МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

Факультет програмної інженерії та бізнесу

Кафедра інженерії програмного забезпечення

Практичні роботи

Minor *«Розробник ігрових додатків»*

дисципліна *«Комп’ютерна графіка з OpenGL»*

(назва дисципліни)

Виконав: студент 3 курсу групи  *326ст*

напряму підготовки (спеціальності):

*122 Комп’ютерні науки*

(шифр і назва напряму підготовки / спеціальності)

*Мисляєв Д.А.*

(прізвище й ініціали студента)

Прийняв: *доц. каф 603, к.т.н, Лучшев П.О.*

(посада, науковий ступінь, прізвище й ініціали)

Національна шкала:

Кількість балів:

Оцінка ECTS:

Зміст

[Практична робота 1. Основні принципи роботи з OpenGL 3](#_Toc4885)

[Завдання, варіант № 13 3](#_Toc237)

[Системна інформація 3](#_Toc24881)

[Теоретичні відомості 3](#_Toc8046)

[Результати виконання практичної роботи 5](#_Toc29254)

[Практична робота 2. Графічні примітиви OpenGL 7](#_Toc4354)

[Завдання, варіант № 13 7](#_Toc28471)

[Системна інформація 9](#_Toc25804)

[Теоретичні відомості 9](#_Toc21792)

[Результати виконання практичної роботи 10](#_Toc19423)

[Практична робота 3. Графік функції однієї змінної 16](#_Toc8099)

[Завдання, варіант № 13 16](#_Toc3775)

[Системна інформація 16](#_Toc2781)

[Теоретичні відомості 17](#_Toc24610)

[Результати виконання практичної роботи 18](#_Toc20904)

[Практична робота 4. Криві другого порядку 22](#_Toc32388)

[Завдання, варіант № 13 22](#_Toc13146)

[Системна інформація 22](#_Toc25478)

[Теоретичні відомості 22](#_Toc9257)

[Результати виконання практичної роботи 23](#_Toc30521)

[Практична робота 5. Квадратичні примітиви. Афінні перетворення у просторі 27](#_Toc26059)

[Завдання, варіант № 13 27](#_Toc11967)

[Системна інформація 27](#_Toc29206)

[Теоретичні відомості 28](#_Toc30949)

[Результати виконання практичної роботи 29](#_Toc12206)

[Загальний перелік посилань 31](#_Toc12454)

[Додаток А. Лістинг програми до практичної роботи №1 32](#_Toc26873)

[Додаток B. Лістинг програми до практичної роботи №2 36](#_Toc19641)

[Додаток C. Лістинг програми до практичної роботи №3 41](#_Toc32765)

[Додаток D. Лістинг програми до практичної роботи №4 55](#_Toc31894)

[Додаток E. Лістинг програми до практичної роботи №5 62](#_Toc28108)

# Практична робота 1. Основні принципи роботи з OpenGL

## Завдання, варіант № 13

За допомогою інструментальних засобів, зазначених викладачем, створити простий програмний проєкт із підтримкою бібліотеки OpenGL. Розробити програму із застосуванням команд OpenGL, яка встановлює анізотропну систему координат, створює та виводить варіант зображення на екран/у вікно з урахуванням заданих примітивів та координат x1, y1 та x2, y2 . Для рисування координатної сітки необхідно використовувати пунктирні лінії. Контур фігури, осі та координатну сітку зобразити лініями різної товщини. Для парних варіантів точки повинні мати квадратну форму, а для непарних – круглу.

|  |  |  |
| --- | --- | --- |
| 13 | Примітиви:  GL\_POINTS, GL\_LINES  x1 = -1; x2 = 8  y1 = -2; y2 = 2 |  |

## Системна інформація

Для розробки та виконання практичних робіт використовувалися наступні апаратні та програмні засоби:

Processor AMD Ryzen 5 2600 3.40 GHz

RAM 16.0 GB (15.9 GB usable)

System type 64-bit operating system, x64-based processor

Edition Windows 11 Version 23H2

IDE Microsoft Visual Studio Enterprise 2022 (64-bit) version 17.11.5

## Теоретичні відомості

Виведення графічних об'єктів на екран може виконуватися в ізотропному (isotropic) або анізотропному (anisotropic) режимі ідображення, кожен з яких допускає зміну напряму осей X та Y, а також зміну масштабу осей координат. В ізотропному режимі відображення масштаб вздовж осей X і Y завжди однаковий (тобто обох осей використовуються однакові логічні одиниці довжини). Анізотропний режим передбачає використання різних масштабів для різних осей (хоч і однакові масштаби допускаються).

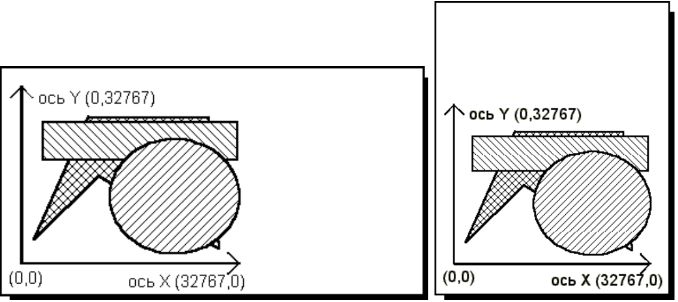


Рис. 1.1. Збереження пропорцій фігур в ізотропному режимі

В ізотропному режимі відображення (рис. 1.1) при зміні розмірів вікна Windows система координат налаштовується таким чином, щоб масштаб по осях X та Y був однаковий. Якщо висота вікна не дорівнює його ширині, то при використанні ізотропного режиму відображення для обчислення масштабу використовується менша зі сторін. При використанні анізотропного режиму налаштування масштабу по кожній осі виконується незалежно, тому область відображення займатиме всю робочу область вікна при будь-якій зміні розмірів цього вікна (рис. 1.2).

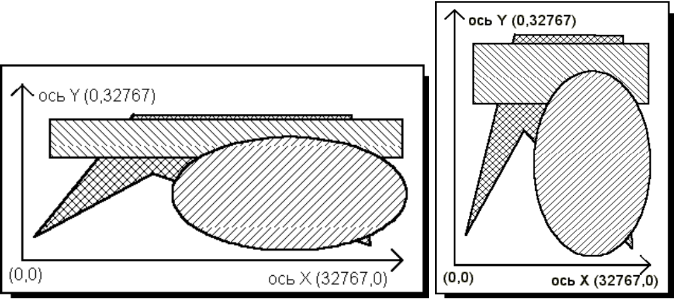


Рис. 1.2. Зміна пропорцій фігур в аніізотропному режимі

Ізотропний режим відображення використовується в тих випадках, коли важливо зберегти пропорції об'єктів (тобто квадрат і окружність повинні залишитися такими, а не трансформуватися в прямокутник і еліпс відповідно) при будь-якій зміні розмірів вікна, в яке виводиться зображення. Анізотропний режим затребуваний у тих випадках, коли зображення повинне займати всю внутрішню поверхню вікна за будь-якої зміни розмірів вікна, коли співвідношення масштабів по осях не має значення.

## Результати виконання практичної роботи

### Розв'язання завдання

Для управління параметрами графічних примітивів було викорастані наступні команди:

* товщина для ліній фігури glLineWidth(), рядок 96, файл Draw.cs;
* тип для курсивних ліній glLineStipple(), glEnable()/glDisable(), рядок 25, файл Draw.cs;
* тип для круглих точок glEnable()/glDisable(), рядок 126, файл Draw.cs;

Коректне відображення завдання під час змінення розмірів/положення вікна наведено у рис 1.3 та 1.4.

Застосування циклів для створення зображень наведено у рядках 50 - 93 файлу Draw.cs.

Використання ООП реалізовано за допомогою розроблення власних класів, які наведено у файлах Draw.cs Додатку А.

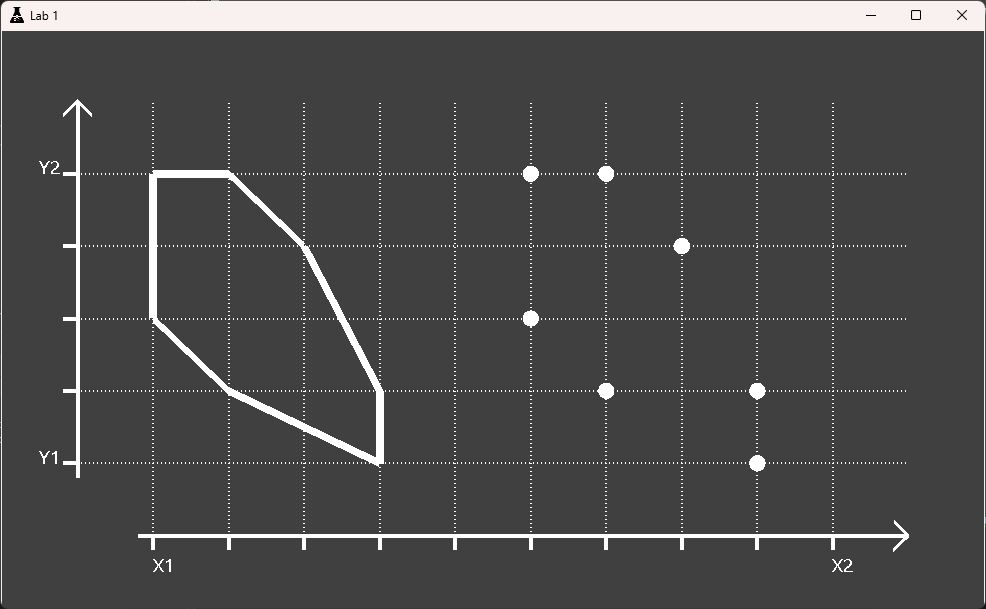


Рисунок 1.3 – Тестування програми при зміні ширини вікна

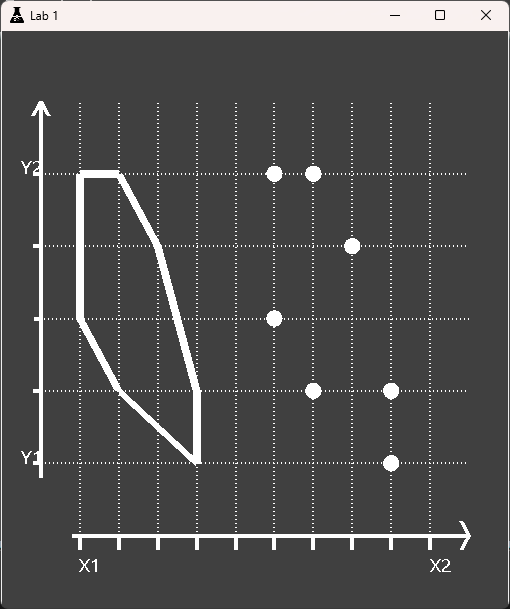


Рисунок 1.4 – Тестування програми при зміні ширини вікна

### Контроль виконання вимог та елементів завдання

В результаті виконання практичної роботи були повністю виконані елементи базового рівня та частково підвищеного рівня складності, що відображено в таблиці 1.1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблиця 1.1 | | | | |
| № з/п | Складність | Вимоги | Бали | Зроблено |
| 1 | Базовий рівень | Використання команд управління параметрами графічних примітивів (колір, тип, товщина) | 2 | **+** |
| 2 | Коректне відображення завдання під час змінення розмірів/положення вікна | 1 | **+** |
| 3 | Розроблення підпрограм для виключення дублювання коду | 1 | **-** |
| 4 | Застосування циклів для створення зображень | 1 | **+** |
| 5 | Підвищений рівень | Формування зображення векторними командами *OpenGL* (*glDrawArrays* и т.п.) | 1 | **-** |
| 6 | Використання ООП (розроблення власних класів) | 2 | **+** |

# Практична робота 2. Графічні примітиви OpenGL

## Завдання, варіант № 13

Використовуючи інструментальні засоби, що вказані викладачем, і беручи до уваги вимого, що наведено в табл. 2.1, створити програмний проєкт з підтримкою OpenGL. За допомогою команд **glOrtho** / **gluOrtho2D** і **glViewport** встановити для робочої області ізотропну систему координат з урахуванням розміру фігури, яку задано у варіанті (табл. 2.2). Після старту застосунок повинен відображати у робочій області одну плитку (**tile**). Приклад початкового стану застосунку показано на рис. 2.1.

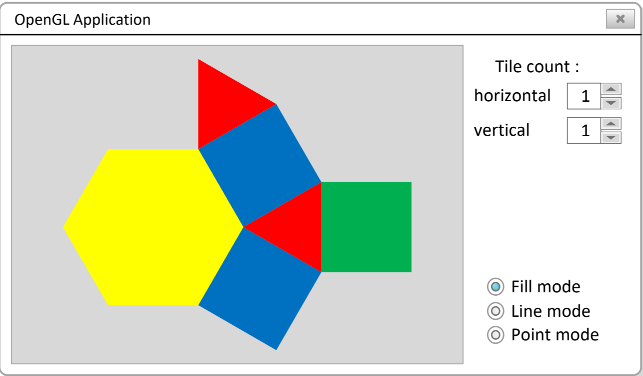
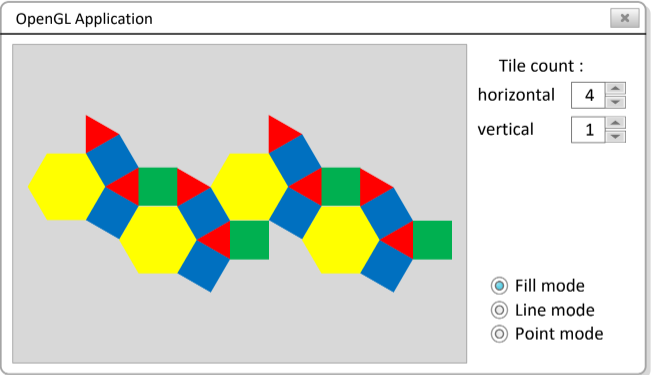


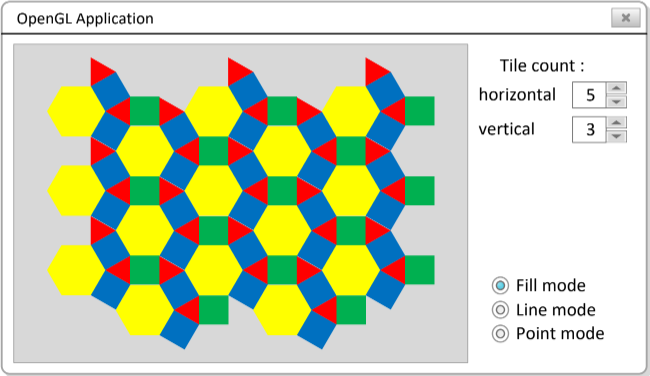
Рисунок 2.1 – Вигляд застосунку після старту

Усі варіанти заданій основані на правильних багатокутниках, розмір яких визначається величиною одного ребра. Для зафарбування пропонується використовувати шість кольорів: білий, сірий (35 %), червоний, зелений, синій и жовтий.

