Державний вищий навчальний заклад  
Ужгородський національний університет  
Факультет інформаційних технологій

Кафедра програмного забезпечення систем

**ЛАБОРАТОРНА РОБОТА № 1**

**Тема:** Моделювання кінематики прямолінійного руху

Виконав:

студент ІІ курсу

спеціальності: 121 інженерія програмного забезпечення

Заяць Дмитро Ігорович

**Ужгород-2025**

**Мета**: отримати уявлення про принципи побудови траєкторії руху тіла.

**Завдання до лабораторної роботи**

1. Використовуючи базові навички програмування побудувати траєкторію руху матеріальної точки в залежності від вхідних параметрів, які будуть задані користувачем:

а ) х0

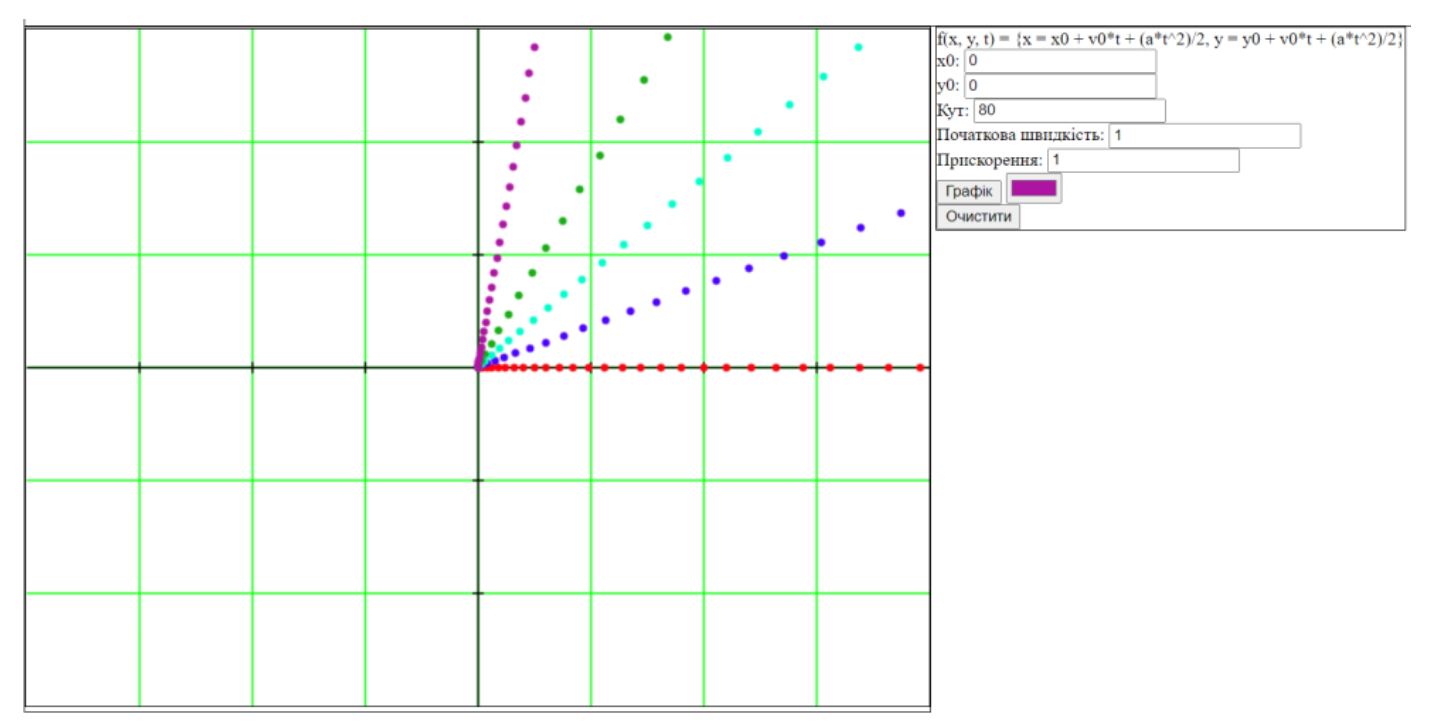
б) у0

в) кут

г) початкова швидкiсть

д) прискорення

Приклад отриманого результату:

