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 № *621пст*

освітньої програми

*121 інженерія програмного забезпечення*

(шифр і назва ОП)

*Лампарт В.М.*

(прізвище й ініціали студента)

Прийняв: доцент каф. 603, *к.т.н*

*Лучшев П.О.*

(посада, науковий ступінь, прізвище й ініціали)

Кількість балів:

Харків – 2024

ЗМІСТ

[ПРАКТИЧНА РОБОТА 1. ОСНОВНІ ПРИНЦИПИ РОБОТИ З OPENGL 3](#_Toc186050431)

[Завдання, варіант № 13 3](#_Toc186050432)

[Системна інформація 3](#_Toc186050433)

[Теоретичні відомості 3](#_Toc186050434)

[Результати виконання практичної роботи 6](#_Toc186050435)

[ПРАКТИЧНА РОБОТА 3. ГРАФІК ФУНКЦІЇ ОДНІЄЇ ЗМІННОЇ 8](#_Toc186050436)

[Завдання, варіант № 13 8](#_Toc186050437)

[Результати виконання практичної роботи 9](#_Toc186050438)

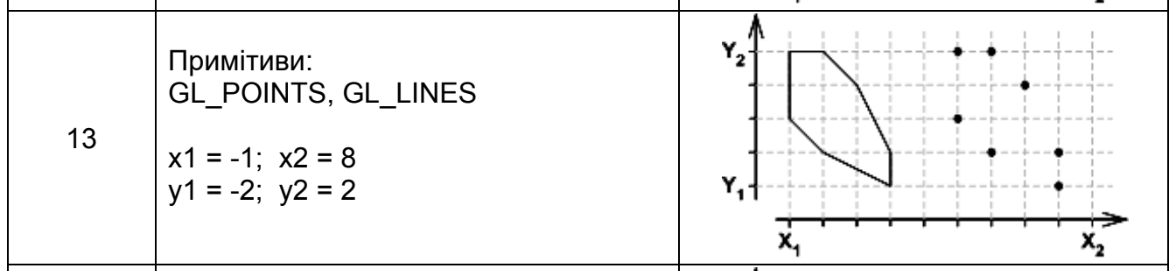
[ДОДАТОК А Лістинг програми до практичної роботи №1 12](#_Toc186050439)

[ДОДАТОК Б Лістинг програми до практичної роботи №3 14](#_Toc186050440)

# 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. Враховуючи систему оцінки, 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.



# Системна інформація

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

Processor AMD Ryzen 5 3600

RAM 16.0 GB

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

Edition Windows 11

IDE Microsoft Visual Studio Enterprise 2022 (64-bit) version 17.11.3

# Теоретичні відомості

***Вершинні масиви***

Використання вершинних масивів зменшує кількість викликів функцій та надмірне використання спільних вершин. Таким чином, ви можете збільшити продуктивність рендерингу.

OpenGL надає функції glEnableClientState() та glDisableClientState() для активації та деактивації 6 різних типів масивів. Крім того, є 6 функцій для визначення точних позицій (адрес) масивів, отже, OpenGL може отримати доступ до масивів у вашому додатку.

̶ glVertexPointer(): вказати покажчик на масив вершинних координат

̶ glNormalPointer(): вказати покажчик на звичайний масив

̶ glColorPointer(): вказати вказівник на масив кольорів RGB

̶ glIndexPointer(): вказати покажчик на індексований кольоровий масив

̶ glTexCoordPointer(): вказати покажчик на масив текстурних шнурів

̶ glEdgeFlagPointer(): вказати вказівник на масив прапорців edge

Для кожної заданої функції потрібні різні параметри. Прапорці ребер використовуються для позначення того, чи знаходиться вершина на граничному ребрі чи ні. Отже, єдині ребра, де ввімкнено прапорці країв, будуть видимими, якщо для glPolygonMode() встановлено значення GL\_LINE.

Для кожної заданої функції потрібні різні параметри. Будь ласка, перегляньте інструкції до API OpenGL. Прапорці ребер використовуються для позначення того, чи знаходиться вершина на граничному ребрі чи ні. Отже, єдині ребра, де ввімкнено прапорці країв, будуть видимими, якщо для glPolygonMode() встановлено значення GL\_LINE.

***Команда glDrawArrays()***

glDrawArrays() зчитує дані вершин з увімкнених масивів, проходячи прямо по масиву без пропусків або стрибків. Оскільки glDrawArrays() не дозволяє стрибати навколо масивів вершин, вам все одно доведеться повторювати спільні вершини один раз на кожну грань.

glDrawArrays() приймає 3 аргументи. По-перше, це примітивний тип. Другий параметр – це початковий виліт масиву. Останній параметр – це кількість вершин, які потрібно передати конвеєру рендерингу OpenGL.

Для наведеного вище прикладу для малювання куба першим параметром є GL\_TRIANGLES, другим - 0, що означає початок роботи масиву. І останній параметр - 36: у куба 6 сторін і кожній стороні потрібно 6 вершин, щоб намалювати 2 трикутника, 6 × 6 = 36.

GLfloat vertices[] = {...}; // 36 of vertex coords

...

// activate and specify pointer to vertex array

glEnableClientState(GL\_VERTEX\_ARRAY);

glVertexPointer(3, GL\_FLOAT, 0, vertices);

// draw a cube

glDrawArrays(GL\_TRIANGLES, 0, 36);

// deactivate vertex arrays after drawing

glDisableClientState(GL\_VERTEX\_ARRAY);

В результаті використання glDrawArrays() ви можете замінити 36 викликів glVertex\*() одним викликом glDrawArrays(). Однак нам все ще потрібно дублювати спільні вершини, тому кількість вершин, визначених у масиві, все ще становить 36 замість 8. glDrawElements() — це рішення для зменшення кількості вершин у масиві, тому воно дозволяє передавати менше даних до OpenGL.