За допомогою клавіатури або маніпулятора «миша» користувач повинен мати можливість виконати замощення (**tessellation**, **tilling**) робочої області по горизонталі і вертикалі. При цьому систему координат необхідно скорегувати таким чином, щоб замощена поверхня розташовувалася у центрі робочої області. Приклад замощення робочої області застосунку показано на рис. 2.2.



a



b

Рисунок 2.2 – Вигляд застосунку під час замощення:

а – тільки по горизонталі; б – по горизонталі та вертикалі

Крім цього, користувач повинен мати можливість змінювати режим відображення графічних примітивів OpenGL: точкове (тільки вершини фігури), контурне (рис. 2.3) і з заповненням кольором (див. рис. 2.2). Передбачається, що перемикання між режимами виконують за подією від клавіатури і/або маніпулятора «миша». При цьому можна використовувати як стандартні елементи керування, так і власні, що реалізовані та відображені засобами OpenGL (для підвищеного рівня складності, див. табл. 2.1).

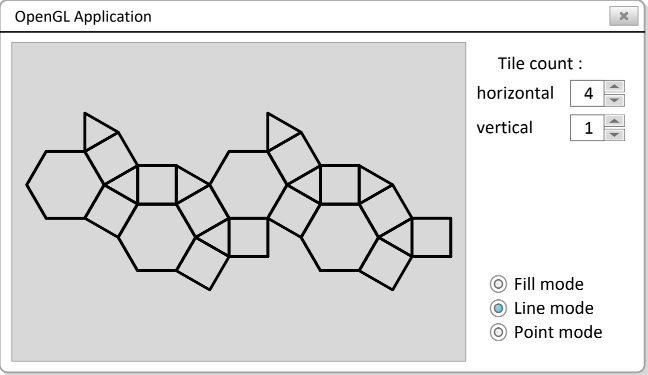


Рисунок 2.3 – Приклад керування режимом виводу графічних примітивів

|  |  |  |
| --- | --- | --- |
| 13 | Сторона фігури a = 0.2  Примітив(и):  GL\_POLYGON |  |

## Системна інформація

Для розробки та виконання практичних робіт використовувалися наступні апаратні та програмні засоби:

Processor AMD Ryzen 5 2600 3.40 GHz

RAM 16.0 GB (15.9 GB usable)

System type 64-bit operating system, x64-based processor

Edition Windows 11 Version 23H2

IDE Microsoft Visual Studio Enterprise 2022 (64-bit) version 17.11.5

## Теоретичні відомості

Під час виводу зображення необхідно враховувати, що кожна поверхня графічного примітиву OpenGL має дві сторони і режим виводу для кожної з них може бути налаштований окремо за допомогою команди **glPolygonMode**.

Для зміни режиму (моделі) розфарбування використовують команду **glShadeModel**. Якщо режим зафарбування напівтонами вимкнений, то колір примітиву визначається кольором тільки однієї вершини. Наприклад, для **GL\_TRIANGLE\_STRIP** колір першого трикутника визначається кольором третьої вершини, другого – четвертої вершини і т. д.

Для встановлення шаблону зафарбування необхідно задіяти команди-перемикачі **glEnable** / **glDisable** (як і для шаблону лінії).

## Результати виконання практичної роботи

### Розв'язання завдання

Під час запуску застосунку зображення відповідає варіанту завдання однією плиткою, рисунок **2.4**, рядки **138 - 214**, файл **RenderControl.cs**, **додаток В**.

Багаторазове заміщення плиткою, рисунок **2.5**, рядки **126 - 130**. Кратність замощення задається користувачем, рядки **66 - 94**. Файл **RenderControl.cs**, **додаток В**.

Коректне відображення завдання під час зміни як розмірів/положення вікна, так і параметрів замощення, рисунок **2.6 - 2.7**, рядки **37 - 64**, **102 - 111**, файл **RenderControl.cs**, **додаток В**.

Організація взаємодії з користувачем одним зі стандартних засобів, рисунок **2.8**, рядки **13 - 50**, файл **MainForm.cs**, **додаток В**.

Застосування мінімальної кількості графічних примітивів, рядок **139**, файл **RenderControl.cs**, **додаток В**.

Керування режимом виводу графічних примітивів, рисунок **2.9 - 3.1**, рядки **132**, файл **RenderControl.cs**, **додаток В**.

Керування режимом шейдингу графічних примітивів, рисунок **3.2 - 3.3**, рядки **133**, файл **RenderControl.cs**, **додаток В**.

