**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

**ІКНІ**

Кафедра **ПЗ**

**ЗВІТ**

До лабораторної роботи №8

**на тему:** *“Робота з мультимедійними даними засобами WPF”*

**з дисципліни:** *“Технології мультимедіа”*

**Лектор:**

доц. кафедри ПЗ

Горечко О.М.

**Виконав:**

студент групи ПЗ-43

Шломʼяк Д.В.

**Прийняв:**

 доц. кафедри ПЗ

Горечко О.М.

Львів – 2025

**Тема роботи:** обота з мультимедійними даними засобами WPF.

**Мета роботи:** отримати практичні навички роботи з мультимедійними даними та анімованими об’єктами з використанням засобів WPF.

# **Теоретичні відомості**

**1. Поняття WPF**

Windows Presentation Foundation (WPF) – це графічна підсистема у складі .NET Framework, призначена для створення клієнтських застосунків з візуально привабливими можливостями взаємодії з користувачем. WPF є високорівневим об'єктно-орієнтованим функціональним шаром, що дає змогу створювати двовимірні та тривимірні інтерфейси.

WPF об'єднала в собі кращі сторони інших технологій:

* З DirectX – тривимірну графіку і апаратне прискорення
* З Windows Forms – продуктивність розробки
* З Adobe Flash – розвинену підтримку анімації
* Зі SMIL та SVG – тегову мову опису анімованої графіки
* З HTML – декларативну розмітку

**2. Основні компоненти WPF**

**XAML (Extensible Application Markup Language)** – мова розмітки для створення екземплярів об'єктів .NET. Головне призначення – конструювання інтерфейсів користувачів WPF. Документи XAML визначають розташування панелей, кнопок та інших елементів керування у вікнах застосунків.

**Canvas** – панель для розташування елементів з абсолютним позиціюванням на основі координат. Використовується для точного розташування графічних об'єктів.

**Ellipse** – графічний елемент класу Shape для відображення еліпсів та кіл. Налаштовується за допомогою властивостей Width, Height, Fill та Stroke.

**DispatcherTimer** – таймер для синхронізації з UI-потоком. Використовується для організації цикличної анімації з контрольованим інтервалом.

**3. Анімація у WPF**

Анімація в WPF – це важлива частина моделі. Основні способи реалізації:

1. **Декларативна анімація (XAML)** – описується у XAML з використанням Storyboard та EventTrigger
2. **Процедурна анімація (C#)** – реалізується через код з використанням таймерів

Необхідний мінімум для анімації:

* **From** – початкове значення
* **To** – кінцеве значення
* **Duration** – час виконання

**4. Обробка подій мишки**

WPF підтримує низку подій мишки:

* **MouseDown** – натиснення кнопки мишки
* **MouseUp** – відпускання кнопки мишки
* **MouseMove** – рух мишки
* **MouseEnter** – входження курсора на елемент
* **MouseLeave** – вихід курсора з елемента

Позиція мишки отримується методом GetPosition(), що повертає координати точки відносно вказаного елемента.

**5. Прозорі вікна у WPF**

Для створення прозорого вікна необхідно:

1. Встановити властивість AllowsTransparency на True
2. Задати властивість WindowStyle на None (приховує рядок заголовку)
3. Встановити Background на Transparent

Такий підхід дозволяє створювати вікна нестандартної форми та взаємодіяти з вмістом без видимої рамки вікна.

**Завдання**

1. Створити програму засобами WPF згідно з варіантом індивідуальних завдань.
2. Допускається використання будь-якого середовища програмування.
3. Продемонструвати викладачеві роботу програми, відповісти на контрольні запитання стосовно побудованого алгоритму та тексту програми.
4. У звіті окрім опису результатів роботи дати розширену відповідь на одне контрольне питання, номер якого відповідає номеру варіанта індивідуального завдання.
5. Результати роботи відобразити у звіті.

Варіант 11: Розробити програму, яка б відображала у прозорому вікні кілька різнокольорових кульок, що неперервно рухаються за колоподібними траєкторіями у площині екрана і реагують на події від мишки (зупиняються, змінюють колір та розмір, продовжують рух).