***Команда glDrawElements()***

glDrawElements() малює послідовність примітивів, перескакуючи навколо вершинних масивів з пов'язаними індексами масивів. При цьому зменшується як кількість викликів функцій, так і кількість вершин для передачі. Крім того, OpenGL може кешувати нещодавно оброблені вершини та повторно використовувати їх без повторного надсилання тих самих вершин у конвеєр перетворення вершин кілька разів.

# Результати виконання практичної роботи

***Розв’язання завдання***

Для управління параметрами графічних примітивів було використано наступні команди (Додаток А):

* колір, glColor3d() у рядках 40, 68, 87 у файлі Figure13.cs;
* тип glLineStipple у рядку 39 файлу Figure13.cs;
* товщина ліній glLineWidth у рядках 37, 67 у файлі Figure13.cs,
* товщина точок glPointSize у рядку 86 файлу Figure.cs.

Застосування циклів для створення зображень наведено у рядках 23–28 файлу Grid.cs, та у рядках 41-48 файлу Figure13.

Використання ООП реалізовано за допомогою розроблення власних класів в файлах Grid.cs та Figure13.cs.

Повний код наведено в Додатку А.

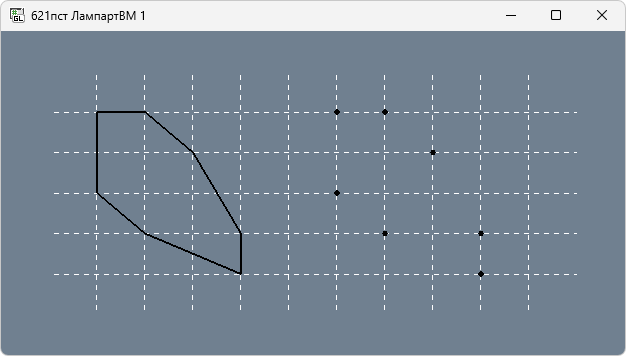


Рисунок 1.1 – Результат роботи програми

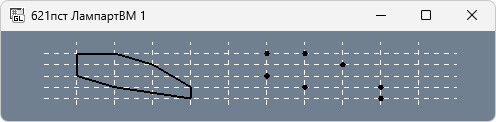


Рисунок 1.2 – Результат роботи програми при зміні розмірів вікна

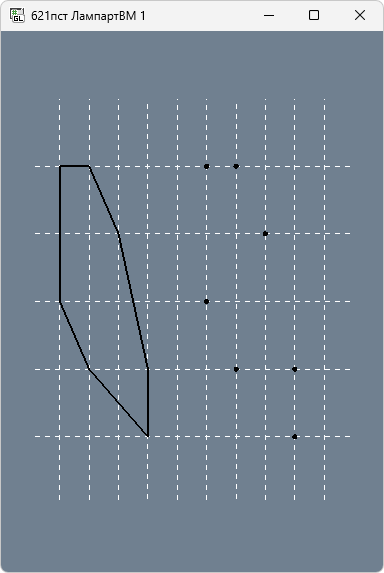


Рисунок 1.3 – Результат роботи програми при зміні розмірів вікна

***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 – Результат виконання практичної роботи №1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Складність | Вимоги до роботи | Бали | Оцінка |
| 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 тощо) | 1 | - |
| 6. | Використання ООП (розроблення власних класів) | 2 | + |

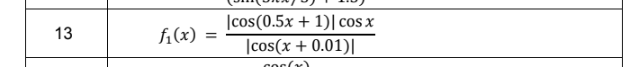
# ПРАКТИЧНА РОБОТА 3. ГРАФІК ФУНКЦІЇ ОДНІЄЇ ЗМІННОЇ

# 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:

* 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.

Для підвищеного рівня складності необхідно додатково реалізувати коректне виведення функції 𝑓2 (𝑥) з урахуванням області визначення функції і відобразити лінії розриву.

****

# Результати виконання практичної роботи

***Розв’язання завдання***

Код реалізації виведення осей координат знаходиться в файлі RendererControl.cs рядки: 66-94.

Код реалізації виведення графіка функції на заданому користувачем інтервалі знаходиться в файлі Function13.cs рядки 38–73.

Код обчислення і виведення на екран точок f1(x)=0 знаходиться в файлі Function13.cs рядки 75–82.

Використання ООП реалізовано за допомогою абстрактного класу AMathFunction через наслідуванням класами FirstFunc та SecondFunc, які містять метод розрахунку функції та значення Y на заданому проміжку Xmin Xmax.

Робота програми продемонстрована на рисунках 3.1–3.2.

Повний код наведено в Додатку Б.

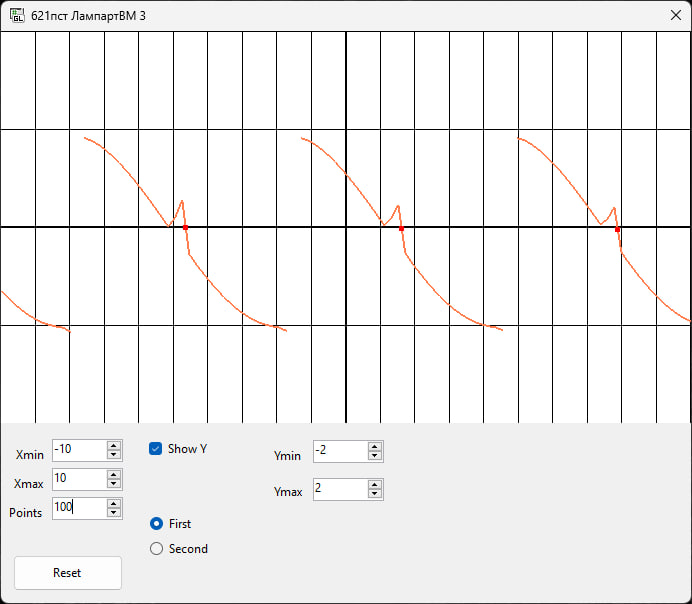


Рисунок 3.1 – Результат роботи програми (Функція 1)

