МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА

Лабораторна робота №4

“Мультипотокове програмування у середовищі Windows Forms”

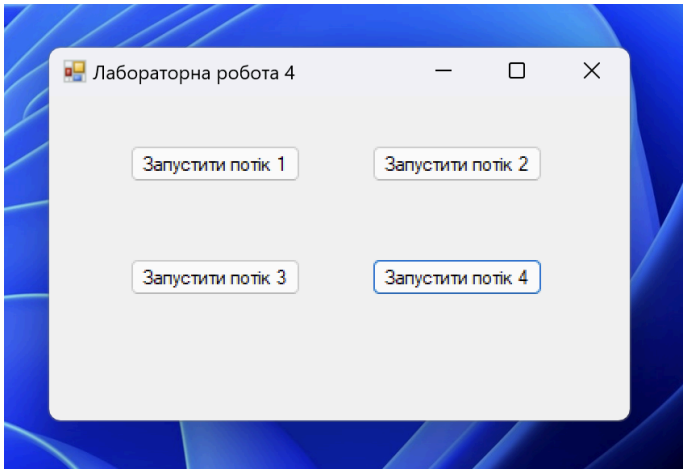
Виконав:
студент групи ПМі-31
Дудинець Олександр

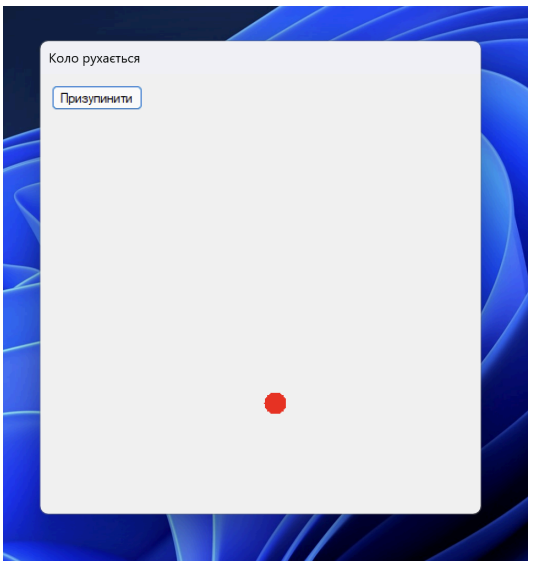
Львів 2024

Мета роботи

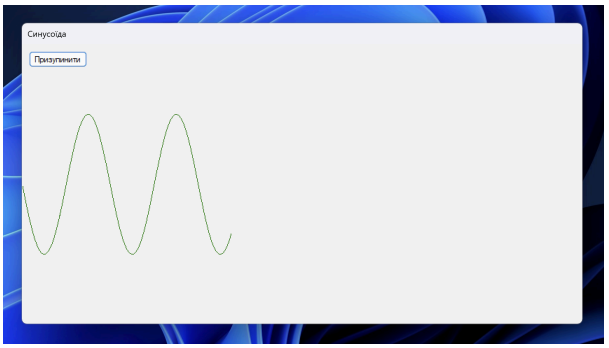
Метою цієї лабораторної роботи було ознайомлення з основами мультипотокового програмування на платформі .NET з використанням Windows Forms. Завдання полягало у створенні програми, яка демонструє роботу декількох потоків, кожен з яких виконує окрему візуальну задачу, використовуючи клас Thread для створення та управління потоками.

Опис ключових частин коду

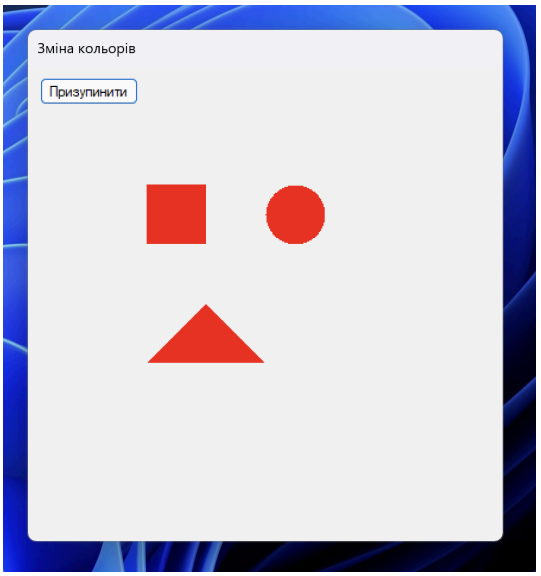
- **Головне вікно (MainForm)**
  - Програма містить головне вікно, в якому розміщені чотири кнопки для запуску та завершення кожного потоку. Кожна кнопка відображає поточний стан відповідного потоку (якщо потік запущений, кнопка відображає "Зупинити потік X", якщо зупинений — "Запустити потік X").
  - **Основні методи:**
    - SetupUI: Ініціалізує інтерфейс головного вікна, створюючи кнопки для управління потоками.
    - BtnThreadX_Click: Метод для кожної кнопки, який запускає новий потік або зупиняє існуючий потік. Потік створюється за допомогою класу Thread та працює в окремому вікні.
    - CloseFormAndStopThread: Метод, який закриває вікно потоку та зупиняє відповідний потік, використовуючи CancellationTokenSource для сигналізації завершення потоку.
  - 

- **Потоки та їхні вікна**
  - Кожен потік виконує візуальну задачу в окремому вікні. Для кожного потоку реалізовані окремі форми (Form1, Form2, Form3, Form4), які виконують наступні завдання:
    - Form1 — анімація кулі, яка рухається по колу.
    - 