**Код програми**

Файл MainWindow.xaml

<Window x:Class="WpfApplication.MainWindow"  
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  
 Title="Animated Balls"  
 WindowState="**Maximized**"  
 WindowStyle="**None**"  
 AllowsTransparency="True"  
 Background="Transparent"  
 MouseLeftButtonDown="Window\_MouseLeftButtonDown"  
 Loaded="Window\_Loaded">  
 <Canvas Name="canvas" Background="Transparent">  
 </Canvas>  
</Window>

Файл MainWindow.xaml

using System;  
using System.Collections.Generic;  
using System.Windows;  
using System.Windows.Controls;  
using System.Windows.Input;  
using System.Windows.Media;  
using System.Windows.Shapes;  
using System.Windows.Threading;  
  
namespace WpfApplication  
{  
 public partial class MainWindow : Window  
 {  
 private DispatcherTimer animationTimer;  
 private List<Ball> balls = new List<Ball>();  
 private Ball selectedBall = null;  
  
 public MainWindow()  
 {  
 InitializeComponent();  
 }  
  
 private void Window\_Loaded(object sender, RoutedEventArgs e)  
 {  
 Canvas canvas = this.FindName("canvas") as Canvas;  
  
 *// 6 кульок з різними кольорами* Color[] colors = { Colors.Red, Colors.Blue, Colors.Green, Colors.Yellow, Colors.Magenta, Colors.Cyan };  
 double[] radii = { 50, 60, 70, 55, 65, 58 };  
 double[] speeds = { 0.007, 0.02, 0.012, 0.018, 0.03, 0.025 };  
 double[] orbitRadii = { 300, 100, 250, 320, 280, 120 };  
  
 for (int i = 0; i < 6; i++)  
 {  
 Ball ball = new Ball(25, colors[i], radii[i], speeds[i], orbitRadii[i], i);  
 balls.Add(ball);  
 canvas.Children.Add(ball.Ellipse);  
 }  
  
 *// Налаштування таймера анімації* animationTimer = new DispatcherTimer();  
 animationTimer.Interval = TimeSpan.FromMilliseconds(16);  
 animationTimer.Tick += AnimationTimer\_Tick;  
 animationTimer.Start();  
  
 *// Обробники подій мишки* canvas.MouseDown += Canvas\_MouseDown;  
 canvas.MouseUp += Canvas\_MouseUp;  
 canvas.MouseMove += Canvas\_MouseMove;  
 }  
  
 private void AnimationTimer\_Tick(object sender, EventArgs e)  
 {  
 Canvas canvas = this.FindName("canvas") as Canvas;  
 double centerX = canvas.ActualWidth / 2;  
 double centerY = canvas.ActualHeight / 2;  
  
 foreach (Ball ball in balls)  
 {  
 if (!ball.IsPaused)  
 {  
 *// Рух по колоподібній траєкторії* ball.Angle += ball.Speed;  
 double x = centerX + ball.OrbitRadius \* Math.Cos(ball.Angle);  
 double y = centerY + ball.OrbitRadius \* Math.Sin(ball.Angle);  
  
 Canvas.SetLeft(ball.Ellipse, x - ball.Radius);  
 Canvas.SetTop(ball.Ellipse, y - ball.Radius);  
 }  
 }  
 }  
  
 private void Canvas\_MouseDown(object sender, MouseButtonEventArgs e)  
 {  
 Canvas canvas = this.FindName("canvas") as Canvas;  
 Point p = e.GetPosition(canvas);  
  
 foreach (Ball ball in balls)  
 {  
 double ballX = Canvas.GetLeft(ball.Ellipse) + ball.Radius;  
 double ballY = Canvas.GetTop(ball.Ellipse) + ball.Radius;  
 double distance = Math.Sqrt(Math.Pow(p.X - ballX, 2) + Math.Pow(p.Y - ballY, 2));  
  
 if (distance <= ball.Radius)  
 {  
 selectedBall = ball;  
 ball.IsPaused = !ball.IsPaused;  
  
 *// Зміна розміру при клацанні* if (ball.IsPaused)  
 {  
 ball.Radius \*= 1.3;  
 }  
 else  
 {  
 ball.Radius /= 1.3;  
 }  
  
