$$S_x = \frac{\Delta i}{\Delta x}, \quad S_y = \frac{\Delta j}{\Delta y}$$

З розвитком алгоритмічного та програмного забезпечення систем КГ з’явилось багато способів побудови різноманітних графічних об’єктів, що в сукупності називають зображенням. Графічні зображення часто мають ієрархічну структуру, яка виникає в результаті процесу побудови знизу-вверх. Простіші базові графічні елементи, з яких будуються графічні об’єкти довільної складності, називаються графічними примітивами. Вони використовуються як будівельні блоки для об’єктів більш високого рівня. Графічні примітиви розглядаються як прості неподільні елементи зображення і генеруються однією окремою командою (методом). Вони можуть мати також апаратний та апаратно-програмний рівень формування. Як правило, множина примітивів містить такі елементи:

- точку (x, y),
- відрізок (A, B) (довжина відрізка з кінцями $A(x_1, y_1)$, $B(x_2, y_2)$ обчислюється за формулою $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Координати точки середини відрізка $O(x_0, y_0)$, обчислюються за формулами: $x_o = \frac{x_1 + x_2}{2}, y_o = \frac{y_1 + y_2}{2}$),
- ламану лінію (замкнену і незамкнену),
- трикутник,
- чотирикутник,
- опуклий многокутник (полігон),
- коло (рівняння кола з центром в точці $O(x_0, y_0)$ та радіусом R обчислюється за формулою $R^2 = (x - x_0)^2 + (y - y_0)^2$).

Параметри примітивів характеризують форму, розміри, місце розташування примітива. Атрибути примітивів – це властивості, які визначають їх вигляд при візуалізації, тобто спосіб зображення заданого примітиву. Для кожного типу об’єктів визначають свій набір атрибутів:

- для точки – розмір та колір;
- для лінії – колір, товщина і тип (лінія, яка використовується для малювання відрізків може бути суцільною, штриховою і т.д.);
- для многокутників – режим контуру чи заповнення. Цей параметр вказує спосіб задання многокутників. Можна малювати тільки вершини або ребра, а можна зображати зафарбовані многокутники, при цьому контур може малюватись одним кольором, а внутрішня область заливатись іншим.

Множина видимих на екрані об'єктів, упорядкованих згідно з просторовими відношеннями (наприклад, за відношенням дальше-ближче, лівіше-правіше), разом з атрибутами елементів поверхонь (колір, текстура, оптичні властивості матеріалів) і з атрибутами оточення (рівень освітлення, туман, димка тощо) утворюють **сцену**.

Область відображення графічних об’єктів у системах КГ прийнято називати Canvas (полотно – з англ.), як правило, такі ж однойменні компоненти застосовуються в середовищах розробки комп’ютерних програм.

4. Інструменти для програмування КГ. Більшість мов програмування мають свої стандартні компоненти (класи та їх методи, функції графічних бібліотех), що дозволяють створювати графічні об’єкти. Далі наведено коротку інформацію про кілька можливих технологій для реалізації завдання лабораторної роботи.

4.1) у мові C++ стандартним модулем є **graphics.h.** У цьому файлі описано всі функції, які потрібні для написання програми, що працює з графікою. Основними функціями є:

initgraph (& grdriver, & grmode, "path") – функція ініціалізації графічного режиму, де &grdriver, &grmode, "path" - параметри завантажуваного режиму.

closegraph () – дана функція без будь-яких параметрів закриває поточний графічний режим;

detectgraph (&grdriver, &grmode) – функція визначення графічного драйвера і відео режиму;

arc (int x, int y, int stangle, int endangle, int radius) – функція малює дугу, за заданими координатами *x, y,* початковим і кінцевим кутом stangle, endangle і заданим радіусом radius.

circle (int x, int y, int radius) – функція малює коло з центром в точці з координатами (*x, y*) і радіусом radius.

drawpoly (int numpoints, int *polypoints) – функція малює полігон з кількістю вершин numpoints і координатами вершин *polypoints. *polypoints є одновимірним масивом.

fillpoly (int numpoints, int *polypoints) – функція малює і зафарбовує заданим кольором полігон;

lineto (int x, int y) – малює лінію від поточної позиції до точки з координатами (*x, y*), потім переносить поточну позицію в (*x, y*);

putpixel (int x, int y, int color) – малює точку з координатами (*x, y*) і кольором color.

rectangle (int left, int top, int right, int bottom) – малює прямокутник від точки з координатами (left, top) до точки з координатами (right, bottom).