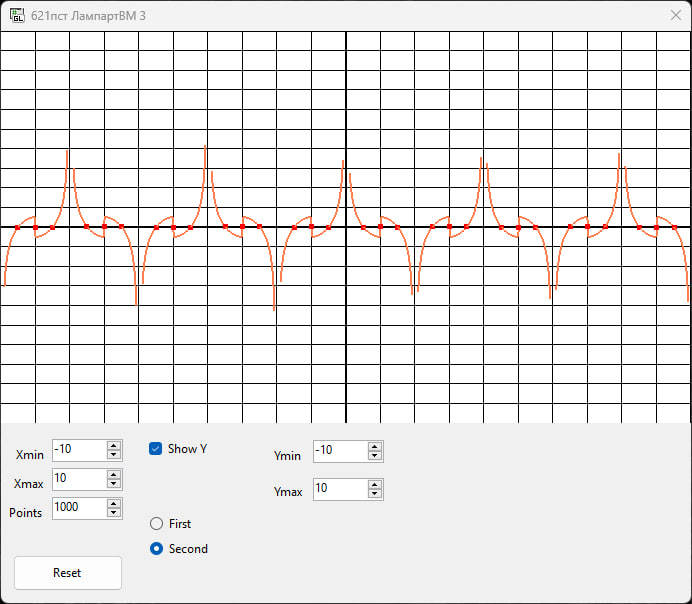


Рисунок 3.2 – Результат роботи програми (Функція 2)

***Контроль виконання вимог та елементів завдання***

В результаті виконання практичної роботи були повністю виконані елементи базового та підвищеного рівня складності, що відображено в таблиці 3.1.

Таблиця 3.1 – Результат виконання практичної роботи №3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Складність | Вимоги до роботи | Бали | Оцінка |
| 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 функцій | 2 | + |
| 5. | Використання ООП (наслідування, використання віртуальних і абстрактних методів) | 1 | + |

# ПРАКТИЧНА РОБОТА 4. КРИВІ ДРУГОГО ПОРЯДКУ

# Завдання, варіант № 13 (Окружність явне подання, Гіпербола параметричне подання)

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.

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] і [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:

Система:

x0 = (x2 − x1)t1 + x1 = (x4 − x3)t2 + x3

y0 = (y2 − y1)t2 + y1 = (y4 − y3)t2 + y3;

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***

Код реалізації виведення осей координат знаходиться в файлі RendererControl.cs рядки: 109-137.

Код реалізації виведення кривих знаходиться в файлах Circle.cs(22-34), HorizontalHyperbola.cs(26-49), VerticalHyperbola.cs(25-47).

Код обчислення і виведення на екран точок перетину з кривими знаходиться в файлах Function13.cs Circle.cs(36-74), HorizontalHyperbola.cs(-49), VerticalHyperbola.cs(51-88).

Використання ООП реалізовано за допомогою інтерфейсу IShape2D через реалізацію класами Circle, HorizontalHyperbola та VerticalHyperbola, які містять метод малювання кривих та розрахунку точок перетину з заданою лінією.

Робота програми продемонстрована на рисунках 4.1–3.3.

Повний код наведено в Додатку В.

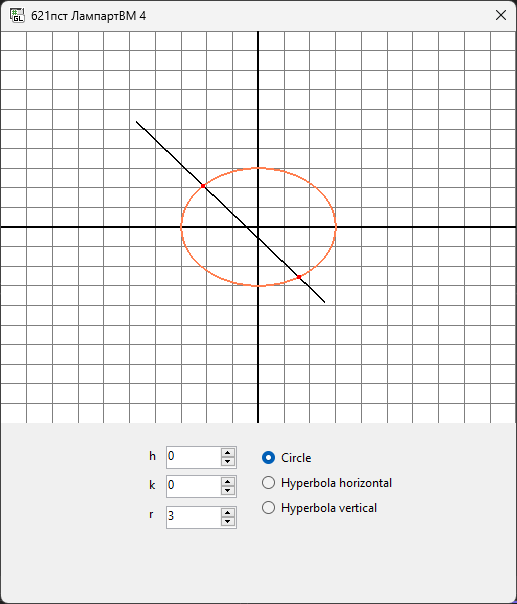
****

Рисунок 4.1 – Результат роботи програми (Окружність)

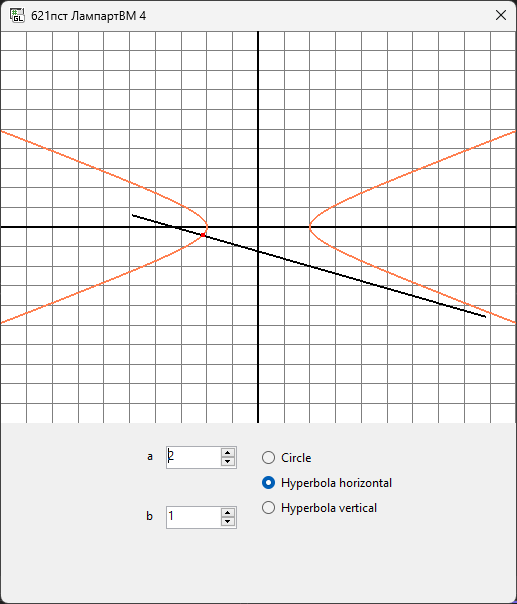


Рисунок 4.2 – Результат роботи програми (Горизонтальна гіпербола)