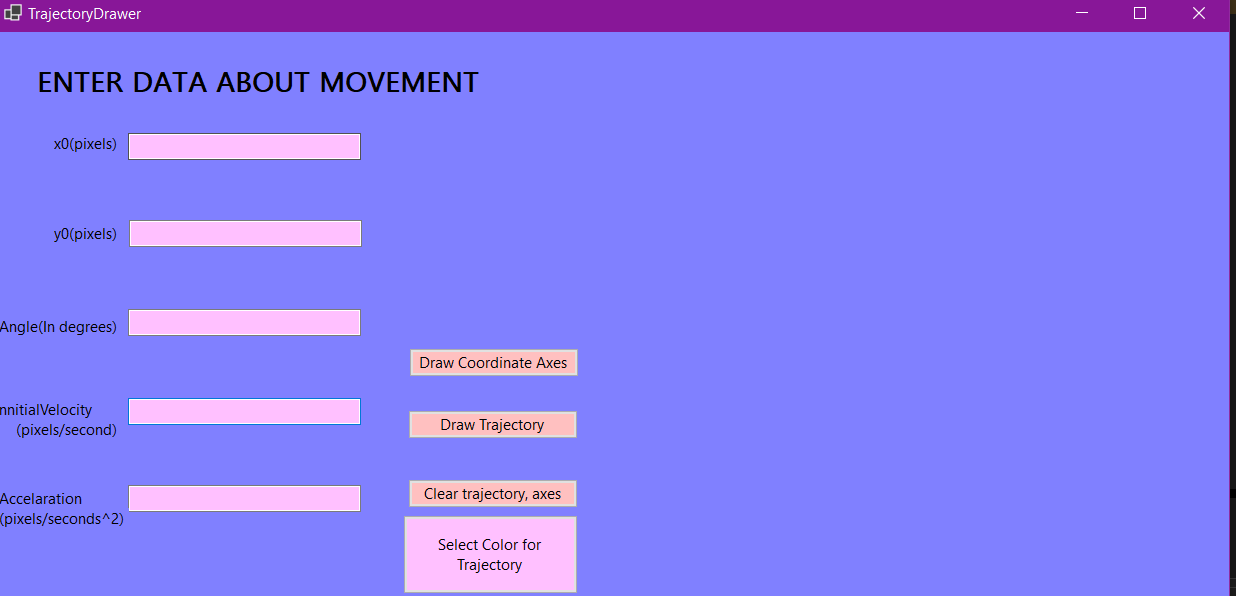


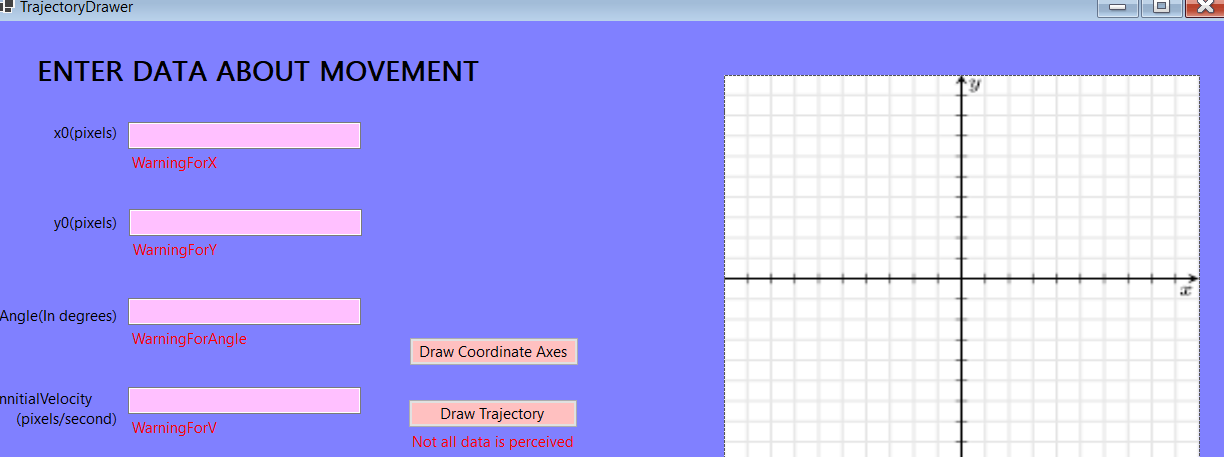
Як видно з графіку, користувач повинен мати змогу задавати колір траєкторії, а також мати можливість очистити графік.