Приклад 1. Наведемо приклад коду C++, що будує графік функції $y=x\sin(x)$ на інтервалі від $x=0$ до $x=10$:

```
#include <graphics.h> // підключення графічної бібліотеки
#include <conio.h>
#include <math.h>      // підключення  математичної бібліотеки (для sin)
int main(void)
{
float x, y;
int gdriver = DETECT, gmode;
initgraph(&gdriver, &gmode, "");
setbkcolor(1);
line(50,0,50,480); line(50,200,640,200);
moveto(50,200);
```

```
x=0;
do
{
y=x*sin(x);
lineto(50+x*50, 100+(100-(y*20
x=x+0.02;
} while(x<10);
getch();
closegraph();
return 0;
}
```

4.2) Для роботи з графікою можна використати засоби Qt – крос-платформеного інструментарію розробки програмного забезпечення мовою програмування C++:

- QGraphicsScene є основою для створення і керування сценою, на якій відбувається відображення графічних об'єктів;
- QGraphicsView дозволяє відображати вміст QGraphicsScene і забезпечує можливість масштабування, переміщення та взаємодії з графічним вмістом. Він слугує вікном, через яке можна переглядати створену сцену;
- QPoint - це клас у бібліотеці Qt, який представляє точку в двовимірному просторі;
- QLine - це клас, який є лінією в двовимірному просторі. Об'єкт класу QLine може бути створений за допомогою координат початкової та кінцевої точок. Клас надає функції для отримання інформації про лінію, такі як довжина, кут нахилу, перевірка перетину з іншою лінією тощо;
- QImage є класом, який надає можливість роботи зі зображеннями у форматі мапи пікселів. QImage може бути використаний для завантаження, збереження та маніпулювання зображеннями. Він підтримує різні формати та операції зображень, такі як масштабування, обрізка, конвертація кольорів та інші;
- QColor - це клас, який представляє колір у бібліотеці Qt. Він дозволяє працювати з кольорами в різних форматах (RGB, HSV тощо) та виконувати різні операції, такі як зміна яскравості, насиченості, отримання окремих компонентів кольору тощо. Клас QColor дуже зручний для роботи з графікою та кольоровою обробкою в Qt-застосунках.

4.3) У C# для роботи з графічними примітивами використовують клас System.Drawing.Graphics, основними структурами якого є: single – це число

одиночної точності з плаваючою крапкою, point – визначає впорядковану пару цілих чисел – координат *x* і *y*, тобто описує точку на двовимірній площині, pen – визначає об'єкт, який використовується для малювання прямих ліній і кривих, rectangle – визначає прямокутник з заданими розмірами та розміщенням.

Основними функціями, що дозволяють малювати примітиви є:

DrawArc (Pen, Rectangle, Single, Single) – малює дугу, яка є частиною еліпса, заданого структурою Rectangle;

DrawBezier (Pen, Point, Point, Point, Point) – малює криву Безє, яка визначається чотирма структурами Point.

DrawClosedCurve (Pen, Point []); - будує замкнуту фундаментальну криву, яка визначається масивом структур Point.

DrawCurve (Pen, Point []) – будує фундаментальну криву через точки зазначеного масиву структур Point;

DrawEllipse (Pen, Rectangle) – малює еліпс, який визначається обмежує структурою Rectangle;

DrawPolygon (Pen, Point []) – малює багатокутник, який визначається масивом структур Point;

DrawRectangle (Pen, Rectangle) – малює прямокутник, який визначається структурою Rectangle.

Приклад 2. Наведемо приклад коду C#, що будує графік функції $y=x\sin(x)$ на інтервалі від $x=0$ до $x=10$:

```
private double F(double x)
      {
         return x* Math.Sin(x);
      }
 private void button1_Click(object sender, EventArgs e)
      {
        Bitmap image = new Bitmap(100, 100,
                                  System.Drawing.Imaging.PixelFormat.Format32bpp
                                  Rgb);
         System.Drawing.Graphics.FromImage(image).Clear(Color.White);
         pictureBox1.Image = image;
         Pen pen = new Pen(Color.Red, 1);
         for (int i = 0; i < 10; i+=0.5)
         {
            System.Drawing.Graphics.FromImage(image).DrawLine(pen, 50 + i,
                     Convert.ToInt32(50 - F(i)), 51 + i, Convert.ToInt32(50 - F(i
               + 1)));
```

```
        }
        pen.Dispose();
        pictureBox1.Invalidate();
    }
```