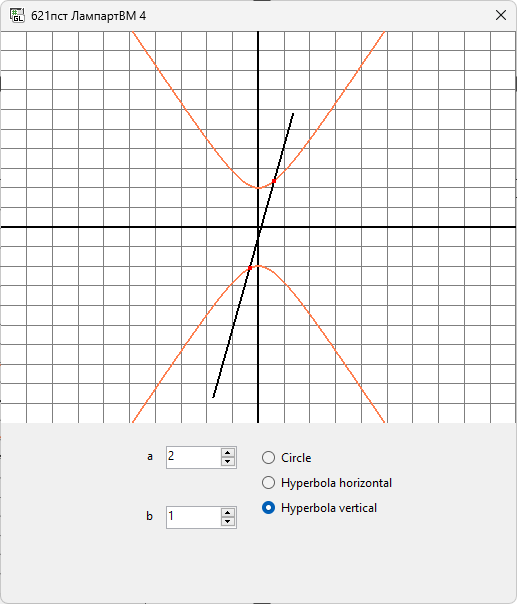


Рисунок 4.3 – Результат роботи програми (Вертикальна гіпербола)

Таблиця 4.1 – Результат виконання практичної роботи №3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Складність | Вимоги до роботи | Бали | Оцінка |
| 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 виведенняпрограми за допомогою маніпулятора «миш» | 2 | + |
| 5. | Використання ООП | 1 | + |

# ДОДАТОК А Лістинг програми до практичної роботи №1

**Код файлу RenderControl:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Diagnostics;

using System.Drawing;

using System.Drawing.Design;

using System.Linq;

namespace glWinForm1

{

public partial class RenderControl : OpenGL

{

public RenderControl()

{

InitializeComponent();

}

private void OnRender(object sender, EventArgs e)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glLoadIdentity();

glViewport(0, 0, Width, Height);

gluOrtho2D(-3, 10, -4, 4);

Grid grid = new Grid();

grid.DrawGrid();

Figure13 figure = new Figure13();

figure.Shape();

figure.Points();

}

}

}

**Код файлу Figure13:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace glWinForm1

{

public partial class RenderControl

{

public class Figure13

{

private readonly (double x, double y)[] shapePoints = new[]

{

(-1.0, 2.0),

(0.0, 2.0),

(1.0, 1.0),

(2.0, -1.0),

(2.0, -2.0),

(0.0, -1.0),

(-1.0, 0.0)

};

private readonly (double x, double y)[] pointPoints = new[]

{

(4.0, 2.0),

(5.0, 2.0),

(6.0, 1.0),

(7.0, -1.0),

(7.0, -2.0),

(5.0, -1.0),

(4.0, 0.0)

};

public void Shape()

{

glLineWidth(2);

glColor3d(0, 0, 0);

glBegin(GL\_LINES);

for (int i = 0; i < shapePoints.Length; i++)

{

var point1 = shapePoints[i];

var point2 = shapePoints[(i + 1) % shapePoints.Length];

glVertex2d(point1.x, point1.y);

glVertex2d(point2.x, point2.y);

}

glEnd();

}

public void Points()

{

glEnable(GL\_POINT\_SMOOTH);

glPointSize(4);

glColor3d(0, 0, 0);

glBegin(GL\_POINTS);

foreach (var point in pointPoints)

{

glVertex2d(point.x, point.y);

}

glEnd();

glDisable(GL\_POINT\_SMOOTH);

}

}

}

}

**Код файлу Grid:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace glWinForm1

{

public partial class RenderControl

{

internal class Grid

{

public void DrawGrid()

{

glLineWidth(1);

glEnable(GL\_LINE\_STIPPLE);

glLineStipple(5, 0xAAAA);

glColor3d(255, 255, 255);

glBegin(GL\_LINES);

for (int i = -1; i <= 8; i++)

{

glVertex2d(i, -3);

glVertex2d(i, 3);

}

for (int j = -2; j <= 2; j++)

{

glVertex2d(-2, j); /

glVertex2d(9, j);

}

glEnd();

glDisable(GL\_LINE\_STIPPLE);

}

}

}

}

# ДОДАТОК Б Лістинг програми до практичної роботи №3

**Код файлу MainForm:**

using System.Drawing;

using System.Windows.Forms;

using static glWinForm1.OpenGL;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.Button;

namespace glWinForm1

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

renderControl1.XMin = (double)xMinNumericUpDown.Value;

renderControl1.XMax = (double)xMaxNumericUpDown.Value;

renderControl1.YMin = (double)yMinNumericUpDown.Value;

renderControl1.YMax = (double)yMaxNumericUpDown.Value;

}

private void xMinNumericUpDown\_ValueChanged(object sender, System.EventArgs e)

{

renderControl1.XMin = (double)xMinNumericUpDown.Value;

renderControl1.Invalidate();

}

private void xMaxNumericUpDown\_ValueChanged(object sender, System.EventArgs e)

{

renderControl1.XMax = (double)xMaxNumericUpDown.Value;

renderControl1.Invalidate();

}

private void pointsNumericUpDown\_ValueChanged(object sender, System.EventArgs e)

{

renderControl1.points = (double)pointsNumericUpDown.Value;

renderControl1.Invalidate();

}

private void checkBox\_CheckedChanged(object sender, System.EventArgs e)