1. Оформити звіт виконаної роботи, згідно взірця
2. Отримані результати (завантажити на **репозиторій**, та **звіт,** завантажити у *папку курсу*, *підпапку* лабораторної роботи та *підпаку* із назвою, що відповідає вашому прізвищу). Якщо код завантажено на репозиторій, то у звіті потрібно вказати посилання на репозиторій.: <https://github.com/ZayatsDmytro/MethodsOfPE>

**Хід роботи**

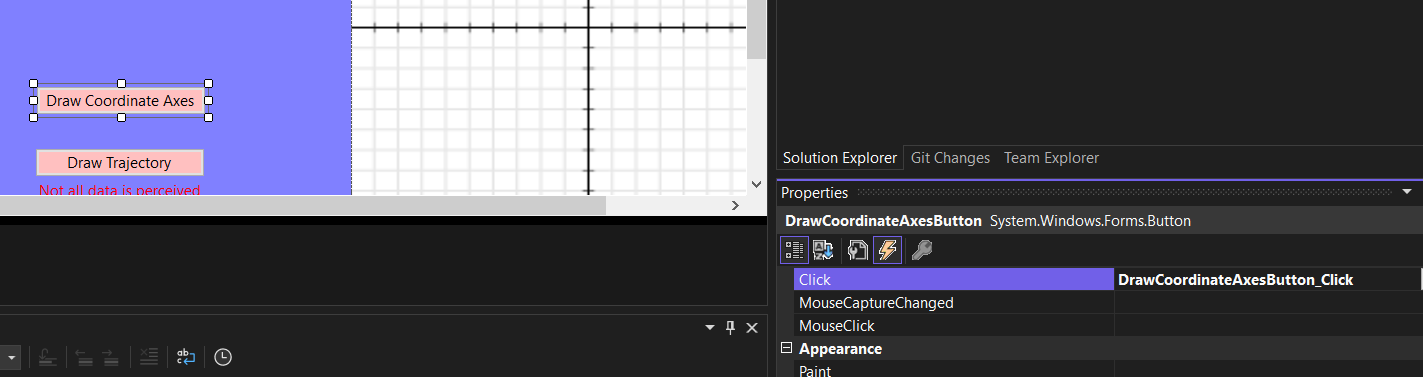
**Інтерфейс та елементи управління**  
На форму були додані всі необхідні елементи управління: кнопки, попередження (Labels), рисунки (PictureBox), TextBox для вводу даних користувача, а також кнопка для вибору кольору траєкторії. Всі ці елементи було налаштовано таким чином, щоб забезпечити зручність користування та відображення необхідної інформації.