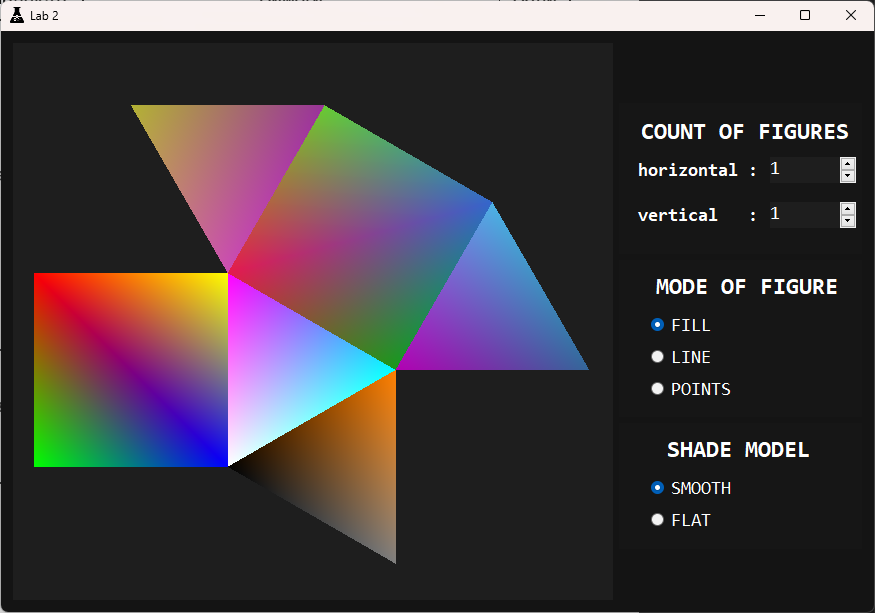


Рисунок 2.4 – Вигляд застосунку після старту

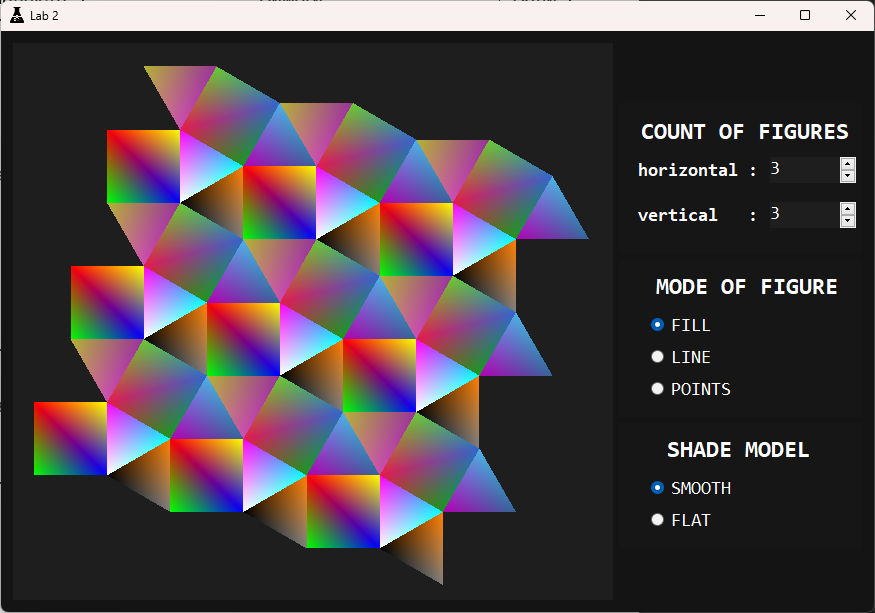


Рисунок 2.5 – Вигляд застосунку під час замощення

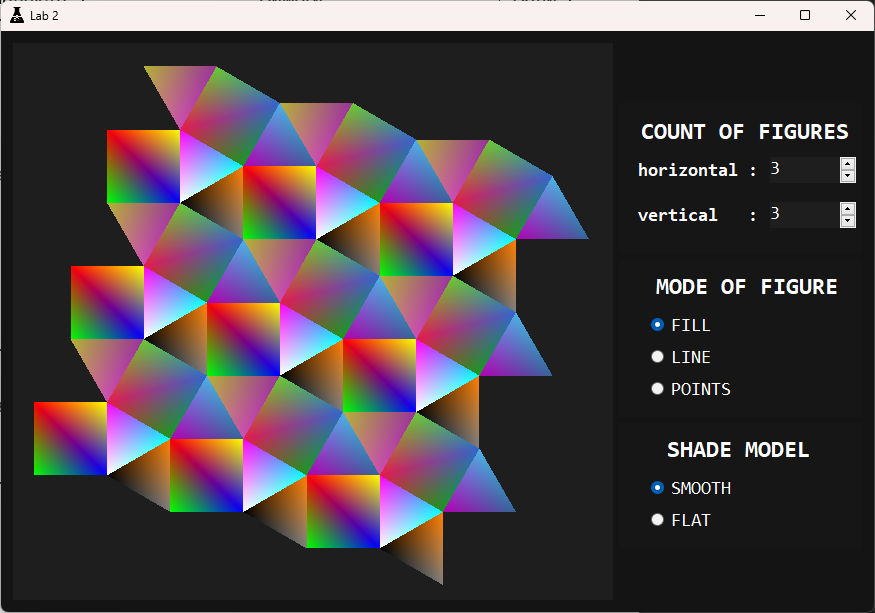


Рисунок 2.6 – Тестування програми при зміні ширини вікна

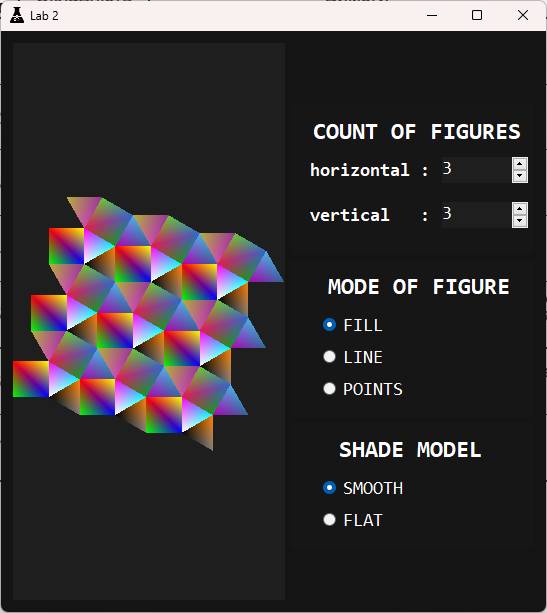


Рисунок 2.7 – Тестування програми при зміні ширини вікна

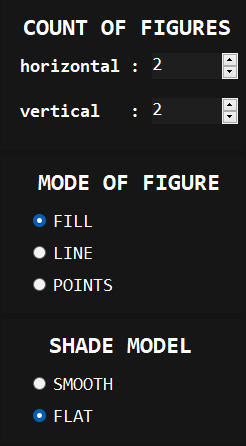


Рисунок 2.8 – Керування параметрами застосунку

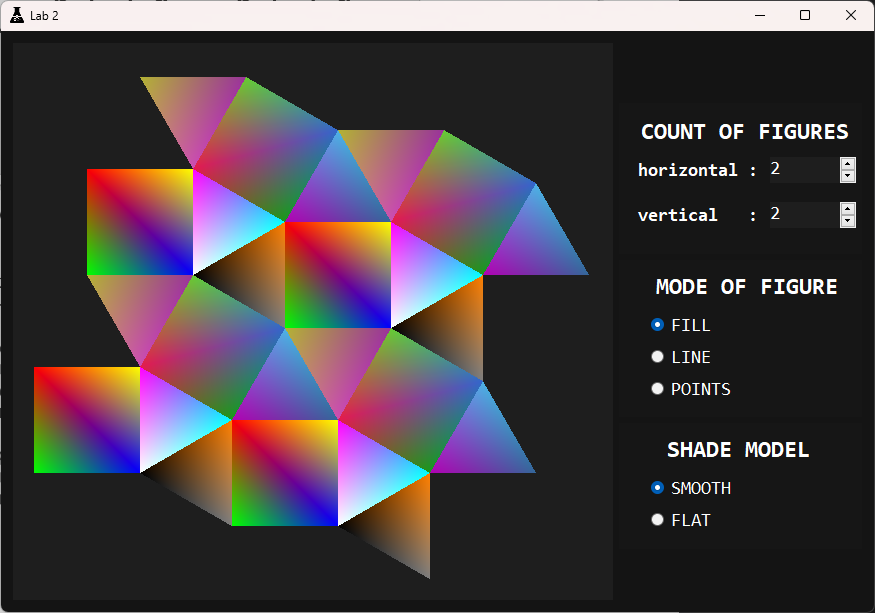


Рисунок 2.9 – Відображення заповнених фігур

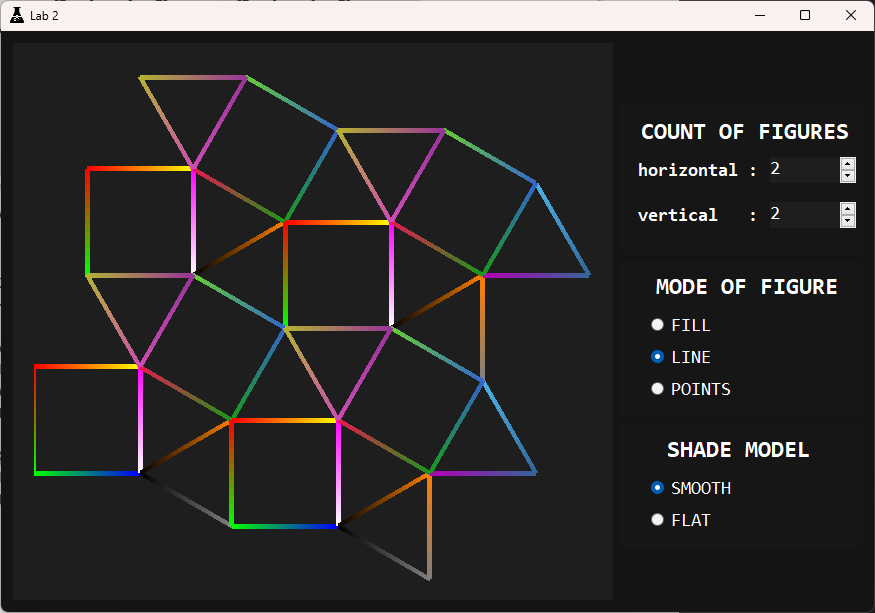


Рисунок 3.0 – Відображення лініями фігур

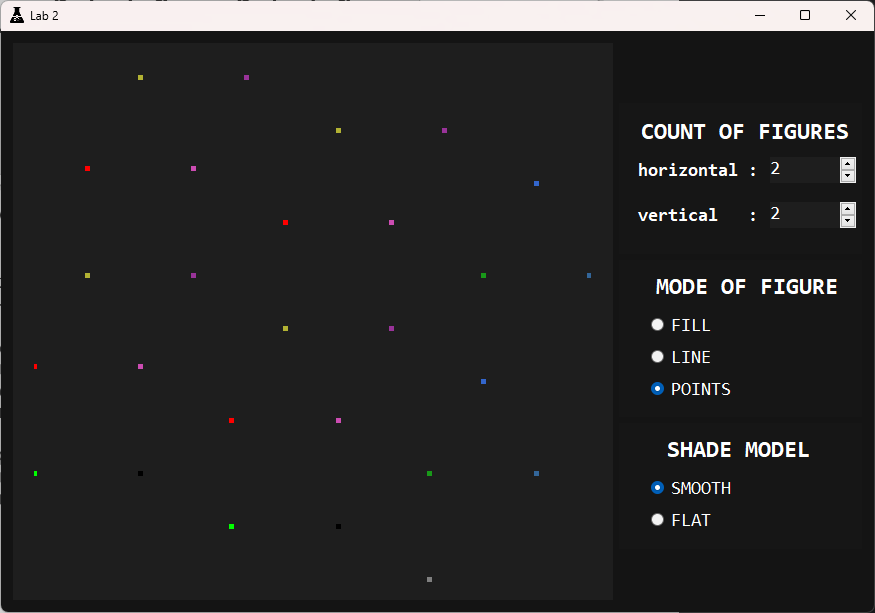


Рисунок 3.1 – Відображення точками фігур

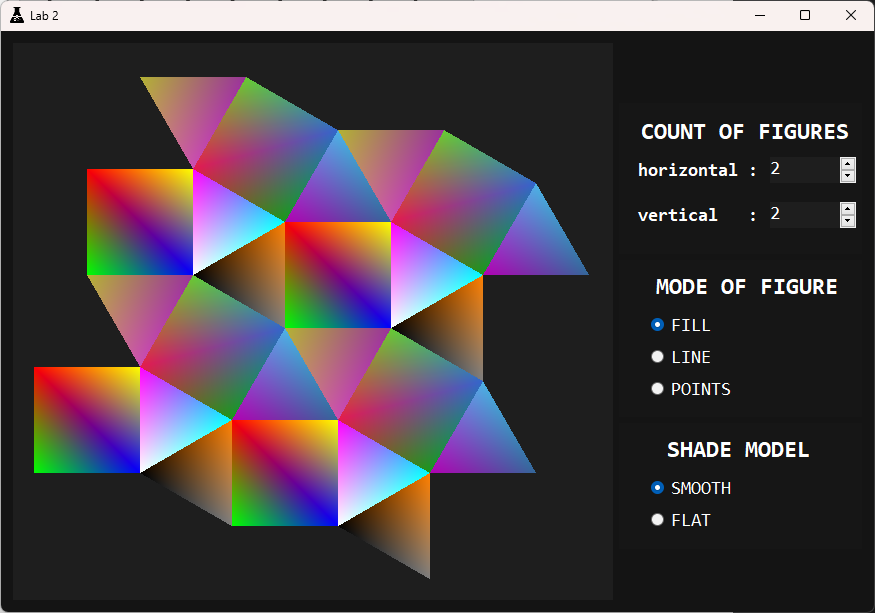


Рисунок 3.2 – Shade model **smooth**

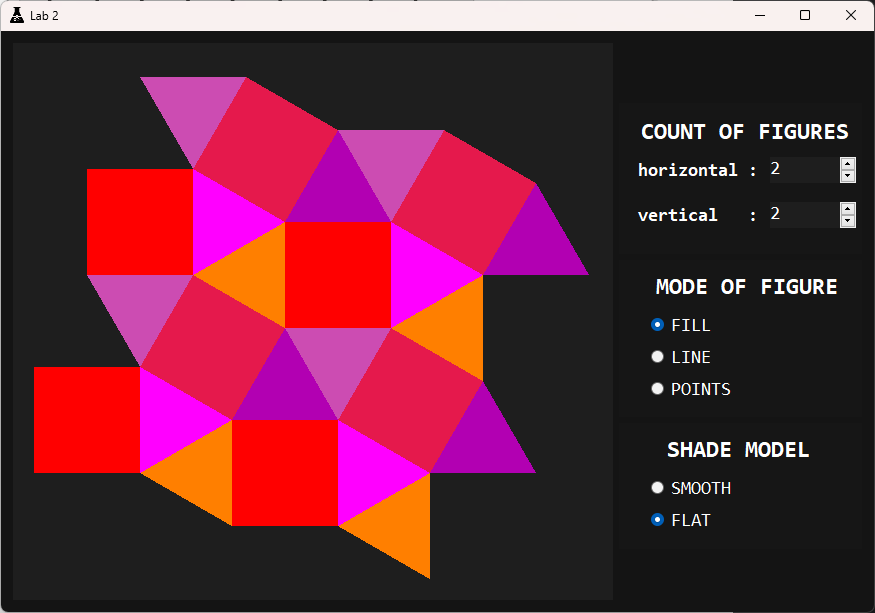


Рисунок 3.3 – Shade model **flat**

### Контроль виконання вимог та елементів завдання

В результаті виконання практичної роботи були повністю виконані елементи базового рівня, що відображено в таблиці 2.1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблиця 2.1 | | | | |
| № з/п | Складність | Вимоги | Бали | Зроблено |
| 1 | Базовий рівень | Під час запуску застосунку зображення відповідає варіанту завдання з однією плиткою (див. рис. 2.1) | 1 | **+** |
| 2 | Багаторазове замощення плиткою (див. рис.2.2). Кратність замощення задається користувачем під час роботи застосунку. | 1 | **+** |
| 3 | Коректне відображення завдання під час зміни як розмірів/положення вікна, так і параметрів замощення | 1 | **+** |
| 4 | Організація взаємодії з користувачем одним зі стандартних засобів (клавіатура, «миша» та ін.) | 1 | **+** |
| 5 | Застосування мінімальної (у рамках варіанту) кількості графічних примітивів для виконання завдання | 1 | **+** |
| 6 | Підвищений рівень | Створення власних елементів інтерфейсу за допомогою OpenGL | 2 | **-** |
| 7 | Використання ООП (розробка власних класів) | 1 | **-** |

# Практична робота 3. Графік функції однієї змінної

## Завдання, варіант № 13

Використовуючи інструментальні засоби, що вказані викладачем,розробити програму для побудови графіка функції виду **y = f(x)** на довільному інтервалі від **Xmin** до **Xmax** і відображення точок перетину функції з віссю абсцис. Крім того, програма повинна мати такі можливості(табл.3.1):

* дозволяти користувачу задавати інтервал від **Xmin** до **Xmax** з перевіркою **Xmin < Xmax**;
* виконувати для завданого користувачем інтервалу від **Xmin** до **Xmax** автоматичне масштабування за віссю **Y** (додатково допускається наявність ручного режиму встановлення **Ymin** і **Ymax**);
* відображати осі координат (та/або координатну сітку) з виводом значень меж видимої області **Xmin**, **Xmax**, **Ymin** і **Ymax**, при цьому система координат повинна бути анізотропною;
* відображати усі точки, де **f(x) = 0**, якщо вони є на завданому інтервалі від **Xmin** до **Xmax**.

Приклад інтерфейсу користувача наведено на рис. 3.1. Варіанти функції **f1(x)** базового рівня складності наведені в табл. 3.2

Для підвищеного рівня складності необхідно додатково реалізувати коректне виведення функції **f2(x)** з урахуванням області визначення функції (варіанти наведені в табл. 3.3) і відобразити лінії розриву (рис. 3.2).

|  |  |
| --- | --- |
| 13 |  |
|  |

## Системна інформація

Для розробки та виконання практичних робіт використовувалися наступні апаратні та програмні засоби:

Processor AMD Ryzen 5 2600 3.40 GHz

RAM 16.0 GB (15.9 GB usable)

System type 64-bit operating system, x64-based processor

Edition Windows 11 Version 23H2

IDE Microsoft Visual Studio Enterprise 2022 (64-bit) version 17.11.5

## Теоретичні відомості

Після того, як користувач у режимі діалогу визначив інтервал по осі X, слід задати кількість точок N, необхідно для побудови графіка функції. Ця кількість може бути задана користувачем явно чи отримана програмно, наприклад, відповідати ширині (кількість пікселів) робочої області. На основі цієї інформації вираховується крок аргументу функції:

і розраховуються координати точок функції в межах завданого користувачем інтервалу:

xi = Xmin + i h; yi = f(xi) ; i ∈ 0. . N − 1.

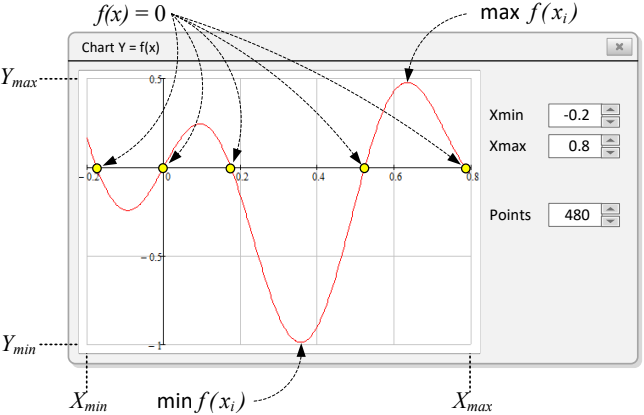


Рис. 3.1. Приклад екранної форми з урахуванням базових вимог до роботи

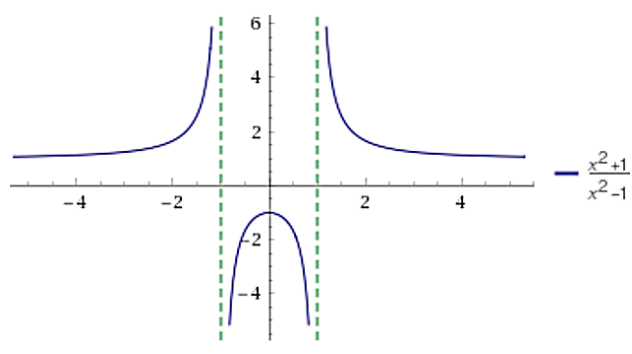


Рис. 3.2. Приклад відображення функції і її ліній розриву (відображені пунктиром)

Використовуючи ці розрахунки, можна знайти межі робочої області по вертикалі:

Ymin = min(yi);

Ymax = max(yi);

i ∈ 0. . N − 1.

Таким чином, знаючи значення меж інтервалу по осі X і обчисливши значення меж по осі Y, можна встановити систему координат (за допомогою команди glOrtho(...)) для виведення графіка y = f(x) на екран.

Знайти корні функції x0 (точки, де f(x0) = 0) можна на основі наступної властивості: якщо на інтервалі від xi до xi+1 є перетин з віссю абсцис, то в результаті примноження відповідних ординат буде виконуватися умова f(xi)f(xi+1) ≤ 0 (рис. 3.3). В цьому випадку координати точок перетину функції з віссю X вираховують спрощеним методом половинного ділення: x0 = (xi + xi+1)⁄2, y0 = f(x0). Якщо брати до уваги дискретність екрану і використовувати для побудови графіка функції кількість точок, що близьке до значення ширини робочої області (в пікселях) чи більше за неї, то спрощений метод половинного ділення дозволить отримати рішення, що візуально не відрізняється від точного.

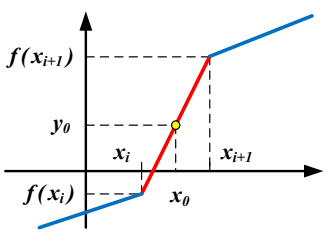


Рис. 3.3. Спрощений варіант методу половинного ділення

Алгоритм коректного відображення на екрані функції f2(x), що має розриви в області визначення, студенти розробляють самостійно. Додатковий аналіз особливостей свого варіанта функції f2(x) можна отримати за допомогою наступних електронних ресурсів [1,2].

## Результати виконання практичної роботи

### Розв'язання завдання

Осі координат і графік функції f1(x) виводяться на заданому користувачем інтервалі від Xmin до Xmax і від Ymin до Ymax. Рисунок **3.4**, рядки **14 - 72** файл **MainForm.cs**, **додаток С**.

Автоматичні обчислення Ymin і Ymax на завданому інтервалі від Xmin до Xmax функції f1(x). Рисунок **3.5**, рядки **81 - 88**, **93 - 103** файл **FuncOne.cs**, **додаток С**.

Обчислення і виведення на екран точок f1(x) = 0. Рисунок **3.6**, рядки **74 - 78**, **106 - 124** файл **FuncOne.cs**, **додаток С**.

Коректне виведення графіка f2(x) (без хибного виводу точок розриву як точок перетину з віссю абсцис) і з відображенням ліній розриву функції. Рисунок **3.7**, рядки **27 - 112**, **114 - 154** файл **FuncTwo.cs**, **додаток С**.

Використання ООП (наслідування). Рядок 1 файл **FuncOne.cs**, Рядок 1 файл **FuncTwo.cs**, **додаток С**.