{

label3.Visible = checkBox.Checked;

label4.Visible = checkBox.Checked;

yMinNumericUpDown.Visible = checkBox.Checked;

yMaxNumericUpDown.Visible = checkBox.Checked;

renderControl1.YMin = -10.0;

renderControl1.YMax = 10.0;

renderControl1.showY = !renderControl1.showY;

renderControl1.Invalidate();

}

private void yMinNumericUpDown\_ValueChanged(object sender, System.EventArgs e)

{

renderControl1.YMin = (double)yMinNumericUpDown.Value;

renderControl1.Invalidate();

}

private void yMaxNumericUpDown\_ValueChanged(object sender, System.EventArgs e)

{

renderControl1.YMax = (double)yMaxNumericUpDown.Value;

renderControl1.Invalidate();

}

private void radioButton1\_CheckedChanged(object sender, System.EventArgs e)

{

renderControl1.choice = 1;

renderControl1.Invalidate();

}

private void radioButton2\_CheckedChanged(object sender, System.EventArgs e)

{

renderControl1.choice = 2;

renderControl1.Invalidate();

}

private void button1\_Click(object sender, System.EventArgs e)

{

xMinNumericUpDown.Value = -10;

xMaxNumericUpDown.Value = 10;

pointsNumericUpDown.Value = 100;

checkBox.Checked = false;

yMinNumericUpDown.Value = -10;

yMaxNumericUpDown.Value = 10;

renderControl1.showY = false;

renderControl1.Invalidate();

}

}

}

**Код файлу RenderControl:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Diagnostics;

using System.Drawing;

using System.Drawing.Design;

using System.Linq;

namespace glWinForm1

{

public partial class RenderControl : OpenGL

{

protected MathFunction currentFunction;

Function13 function13 = new Function13();

public double points = 100;

public bool showY { get; set; } = false;

public double XMin { get; set; } = -3.0;

public double XMax { get; set; } = 3.0;

public double YMin { get; set; } = -3.0;

public double YMax { get; set; } = 3.0;

public byte choice { get; set; } = 1;

private byte functionChoice = 1;

public RenderControl()

{

InitializeComponent();

InitializeFunction(functionChoice);

}

protected void InitializeFunction(byte choice)

{

currentFunction = choice switch

{

1 => new FirstFunc(),

2 => new SecondFunc(),

\_ => new FirstFunc()

};

}

public void SetFunction(byte choice)

{

functionChoice = choice;

InitializeFunction(choice);

}

private void OnRender(object sender, EventArgs e)

{

glClearColor(1, 1, 1, 1);

glClear(GL\_COLOR\_BUFFER\_BIT);

glLoadIdentity();

glViewport(0, 0, Width, Height);

if (!showY)

(YMin, YMax) = function13.CalculateY(XMin, XMax, points, YMin, YMax, choice);

YMax = Math.Min(YMax, 100);

YMin = Math.Max(YMin, -100);

gluOrtho2D(XMin, XMax, YMin, YMax);

Grid(XMin, XMax, YMin, YMax);

Axis(XMin, XMax, YMin, YMax);

function13.Graph(XMin, XMax, points, YMin, YMax, choice);

}

public void Grid(double XMin, double XMax, double YMin, double YMax)

{

glLineWidth(0.3f);

glColor3d(0, 0, 0);

glBegin(GL\_LINES);

for (double i = XMin; i <= XMax; i++)

{

glVertex2d(i, YMin);

glVertex2d(i, YMax);

}

for (double i = YMin; i <= YMax; i++)

{

glVertex2d(XMin, i);

glVertex2d(XMax, i);

}

glEnd();

}

public void Axis(double xMin, double xMax, double yMin, double yMax)

{

glLineWidth(2);

glColor3d(0, 0, 0);

glBegin(GL\_LINES);

glVertex2d(xMin, 0);

glVertex2d(xMax, 0);

glVertex2d(0, yMin);

glVertex2d(0, yMax);

glEnd();

}

}

}

**Код файлу Function13:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace glWinForm1

{

public partial class RenderControl

{

internal class Function13

{

private FirstFunc firstFunc = new FirstFunc();

private SecondFunc secondFunc = new SecondFunc();

public double Calculate(double x, byte choice)

{

return choice switch

{

1 => firstFunc.Calculate(x),

2 => secondFunc.Calculate(x),

\_ => 0

};

}

public (double, double) CalculateY(double XMin, double XMax, double points, double YMin, double YMax, byte choice)

{

MathFunction currentFunction = choice switch

{

1 => firstFunc,

2 => secondFunc,

\_ => firstFunc

};

return currentFunction.GetYRange(XMin, XMax, points);

}

public void Graph(double XMin, double XMax, double points, double YMin, double YMax, byte choice)

{

glLineWidth(2);

glColor3ub(255, 127, 80);

glBegin(GL\_LINE\_STRIP);

double prevY = double.NaN;

double x = XMin;

double step = (XMax - XMin) / (points - 1);

double y = Calculate(x, choice);

glVertex2d(x, y);

for (int i = 1; i < points; i++)

{

prevY = y;

x = XMin + i \* step;

y = Calculate(x, choice);

if (!double.IsNaN(prevY) && Math.Abs(y - prevY) > 1.0)

{

glEnd();

glBegin(GL\_LINE\_STRIP);

continue;

}

glVertex2d(x, y);

if ((prevY \* y) <= 0 && prevY != 0)

{

glEnd();

CrossingX(prevY, x, step, y);

glBegin(GL\_LINE\_STRIP);

glColor3ub(255, 127, 80);

glVertex2d(x, y);

}

}

glEnd();

}

private void CrossingX(double prevY, double x, double step, double y)