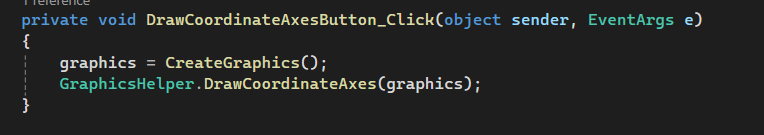


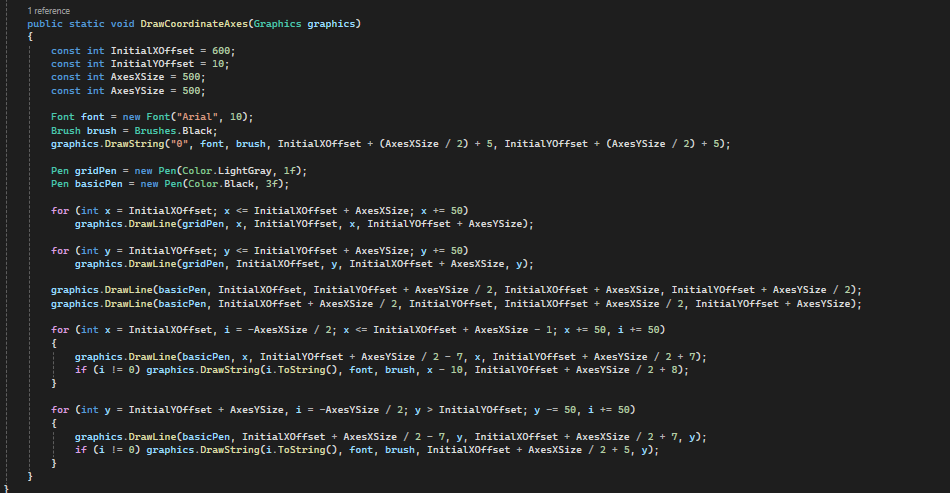


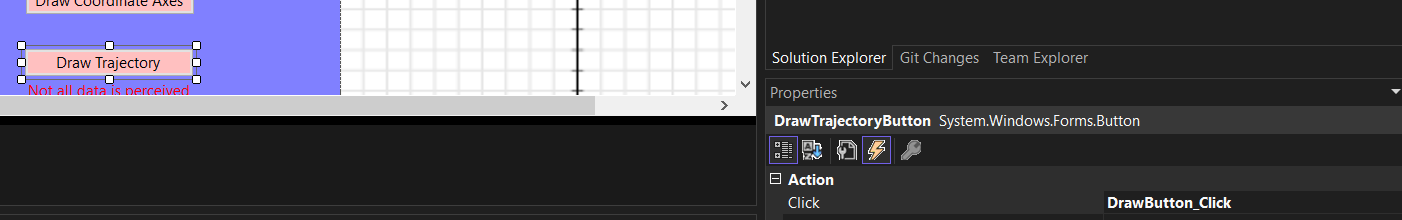
**Прив'язка подій до елементів**  
До деяких елементів було прив'язано обробники подій, що забезпечують виконання певних дій:

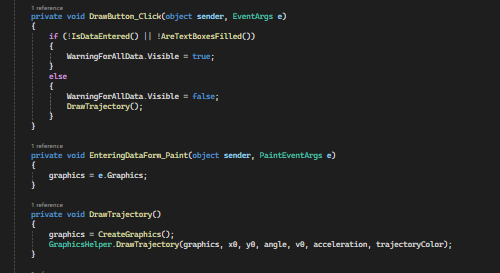
* **Виведення траєкторії:** після того, як користувач введе необхідні дані та натисне кнопку "Намалювати", програма перевіряє коректність вводу і малює траєкторію.
* **Очищення форми:** натискання кнопки "Очистити" дозволяє очистити всю форму, включаючи попередження і всі виведені графічні елементи.
* **Зміна кольору траєкторії:** користувач має можливість змінювати колір траєкторії через стандартний діалог для вибору кольору.