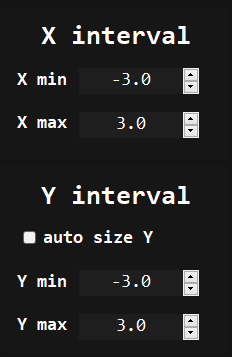


Рисунок 3.4 – Введення інтервалів для X та Y через інтерфейс

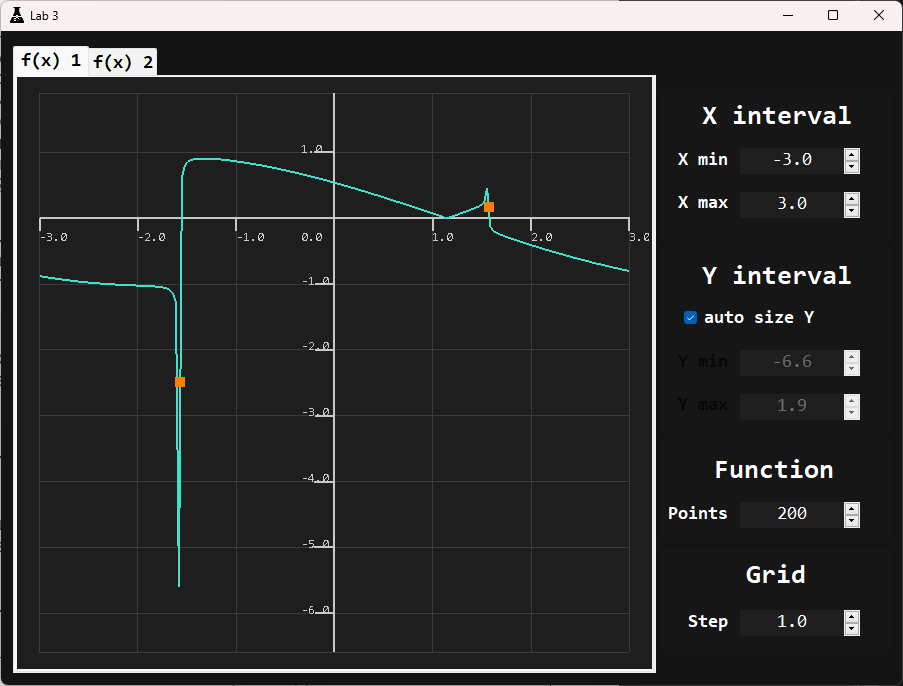


Рисунок 3.5 – Автоматичний розрахунок інтервалів Y

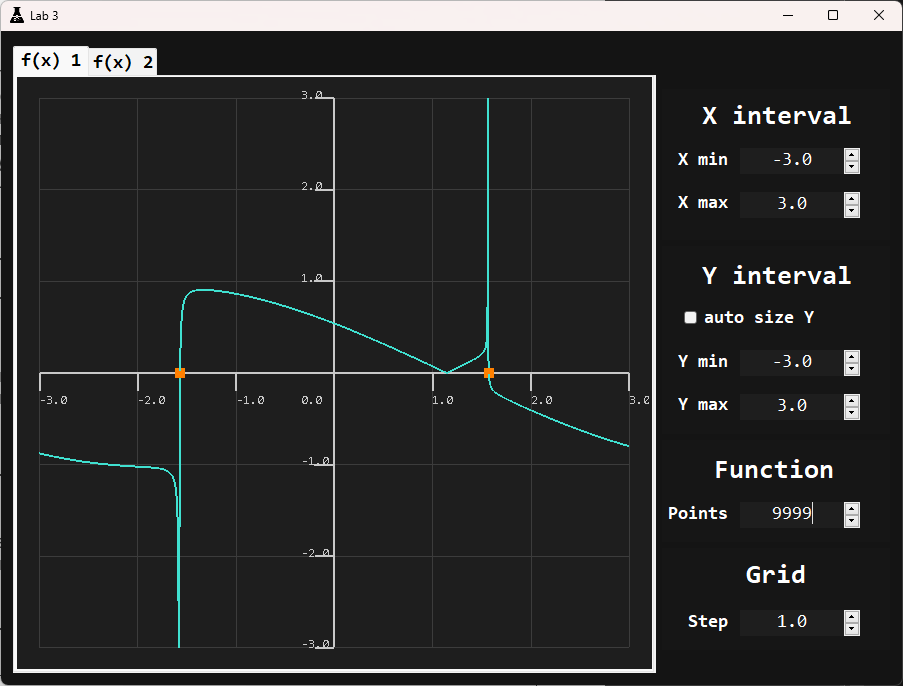


Рисунок 3.6 – Відображення точок (помаранчеві), коли у = 0

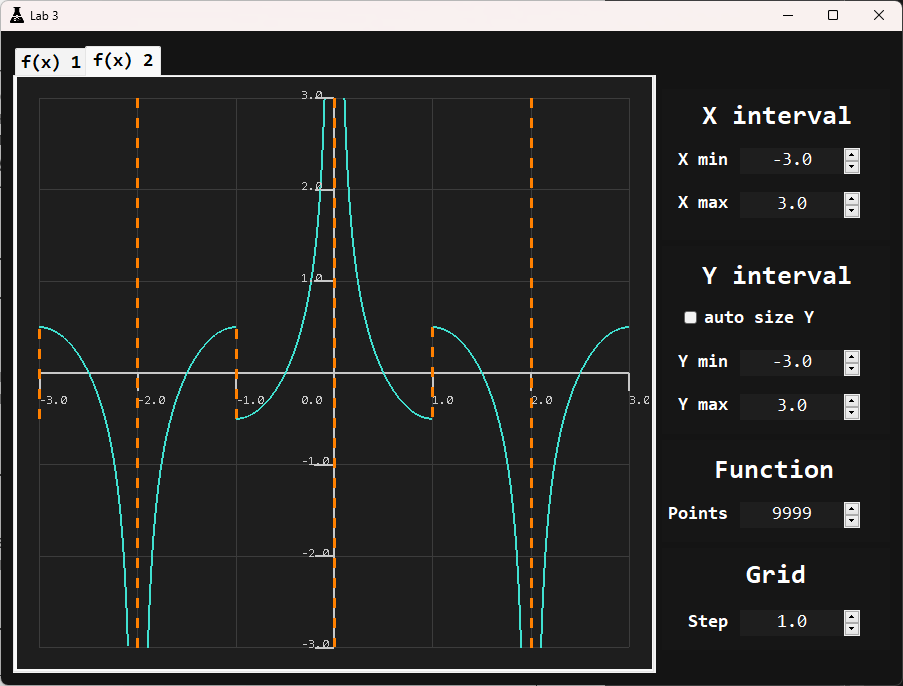


Рисунок 3.7 – Відображення другої функції з лініями разриву (помаранчеві пунктиром)

### Контроль виконання вимог та елементів завдання

В результаті виконання практичної роботи були повністю виконані елементи базового рівня та частково підвищеного рівня складності, що відображено в таблиці 1.1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблиця 3.1 | | | | |
| № з/п | Складність | Вимоги | Бали | Зроблено |
| 1 | Базовий рівень | Осі координат і графік функції f1(x) виводяться на заданому користувачем  інтервалі від Xmin до Xmax і від Ymin до Ymax | 1 | **+** |
| 2 | Автоматичні обчислення Ymin і Ymax на завданому інтервалі від Xmin до Xmax функції f1(x) | 2 | **+** |
| 3 | Обчислення і виведення на екран точок f1(x) = 0 | 2 | **+** |
| 4 | Підвищений рівень | Коректне виведення графіка f2(x) (без хибного виводу точок розриву як точок  перетину з віссю абсцис) і з відображенням ліній розриву функції | 2 | **+** |
| 5 | Використання ООП (наслідування, використання віртуальних і абстрактних методів) | 1 | **+** |

# Практична робота 4. Криві другого порядку

## Завдання, варіант № 13

Використовуючи інструментальні засоби, вказані викладачем, розробити програму для виведення кривих другого порядку на екран (у вікно Windows) за допомогою відрізків. Систему оцінювання наведено в табл. 4.1, а варіанти завдань – в табл. 4.2. Для кривих, які у варіанті відмічені «++», знайти та вивести на екран точки перетину, якщо такі є, з довільним відрізком, координати якого задає користувач.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Окружніть | | Гіпербола | |
| Попадання | | Попадання | |
| явне | параметричне | явне | параметричне |
| 13 | **++** |  |  | **+** |

## Системна інформація

Для розробки та виконання практичних робіт використовувалися наступні апаратні та програмні засоби:

Processor AMD Ryzen 5 2600 3.40 GHz

RAM 16.0 GB (15.9 GB usable)

System type 64-bit operating system, x64-based processor

Edition Windows 11 Version 23H2

IDE Microsoft Visual Studio Enterprise 2022 (64-bit) version 17.11.5

## Теоретичні відомості

Кожну криву другого порядку можна подати як послідовність відрізків. У цьому випадку перетин кривої другого порядку та довільного відрізка можна розглядати як пошук спільної точки [x0, y0] двох відрізків[x1, y1], [x2, y2] и [x3, y3], [x4, y4], заданих у параметричному вигляді (один з яких є фрагментом кривої). Цю задачу можна подати у вигляді системи, що складається з двох лінійних рівнянь з невідомими параметрами t1першого і t2 другого відрізків:

І за умови, що результат рішення буде задовольняти наступні дві умови: 0 ≤ t1 ≤ 1 і 0 ≤ t2 ≤ 1, інакше відрізки або паралельні, або перетинаються тільки прямі, на яких вони лежать [1, 2].

## Результати виконання практичної роботи

### Розв'язання завдання

Установлення ізотропної системи координат для вікна з змінюваними розмірами. Рисунок **4.1 - 4.2**, рядки **3 - 11**, **83** файл **RenderControl.cs**, **додаток D**.

Виведення кривих другого порядку відповідно до варіанту завдання. Файл **RenderControl.cs**, **додаток D**.

* Окружність, рисунок **4.3**, рядки **211 - 251**
* Гіпербола, рисунок **4.4**, рядки **318 - 357**

Виведення відрізка та обчислення його точок перетину з кривою другого порядку відповідно до варіанту. Рисунок **4.5**, файл **RenderControl.cs**, **додаток D**.

* Лінія, рядки **253 - 264**
* Точка, рядки **266 - 316**
* Малювання лінії в окружності, рядки **239 - 243**

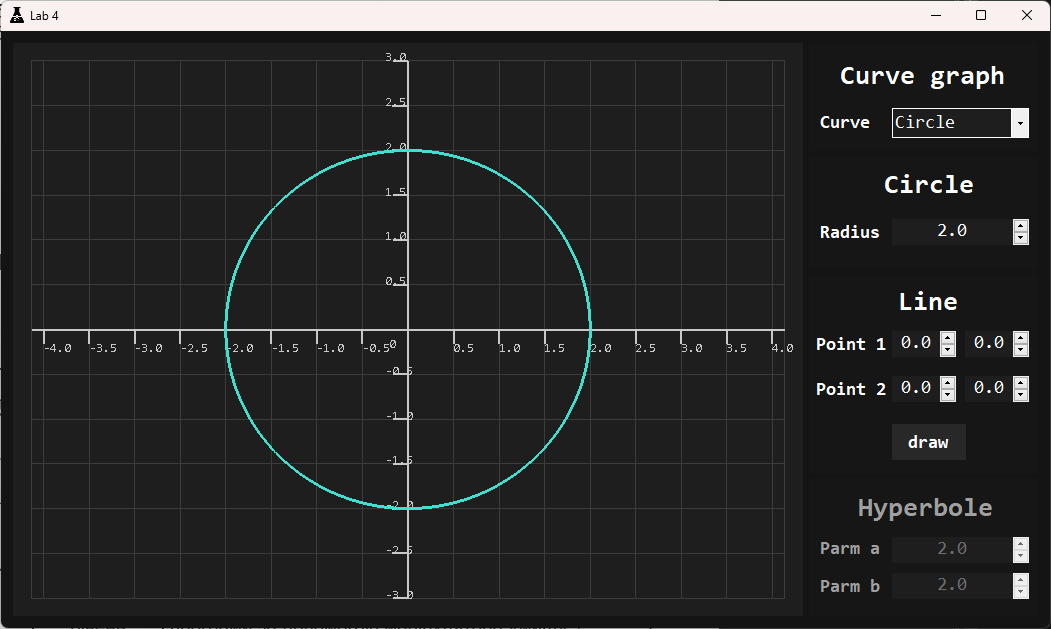


Рисунок 4.1 – Тестування програми при зміні ширини вікна

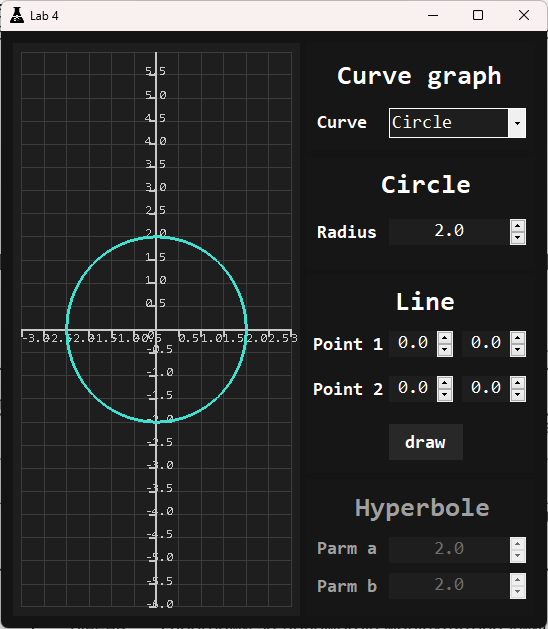


Рисунок 4.2 – Тестування програми при зміні ширини вікна

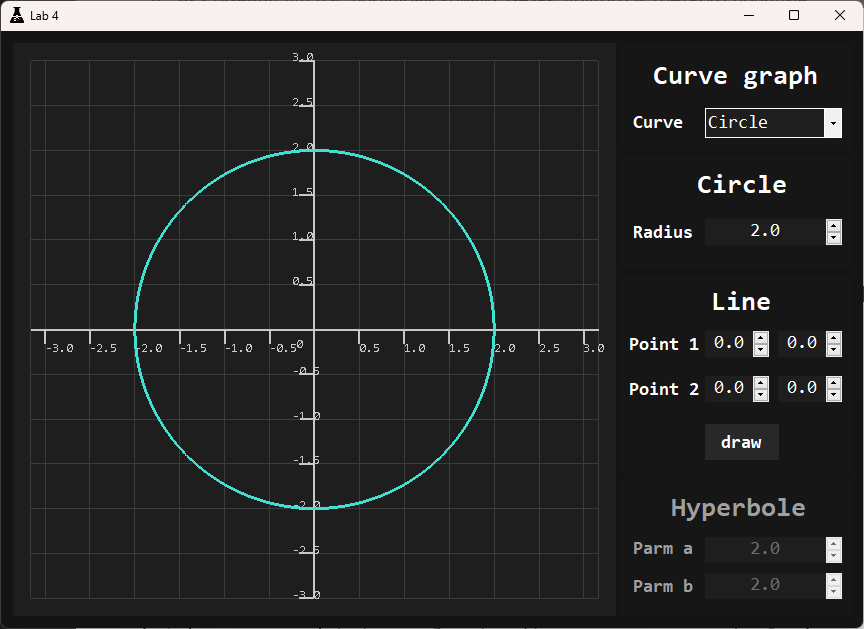


Рисунок 4.3 – Крива другого порядку, окружність

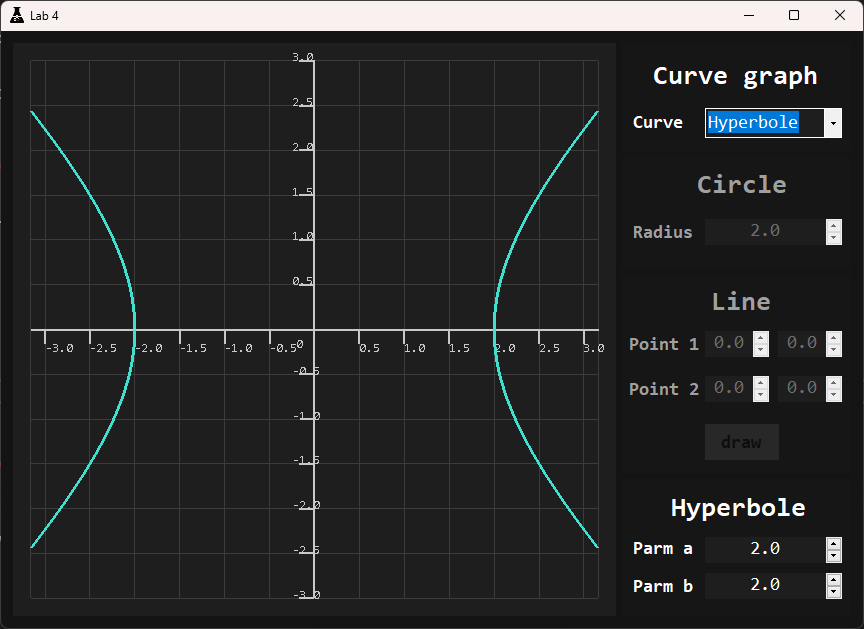


Рисунок 4.3 – Крива другого порядку, гіпербола

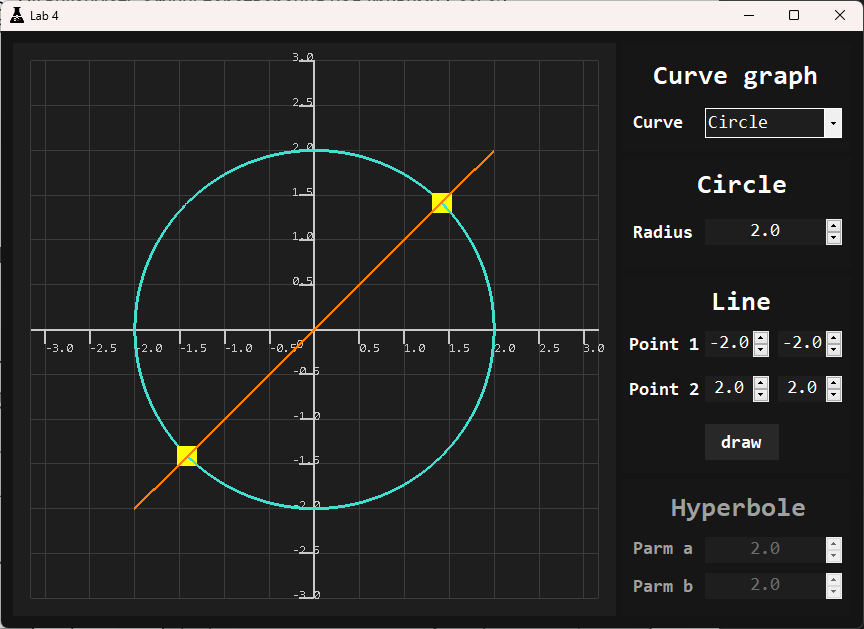


Рисунок 4.4 – Відображення відрізка та точок перетину з окружністю

### Контроль виконання вимог та елементів завдання

В результаті виконання практичної роботи були повністю виконані елементи базового рівня та частково підвищеного рівня складності, що відображено в таблиці 4.1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблиця 4.1 | | | | |
| № з/п | Складність | Вимоги | Бали | Зроблено |
| 1 | Базовий рівень | Установлення ізотропної системи координат для вікна з змінюваними розмірами | 1 | **+** |
| 2 | Виведення кривих другого порядку відповідно до варіанту завдання | 2 | **+** |
| 3 | Виведення відрізка та обчислення його точок перетину з кривою другого порядку відповідно до варіанту | 2 | **+** |
| 4 | Підвищений рівень | Вказання положення точок, що формують відрізок, в області графічного виведення програми за допомогою маніпулятора «миш» | 2 | **-** |
| 5 | Використання ООП | 1 | **-** |