 *// Зміна кольору при клацанні* Random rand = new Random();  
 ball.SetColor(Color.FromRgb(  
 (byte)rand.Next(100, 256),  
 (byte)rand.Next(100, 256),  
 (byte)rand.Next(100, 256)  
 ));  
  
 ball.Ellipse.Width = ball.Radius \* 2;  
 ball.Ellipse.Height = ball.Radius \* 2;  
  
 e.Handled = true;  
 break;  
 }  
 }  
 }  
  
 private void Canvas\_MouseUp(object sender, MouseButtonEventArgs e)  
 {  
 selectedBall = null;  
 }  
  
 private void Canvas\_MouseMove(object sender, MouseEventArgs e)  
 {  
 if (selectedBall != null)  
 {  
 Canvas canvas = this.FindName("canvas") as Canvas;  
 Point p = e.GetPosition(canvas);  
 Canvas.SetLeft(selectedBall.Ellipse, p.X - selectedBall.Radius);  
 Canvas.SetTop(selectedBall.Ellipse, p.Y - selectedBall.Radius);  
 }  
 }  
  
 private void Window\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)  
 {  
 this.DragMove();  
 }  
 }  
  
 public class Ball  
 {  
 public Ellipse Ellipse { get; set; }  
 public double Radius { get; set; }  
 public double Angle { get; set; }  
 public double Speed { get; set; }  
 public double OrbitRadius { get; set; }  
 public bool IsPaused { get; set; }  
  
 public Ball(double radius, Color color, double orbitRadius, double speed, double initialOrbitRadius, int index)  
 {  
 Radius = radius;  
 Speed = speed;  
 OrbitRadius = initialOrbitRadius;  
 IsPaused = false;  
 Angle = (Math.**PI** \* 2 / 4) \* index;  
  
 Ellipse = new Ellipse  
 {  
 Width = radius \* 2,  
 Height = radius \* 2,  
 Fill = new SolidColorBrush(color),  
 Stroke = new SolidColorBrush(Colors.Black),  
 StrokeThickness = 2  
 };  
 }  
  
 public void SetColor(Color color)  
 {  
 Ellipse.Fill = new SolidColorBrush(color);  
 }  
 }  
}

# **Результат виконання**

**1. Етап розробки**

**Крок 1:** Проектування архітектури

Програма складається з двох основних компонентів:

MainWindow.xaml – XAML-файл з описом інтерфейсу

MainWindow.xaml.cs – C# файл з реалізацією логіки

**Крок 2:** Створення XAML-макету

<Window WindowStyle="None" AllowsTransparency="True"

Background="Transparent" WindowState="Maximized">

<Canvas Name="canvas" Background="Transparent"/>

</Window>

Прозоре вікно максимізованого розміру з Canvas контейнером для розташування кульок.

**Крок 3:** Реалізація класу Ball

Створений клас для представлення кульки з властивостями:

* Ellipse – графічний елемент
* Radius – радіус кульки
* Angle – поточний кут на орбіті
* Speed – швидкість обертання
* OrbitRadius – радіус орбіти
* IsPaused – стан паузи

**Крок 4:** Налаштування анімації

DispatcherTimer налаштований на інтервал 16 мс (≈60 FPS). У кожному тіку таймера виконується розрахунок нової позиції кульки за формулами:

x = centerX + OrbitRadius \* cos(Angle)

y = centerY + OrbitRadius \* sin(Angle)

**Крок 5:** Обробка подій мишки

Реалізовані три обробники подій:

Canvas\_MouseDown – перевірка попадання мишки на кульку; при попаданні: зупинка/продовження руху, зміна розміру (+30%), зміна кольору на випадковий

Canvas\_MouseMove – при утриманні кнопки мишки переміщення паузованої кульки слідом за курсором

Canvas\_MouseUp – звільнення вибраної кульки

**Крок 6:** Налаштування параметрів

Створено 6 кульок з такими параметрами:

Таблиця 1. Параметри кульок

| Кулька | Колір | Радіус орбіти | Швидкість |
| --- | --- | --- | --- |
| 1 | Червона | 300 | 0.015 |
| 2 | Синя | 350 | 0.02 |
| 3 | Зелена | 250 | 0.012 |
| 4 | Жовта | 320 | 0.018 |
| 5 | Фіолетова | 280 | 0.016 |
| 6 | Голуба | 310 | 0.014 |

**2. Функціональність програми**

Основна функція: Кульки обертаються навколо центру екрана по колоподібних траєкторіях з різними радіусами та швидкостями.