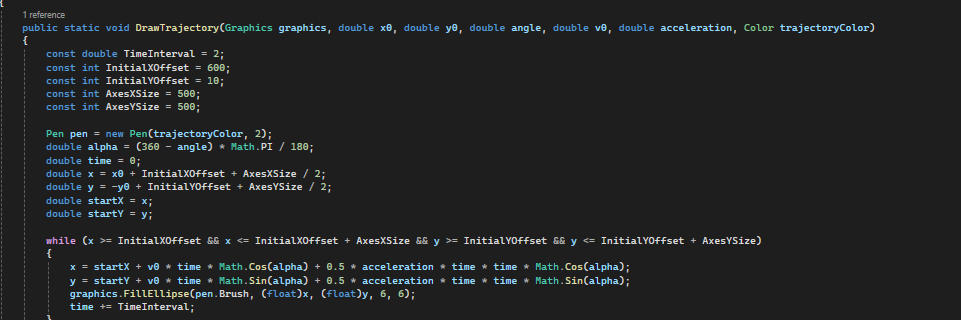


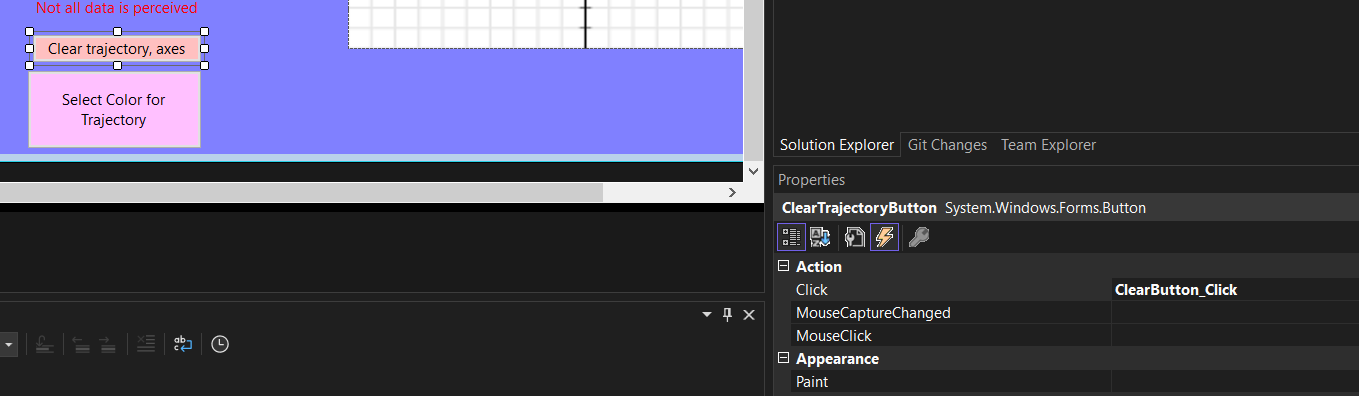


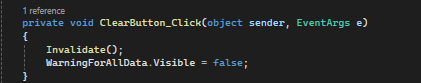




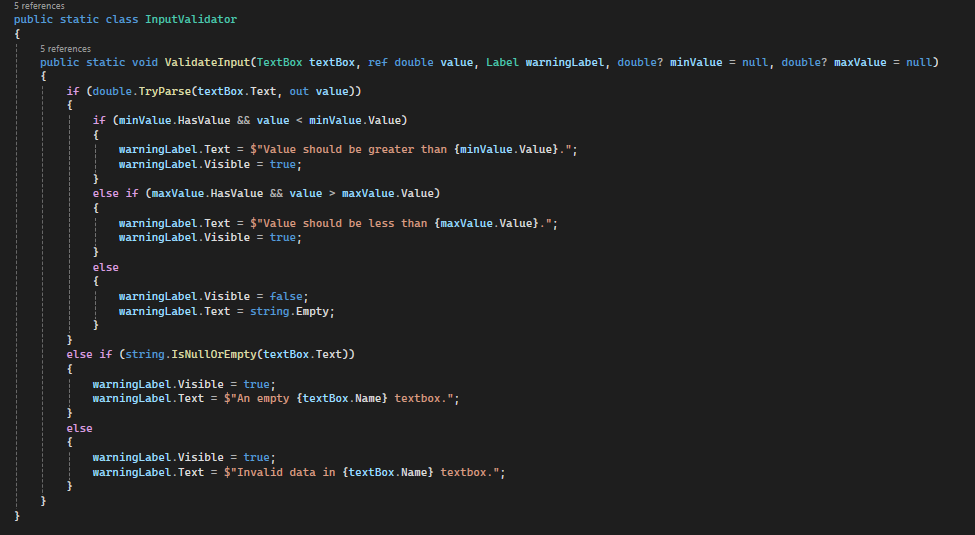




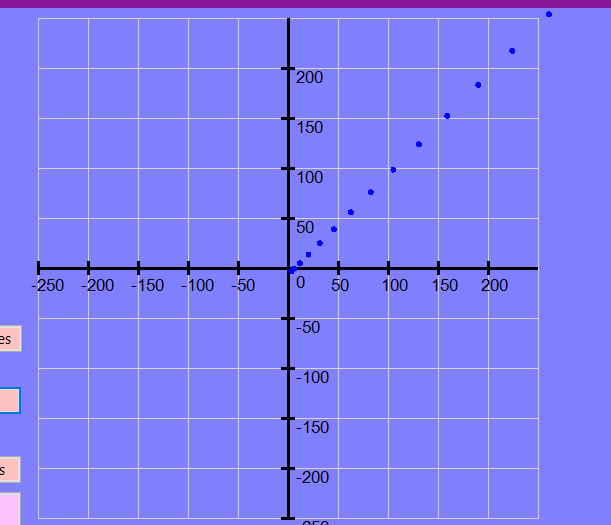




**Перевірка вводу даних**  
Для кожного текстового поля (TextBox) реалізована валідація вводу. Введення даних перевіряється на коректність і відповідність заданим межам (наприклад, координати повинні бути в межах від -250 до 250). У разі помилки або неправильного вводу користувач отримує попередження у вигляді повідомлення, яке відображається поруч з відповідним полем вводу.



**Робота з даними та малювання траєкторії**  
При натисканні кнопки "Намалювати траєкторію", програма використовує введені користувачем дані (початкові координати, кут, швидкість, прискорення) для обчислення траєкторії руху. Це здійснюється за допомогою математичних розрахунків фізичних параметрів руху, таких як час, швидкість, прискорення, а також врахуванням напрямку траєкторії через кут. Результат зображується на координатній площині.



**Рефакторинг коду**  
В процесі роботи над проектом був проведений рефакторинг:

* Зайві методи були видалені, що дозволило зменшити складність і полегшити підтримку коду.
* Клас форми був розбитий на окремі класи відповідно до принципу **Single Responsibility Principle (SRP)**. Це дозволило зробити код більш модульним та легким для подальших змін та доповнень.