# Практична робота 5. Квадратичні примітиви. Афінні перетворення у просторі

## Завдання, варіант № 13

Використовуючи інструментальні засоби, вказані викладачем, розробити програму з використання засобів OpenGL, яка встановлює ізотропну систему координат, створює і виводить зображення тривимірної сцени з такими елементами (систему оцінки наведено в табл. 5.1, а варіанти завдань – в табл.5.2):

* осі координат з нулем у центрі екрана та вказанням осі та додатного напрямку;
* координатна сітка (**grid**) в одній з площин (**X0Y**, **X0Z** чи **Y0Z**);
* три квадратичні фігури – **gluDisk** / **gluPartialDisk**, **gluSphere**, **gluCylinder** в режимі відображення каркаса і з спрощеною моделлю освітлення **glEnable(GL\_COLOR\_MATERIAL)** для базового рівня складності;
* площина відтину для однієї з фігур (сфера, циліндр чи конус);
* повноцінна модель освітлення та/або текстурами для реалізації завдання з підвищеною складністю.

Мінімальний інтерфейс користувача повинен забезпечувати можливості повороту сцени відносно осей **OX** і **OY** за допомогою маніпулятора «миш» і керування параметрами площини відтину [1, 2]. Параметри деталізації об’єктів (**slices**, **stacks**), кольору, товщини і типу ліній обирають самостійно. Приклад сцени показано на рис. 5.1.

|  |  |  |
| --- | --- | --- |
| 13 | Примітиви:  GL\_POINTS, GL\_LINES  x1 = -1; x2 = 8  y1 = -2; y2 = 2 |  |

## Системна інформація

Для розробки та виконання практичних робіт використовувалися наступні апаратні та програмні засоби:

Processor AMD Ryzen 5 2600 3.40 GHz

RAM 16.0 GB (15.9 GB usable)

System type 64-bit operating system, x64-based processor

Edition Windows 11 Version 23H2

IDE Microsoft Visual Studio Enterprise 2022 (64-bit) version 17.11.5

## Теоретичні відомості

Ізотропну систему координат можна встановлювати двома способами. В першому випадку за допомогою команди glViewport задають робочу область відповідного меншого значення ширини/висоти вікна, а у другому – вводять множник (або дільник), що коректує і рівний відношенню ширини і висоти вікна під час встановлення системи координат (наприклад, командою glOrtho). Глибину потрібно задавати таким чином, щоб за будь-якого розміщення заданої сцени усі об’єкти знаходилися всередині видимої області.

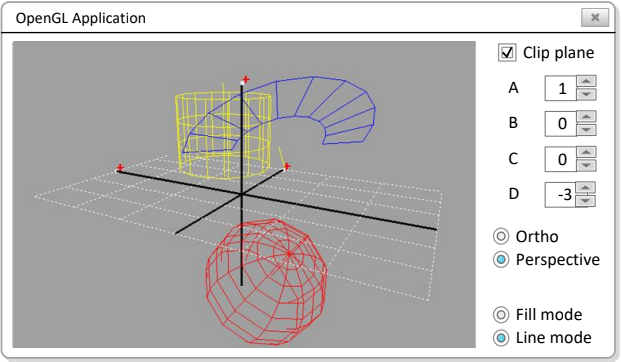


Рис. 5.1. Приклад сцени з квадратичними об’єктами і плоскістю відтину для сфери

Для відображення квадратичних примітивів використовують точку прив’язки об’єкта. В загальному випадку за відсутності афінних перетворень ця точка прив’язки об’єкта збігається з початком системи координат. Точкою прив’язки для сфери і диска (повного і часткового) виступає їхній центр, а для циліндра (конуса) – центр однієї з основ. Для розміщення кожного з трьох квадратичних об’єктів відповідно варіанту завдання необхідно скористатися одним з афінних перетворень або їхніх комбінацій: поворот, перенос, масштабування. Після виконання перетворень точка прив’язки повинна знаходитися на координатах x0, y0, z0, а вісь об’єкта паралельна (∥) для сфери/диска або колінеарна (⇈, ⇅) для циліндра / конуса з урахуванням варіанта. Під час формування зображення квадратичних примітивів використовують такі параметри:

* **x0, y0, z0** – координати точки прив’язки фігури;
* **R** – радіус сфери або радіус основи циліндра/конуса з центром у точці прив’язки або зовнішній радіус диска;
* **r** – радіус другої основи циліндра / усіченого конуса; внутрішній радіус диска;
* **h** – висота циліндра / конуса / усіченого конуса;
* **∠start** – початковий кут частинного диска;
* **∠sweep** – кут розгортки для частинного диска;
* ось **||**,**⇈**,**⇅** – паралельність / колінеарність осей координат і фігури.

## Результати виконання практичної роботи

### Розв'язання завдання

Рисунок 1.3 – Тестування програми при зміні ширини вікна

Рисунок 1.4 – Тестування програми при зміні ширини вікна

### Контроль виконання вимог та елементів завдання

В результаті виконання практичної роботи були повністю виконані елементи базового рівня та частково підвищеного рівня складності, що відображено в таблиці 1.1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблиця 1.1 | | | | |
| № з/п | Складність | Вимоги | Бали | Зроблено |
| 1 | Базовий рівень | Коректне (ізотропне) відображення завдання (під час зміни розмірів вікна) у ортографічній проекції | 1 | **+** |
| 2 | Під час запуску застосунку відображаються осі 0X, 0Y, 0Z, координатна сітка і каркас квадратичних об’єктів | 1 | **+** |
| 3 | Інтерфейс керування параметрами площини відтину | 1 | **-** |
| 4 | Використання джерел світла для освітлення об’єктів сцени сумісно з командою glColorMaterial | 1 |  |
| 5 | Використання списків відображення (Display Lists) | 1 | **+** |
| 6 | Підвищений рівень | Створення зображення сцени в перспективній проекції | 1 | **-** |
| 7 | Накладення текстури на поверхню завданих у варіанті фігур | 1 |  |
| 8 | Застосування команди glMaterial для налаштування параметрів відбиття поверхонь об’єктів сцени | 1 | **+** |

# Загальний перелік посилань

1. Microsoft. glDrawArrays function [Електронний ресурс] / Microsoft – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/windows/win32/opengl/gldrawarrays>.
2. Вивчення розривних функцій за допомогою Wolfram | Alpha [Електронний ресурс] // Математика, статистика, анализ данных в WolframAlpha. – Режим доступу : <http://www.wolframalpha-ru.com/2012/06/wolframalpha_16.html.>
3. Discontinuities [Електронний ресурс] // WolframAlpha Computational intelligence. – Режим доступу : <https://www.wolframalpha.com/input/?i=discontinuities+1%2F(x-1).>
4. Роджерс, Д. Математические основы машинной графики / Д. Роджерс, Дж. Адамс ; пер. с англ. – М. : Мир, 2001. – 604 с.
5. Препарата, Ф. Вычислительная геометрия : Введение / Ф. Препарата, М. Шеймос. – М. : Мир, 1989. – 478 с.
6. Аммерал, Л. Интерактивная трехмерная машинная графика / Л. Аммерал. – М. : «Сол Систем», 1992. – 315 с.
7. Шикин, Е. В. Компьютерная графика. Динамика, реалистические изображения / Е. В. Шикин, А. В. Боресков. – М. : Диалог-МИФИ, 1996. – 288 с.

# Додаток А. Лістинг програми до практичної роботи №1

### Код файлу (MainForm.cs)

1. public partial class MainForm : Form
2. {
3. public MainForm()
4. {
5. InitializeComponent();
6. }

### Код файлу (RenderControl.cs)

1. public partial class RenderControl : OpenGL
2. {
3. int min\_x = -1, max\_x = 8;
4. int min\_y = -2, max\_y = 2;
5. Draw draw
6. public static OutText OutText { get; set; }
8. public RenderControl()
9. {
10. InitializeComponent();
11. }
12. private void RenderControl\_Render(object sender, EventArgs e)
13. {
14. glClear(GL\_COLOR\_BUFFER\_BIT);
15. glLoadIdentity();
16. glViewport(0, 0, Width, Height);
17. glOrtho(min\_x - 2, max\_x + 2, min\_y - 2, max\_y + 2, -1, 1);
18. draw.Drawing();
19. DrawText("X1", min\_x, min\_y - 1.5);
20. DrawText("X2", max\_x, min\_y - 1.5);
21. DrawText("Y1", min\_x - 1.5, min\_y);
22. DrawText("Y2", min\_x - 1.5, max\_y);
23. }
24. private void RenderControl\_ContextCreated(object sender, EventArgs e)
25. {
26. draw = new Draw(min\_x, max\_x, min\_y, max\_y);
27. }
28. }

### Код файлу (Draw.cs)

1. public class Draw
2. {
3. private int min\_x, max\_x, min\_y, max\_y;
4. public Draw(int MinX, int MaxX, int MinY, int MaxY)
5. {
6. min\_x = MinX;
7. max\_x = MaxX;
8. min\_y = MinY;
9. max\_y = MaxY;
10. }
11. public void Drawing()
12. {
13. //glColor3f(0.0f / 255.0f, 0.0f / 255.0f, 0.0f / 255.0f);
14. DrawGrid();
15. DrawAxis();
16. DrawFigure();
17. DrawPorintsFigure();
18. }
19. private void DrawGrid()
20. {
21. glLineStipple(1, 4369);
22. glEnable(GL\_LINE\_STIPPLE);
23. glLineWidth(2);
24. glBegin(GL\_LINES);
25. // draw horizontal lines
26. for (float i = min\_x; i <= max\_x; i += 1)
27. {
28. glVertex2d(i, min\_y - 1);
29. glVertex2d(i, max\_y + 1);
30. }
31. // draw vertical lines
32. for (float i = min\_y; i <= max\_y; i += 1)
33. {
34. glVertex2d(min\_x - 1, i);
35. glVertex2d(max\_x + 1, i);
36. }
37. glEnd();
38. glDisable(GL\_LINE\_STIPPLE);
39. glLineWidth(1);
40. }
41. private void DrawAxis()
42. {
43. glLineWidth(4);
44. glBegin(GL\_LINES);
45. // Y
46. glVertex2d(min\_x - 1, min\_y - 0.2);
47. glVertex2d(min\_x - 1, max\_y + 1);
48. // arrow
49. glVertex2d(min\_x - 1, max\_y + 1);
50. glVertex2d(min\_x - 1 - 0.2, max\_y + 1 - 0.2);
51. glVertex2d(min\_x - 1, max\_y + 1);
52. glVertex2d(min\_x - 1 + 0.2, max\_y + 1 - 0.2);
53. // strokes
54. for (int i = min\_y; i <= max\_y; i++)
55. {
56. glVertex2d(min\_x - 1, i);
57. glVertex2d(min\_x - 1 - 0.2, i);
58. }
59. // X
60. glVertex2d(min\_x - 0.2, min\_y - 1);
61. glVertex2d(max\_x + 1, min\_y - 1);
62. // arrow
63. glVertex2d(max\_x + 1, min\_y - 1);
64. glVertex2d(max\_x + 1 - 0.2, min\_y - 1 - 0.2);
65. glVertex2d(max\_x + 1, min\_y - 1);
66. glVertex2d(max\_x + 1 - 0.2, min\_y - 1 + 0.2);
67. // strokes
68. for (int i = min\_x; i <= max\_x; i++)
69. {
70. glVertex2d(i, min\_y - 1);
71. glVertex2d(i, min\_y - 1 - 0.2);
72. }
73. glEnd();
74. glLineWidth(1);
75. }
76. private void DrawFigure()
77. {
78. glLineWidth(8);
79. glBegin(GL\_LINES);
80. glVertex2d(-1, 2);
81. glVertex2d(-1, 0);
82. glVertex2d(-1, 0);
83. glVertex2d(0, -1);
84. glVertex2d(0, -1);
85. glVertex2d(2, -2);
86. glVertex2d(2, -2);
87. glVertex2d(2, -1);
88. glVertex2d(2, -1);
89. glVertex2d(1, 1);
90. glVertex2d(1, 1);
91. glVertex2d(0, 2);
92. glVertex2d(0, 2);
93. glVertex2d(-1, 2);
94. glEnd();
95. glLineWidth(1);
96. }
97. private void DrawPorintsFigure()
98. {
99. glPointSize(16);
100. glEnable(GL\_POINT\_SMOOTH);
101. glBegin(GL\_POINTS);
102. glVertex2d(4, 2);
103. glVertex2d(4, 0);
104. glVertex2d(5, -1);
105. glVertex2d(7, -2);
106. glVertex2d(7, -1);
107. glVertex2d(6, 1);
108. glVertex2d(5, 2);
109. glEnd();
110. glDisable(GL\_POINT\_SMOOTH);
111. glPointSize(1);
112. }
113. }

# Додаток B. Лістинг програми до практичної роботи №2

### Код файлу (MainForm.cs)

1. public partial class MainForm : Form
2. {
3. public MainForm()
4. {
5. InitializeComponent();
6. }
7. private void MainForm\_Load(object sender, System.EventArgs e)
8. {
9. renderControl1.Start();
10. }
11. private void horValue\_ValueChanged(object sender, System.EventArgs e)
12. {
13. renderControl1.SetHorValue((int)horValue.Value);
14. renderControl1.Refresh();
15. }
16. private void verValue\_ValueChanged(object sender, System.EventArgs e)
17. {
18. renderControl1.SetVerValue((int)verValue.Value);
19. renderControl1.Refresh();
20. }
21. private void fillMode\_CheckedChanged(object sender, EventArgs e)
22. {
23. renderControl1.SetPolygonMode(GL\_FILL);
24. }
25. private void lineMode\_CheckedChanged(object sender, EventArgs e)
26. {
27. renderControl1.SetPolygonMode(GL\_LINE);
28. }
29. private void pointsMode\_CheckedChanged(object sender, EventArgs e)
30. {
31. renderControl1.SetPolygonMode(GL\_POINT);
32. }
33. private void smoothModel\_CheckedChanged(object sender, EventArgs e)
34. {
35. renderControl1.SetShadeMode(GL\_SMOOTH);
36. }
37. private void flatModel\_CheckedChanged(object sender, EventArgs e)
38. {
39. renderControl1.SetShadeMode(GL\_FLAT);
40. }
41. }