{

glPointSize(5);

glColor3d(1, 0, 0);

glBegin(GL\_POINTS);

glVertex2d(x - step / 2, (prevY + y) / 2);

glEnd();

}

}

}

}

**Код файлу AMathFunc:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace glWinForm1

{

public abstract class MathFunction

{

public abstract double Calculate(double x);

public abstract (double min, double max) GetYRange(double xMin, double xMax, double points);

}

}

**Код файлу FirstFunc:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace glWinForm1

{

internal class FirstFunc: MathFunction

{

public override double Calculate(double x)

{

return (Math.Abs(Math.Cos(0.5 \* x + 1)) \* Math.Cos(x)) / Math.Abs(Math.Cos(x + 0.01));

}

public override (double min, double max) GetYRange(double xMin, double xMax, double points)

{

double min = double.MaxValue;

double max = double.MinValue;

double step = (xMax - xMin) / (points - 1);

for (int i = 0; i < points; i++)

{

double x = xMin + i \* step;

double y = Calculate(x);

if (!double.IsNaN(y) && !double.IsInfinity(y))

{

min = Math.Min(min, y);

max = Math.Max(max, y);

}

}

return (min, max);

}

}

}

**Код файлу SecondFunc:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace glWinForm1

{

internal class SecondFunc : MathFunction

{

public override double Calculate(double x)

{

return (Math.Cos(Math.PI \* x / 2) \* Math.Cos(Math.Abs(Math.PI \* x))) / Math.Abs(Math.Sin(Math.PI \* x));

}

public override (double min, double max) GetYRange(double xMin, double xMax, double points)

{

double min = double.MaxValue;

double max = double.MinValue;

double step = (xMax - xMin) / (points - 1);

for (int i = 0; i < points; i++)

{

double x = xMin + i \* step;

double y = Calculate(x);

if (!double.IsNaN(y) && !double.IsInfinity(y))

{

min = Math.Min(min, y);

max = Math.Max(max, y);

}

}

return (min, max);

}

}

}

# ДОДАТОК В Лістинг програми до практичної роботи №4

**Код файлу RenderControl:**

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using static glWinForm1.OpenGL;

namespace glWinForm1

{

public partial class RenderControl : OpenGL

{

// Тепер тут зберігається поточна фігура (коло / гіпербола).

public IShape2D CurrentShape { get; set; }

// Межі області малювання

private double margin = 10.0;

// Дані для відрізка

public double XStart { get; set; }

public double YStart { get; set; }

public double XEnd { get; set; }

public double YEnd { get; set; }

public bool DrawLine { get; set; } = false;

public RenderControl()

{

InitializeComponent();

this.MouseClick += RenderControl\_MouseClick;

}

private void OnRender(object sender, EventArgs e)

{

glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

glClear(GL\_COLOR\_BUFFER\_BIT);

glLoadIdentity();

glViewport(0, 0, Width, Height);

// Ортонормальне проєціювання

gluOrtho2D(-margin, margin, -margin, margin);

// Малюємо сітку та осі

Grid(-margin, margin, -margin, margin);

Axis(-margin, margin, -margin, margin);

// Малюємо поточну фігуру, якщо вона задана

CurrentShape?.Draw();

// Якщо користувач клацнув двічі і є відрізок – малюємо його

if (DrawLine)

{

glColor3ub(0, 0, 0);

glBegin(GL\_LINES);

glVertex2d(XStart, YStart);

glVertex2d(XEnd, YEnd);

glEnd();

// Знаходимо точки перетину з поточною фігурою

var points = CurrentShape?.GetIntersections(XStart, YStart, XEnd, YEnd);

if (points != null && points.Count > 0)

{

glColor3d(1, 0, 0);

glPointSize(4);

glBegin(GL\_POINTS);

foreach (var p in points)

{

glVertex2d(p.x, p.y);

}

glEnd();

}

}

}

private void RenderControl\_MouseClick(object sender, MouseEventArgs e)

{

double viewportSize = (Width > Height) ? Height : Width;

double xOffset = (Width > Height) ? (Width - Height) / 2.0 : 0;

double yOffset = (Width > Height) ? 0 : (Height - Width) / 2.0;

// Переводимо піксельні координати в Normalized Device Coordinates (-1..1)

double normalizedX = (e.X - xOffset) / viewportSize \* 2.0 - 1.0;

// Увага, для Y йде «дзеркало»:

double normalizedY = -((e.Y - yOffset) / viewportSize \* 2.0 - 1.0);

// Тепер маштабуємо під нашу margin

double x = normalizedX \* margin;

double y = normalizedY \* margin;

if (e.Button == MouseButtons.Left)

{

// Перша точка відрізка

XStart = x;

YStart = y;

}

else if (e.Button == MouseButtons.Right)

{

// Друга точка відрізка

XEnd = x;

YEnd = y;

DrawLine = true;

}

Invalidate();

}

private void Grid(double xMin, double xMax, double yMin, double yMax)

{

glLineWidth(0.3f);

glColor3d(0.5, 0.5, 0.5);

glBegin(GL\_LINES);

for (double i = xMin; i <= xMax; i++)

{

glVertex2d(i, yMin);

glVertex2d(i, yMax);

}

for (double i = yMin; i <= yMax; i++)

{

glVertex2d(xMin, i);

glVertex2d(xMax, i);

}

glEnd();

}

private void Axis(double xMin, double xMax, double yMin, double yMax)

{

glLineWidth(2);

glColor3d(0, 0, 0);

glBegin(GL\_LINES);

glVertex2d(xMin, 0);

glVertex2d(xMax, 0);

glVertex2d(0, yMin);

glVertex2d(0, yMax);

glEnd();