У проекті були створені два допоміжні класи:

* **GraphicsHelper:** цей клас відповідає за малювання на формі, зокрема для відображення траєкторії руху та координатних осей. Він реалізує логіку малювання траєкторії, координатних осей і сітки.
* **InputValidator:** клас, що відповідає за перевірку вводу даних. Він здійснює перевірку на наявність помилок у введених даних, а також перевіряє, чи відповідають дані заданим межам.

Додавання другого способу:  
Створили окремий клас для намалювання траєкторії на малюнку і збереження у підпапки проекту.  
public static class ImageGraphicsHelper

{

public static void DrawTrajectoryAndSaveToFile(double x0, double y0, double angle, double v0, double acceleration, Color trajectoryColor, string filePath)

{

const double TimeInterval = 2;

const int InitialXOffset = 600;

const int InitialYOffset = 10;

const int AxesXSize = 500;

const int AxesYSize = 500;

Bitmap bitmap = new Bitmap(InitialXOffset + AxesXSize, InitialYOffset + AxesYSize);

using (Graphics graphics = Graphics.FromImage(bitmap))

{

DrawCoordinateAxes(graphics);

Pen pen = new Pen(trajectoryColor, 2);

double alpha = (360 - angle) \* Math.PI / 180;

double time = 0;

double x = x0 + InitialXOffset + AxesXSize / 2;

double y = -y0 + InitialYOffset + AxesYSize / 2;

double startX = x;

double startY = y;

while (x >= InitialXOffset && x <= InitialXOffset + AxesXSize && y >= InitialYOffset && y <= InitialYOffset + AxesYSize)