### Код файлу (RenderControl.cs)

1. public partial class RenderControl : OpenGL
2. {
3. private int count\_hor;
4. private int count\_vert;
5. private float a;
6. private float size\_plit;
7. private float size;
8. private uint polygonMode;
9. private uint shadeModel;
10. private float min\_x, max\_x;
11. private float min\_y, max\_y;
12. public RenderControl()
13. {
14. InitializeComponent();
15. }
16. // start value
17. public void Start()
18. {
19. a = 0.2f; // size side of figure
20. size\_plit = 0.575f;
21. count\_hor = 1;
22. count\_vert = 1;
23. polygonMode = GL\_FILL;
24. shadeModel = GL\_SMOOTH;
25. UpdateSizeScreen();
26. }
27. // set sizze screen
28. public void UpdateSizeScreen()
29. {
30. float modX = 0, modY = 0, exMod = 0;
31. if(count\_hor >= count\_vert)
32. {
33. exMod = (a / 2) \* (count\_vert - 1);
34. modX = (0.375f \* (count\_hor - 1));
35. modY = modX + exMod;
36. min\_x = -a - exMod;
37. max\_x = 0.375f + modX;
38. min\_y = -0.3375f - modY;
39. max\_y = 0.2375f;
40. }
41. else if(count\_hor < count\_vert)
42. {
43. modX = (a / 2) \* (count\_vert - 1);
44. modY = (0.375f \* (count\_vert - 1)) + modX;
45. min\_x = -a - modX;
46. max\_x = 0.375f + (modY - modX);
47. min\_y = -0.3375f - modY;
48. max\_y = 0.2375f;
49. }
50. }
51. // set count horizontal
52. public void SetHorValue(int value)
53. {
54. count\_hor = value;
55. UpdateSizeScreen();
56. }
58. // set count vertival
59. public void SetVerValue(int value)
60. {
61. count\_vert = value;
62. UpdateSizeScreen();
63. }
64. public void SetPolygonMode(uint value)
65. {
66. polygonMode = value;
67. Refresh();
68. }
69. public void SetShadeMode(uint value)
70. {
71. shadeModel = value;
72. Refresh();
73. }
74. private void RenderControl\_Render(object sender, EventArgs e)
75. {
76. glClear(GL\_COLOR\_BUFFER\_BIT);
77. glLoadIdentity();
78. // Устанавливаем квадратную область просмотра, сохраняя пропорции
79. if (Width > Height)
80. {
81. // Если ширина больше высоты, центрируем по горизонтали
82. glViewport((Width - Height) / 2, 0, Height, Height);
83. }
84. else
85. {
86. // Если высота больше ширины, центрируем по вертикали
87. glViewport(0, (Height - Width) / 2, Width, Width);
88. }
89. glOrtho(min\_x, max\_x, min\_y, max\_y, -1, 1);
90. for (int vert = 0; vert < count\_vert; vert++)
91. {
92. for (int hor = 0; hor < count\_hor; hor++)
93. {
94. DrawFidures(vert, hor);
95. }
96. }
97. }
99. private void DrawFidures(int vert, int hor)
100. {
101. float coef\_x = ((((a \* MathF.Sqrt(3)) / 2) + a) \* hor) - ((a / 2) \* vert);
102. float coef\_y = ((-a / 2) \* hor) - ((((a \* MathF.Sqrt(3)) / 2) + a) \* vert);
103. float pos\_x = 0 + coef\_x;
104. float pos\_y = 0 + coef\_y;
105. glPolygonMode(GL\_FRONT\_AND\_BACK, polygonMode);
106. glShadeModel(shadeModel);
107. glPointSize(5);
108. glLineWidth(5);
109. // figure 1
110. glBegin(GL\_POLYGON);
111. glColor3f(1f, 0f, 0f); // Красный
112. glVertex2d(pos\_x - a, pos\_y);
113. glColor3f(0f, 1f, 0f); // Зеленый
114. glVertex2d(pos\_x - a, pos\_y - a);
115. glColor3f(0f, 0f, 1f); // Синий
116. glVertex2d(pos\_x, pos\_y - a);
117. glColor3f(1f, 1f, 0f); // Желтый
118. glVertex2d(pos\_x, pos\_y);
119. glEnd();
120. // figure 2
121. glBegin(GL\_POLYGON);
122. glColor3f(1f, 0f, 1f); // Фиолетовый
123. glVertex2d(pos\_x, pos\_y);
124. glColor3f(0f, 1f, 1f); // Голубой
125. glVertex2d(pos\_x + ((a \* Math.Sqrt(3)) / 2), pos\_y - (a / 2));
126. glColor3f(1f, 1f, 1f); // Белый
127. glVertex2d(pos\_x, pos\_y - a);
128. glEnd();
129. // figure 3
130. glBegin(GL\_POLYGON);
131. glColor3f(1f, 0.5f, 0f); // Оранжевый
132. glVertex2d(pos\_x + ((a \* Math.Sqrt(3)) / 2), pos\_y - (a / 2));
133. glColor3f(0.5f, 0.5f, 0.5f); // Серый
134. glVertex2d(pos\_x + ((a \* Math.Sqrt(3)) / 2), pos\_y + ((-a / 2) - a));
135. glColor3f(0f, 0f, 0f); // Черный
136. glVertex2d(pos\_x, pos\_y - a);
137. glEnd();
138. // figure 4
139. glBegin(GL\_POLYGON);
140. glColor3f(0.7f, 0f, 0.7f); // Фиолетово-розовый
141. glVertex2d(pos\_x + ((a \* Math.Sqrt(3)) / 2), pos\_y - (a / 2));
142. glColor3f(0.2f, 0.4f, 0.6f); // Сине-зеленый
143. glVertex2d(pos\_x + (((a \* Math.Sqrt(3)) / 2) + a), pos\_y - (a / 2));
144. glColor3f(0.3f, 0.7f, 0.9f); // Голубовато-зеленый
145. glVertex2d(pos\_x + (((a \* Math.Sqrt(3)) / 2) + a / 2), (pos\_y - (a / 2)) + ((a \* Math.Sqrt(3)) / 2));
146. glEnd();
147. // figure 5
148. glBegin(GL\_POLYGON);
149. glColor3f(0.9f, 0.1f, 0.3f); // Темно-красный
150. glVertex2d(pos\_x, pos\_y);
151. glColor3f(0.4f, 0.8f, 0.2f); // Светло-зеленый
152. glVertex2d(pos\_x + (a / 2), pos\_y + ((a \* Math.Sqrt(3)) / 2));
153. glColor3f(0.2f, 0.4f, 0.8f); // Синий
154. glVertex2d(pos\_x + (((a \* Math.Sqrt(3)) / 2) + a / 2), (pos\_y - (a / 2)) + ((a \* Math.Sqrt(3)) / 2));
155. glColor3f(0.1f, 0.6f, 0.1f); // Зеленый
156. glVertex2d(pos\_x + ((a \* Math.Sqrt(3)) / 2), pos\_y - (a / 2));
157. glEnd();
158. // figure 6
159. glBegin(GL\_POLYGON);
160. glColor3f(0.8f, 0.3f, 0.7f); // Лиловый
161. glVertex2d(pos\_x, pos\_y);
162. glColor3f(0.7f, 0.7f, 0.2f); // Желтоватый
163. glVertex2d(pos\_x - (a / 2), pos\_y + ((a \* Math.Sqrt(3)) / 2));
164. glColor3f(0.6f, 0.2f, 0.6f); // Фиолетовый
165. glVertex2d(pos\_x + (a / 2), pos\_y + ((a \* Math.Sqrt(3)) / 2));
166. glEnd();
167. }
168. }

# Додаток C. Лістинг програми до практичної роботи №3

### Код файлу (MainForm.cs)

1. public partial class MainForm : Form
2. {
3. private float step = 0.1f;
4. public MainForm()
5. {
6. InitializeComponent();
7. // set mix max value for X numerics
8. Xmax\_numeric.Minimum = Xmin\_numeric.Value + (decimal)step;
9. Xmin\_numeric.Maximum = Xmax\_numeric.Value - (decimal)step;
10. }
11. private void Xmin\_numeric\_ValueChanged(object sender, System.EventArgs e)
12. {
13. Xmax\_numeric.Minimum = Xmin\_numeric.Value + (decimal)step;
14. switch (tabControl1.SelectedIndex)
15. {
16. case 0:
17. renderControl1.SetXmin((float)Xmin\_numeric.Value);
18. break;
19. case 1:
20. renderControl2.SetXmin((float)Xmin\_numeric.Value);
21. break;
22. }
23. }
24. private void Xmax\_numeric\_ValueChanged(object sender, System.EventArgs e)
25. {
26. Xmin\_numeric.Maximum = Xmax\_numeric.Value - (decimal)step;
27. switch (tabControl1.SelectedIndex)
28. {
29. case 0:
30. renderControl1.SetXmax((float)Xmax\_numeric.Value);
31. break;
32. case 1:
33. renderControl2.SetXmax((float)Xmax\_numeric.Value);
34. break;
35. }
36. }
37. private void Ymin\_numeric\_ValueChanged(object sender, System.EventArgs e)
38. {
39. switch (tabControl1.SelectedIndex)
40. {
41. case 0:
42. renderControl1.SetYmin((float)Ymin\_numeric.Value);
43. break;
44. case 1:
45. renderControl2.SetYmin((float)Ymin\_numeric.Value);
46. break;
47. }
48. }
49. private void Ymax\_numeric\_ValueChanged(object sender, System.EventArgs e)
50. {
51. switch (tabControl1.SelectedIndex)
52. {
53. case 0:
54. renderControl1.SetYmax((float)Ymax\_numeric.Value);
55. break;
56. case 1:
57. renderControl2.SetYmax((float)Ymax\_numeric.Value);
58. break;
59. }
60. }
61. private void Points\_numeric\_ValueChanged(object sender, System.EventArgs e)
62. {
63. switch (tabControl1.SelectedIndex)
64. {
65. case 0:
66. renderControl1.SetPoints((int)Points\_numeric.Value);
67. break;
68. case 1:
69. renderControl2.SetPoints((int)Points\_numeric.Value);
70. break;
71. }
72. }
73. private void Step\_numeric\_ValueChanged(object sender, System.EventArgs e)
74. {
75. switch (tabControl1.SelectedIndex)
76. {
77. case 0:
78. renderControl1.SetStep((float)Step\_numeric.Value);
79. break;
80. case 1:
81. renderControl2.SetStep((float)Step\_numeric.Value);
82. break;
83. }
84. }
85. private void tabControl1\_SelectedIndexChanged(object sender, System.EventArgs e)
86. {
87. switch (tabControl1.SelectedIndex)
88. {
89. case 0:
90. renderControl1.Start(Ymin\_numeric,
91. Ymax\_numeric,
92. tabControl1.SelectedIndex,
93. autoSizeY\_checkBox.Checked,
94. (float)(Xmin\_numeric.Value),
95. (float)(Xmax\_numeric.Value),
96. (float)(Ymin\_numeric.Value),
97. (float)(Ymax\_numeric.Value),
98. 0.2f,
99. (float)(Step\_numeric.Value),
100. (int)(Points\_numeric.Value));
101. break;
102. case 1:
103. renderControl2.Start(Ymin\_numeric,
104. Ymax\_numeric,
105. tabControl1.SelectedIndex,
106. autoSizeY\_checkBox.Checked,
107. (float)(Xmin\_numeric.Value),
108. (float)(Xmax\_numeric.Value),
109. (float)(Ymin\_numeric.Value),
110. (float)(Ymax\_numeric.Value),
111. 0.2f,
112. (float)(Step\_numeric.Value),
113. (int)(Points\_numeric.Value));
114. break;
115. }
116. }
117. private void MainForm\_Load(object sender, EventArgs e)
118. {
119. renderControl1.Start(Ymin\_numeric,
120. Ymax\_numeric,
121. tabControl1.SelectedIndex,
122. autoSizeY\_checkBox.Checked,
123. (float)(Xmin\_numeric.Value),
124. (float)(Xmax\_numeric.Value),
125. (float)(Ymin\_numeric.Value),
126. (float)(Ymax\_numeric.Value),
127. 0.2f,
128. (float)(Step\_numeric.Value),
129. (int)(Points\_numeric.Value));
130. }
131. private void autoSizeY\_checkBox\_CheckedChanged(object sender, EventArgs e)
132. {
133. switch (tabControl1.SelectedIndex)
134. {
135. case 0:
136. renderControl1.SetAutoSizeY(autoSizeY\_checkBox.Checked);
137. break;
138. case 1:
139. renderControl2.SetAutoSizeY(autoSizeY\_checkBox.Checked);
140. break;
141. }
142. if (autoSizeY\_checkBox.Checked)
143. {
144. Ymin\_numeric.Enabled = false;
145. Ymax\_numeric.Enabled = false;
146. label5.Enabled = false;
147. label4.Enabled = false;
148. }
149. else
150. {
151. Ymin\_numeric.Enabled = true;
152. Ymax\_numeric.Enabled = true;
153. label5.Enabled = true;
154. label4.Enabled = true;
155. }
156. }
157. }