}

}

}

**Код файлу IShape2D:**

using System;

using System.Collections.Generic;

namespace glWinForm1

{

public interface IShape2D

{

void Draw();

List<(double x, double y)> GetIntersections(double x1, double y1, double x2, double y2);

}

}

**Код файлу Circle:**

using System;

using System.Collections.Generic;

using static glWinForm1.OpenGL;

namespace glWinForm1

{

public class Circle : IShape2D

{

public double H { get; set; }

public double K { get; set; }

public double R { get; set; }

private const int SEGMENTS = 100;

public Circle(double h, double k, double r)

{

H = h;

K = k;

R = r;

}

public void Draw()

{

glColor3ub(255, 127, 80); // колір

glBegin(GL\_LINE\_LOOP);

for (int i = 0; i <= SEGMENTS; i++)

{

double theta = 2.0 \* Math.PI \* i / SEGMENTS;

double x = H + R \* Math.Cos(theta);

double y = K + R \* Math.Sin(theta);

glVertex2d(x, y);

}

glEnd();

}

public List<(double x, double y)> GetIntersections(double x1, double y1, double x2, double y2)

{

// Перетворимо задачу на знаходження t у рівнянні прямої

double dx = x2 - x1;

double dy = y2 - y1;

// Коефіцієнти квадратного рівняння A t^2 + B t + C = 0

double A = dx \* dx + dy \* dy;

double B = 2 \* (dx \* (x1 - H) + dy \* (y1 - K));

double C = (x1 - H) \* (x1 - H) + (y1 - K) \* (y1 - K) - R \* R;

double D = B \* B - 4 \* A \* C; // дискримінант

List<(double x, double y)> result = new List<(double x, double y)>();

if (D < 0)

return result;

// Знаходимо два розв’язки

double sqrtD = Math.Sqrt(D);

double t1 = (-B - sqrtD) / (2 \* A);

double t2 = (-B + sqrtD) / (2 \* A);

// Перевіряємо, чи лежать ці точки на відрізку (0 <= t <= 1)

if (t1 >= 0 && t1 <= 1)

{

double ix = x1 + t1 \* dx;

double iy = y1 + t1 \* dy;

result.Add((ix, iy));

}

if (t2 >= 0 && t2 <= 1)

{

double ix = x1 + t2 \* dx;

double iy = y1 + t2 \* dy;

result.Add((ix, iy));

}

return result;

}

}

}

**Код файлуHorizontalHyperbola:**

using System;

using System.Collections.Generic;

using static glWinForm1.OpenGL;

namespace glWinForm1

{

public class HorizontalHyperbola : IShape2D

{

public double H { get; set; }

public double K { get; set; }

public double A { get; set; }

public double B { get; set; }

// Наскільки далеко малювати гілки гіперболи

private const double TMax = 5.0;

private const double Step = 0.1;

public HorizontalHyperbola(double h, double k, double a, double b)

{

H = h;

K = k;

A = a;

B = b;

}

public void Draw()

{

glColor3ub(255, 127, 80);

// Права гілка

glBegin(GL\_LINE\_STRIP);

for (double t = -TMax; t <= TMax; t += Step)

{

double x = H + A \* Math.Cosh(t);

double y = K + B \* Math.Sinh(t);

glVertex2d(x, y);

}

glEnd();

// Ліва гілка (змінюємо знак)

glBegin(GL\_LINE\_STRIP);

for (double t = -TMax; t <= TMax; t += Step)

{

double x = H - A \* Math.Cosh(t);

double y = K - B \* Math.Sinh(t);

glVertex2d(x, y);

}

glEnd();

}

public List<(double x, double y)> GetIntersections(double x1, double y1, double x2, double y2)

{

// Переводимо відрізок у локальні координати (щоб центр гіперболи був у (0,0))

x1 -= H; y1 -= K;

x2 -= H; y2 -= K;

double dx = x2 - x1;

double dy = y2 - y1;

// Коефіцієнти рівняння для x^2/a^2 - y^2/b^2 = 1

// У параметричному вигляді x(t) = x1 + t\*dx, y(t) = y1 + t\*dy

double A\_ = (dx \* dx) / (A \* A) - (dy \* dy) / (B \* B);

double B\_ = 2 \* ((x1 \* dx) / (A \* A) - (y1 \* dy) / (B \* B));

double C\_ = (x1 \* x1) / (A \* A) - (y1 \* y1) / (B \* B) - 1;

List<(double x, double y)> result = new List<(double x, double y)>();

double D = B\_ \* B\_ - 4 \* A\_ \* C\_;

if (D < 0) return result;

double sqrtD = Math.Sqrt(D);

double t1 = (-B\_ - sqrtD) / (2 \* A\_);

double t2 = (-B\_ + sqrtD) / (2 \* A\_);

if (t1 >= 0 && t1 <= 1)

{

double ix = x1 + t1 \* dx + H;

double iy = y1 + t1 \* dy + K;

result.Add((ix, iy));

}

if (t2 >= 0 && t2 <= 1)

{

double ix = x1 + t2 \* dx + H;

double iy = y1 + t2 \* dy + K;

result.Add((ix, iy));

}

return result;

}

}

}

**Код файлуVerticallHyperbola:**

using System;

using System.Collections.Generic;

using static glWinForm1.OpenGL;

namespace glWinForm1

{

public class VerticalHyperbola : IShape2D

{

public double H { get; set; }

public double K { get; set; }

public double A { get; set; }

public double B { get; set; }

private const double TMax = 5.0;

private const double Step = 0.1;

public VerticalHyperbola(double h, double k, double a, double b)

{

H = h;

K = k;

A = a;

B = b;

}

public void Draw()

{

glColor3ub(255, 127, 80);