4.4) Для відтворення графіки на вебсторінці можна використовувати HTML-елемент <canvas>. Canvas (англійською «полотно») – елемент HTML5, який застосовують для створення графіки, використовуючи скрипти (переважно JavaScript). Однією з ключових операцій є отримання контексту полотна виведення за допомогою методу getContext(). Цей контекст надає методи та властивості для ефективного малювання в межах Canvas. Полотно спочатку порожнє і прозоре.

Для отримання контексту використовують getContext, наприклад, таким чином:

```
var a = document.getElementById('name');
var b = a.getContext('2d');
```

Прямокутник з властивостями: x, y – координати верхнього лівого кута; width – ширина; height – висота; малюють за допомогою таких методів контексту:

fillRect(x, y, width, height) – заповнений прямокутник;

strokeRect(x, y, width, height) – прямокутний контур;

clearRect(x, y, width, height) – очищення й прозорість прямокутної області.

Контур малюють за допомогою таких методів:

beginPath() – створити контур;

closePath() – закрити контур;

stroke() – обведення контуру;

fill() – заповнення внутрішньої області;

lineTo(x,y) – проведення лінії з поточної точки до точки (x, y);

moveTo(x, y) – переміщення у точку (x, y) без проведення лінії;

arc(x, y, r, φ1, φ2, anticlockwise) – малювання дуги кола з центром у точці (x, y), радіусом r і кутом у межах від φ0 до φ1 у напрямку проти руху годинникової стрілки при значенні true параметра anticlockwise, інакше – у напрямку руху годинникової стрілки.

Робота з елементом <canvas> досить проста, вимагає базових знань HTML та JavaScript. Елемент <canvas> є одним із найпоширеніших інструментів для візуалізації 2D-графіки в Інтернеті.

Хід виконання роботи

1. Опрацювати теоретичні відомості та відповідні лекційні матеріали.
2. Проаналізувати можливі засоби для реалізації завдання.

3. Ознайомитися з можливостями обраних мови програмування та середовища розробки для створення програм візуалізації двовимірних фігур.

4. Розробити програму згідно індивідуального варіанту.

5. Протестувати програму на даних різних класах еквівалентності. При необхідності виправити помилки.

6. Провести самоконтроль знань з використанням контрольних питань.

7. Після захисту роботи оформити звіт згідно вимог та завантажити його у ВНС.

Завдання

Написати програму згідно індивідуального варіанту вибраною мовою програмування з використанням її базових графічних примітивів. Використовувати засоби ШІ для генерування програмного коду не можна.

Загальні вимоги до програми:

1. Система координат

- Початок координат – у центрі області виведення.
- Осі координат з підписами та одиничними відрізками.
- Масштаб координатної системи повинен автоматично підбиратися так, щоб усі побудовані фігури залишалися видимими.

2. Побудова фігур

- Фігури задаються у математичних координатах, а не координатах екрану.
- Всі допоміжні точки (вершини, центри, висоти, діагоналі) повинні бути обчислені програмою та відображені на екрані.
- Програма повинна дозволяти будувати декілька фігур без перезапуску.
- Якщо за умовою варіанту вхідні дані не визначають однозначно фігуру, передбачити обґрунтоване уведення додатково необхідних даних.
- Дані про фігури повинні зберігатися у файлі.

3. Введення даних

- Вводяться у зручному вигляді параметри фігур (координати, довжини, висоти тощо).
- Користувач може обирати кольори ліній та/або заливки кожної фігури.
- Передбачено обробку некоректного введення (нечислові значення, геометрично неможливі фігури, вихід за область координат).

4. Динамічна демонстрація

- Додавати нові фігури під час роботи;
- Очищати область побудови;
- Зберігати побудовані фігури видимими після змін масштабу.

Варіанти

1. Програма «Рівносторонні трикутники (через центр)»

Побудувати рівносторонні трикутники за координатами центра та довжини сторони. Автоматично побудувати медіани. Забезпечити можливість вибору кольору заливки та кольору ліній медіан.

2. Програма «Паралелограми (3 вершини)»

Побудувати паралелограми за координатами трьох вершин. Автоматично обчислити четверту вершину та побудувати діагоналі. Користувач обирає кольори заливки та діагоналей.