{

x = startX + v0 \* time \* Math.Cos(alpha) + 0.5 \* acceleration \* time \* time \* Math.Cos(alpha);

y = startY + v0 \* time \* Math.Sin(alpha) + 0.5 \* acceleration \* time \* time \* Math.Sin(alpha);

graphics.FillEllipse(pen.Brush, (float)x, (float)y, 6, 6);

time += TimeInterval;

}

bitmap.Save(filePath, System.Drawing.Imaging.ImageFormat.Png);

}

}

public static void DrawCoordinateAxes(Graphics graphics)

{

const int InitialXOffset = 600;

const int InitialYOffset = 10;

const int AxesXSize = 500;

const int AxesYSize = 500;

Font font = new Font("Arial", 10);

Brush brush = Brushes.Black;

graphics.DrawString("0", font, brush, InitialXOffset + (AxesXSize / 2) + 5, InitialYOffset + (AxesYSize / 2) + 5);

Pen gridPen = new Pen(Color.LightGray, 1f);

Pen basicPen = new Pen(Color.Black, 3f);

for (int x = InitialXOffset; x <= InitialXOffset + AxesXSize; x += 50)

graphics.DrawLine(gridPen, x, InitialYOffset, x, InitialYOffset + AxesYSize);

for (int y = InitialYOffset; y <= InitialYOffset + AxesYSize; y += 50)

graphics.DrawLine(gridPen, InitialXOffset, y, InitialXOffset + AxesXSize, y);

graphics.DrawLine(basicPen, InitialXOffset, InitialYOffset + AxesYSize / 2, InitialXOffset + AxesXSize, InitialYOffset + AxesYSize / 2);

graphics.DrawLine(basicPen, InitialXOffset + AxesXSize / 2, InitialYOffset, InitialXOffset + AxesXSize / 2, InitialYOffset + AxesYSize);

for (int x = InitialXOffset, i = -AxesXSize / 2; x <= InitialXOffset + AxesXSize - 1; x += 50, i += 50)

{

graphics.DrawLine(basicPen, x, InitialYOffset + AxesYSize / 2 - 7, x, InitialYOffset + AxesYSize / 2 + 7);

if (i != 0) graphics.DrawString(i.ToString(), font, brush, x - 10, InitialYOffset + AxesYSize / 2 + 8);

}

for (int y = InitialYOffset + AxesYSize, i = -AxesYSize / 2; y > InitialYOffset; y -= 50, i += 50)

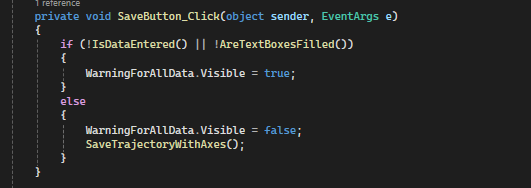
{

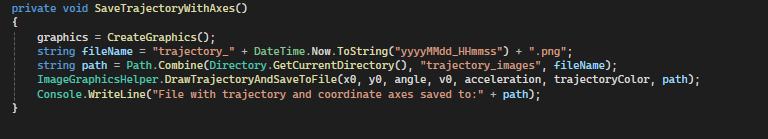
graphics.DrawLine(basicPen, InitialXOffset + AxesXSize / 2 - 7, y, InitialXOffset + AxesXSize / 2 + 7, y);

if (i != 0) graphics.DrawString(i.ToString(), font, brush, InitialXOffset + AxesXSize / 2 + 5, y);

}

}

}  




Використав методи для роботи з рядками, назва файлу визначається як поточний час в точності до секунд з приставкою trajectory\_ і зберігається у поточній директорії у папці trajectory\_images.

**Висновок:** студент отримав уявлення про принципи побудови траєкторії руху тіла, виконав практичне завдання де рух матеріальної точки залежав від вхідних параметрів заданих користувачем. Робота була виконана за допомогою платформи .Net, а саме WinForms на основі мови програмування С# з використанням простору імен System.Drawing.