// Верхня й нижня гілки

glBegin(GL\_LINE\_STRIP);

for (double t = -TMax; t <= TMax; t += Step)

{

double x = H + B \* Math.Sinh(t);

double y = K + A \* Math.Cosh(t);

glVertex2d(x, y);

}

glEnd();

glBegin(GL\_LINE\_STRIP);

for (double t = -TMax; t <= TMax; t += Step)

{

double x = H - B \* Math.Sinh(t);

double y = K - A \* Math.Cosh(t);

glVertex2d(x, y);

}

glEnd();

}

public List<(double x, double y)> GetIntersections(double x1, double y1, double x2, double y2)

{

// Локальна трансляція

x1 -= H; y1 -= K;

x2 -= H; y2 -= K;

double dx = x2 - x1;

double dy = y2 - y1;

// Для "вертикальної" гіперболи: x^2/b^2 - y^2/a^2 = -1

// або (y^2 / a^2) - (x^2 / b^2) = 1

double A\_ = (dy \* dy) / (A \* A) - (dx \* dx) / (B \* B);

double B\_ = 2 \* ((y1 \* dy) / (A \* A) - (x1 \* dx) / (B \* B));

double C\_ = (y1 \* y1) / (A \* A) - (x1 \* x1) / (B \* B) - 1;

List<(double x, double y)> result = new List<(double x, double y)>();

double D = B\_ \* B\_ - 4 \* A\_ \* C\_;

if (D < 0) return result;

double sqrtD = Math.Sqrt(D);

double t1 = (-B\_ - sqrtD) / (2 \* A\_);

double t2 = (-B\_ + sqrtD) / (2 \* A\_);

if (t1 >= 0 && t1 <= 1)

{

double ix = x1 + t1 \* dx + H;

double iy = y1 + t1 \* dy + K;

result.Add((ix, iy));

}

if (t2 >= 0 && t2 <= 1)

{

double ix = x1 + t2 \* dx + H;

double iy = y1 + t2 \* dy + K;

result.Add((ix, iy));

}

return result;

}

}

}

**Код файлу MainForm:**

using System;

using System.Windows.Forms;

namespace glWinForm1

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

hNumericUpDown.ValueChanged += Nud\_ValueChanged;

kNumericUpDown.ValueChanged += Nud\_ValueChanged;

rNumericUpDown.ValueChanged += Nud\_ValueChanged;

aNumericUpDown.ValueChanged += Nud\_ValueChanged;

bNumericUpDown.ValueChanged += Nud\_ValueChanged;

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton1.Checked)

{

// Коло

ShowCircleControls(true);

ShowHyperbolaControls(false);

// Створюємо екземпляр кола

var h = (double)hNumericUpDown.Value;

var k = (double)kNumericUpDown.Value;

var r = (double)rNumericUpDown.Value;

renderControl1.CurrentShape = new Circle(h, k, r);

renderControl1.Invalidate();

}

}

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton2.Checked)

{

// Горизонтальна гіпербола

ShowCircleControls(false);

ShowHyperbolaControls(true);

var a = (double)aNumericUpDown.Value;

var b = (double)bNumericUpDown.Value;

// h, k у цьому прикладі не використовуються явно,

// або ж можна додати NumericUpDown для h і k,

// якщо хочете зсунути центр гіперболи.

// Тут, щоб показати, як використати h, k, можна зробити так:

var h = (double)hNumericUpDown.Value;

var k = (double)kNumericUpDown.Value;

renderControl1.CurrentShape = new HorizontalHyperbola(h, k, a, b);

renderControl1.Invalidate();

}

}

private void radioButton3\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton3.Checked)

{

// Вертикальна гіпербола

ShowCircleControls(false);

ShowHyperbolaControls(true);

var a = (double)aNumericUpDown.Value;

var b = (double)bNumericUpDown.Value;

var h = (double)hNumericUpDown.Value;

var k = (double)kNumericUpDown.Value;

renderControl1.CurrentShape = new VerticalHyperbola(h, k, a, b);

renderControl1.Invalidate();

}

}

private void Nud\_ValueChanged(object sender, EventArgs e)

{

// Перевіряємо, яка фігура вибрана зараз (через radioButton)

if (radioButton1.Checked)

{

// Якщо коло

renderControl1.CurrentShape = new Circle(

(double)hNumericUpDown.Value,

(double)kNumericUpDown.Value,

(double)rNumericUpDown.Value

);

}

else if (radioButton2.Checked)

{

// Горизонтальна гіпербола

renderControl1.CurrentShape = new HorizontalHyperbola(

(double)hNumericUpDown.Value,

(double)kNumericUpDown.Value,

(double)aNumericUpDown.Value,

(double)bNumericUpDown.Value

);

}

else if (radioButton3.Checked)

{

// Вертикальна гіпербола

renderControl1.CurrentShape = new VerticalHyperbola(

(double)hNumericUpDown.Value,

(double)kNumericUpDown.Value,

(double)aNumericUpDown.Value,

(double)bNumericUpDown.Value

);

}

renderControl1.Invalidate();

}

private void ShowCircleControls(bool show)

{

hLabel.Visible = show;

kLabel.Visible = show;

rLabel.Visible = show;

hNumericUpDown.Visible = show;

kNumericUpDown.Visible = show;

rNumericUpDown.Visible = show;

}

private void ShowHyperbolaControls(bool show)

{

aLabel.Visible = show;

bLabel.Visible = show;

aNumericUpDown.Visible = show;

bNumericUpDown.Visible = show;

}

private void resetBtn\_Click(object sender, EventArgs e)

{

ShowCircleControls(false);

ShowHyperbolaControls(false);

}

}

}