### Код файлу (RenderControl.cs)

1. public partial class RenderControl : OpenGL
2. {
3. private float Xmin, Xmax;
4. private float Ymin, Ymax;
5. private float margin;
6. private float step;
7. private float \_heigth;
8. private float \_width;
9. private int points;
10. private int tabIndex;
11. private bool autoSizeY;
12. private NumericUpDown Ymin\_numeric;
13. private NumericUpDown Ymax\_numeric;
14. private DrawConfig config;
15. private FuncOne \_funcOne;
16. private FuncTwo \_funcTwo;
17. public RenderControl()
18. {
19. InitializeComponent();
20. }
21. public void Start(NumericUpDown \_Ymin\_numeric, NumericUpDown \_Ymax\_numeric, int \_tabIndex, bool \_autoSizeY, float \_Xmin, float \_Xmax, float \_Ymin, float \_Ymax, float \_margin, float \_step, int \_points)
22. {
23. Ymin\_numeric = \_Ymin\_numeric;
24. Ymax\_numeric = \_Ymax\_numeric;
25. tabIndex = \_tabIndex;
26. autoSizeY = \_autoSizeY;
27. Xmin = \_Xmin;
28. Xmax = \_Xmax;
29. Ymin = \_Ymin;
30. Ymax = \_Ymax;
31. margin = \_margin;
32. step = \_step;
33. points = \_points;
34. UpdateHeightWidth();
35. UpdateConfig();
36. }
37. private void UpdateHeightWidth()
38. {
39. \_heigth = (-Ymin) + Ymax;
40. \_width = (-Xmin) + Xmax;
41. }
42. private void UpdateConfig()
43. {
44. config = new DrawConfig(Xmin, Xmax, Ymin, Ymax, margin, step, \_heigth, \_width);
45. switch(tabIndex)
46. {
47. case 0:
48. \_funcOne = new FuncOne(points, Ymin\_numeric, Ymax\_numeric, config, autoSizeY);
49. break;
50. case 1:
51. \_funcTwo = new FuncTwo(points, Ymin\_numeric, Ymax\_numeric, config, autoSizeY);
52. break;
53. }
54. }
55. public void SetAutoSizeY(bool \_autoSizeY)
56. {
57. autoSizeY = \_autoSizeY;
58. UpdateConfig();
59. Refresh();
60. }
61. public void SetXmin(float \_Xmin)
62. {
63. Xmin = \_Xmin;
64. UpdateHeightWidth();
65. UpdateConfig();
66. Refresh();
67. }
68. public void SetXmax(float \_Xmax)
69. {
70. Xmax = \_Xmax;
71. UpdateHeightWidth();
72. UpdateConfig();
73. Refresh();
74. }
75. public void SetYmin(float \_Ymin)
76. {
77. Ymin = \_Ymin;
78. UpdateHeightWidth();
79. UpdateConfig();
80. Refresh();
81. }
82. public void SetYmax(float \_Ymax)
83. {
84. Ymax = \_Ymax;
85. UpdateHeightWidth();
86. UpdateConfig();
87. Refresh();
88. }
89. public void SetPoints(int \_points)
90. {
91. points = \_points;
92. UpdateHeightWidth();
93. UpdateConfig();
94. Refresh();
95. }
96. public void SetStep(float \_step)
97. {
98. step = \_step;
99. UpdateHeightWidth();
100. UpdateConfig();
101. Refresh();
102. }
103. private void RenderControl\_Render(object sender, EventArgs e)
104. {
105. glClear(GL\_COLOR\_BUFFER\_BIT);
106. glLoadIdentity();
107. glViewport(0, 0, Width, Height);
108. glOrtho(Xmin - margin, Xmax + margin, Ymin - margin, Ymax + margin, -1, 1);
110. switch(tabIndex)
111. {
112. case 0:
113. if(\_funcOne != null)
114. {
115. \_funcOne.DrawCoordinateSystem();
116. \_funcOne.DrawFunction();
117. }
118. break;
119. case 1:
120. if(\_funcTwo != null)
121. {
122. \_funcTwo.DrawCoordinateSystem();
123. \_funcTwo.DrawFunction();
124. }
125. break;
126. }
127. DrawTextAxis();
128. }
129. private void DrawTextAxis()
130. {
131. glColor3d(200f / 255f, 200f / 255f, 200f / 255f);
132. // X axis
133. float \_start = step \* (((-Xmin) / step) % 1);
134. for (float colum = \_start; colum < \_width + step; colum += step)
135. {
136. if ((Xmin + colum != 0) && (Xmin + colum >= Xmin && Xmin + colum <= Xmax))
137. {
138. DrawText((Xmin + colum).ToString("F1"), Xmin + colum, -step / 3);
139. }
140. }
141. // Y axis
142. \_start = step \* ((Ymax / step) % 1);
143. for (float row = \_start; row < \_heigth + step; row += step)
144. {
145. if ((Ymax - row != 0) && (Ymax - row <= Ymax && Ymax - row >= Ymin))
146. {
147. DrawText((Ymax - row).ToString("F1"), -step / 3, Ymax - row);
148. }
149. }
150. // 0
151. DrawText((0).ToString("F1"), -step / 3, -step / 3);
152. }
153. }

### Код файлу (DrawConfig.cs)

1. public class DrawConfig
2. {
3. public float Xmin { get; set; }
4. public float Xmax { get; set; }
5. public float Ymin { get; set; }
6. public float Ymax { get; set; }
7. public float margin { get; set; }
8. public float step { get; set; }
9. public float height { get; set; }
10. public float width { get; set; }
11. public DrawConfig(float xmin, float xmax, float ymin, float ymax, float \_margin, float \_step, float \_height, float \_width)
12. {
13. Xmin = xmin;
14. Xmax = xmax;
15. Ymin = ymin;
16. Ymax = ymax;
17. margin = \_margin;
18. step = \_step;
19. height = \_height;
20. width = \_width;
21. }
22. }

### Код файлу (Draw.cs)

1. public class Draw
2. {
3. protected float Xmin, Xmax;
4. protected float Ymin, Ymax;
5. protected float margin;
6. protected float step;
7. protected float \_heigth;
8. protected float \_width;
9. public Draw(DrawConfig config)
10. {
11. Xmin = config.Xmin;
12. Xmax = config.Xmax;
13. Ymin = config.Ymin;
14. Ymax = config.Ymax;
15. margin = config.margin;
16. step = config.step;
17. \_heigth = config.height;
18. \_width = config.width;
19. }
20. public void DrawCoordinateSystem()
21. {
22. DrawGrid();
23. DrawAxis();
24. }
26. private void DrawGrid()
27. {
28. glLineWidth(1);
29. glBegin(GL\_LINES);
30. glColor3d(60f / 255f, 60f / 255f, 60f / 255f);
31. float \_start = step \* ((Ymax / step) % 1);
32. // draw horizontal lines
33. for (float row = \_start; row < \_heigth; row += step)
34. {
35. glVertex2d(Xmin, Ymax - row);
36. glVertex2d(Xmax, Ymax - row);
37. }
38. \_start = step \* (((-Xmin) / step) % 1);
39. // draw vertical lines
40. for (float colum = \_start; colum < \_width; colum += step)
41. {
42. glVertex2d(Xmin + colum, Ymax);
43. glVertex2d(Xmin + colum, Ymin);
44. }
45. // draw borders
46. glVertex2d(Xmin, Ymax);
47. glVertex2d(Xmax, Ymax);
48. glVertex2d(Xmin, Ymin);
49. glVertex2d(Xmax, Ymin);
50. glVertex2d(Xmax, Ymax);
51. glVertex2d(Xmax, Ymin);
52. glVertex2d(Xmin, Ymax);
53. glVertex2d(Xmin, Ymin);
54. glEnd();
55. }
56. private void DrawAxis()
57. {
58. glLineWidth(2);
59. glBegin(GL\_LINES);
60. glColor3d(200f / 255f, 200f / 255f, 200f / 255f);
61. // X axis
62. glVertex2d(Xmax, 0f);
63. glVertex2d(Xmin, 0f);
64. float \_start = step \* (((-Xmin) / step) % 1);
65. // draw vertical lines
66. for (float colum = \_start; colum < \_width + step; colum += step)
67. {
68. if(Xmin + colum >= Xmin && Xmin + colum <= Xmax)
69. {
70. glVertex2d(Xmin + colum, 0);
71. glVertex2d(Xmin + colum, -step / 5);
72. }
73. }
74. // Y axis
75. glVertex2d(0f, Ymax);
76. glVertex2d(0f, Ymin);
77. \_start = step \* ((Ymax / step) % 1);
78. // draw horizontal lines
79. for (float row = \_start; row < \_heigth + step; row += step)
80. {
81. if(Ymax - row <= Ymax && Ymax - row >= Ymin)
82. {
83. glVertex2d(-step / 5, Ymax - row);
84. glVertex2d(0, Ymax - row);
85. }
87. }
88. glLineWidth(1);
89. glEnd();
90. }
91. }

### Код файлу (FuncOne.cs)

1. internal class FuncOne : Draw
2. {
3. private int points;
4. private bool autoSizeY;
5. private NumericUpDown Ymin\_numeric;
6. private NumericUpDown Ymax\_numeric;
7. public FuncOne(int \_points, NumericUpDown \_Ymin\_numeric, NumericUpDown \_Ymax\_numeric, DrawConfig config, bool \_autoSizeY)
8. : base(config)
9. {
10. points = \_points;
11. autoSizeY = \_autoSizeY;
12. Ymin\_numeric = \_Ymin\_numeric;
13. Ymax\_numeric = \_Ymax\_numeric;
14. }
15. private float Func(float x)
16. {
17. float numerator = MathF.Abs(MathF.Cos(0.5f \* x + 1.0f)) \* MathF.Cos(x);
18. float denominator = MathF.Abs(MathF.Cos(x + 0.01f));
19. return numerator / denominator;
20. }
21. public void DrawFunction()
22. {
23. float h = \_width / (points - 1);
24. float x = 0;
25. float y = 0;
26. float \_x = 0;
27. float \_y = 0;
28. float \_Ymin = float.MaxValue;
29. float \_Ymax = float.MinValue;
30. glLineWidth(2);
31. glBegin(GL\_LINES);
32. glColor3d(64f / 255f, 224f / 255f, 208f / 255f);
33. for (int i = 0; i < points; i++)
34. {
35. if (i > 0)
36. {
37. \_x = x;
38. \_y = y;
39. }
40. x = Xmin + i \* h;
41. y = Func(x);
42. if (!autoSizeY)
43. {
44. if (y > Ymax)
45. {
46. y = Ymax;
47. continue;
48. }
49. else if (y < Ymin)
50. {
51. y = Ymin;
52. continue;
53. }
54. }
55. if (i > 0)
56. {
57. glVertex2d(\_x, \_y);
58. glVertex2d(x, y);
59. if (\_y \* y <= 0)
60. {
61. DrawPoint((\_x + x) / 2, (\_y + y) / 2);
62. }
63. }
64. if (y < \_Ymin && y < 0)
65. {
66. \_Ymin = y;
67. }
68. if (y > \_Ymax && y > 0)
69. {
70. \_Ymax = y;
71. }
72. }
73. glEnd();
74. \_Ymin = MathF.Round(\_Ymin, 1);
75. \_Ymax = MathF.Round(\_Ymax, 1);
76. if (\_Ymin < -100) \_Ymin = -99.0f;
77. if (\_Ymax > 100) \_Ymax = 99.0f;
78. if (autoSizeY)
79. {
80. Ymin\_numeric.Value = (decimal)(\_Ymin - step);
81. Ymax\_numeric.Value = (decimal)(\_Ymax + step);
82. }
83. }
84. private void DrawPoint(float x, float y)
85. {
86. // end draw function
87. glEnd();
88. // draw point
89. glPointSize(10);
90. glBegin(GL\_POINTS);
91. glColor3d(255f / 255f, 128f / 255f, 0f / 255f);
92. glVertex2d(x, y);
93. glEnd();
94. // continue draw function
95. glBegin(GL\_LINES);
96. glColor3d(64f / 255f, 224f / 255f, 208f / 255f);
97. }
98. }

### Код файлу (FuncTwo.cs)

1. internal class FuncTwo : Draw
2. {
3. private int points;
4. private bool autoSizeY;
5. private NumericUpDown Ymin\_numeric;
6. private NumericUpDown Ymax\_numeric;
7. public FuncTwo(int \_points, NumericUpDown \_Ymin\_numeric, NumericUpDown \_Ymax\_numeric, DrawConfig config, bool \_autoSizeY)
8. : base(config)
9. {
10. points = \_points;
11. autoSizeY = \_autoSizeY;
12. Ymin\_numeric = \_Ymin\_numeric;
13. Ymax\_numeric = \_Ymax\_numeric;
14. }
15. private float Func(float x)
16. {
17. float numerator = MathF.Cos(float.Pi \* x / 2) \* MathF.Cos(MathF.Abs(float.Pi \* x));
18. float denominator = MathF.Abs(MathF.Sin(float.Pi \* x));
19. return numerator / denominator;
20. }
21. public void DrawFunction()
22. {
23. float h = \_width / (points - 1);
24. float x = 0;
25. float y = 0;
26. float \_x = 0;
27. float \_y = 0;
28. float \_Ymin = float.MaxValue;
29. float \_Ymax = float.MinValue;
30. float breakdown = 1f / points \* \_width;
31. glLineWidth(2);
32. glBegin(GL\_LINES);
33. glColor3d(64f / 255f, 224f / 255f, 208f / 255f);
34. for (int i = 0; i < points; i++)
35. {
36. if (i > 0)
37. {
38. \_x = x;
39. \_y = y;
40. }
41. x = Xmin + i \* h;
42. y = Func(x);
43. if (MathF.Abs(x - MathF.Round(x)) < breakdown)
44. {
45. continue;
46. }
47. else if (MathF.Abs(\_x - MathF.Round(\_x)) < breakdown)
48. {
49. DrawLinesBreakdown(\_x, \_y);
50. continue;
51. }
52. if (!autoSizeY)
53. {
54. if (y > Ymax)
55. {
56. y = Ymax;
57. continue;
58. }
59. else if (y < Ymin)
60. {
61. y = Ymin;
62. continue;
63. }
64. }
65. if (i > 0)
66. {
67. glVertex2d(\_x, \_y);
68. glVertex2d(x, y);
69. }
70. if (y < \_Ymin && y < 0)
71. {
72. \_Ymin = y;
73. }
74. if (y > \_Ymax && y > 0)
75. {
76. \_Ymax = y;
77. }
78. }
79. \_Ymin = MathF.Round(\_Ymin, 1);
80. \_Ymax = MathF.Round(\_Ymax, 1);
81. if (\_Ymin < -100) \_Ymin = -99.0f;
82. if (\_Ymax > 100) \_Ymax = 99.0f;
83. if (autoSizeY && (\_Ymin < -0.1f && \_Ymax > 0.1f))
84. {
85. Ymin\_numeric.Value = (decimal)(\_Ymin - step);
86. Ymax\_numeric.Value = (decimal)(\_Ymax + step);
87. }
88. glEnd();
89. }
90. private void DrawLinesBreakdown(float x, float y)
91. {
92. // end draw function
93. glEnd();
94. glLineStipple(10, 21845);
95. glEnable(GL\_LINE\_STIPPLE);
96. glLineWidth(3);
97. glBegin(GL\_LINES);
98. glColor3d(255f / 255f, 128f / 255f, 0f / 255f);
99. if (y > Ymax)
100. {
101. y = Ymax;
102. }
103. else if (y < Ymin)
104. {
105. y = Ymin;
106. }
107. glVertex2d(x, y);
108. glVertex2d(x, -y);
109. glEnd();
110. glDisable(GL\_LINE\_STIPPLE);
111. glLineWidth(2);
112. glBegin(GL\_LINES);
113. glColor3d(64f / 255f, 224f / 255f, 208f / 255f);
114. }
115. }

# Додаток D. Лістинг програми до практичної роботи №4

### Код файлу (MainForm.cs)

1. public partial class MainForm : Form
2. {
3. public MainForm()
4. {
5. InitializeComponent();
6. }
7. private void curve\_comboBox\_SelectedIndexChanged(object sender, System.EventArgs e)
8. {
9. renderControl1.SetDrawCurve(curve\_comboBox.SelectedIndex);
10. switch (curve\_comboBox.SelectedIndex)
11. {
12. case 0:
13. hyperbole\_panel.Enabled = false;
14. circle\_panel.Enabled = true;
15. line\_panel.Enabled = true;
16. break;
17. case 1:
18. circle\_panel.Enabled = false;
19. line\_panel.Enabled = false;
20. hyperbole\_panel.Enabled = true;
21. break;
22. }
23. }
24. private void drawLine\_button\_Click(object sender, System.EventArgs e)
25. {
26. renderControl1.SetPointsLine(
27. [(float)point1\_x\_numeric.Value, (float)point1\_y\_numeric.Value],
28. [(float)point2\_x\_numeric.Value, (float)point2\_y\_numeric.Value]);
29. }
30. private void radius\_numeric\_ValueChanged(object sender, System.EventArgs e)
31. {
32. renderControl1.SetRadius((float)radius\_numeric.Value);
33. }
34. private void hyp\_a\_numeric\_ValueChanged(object sender, System.EventArgs e)
35. {
36. renderControl1.SetHypA((float)hyp\_a\_numeric.Value);
37. }
38. private void hyp\_b\_numeric\_ValueChanged(object sender, System.EventArgs e)
39. {
40. renderControl1.SetHypB((float)hyp\_b\_numeric.Value);
41. }
42. }