Взаємодія з мишкою:

* Клацання на кульці:
  + Зупиняє рух (першого разу)
  + Змінює розмір на 130%
  + Змінює колір на випадковий
  + При повторному клацанні продовжує рух, повертає розмір
* Перетягування паузованої кульки:
* Можна переміщати кульку в межах вікна
* При відпусканні кнопки мишки кулька залишається на місці
* Переміщення вікна:
* Клацання на порожній площі дозволяє переміщувати все вікно

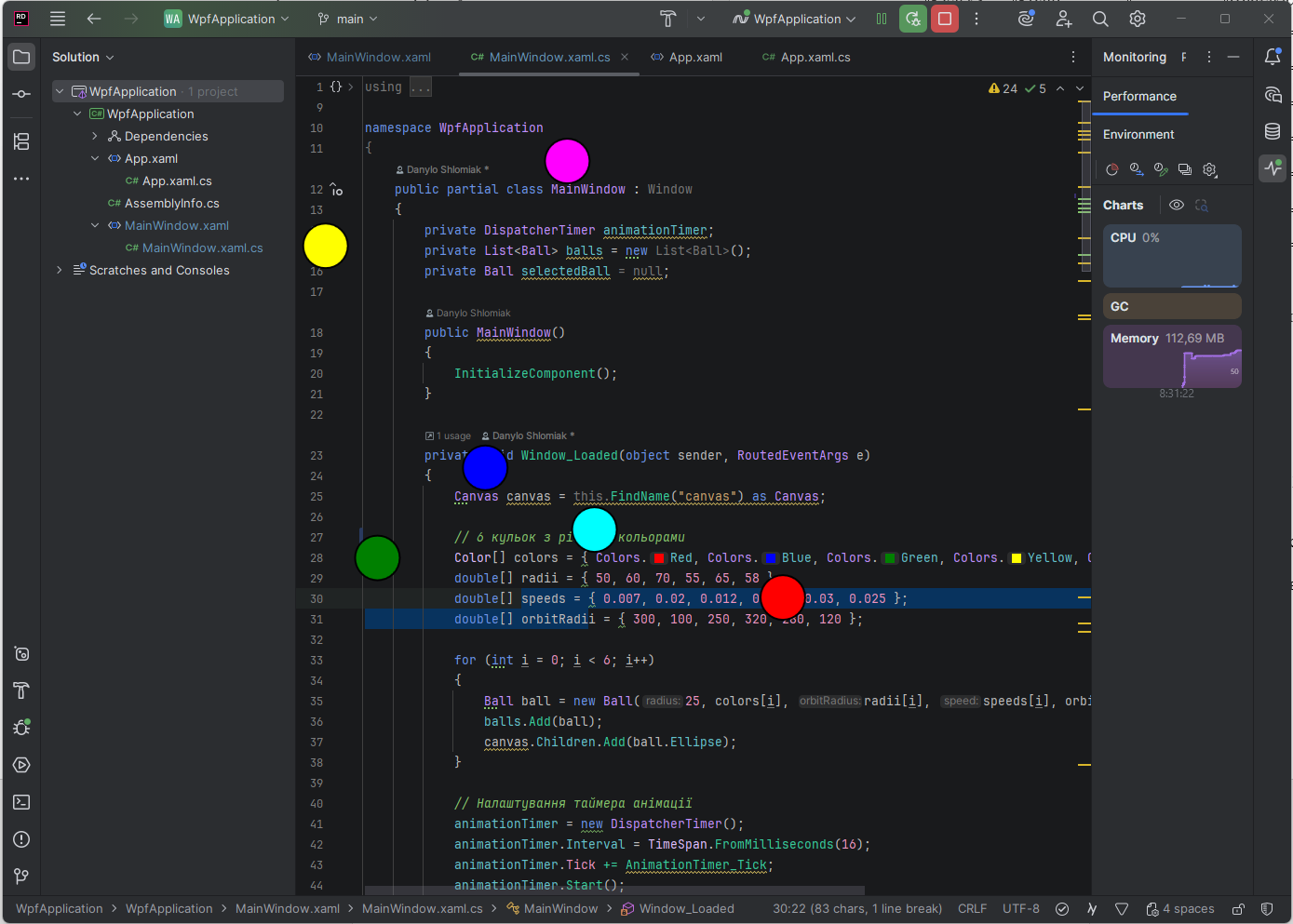


Рис. 1. Результат запущеної рограми

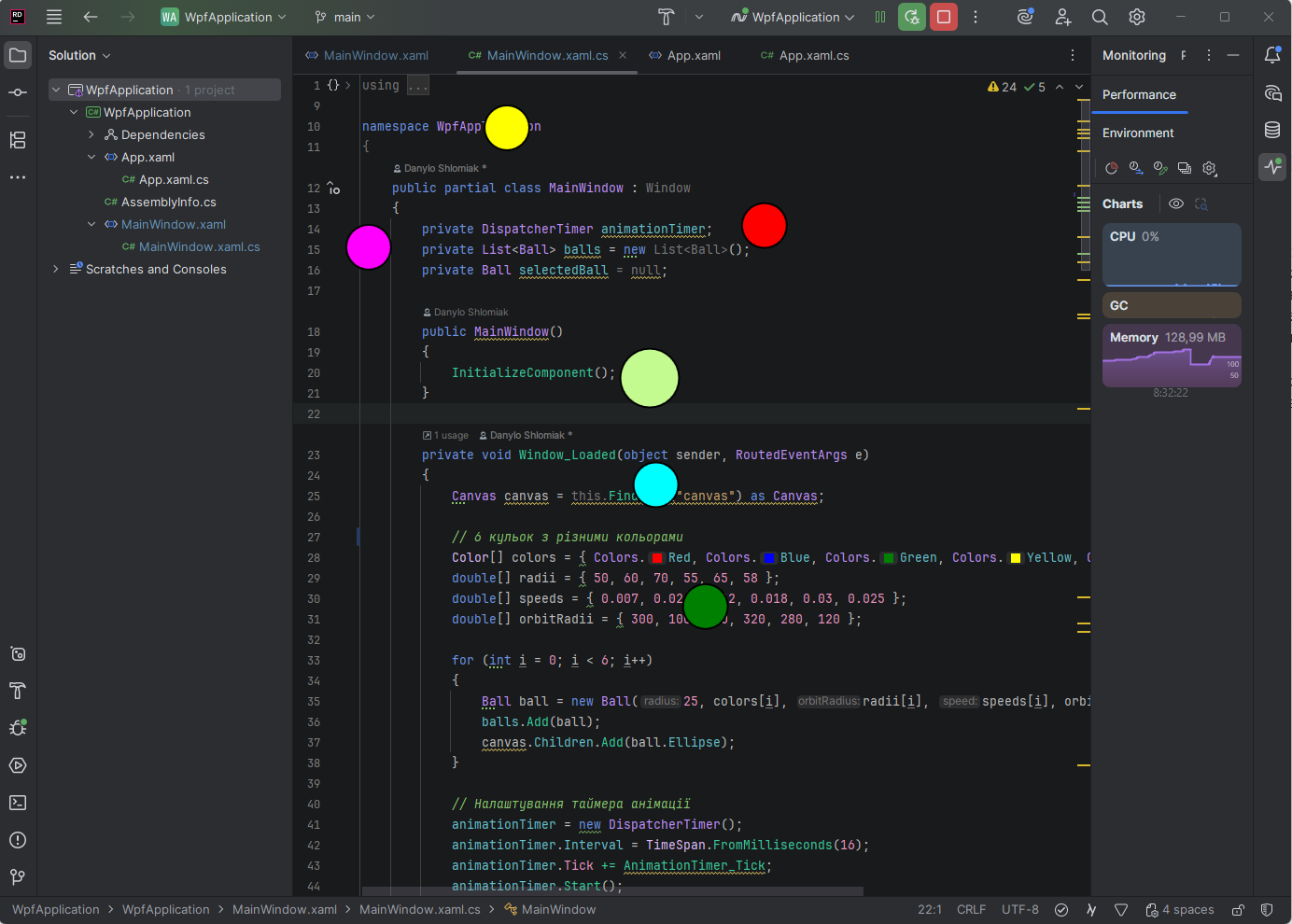


Рис. 2. Результат синьої кульки, яку зупинили і змінили колір та розмір

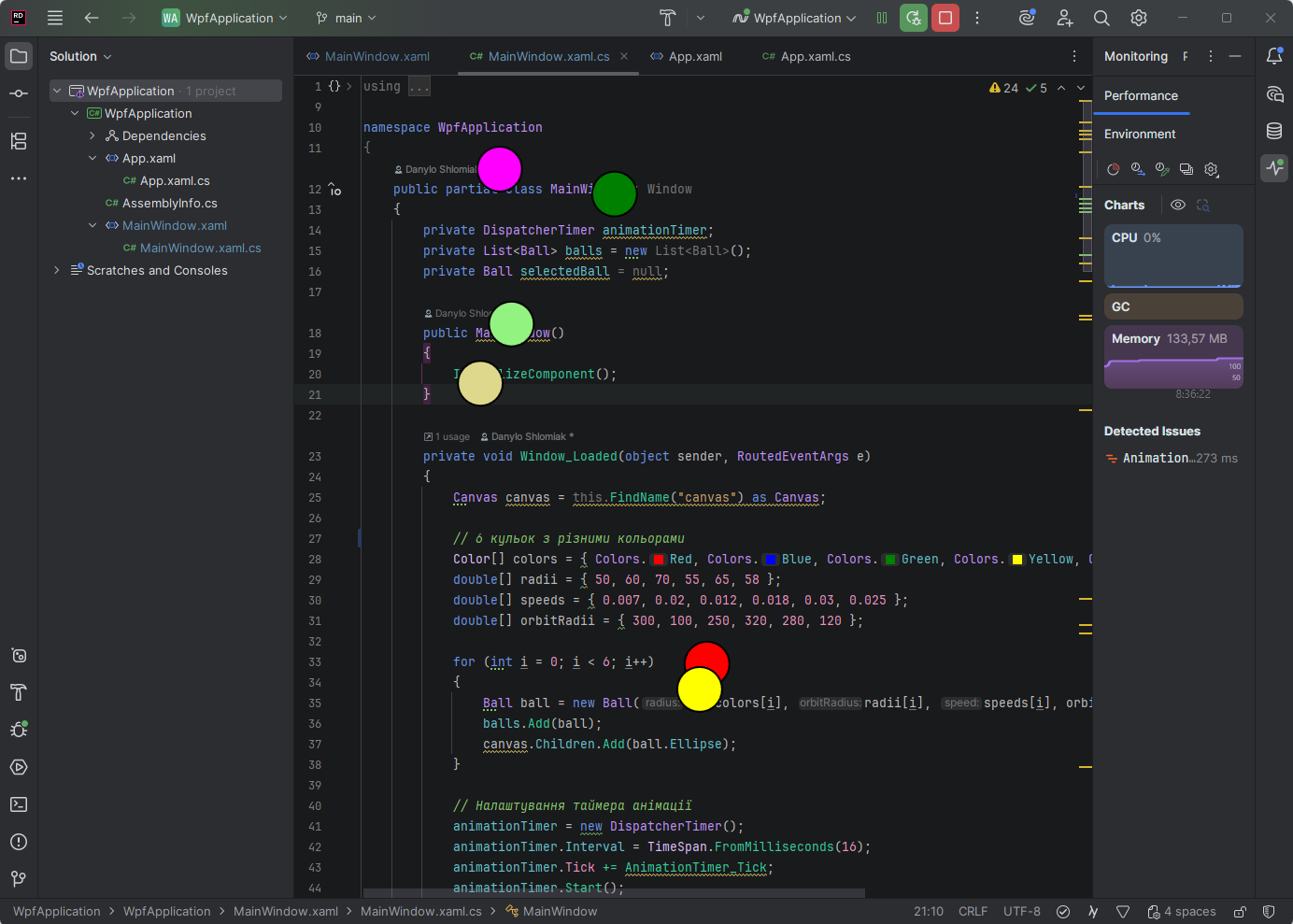


Рис. 3. Результат продовження руху початково синьої кульки