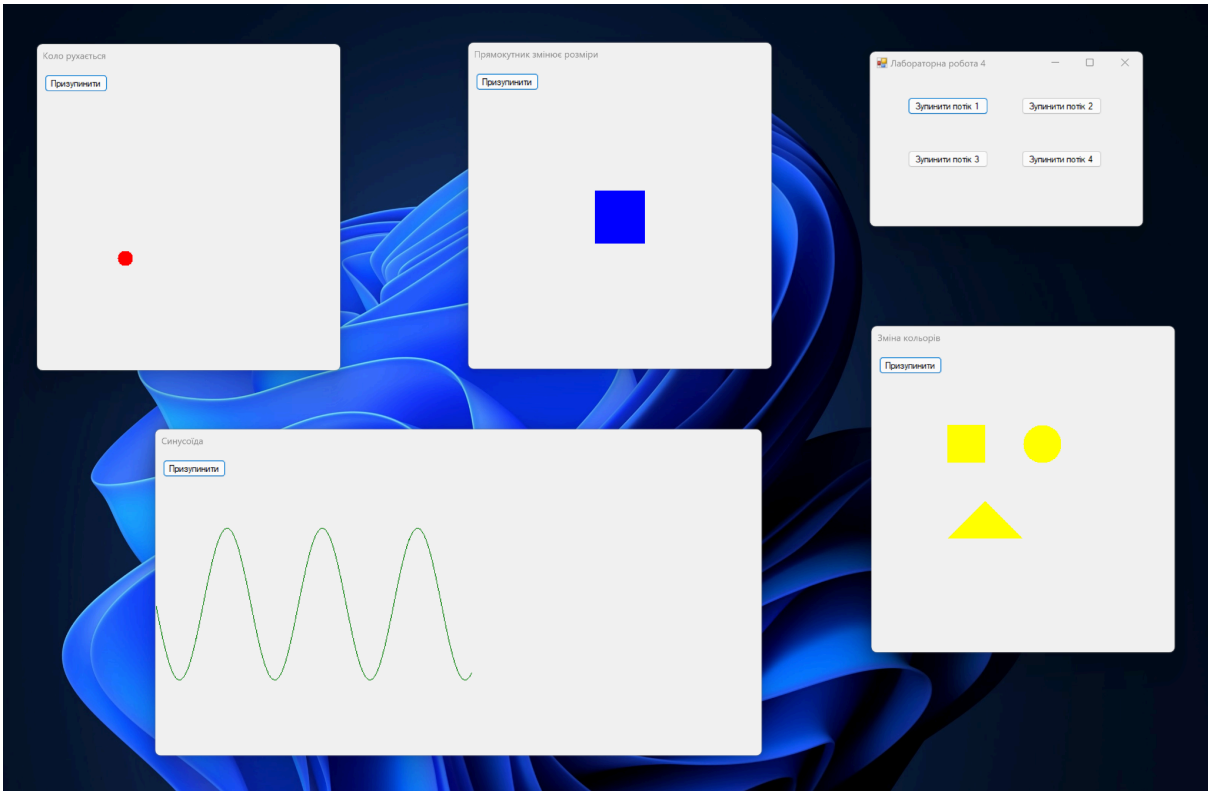
    - Form2 — змінення розмірів прямокутника.
    - 

    - Form3 — малювання графіку синусоїди.
    - 

- Form4 — зміна кольорів декількох фігур.



  - Кожна форма працює у власному потоці, запущеному з використанням класу Thread. Потоки працюють незалежно один від одного, завдяки чому візуальні ефекти відображаються плавно.



Управління потоками

Для управління зупинкою потоків використовується клас CancellationTokenSource. В потоці періодично перевіряється стан токену (token.IsCancellationRequested), і якщо була вимога на зупинку, потік завершиться.

```
private void BtnThread1_Click(object sender, EventArgs e)
{
    Button btn = sender as Button;
    if (!isRunning1)
    {
        cts1 = new CancellationTokenSource();
        thread1 = new Thread(() => Thread1Proc(cts1.Token));
        thread1.SetApartmentState(ApartmentState.STA);
        thread1.Start();
        isRunning1 = true;
        btn.Text = "Зупинити потік 1";
    }
    else
    {
        cts1.Cancel();
        CloseForm(form1);
        isRunning1 = false;
        btn.Text = "Запустити потік 1";
    }
}
```

Кожен потік працює з окремим вікном, яке запускається з використанням Application.Run(). Коли користувач закриває вікно потоку, програма відслідковує подію FormClosed, сигналізуючи про необхідність зупинки відповідного потоку.

Запуск потоків

Потоки запускаються за допомогою кнопок у головному вікні. Якщо потік запущений, його вікно відкривається і виконується відповідне завдання. Користувач може зупинити потік, натиснувши на кнопку "Зупинити потік X", або просто закрити вікно, після чого програма автоматично оновить стан кнопки до "Запустити потік X".

Окрім цього, у кожному вікні є можливість призупинити поточний потік без повної його зупинки.

Проблеми під час розробки та їх вирішення

Спочатку спостерігалася проблема зависання програми під час закриття потоків з головного меню. Це було пов'язано з неправильним використанням методу thread.Join(), який блокував головний потік. Проблему було вирішено шляхом використання CancellationTokenSource для асинхронної зупинки потоків.

Висновок

У результаті виконання цієї лабораторної роботи я навчився працювати з мультипотоковим програмуванням у середовищі Windows Forms з використанням класу Thread. Було створено програму, яка демонструє одночасну роботу кількох потоків, кожен з яких виконує свою задачу в окремому вікні. Програма коректно керує запуском та зупинкою потоків, забезпечуючи стабільну роботу навіть при одночасній роботі 4 потоків.

Варто враховувати, що ефективність мультипотокових програм може знижуватися через накладні витрати на синхронізацію потоків та обмеження апаратних ресурсів. Тому при розробці паралельних алгоритмів важливо балансувати між кількістю потоків та доступними ресурсами системи.