### Код файлу (RenderControl.cs)

1. public partial class RenderControl : OpenGL
2. {
3. private float size = 3.0f;
4. private float AspectRatio { get => (float)Width / Height; }
5. private float Xmin { get => (AspectRatio > 1) ? -size \* AspectRatio : -size; }
6. private float Xmax { get => (AspectRatio > 1) ? +size \* AspectRatio : +size; }
7. private float Ymin { get => (AspectRatio < 1) ? -size / AspectRatio : -size; }
8. private float Ymax { get => (AspectRatio < 1) ? +size / AspectRatio : +size; }
9. private float margin = 0.2f;
10. private float step = 0.5f;
11. private float \_heigth;
12. private float \_width;
13. private int drawCurve = 0;
14. private float radius = 2f;
16. private float[] point1;
17. private float[] point2;
18. private bool isDrawLine = false;
19. private float a = 2f;
20. private float b = 2f;
21. public RenderControl()
22. {
23. InitializeComponent();
24. }
25. public void SetDrawCurve(int indexCurve)
26. {
27. drawCurve = indexCurve;
28. Refresh();
29. }
30. private void UpdateHeightWidth()
31. {
32. \_heigth = (-Ymin) + Ymax;
33. \_width = (-Xmin) + Xmax;
34. }
35. public void SetPointsLine(float[] \_point1, float[] \_point2)
36. {
37. point1 = \_point1;
38. point2 = \_point2;
39. isDrawLine = true;
40. Refresh();
41. }
42. public void SetRadius(float \_radius)
43. {
44. radius = \_radius;
45. Refresh();
46. }
47. public void SetHypA(float \_a)
48. {
49. a = \_a;
50. Refresh();
51. }
52. public void SetHypB(float \_b)
53. {
54. b = \_b;
55. Refresh();
56. }
57. private void RenderControl\_Render(object sender, EventArgs e)
58. {
59. glClear(GL\_COLOR\_BUFFER\_BIT);
60. glLoadIdentity();
61. glViewport(0, 0, Width, Height);
62. glOrtho(Xmin - margin, Xmax + margin, Ymin - margin, Ymax + margin, -1, 1);
64. UpdateHeightWidth();
65. DrawGrid();
66. DrawAxis();
67. DrawTextAxis();
68. switch(drawCurve)
69. {
70. case 0:
71. DrawCircle();
72. if(isDrawLine) DrawLine();
73. break;
74. case 1:
75. DrawHyperbole();
76. break;
77. }
78. }
79. private void DrawGrid()
80. {
81. glLineWidth(1);
82. glBegin(GL\_LINES);
83. glColor3d(60f / 255f, 60f / 255f, 60f / 255f);
84. float \_start = step \* ((Ymax / step) % 1);
85. // draw horizontal lines
86. for (float row = \_start; row < \_heigth; row += step)
87. {
88. glVertex2d(Xmin, Ymax - row);
89. glVertex2d(Xmax, Ymax - row);
90. }
91. \_start = step \* (((-Xmin) / step) % 1);
92. // draw vertical lines
93. for (float colum = \_start; colum < \_width; colum += step)
94. {
95. glVertex2d(Xmin + colum, Ymax);
96. glVertex2d(Xmin + colum, Ymin);
97. }
98. // draw borders
99. glVertex2d(Xmin, Ymax);
100. glVertex2d(Xmax, Ymax);
101. glVertex2d(Xmin, Ymin);
102. glVertex2d(Xmax, Ymin);
103. glVertex2d(Xmax, Ymax);
104. glVertex2d(Xmax, Ymin);
105. glVertex2d(Xmin, Ymax);
106. glVertex2d(Xmin, Ymin);
107. glEnd();
108. }
109. private void DrawAxis()
110. {
111. glLineWidth(2);
112. glBegin(GL\_LINES);
113. glColor3d(200f / 255f, 200f / 255f, 200f / 255f);
114. // X axis
115. glVertex2d(Xmax, 0f);
116. glVertex2d(Xmin, 0f);
117. float \_start = step \* (((-Xmin) / step) % 1);
118. // draw vertical lines
119. for (float colum = \_start; colum < \_width + step; colum += step)
120. {
121. glVertex2d(Xmin + colum, 0);
122. glVertex2d(Xmin + colum, -step / 3);
123. }
124. // Y axis
125. glVertex2d(0f, Ymax);
126. glVertex2d(0f, Ymin);
127. \_start = step \* ((Ymax / step) % 1);
128. // draw horizontal lines
129. for (float row = \_start; row < \_heigth + step; row += step)
130. {
131. glVertex2d(-step / 3, Ymax - row);
132. glVertex2d(0, Ymax - row);
133. }
134. glLineWidth(1);
135. glEnd();
136. }
137. private void DrawTextAxis()
138. {
139. glColor3d(200f / 255f, 200f / 255f, 200f / 255f);
140. // X axis
141. float \_start = step \* (((-Xmin) / step) % 1);
142. for (float colum = \_start; colum < \_width + step; colum += step)
143. {
144. if (Xmin + colum >= 0.1f || Xmin + colum <= -0.1f)
145. {
146. DrawText((Xmin + colum).ToString("F1"), Xmin + colum, -step / 2);
147. }
148. }
149. // Y axis
150. \_start = step \* ((Ymax / step) % 1);
151. for (float row = \_start; row < \_heigth + step; row += step)
152. {
153. if (Ymax - row >= 0.1f || Ymax - row <= -0.1f)
154. {
155. DrawText((Ymax - row).ToString("F1"), -step / 2, Ymax - row);
156. }
157. }
158. // 0
159. DrawText((0).ToString(), -step / 2.5f, -step / 2.5f);
160. }
161. private void DrawCircle()
162. {
163. float x = 0;
164. float y = 0;
165. float \_x = 0;
166. float \_y = 0;
167. float c = 0.001f;
168. glLineWidth(3);
169. glBegin(GL\_LINES);
170. glColor3d(64f / 255f, 224f / 255f, 208f / 255f);
171. for(x = -radius; x < radius + c; x += c)
172. {
173. x = MathF.Round(x, 3);
174. y = MathF.Sqrt((radius\*radius) - (x\*x));
175. if (x > -radius)
176. {
177. glVertex2d(\_x, \_y);
178. glVertex2d(x, y);
179. glVertex2d(\_x, -\_y);
180. glVertex2d(x, -y);
181. if(isDrawLine)
182. {
183. DrawPoint([\_x, \_y], [x, y]);
184. DrawPoint([\_x, -\_y], [x, -y]);
185. }
186. }
187. \_y = y;
188. \_x = x;
189. }
190. glEnd();
191. }
192. private void DrawLine()
193. {
194. glLineWidth(3);
195. glBegin(GL\_LINES);
196. glColor3d(255f / 255f, 128f / 255f, 0f / 255f);
197. glVertex2d(point1[0], point1[1]);
198. glVertex2d(point2[0], point2[1]);
199. glEnd();
200. }
201. private void DrawPoint(float[] \_point1, float[] \_point2)
202. {
203. // circle
204. float x1 = \_point1[0];
205. float y1 = \_point1[1];
206. float x2 = \_point2[0];
207. float y2 = \_point2[1];
208. // line
209. float x3 = point1[0];
210. float y3 = point1[1];
211. float x4 = point2[0];
212. float y4 = point2[1];
213. // point
214. float x0 = 0;
215. float y0 = 0;
216. float D = (x2 - x1) \* (y4 - y3) - (y2 - y1) \* (x4 - x3);
217. float t1 = 0;
218. float t2 = 0;
219. if (D != 0)
220. {
221. t1 = ((x3 - x1) \* (y4 - y3) - (y3 - y1) \* (x4 - x3)) / D;
222. t2 = ((x3 - x1) \* (y2 - y1) - (y3 - y1) \* (x2 - x1)) / D;
223. if ((t1 >= 0 && t1 <= 1) && (t2 >= 0 && t2 <= 1))
224. {
225. x0 = (x2 - x1) \* t1 + x1;
226. y0 = (y2 - y1) \* t1 + y1;
227. // end draw function
228. glEnd();
229. glPointSize(20);
230. glBegin(GL\_POINTS);
231. glColor3d(255f / 255f, 255f / 255f, 0f / 255f);
232. glVertex2d(x0, y0);
233. glEnd();
234. // continue draw function
235. glBegin(GL\_LINES);
236. glColor3d(64f / 255f, 224f / 255f, 208f / 255f);
237. }
238. }
239. }
240. private void DrawHyperbole()
241. {
242. float x = 0, y = 0;
243. float \_x = 0, \_y = 0;
244. float c = 0.001f;
245. glLineWidth(3);
246. glBegin(GL\_LINES);
247. glColor3d(64f / 255f, 224f / 255f, 208f / 255f);
248. for(x = Xmin; x <= Xmax; x += c)
249. {
250. x = MathF.Round(x, 3);
251. y = MathF.Sqrt((((x\*x) / (a\*a)) - 1) \* (b\*b));
252. if(y > Ymax || y < Ymin)
253. {
254. \_x = x;
255. \_y = y;
256. continue;
257. }
258. if(x > Xmin)
259. {
260. glVertex2d(\_x, \_y);
261. glVertex2d(x, y);
262. glVertex2d(\_x, -\_y);
263. glVertex2d(x, -y);
264. }
265. \_x = x;
266. \_y = y;
267. }
268. glEnd();
269. }
270. }

# Додаток E. Лістинг програми до практичної роботи №5

### Код файлу (MainForm.cs)

1. public partial class MainForm : Form
2. {
3. public MainForm()
4. {
5. InitializeComponent();
6. }

### Код файлу (RenderControl.cs)

1. public partial class RenderControl : OpenGL
2. {
3. int min\_x = -1, max\_x = 8;
4. int min\_y = -2, max\_y = 2;
5. Draw draw
6. public static OutText OutText { get; set; }
8. public RenderControl()
9. {
10. InitializeComponent();
11. }
12. private void RenderControl\_Render(object sender, EventArgs e)
13. {
14. glClear(GL\_COLOR\_BUFFER\_BIT);
15. glLoadIdentity();
16. glViewport(0, 0, Width, Height);
17. glOrtho(min\_x - 2, max\_x + 2, min\_y - 2, max\_y + 2, -1, 1);
18. draw.Drawing();
19. DrawText("X1", min\_x, min\_y - 1.5);
20. DrawText("X2", max\_x, min\_y - 1.5);
21. DrawText("Y1", min\_x - 1.5, min\_y);
22. DrawText("Y2", min\_x - 1.5, max\_y);
23. }
24. private void RenderControl\_ContextCreated(object sender, EventArgs e)
25. {
26. draw = new Draw(min\_x, max\_x, min\_y, max\_y);
27. }
28. }

### Код файлу (Draw.cs)

1. public class Draw
2. {
3. private int min\_x, max\_x, min\_y, max\_y;
4. public Draw(int MinX, int MaxX, int MinY, int MaxY)
5. {
6. min\_x = MinX;
7. max\_x = MaxX;
8. min\_y = MinY;
9. max\_y = MaxY;
10. }
11. public void Drawing()
12. {
13. //glColor3f(0.0f / 255.0f, 0.0f / 255.0f, 0.0f / 255.0f);
14. DrawGrid();
15. DrawAxis();
16. DrawFigure();
17. DrawPorintsFigure();
18. }
19. private void DrawGrid()
20. {
21. glLineStipple(1, 4369);
22. glEnable(GL\_LINE\_STIPPLE);
23. glLineWidth(2);
24. glBegin(GL\_LINES);
25. // draw horizontal lines
26. for (float i = min\_x; i <= max\_x; i += 1)
27. {
28. glVertex2d(i, min\_y - 1);
29. glVertex2d(i, max\_y + 1);
30. }
31. // draw vertical lines
32. for (float i = min\_y; i <= max\_y; i += 1)
33. {
34. glVertex2d(min\_x - 1, i);
35. glVertex2d(max\_x + 1, i);
36. }
37. glEnd();
38. glDisable(GL\_LINE\_STIPPLE);
39. glLineWidth(1);
40. }
41. private void DrawAxis()
42. {
43. glLineWidth(4);
44. glBegin(GL\_LINES);
45. // Y
46. glVertex2d(min\_x - 1, min\_y - 0.2);
47. glVertex2d(min\_x - 1, max\_y + 1);
48. // arrow
49. glVertex2d(min\_x - 1, max\_y + 1);
50. glVertex2d(min\_x - 1 - 0.2, max\_y + 1 - 0.2);
51. glVertex2d(min\_x - 1, max\_y + 1);
52. glVertex2d(min\_x - 1 + 0.2, max\_y + 1 - 0.2);
53. // strokes
54. for (int i = min\_y; i <= max\_y; i++)
55. {
56. glVertex2d(min\_x - 1, i);
57. glVertex2d(min\_x - 1 - 0.2, i);
58. }
59. // X
60. glVertex2d(min\_x - 0.2, min\_y - 1);
61. glVertex2d(max\_x + 1, min\_y - 1);
62. // arrow
63. glVertex2d(max\_x + 1, min\_y - 1);
64. glVertex2d(max\_x + 1 - 0.2, min\_y - 1 - 0.2);
65. glVertex2d(max\_x + 1, min\_y - 1);
66. glVertex2d(max\_x + 1 - 0.2, min\_y - 1 + 0.2);
67. // strokes
68. for (int i = min\_x; i <= max\_x; i++)
69. {
70. glVertex2d(i, min\_y - 1);
71. glVertex2d(i, min\_y - 1 - 0.2);
72. }
73. glEnd();
74. glLineWidth(1);
75. }
76. private void DrawFigure()
77. {
78. glLineWidth(8);
79. glBegin(GL\_LINES);
80. glVertex2d(-1, 2);
81. glVertex2d(-1, 0);
82. glVertex2d(-1, 0);
83. glVertex2d(0, -1);
84. glVertex2d(0, -1);
85. glVertex2d(2, -2);
86. glVertex2d(2, -2);
87. glVertex2d(2, -1);
88. glVertex2d(2, -1);
89. glVertex2d(1, 1);
90. glVertex2d(1, 1);
91. glVertex2d(0, 2);
92. glVertex2d(0, 2);
93. glVertex2d(-1, 2);
94. glEnd();
95. glLineWidth(1);
96. }
97. private void DrawPorintsFigure()
98. {
99. glPointSize(16);
100. glEnable(GL\_POINT\_SMOOTH);
101. glBegin(GL\_POINTS);
102. glVertex2d(4, 2);
103. glVertex2d(4, 0);
104. glVertex2d(5, -1);
105. glVertex2d(7, -2);
106. glVertex2d(7, -1);
107. glVertex2d(6, 1);
108. glVertex2d(5, 2);
109. glEnd();
110. glDisable(GL\_POINT\_SMOOTH);
111. glPointSize(1);
112. }
113. }