# **Висновки**

У ході виконання лабораторної роботи я здобув практичні навички роботи з професійним відеоредактором Adobe Premiere Pro та засвоїв основні принципи видеомонтажу, редагування аудіо та створення мультимедійного контенту. Комплексний проект, в якому я демонстрував процес роботи в Blender, дозволив мені застосувати теоретичні знання на практиці та розвинути креативний підхід до вирішення технічних завдань.

Під час роботи я опанував ключові техніки видеомонтажу, включаючи синхронізацію декількох відеопотоків, налаштування ефекту Picture-in-Picture та управління часовою шкалою (Timeline). Я навчився ефективно редагувати вихідний матеріал, видаляючи непотрібні фрагменти та прискорюючи нудні сцени без втрати якості пояснень. Особливе значення мала розробка креативних текстових ефектів, у тому числі анімація з використанням масок та ключових кадрів, а також оригінальна реалізація ефекту «друкарської машинки» через послідовне встановлення ключових кадрів.

Робота зі звуком продемонструвала важливість правильного мікшування аудіо-доріжок. Я розумію, як збалансувати основний голос з фоновою музикою та як важливо контролювати гучність кожної доріжки для досягнення професійного звучання. Багатошарове мікшування дозволило мені створити якісний аудіопродукт, який підтримує видеоконтент і не відвертає від нього.

Фінальний експорт проекту у форматі MP4 забезпечив універсальну сумісність матеріалу з різними платформами та пристроями, що є важливим для розповсюдження навчального контенту. Через цю лабораторну роботу я зрозумів, що видеомонтаж — це не просто технічна навичка, а мистецтво поєднання різних елементів в єдину, органічну композицію, яка передає задуману ідею та утримує увагу глядача.