3. Програма «Рівнобедрені трикутники»

Побудувати рівнобедрені трикутники за довжиною висоти та координатами основи. Відобразити висоту та побудувати описане коло. Забезпечити вибір кольору заливки кола та ліній.

4. Програма «Квадрати (через діагональ)»

Побудувати квадрати за координатами однієї вершини та довжини діагоналі. Автоматично побудувати описане коло. Користувач обирає кольори контурів і заливки кола.

5. Програма «Правильні шестикутники»

Побудувати декілька шестикутників за координатами центра та радіуса описаного кола. Користувач обирає кольори ліній та заливки.

6. Програма «Трикутники (три вершини)»

Побудувати трикутники за координатами трьох вершин. Автоматично побудувати центр мас та описане коло. Користувач обирає кольори ліній та заливки.

7. Програма «Кола (центр + точка на колі)»

Побудувати кола за координатами центра та точки на колі. Автоматично відобразити:

- діаметр, якому належить введена точка,
- відрізок дотичної довжини 2 радіуса, проведеної в іншій точці відображеного діаметра.

Користувач обирає кольори контуру та заливки кола.

8. Програма «Трапеції»

Побудувати трапеції за координатами її центра, висоти та довжин основ. Автоматично побудувати середню лінію. Користувач обирає кольори заливки та ліній середньої лінії.

9. Програма «Квадрати + повернений квадрат»

Побудувати квадрати за координатами центра та довжини сторони. Автоматично побудувати квадрат, повернутий на 45°. Користувач обирає кольори обох квадратів.

10. Програма «Ромби»

Побудувати ромби за координатами центра та довжин діагоналей. Автоматично побудувати вписане коло. Користувач обирає кольори діагоналей та заливки кола.

11. Програма «Прямокутники»

Побудувати прямокутники за шириною, висотою та координатами центра. Вважати, що сторони прямокутника паралельні осям координат. Автоматично побудувати обидві діагоналі та візуалізувати центр фігури. Користувач обирає кольори заливки та діагоналей.

12. Програма «Квадрати та кола»

Побудувати квадрати за координатами центра та довжини сторони. Програма повинна автоматично побудувати для кожного квадрата вписане коло та описане коло.Користувач обирає кольори контурів квадрата, вписаного та описаного кіл, а також кольори заливки кіл.

13. Програма «Рівнобічні трапеції»

Побудувати рівнобічні трапеції за координатами центра, висоти та довжин основ. Автоматично побудувати діагоналі. Користувач обирає кольори ліній та заливки.

14. Програма «Паралелограми»

Побудувати паралелограми за координатами центра, довжин сторін та кута між сторонами. Автоматично побудувати описаний прямокутник. Користувач обирає кольори контурів та заливки.

15. Програма «Квадрати»

Побудувати квадрати за координатами центра та довжини сторони. Автоматично побудувати вписане коло. Забезпечити можливість вибору кольору заливки квадрата та кола.

Вимоги до звіту

Звіт формується в електронному вигляді та завантажується у ВНС з назвою *Прізвище_№1.pdf*. Формат сторінок звіту – А4. Сторінки нумеруються, крім титульної.

Рекомендується використовувати шрифт розміром 14 пт, міжрядковий інтервал 1,5, абзацний відступ 1 см. Заголовки у тексті центруються, усі тексти вирівнюються по ширині сторінки.

Грамотність – обов'язкова вимога до текстів звіту.

Структура звіту:

1. Титульний аркуш
2. Тема
3. Мета
4. Теоретичні відомості

У теоретичних відомостях описати переваги обраної технології для реалізації завдання та основні інструменти (функції, методи), які використано під час виконання лабораторної роботи. Обсяг 1,5-2 сторінки.

5. Формулювання завдання
6. Текст програми з коментарями
7. Результати виконання програми
8. Висновки

У висновках вказати чи виконано завдання (цілком, частково), що було вивчено. Записати, яку функцію (компоненту) програми було найважче реалізувати.

Критерії оцінювання лабораторної роботи

Рекомендований термін виконання та захисту лабораторної роботи №1 включає перші три тижні навчання семестру – № 1,2,3.

Максимальна оцінка – 5 балів. Робота оцінюється в 0 балів з 9-го тижня.

5 балів розподіляється за такими видами робіт:

- 3,5 балів за програму;
- 1 бал з 5 балів – за відповіді на 2-3 теоретичні питання;
- 0,5 бал за вчасно підготовлений і завантажений у ВНС змістовний звіт (зокрема, у першу чергу – змістовне наповнення теоретичних відомостей).

Під час захисту роботи студент повинен пояснити:

- алгоритм перетворення математичних координат у координати екрану;
- обчислення допоміжних точок (вершин, центрів, діагоналей тощо).

3,5 балів – відмінна САМОСТІЙНА робота, програма повністю виконує все завдання (-/мінус 0,5 при невиконанні кожної некритичної умови завдання), передбачає виняткові та екстремальні ситуації (-/мінус 0,5 при неврахуванні цих ситуацій), має зручний інтерфейс користувача (-/мінус 0,5 інтерфейс побудований непрофесійно).

Самостійність виконання студентом роботи перевіряється через виконання завдання у присутності викладача (а саме, необхідно внести зміну у програму на вимогу викладача), відповіді на питання як працюють частини програми, розуміння і відповідні пояснення коду. Якщо завдання при викладачеві не виконано, тоді виконання роботи можна вважати частково самостійним лише при умові, що є пояснення як працює програма та розуміння коду студентом. У такому випадку студент може отримати максимум 1,5 балів (з 3,5).

Якщо студент не орієнтується в коді програми, робота не може бути зарахована, навіть якщо є відмінні відповіді на теоретичні питання.

Якщо програма має суттєві недоліки (сумарна оцінка менше 0 з 3,5 балів), робота не зараховується. У такому випадку проводиться повторна здача роботи після того, як здадуть цю роботу усі студенти, які її підготували в зазначені у графіку терміни.

Звіт – обов’язкова компонента лабораторної роботи. При відсутності звіту впродовж семестру робота не зараховується.

Якщо студент не завантажив звіт до початку наступного заняття після захисту роботи, можливість отримати 0,5 балів за звіт втрачається. Якщо звіт не завантажено перед останнім лабораторним заняттям, робота не зараховується і потрібно повторно захищати роботу.

Контрольні питання

1. Дайте означення КГ? Що таке інтерактивна КГ?
2. Які сфери застосування комп’ютерної графіки?
3. Наведіть приклади застосування тривимірної КГ?
4. Наведіть приклади застосування растрової КГ?
5. Наведіть приклади застосування векторної КГ?
6. Які типи графіки існують?
7. Що таке растр та його роздільна здатність?
8. Що таке графічний примітив? Наведіть приклади графічних примітивів.
9. Що таке атрибути графічних примітивів? Наведіть приклади.
10. Що таке сцена в КГ?
11. У чому полягає обробка зображень?

12. У чому полягає розпізнавання зображень?

13. Які основні етапи розвитку КГ?

14. Що таке канва в КГ? Які є основні інструменти роботи з канвою?

15. Чим принципово відрізняється декартова система координат від системи координат області виведення у КГ?

16. Які перетворення необхідно виконати, щоб декартова система координат реального світу співпадала з системою координат області виведення КГ?

СПИСОК ЛІТЕРАТУРИ

1. Навчально-методичний комплекс «Комп’ютерна графіка» [Електронний ресурс]/ Укладач Є.Левус. – Режим доступу: https://vns.lpnu.ua/course/view.php?id=17471

2. Василюк А. С. Комп’ютерна графіка: навчальний посібник/ А. С. Василюк, Н. І. Мельникова. Львів: Видавництво Львівської політехніки, 2016. - 308 с.

3. Є.В. Бородавка. Комп'ютерна графіка: навчальний посібник/ О.О. Терентьєв. Київ: КНУБА, 2023. - 132 с.

4. Кривцова О. П. Програмування мовою С++. Технологія візуального програмування: навчальний посібник – Полтава : ПНПУ імені В.Г. Короленка, 2020. – 144 с.

5. Canvas tutorial [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/ Canvas_API/ Tutorial?retiredLocale=uk

НАВЧАЛЬНЕ ВИДАННЯ

ПОБУДОВА ДВОВИМІРНИХ ФІГУР ЗАСОБАМИ МОВИ ПРОГРАМУВАННЯ

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторної роботи №1
з дисципліни „ Комп’ютерна графіка ”
для студентів спеціальності
„Інженерія програмного забезпечення”

Укладач Левус Євгенія Василівна