**Виконав: Мельник Б. В.; КН-922в**

**Лабораторна робота №13**

**Тема:** Використання засобів графіки Windows Forms.

**Мета роботи:** набути навички у створенні застосунків із використанням графічних можливостей фреймворку Windows Forms.

**Індивідуальні завдання**

**Варіант №12**

**Завдання 1**

Використовуючи компонент PictureBox та його графічні можливості, намалювати пpaпop України в місці компоненту, завданому власним варіантом (для оріснтацїі треба встановити компонентний контур)

|  |  |
| --- | --- |
| № варіанту | Відносна локація розташування пpaпopy на Picture Box |
| 1 | 2 |
| 12. |  |

**Зауваження:** рисунок мас бути створений за допомогою графічних примітивів, використання зовнішніх зображень чи фото ЗАБОРОНЯЄТЬСЯ

**Текст програми:**

***Task\_1.cs:***

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Task\_1

{

public partial class Task\_1 : Form

{

public Task\_1()

{

InitializeComponent();

}

private void pictureBox1\_Paint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics; // Отримання графічного об'єкта

// Розміри PictureBox

int pictureBoxWidth = pictureBox1.ClientSize.Width;

int pictureBoxHeight = pictureBox1.ClientSize.Height;

// Розмір прапора

int flagWidth = pictureBoxWidth / 4;

int flagHeight = pictureBoxHeight / 4;

// Координати прапора

int x = pictureBoxWidth - flagWidth - 10;

int y = pictureBoxHeight - flagHeight - 10;

// Малювання прапора України

g.FillRectangle(Brushes.Blue, new Rectangle(x, y, flagWidth, flagHeight / 2)); // Синя частина

g.FillRectangle(Brushes.Yellow, new Rectangle(x, y + flagHeight / 2, flagWidth, flagHeight / 2)); // Жовта частина

}

}

}

**Результат роботи програми:**

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

**Алгоритм побудови коду:**

1. **Ініціалізація форми**:
   * Завантаження форми через InitializeComponent().
   * Додавання компонента PictureBox до форми.
2. **Реалізація події малювання**:
   * Обробник події pictureBox1\_Paint:
     + Отримання графічного об'єкта Graphics для малювання.
     + Визначення розмірів компонента PictureBox.
     + Розрахунок координат і розмірів прапора відносно розмірів PictureBox.
     + Малювання синьої частини прапора методом FillRectangle.
     + Малювання жовтої частини прапора нижче синьої.
3. **Розташування прапора**:
   * Визначення прапора у нижньому правому куті PictureBox із відступом у 10 пікселів.

**Функціональність:**

1. **Малювання прапора**:
   * Прапор України складається з двох частин:
     + Верхня половина – синя (колір Brushes.Blue).
     + Нижня половина – жовта (колір Brushes.Yellow).
2. **Динамічне розташування**:
   * Прапор автоматично масштабується та розташовується відповідно до розмірів PictureBox.
3. **Інтерактивність**:
   * Прапор перемальовується автоматично при зміні розмірів вікна або PictureBox.

**Висновок:**

Програма успішно реалізує малювання прапора України без використання зовнішніх зображень. Графічні примітиви забезпечують чіткий і коректний вигляд прапора. Розташування прапора залежить від розмірів PictureBox, що робить програму адаптивною до різних налаштувань інтерфейсу. Реалізація відповідає вимогам завдання та легко розширюється.

**Завдання 2**

Модифікувати минуле завдання так, щоб під час наведения курсору на компонент Picture Box поруч (ліворуч a6o праворуч в залежності від локації i бажання) із прапором України з'являвся пpaпop країни згідно до свого варіанту, а після видалення курсору— пpaпop даної країни зникав би.

|  |  |
| --- | --- |
| № варіанту | Відносна локація розташування пpaпopy на Picture Box |
| 1 | 2 |
| 12. |  |

**Зауваження:** рисунки мають бути створені за допомогою графічних примітивів, використання зовнішніх зображень чи фото ЗАБОРОНЯЄТЬСЯ

**Текст програми:**

***Task\_2.cs:***

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Task\_2

{

public partial class Task\_2 : Form

{

public Task\_2()

{

InitializeComponent();

}

private void pictureBox1\_Paint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics; // Отримання графічного об'єкта

// Малювання прапора України

int flagWidth = pictureBox1.Width;

int flagHeight = pictureBox1.Height;

g.FillRectangle(Brushes.Blue, new Rectangle(0, 0, flagWidth, flagHeight / 2)); // Верхня синя частина

g.FillRectangle(Brushes.Yellow, new Rectangle(0, flagHeight / 2, flagWidth, flagHeight / 2)); // Нижня жовта частина

}

private void pictureBoxJapan\_Paint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics; // Отримання графічного об'єкта

// Малювання прапора Японії

int flagWidth = pictureBoxJapan.Width;

int flagHeight = pictureBoxJapan.Height;

g.FillRectangle(Brushes.White, new Rectangle(0, 0, flagWidth, flagHeight)); // Білий фон

// Червоне коло в центрі

int circleDiameter = Math.Min(flagWidth, flagHeight) / 2;

int x = (flagWidth - circleDiameter) / 2;

int y = (flagHeight - circleDiameter) / 2;

g.FillEllipse(Brushes.Red, x, y, circleDiameter, circleDiameter);

}

private void pictureBox1\_MouseEnter(object sender, EventArgs e)

{

pictureBoxJapan.Visible = true; // Показати прапор Японії

}

private void pictureBox1\_MouseLeave(object sender, EventArgs e)

{

pictureBoxJapan.Visible = false; // Сховати прапор Японії

}

}

}

**Результат роботи програми:**

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, диаграмма, дизайн

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, диаграмма, дизайн

Автоматически созданное описание

**Алгоритм побудови коду:**

1. **Ініціалізація форми**:
   * Завантаження форми через InitializeComponent().
   * Додавання двох компонентів PictureBox:
     + Основний (pictureBox1) для прапора України.
     + Додатковий (pictureBoxJapan) для прапора Японії (початково невидимий).
2. **Малювання прапорів**:
   * У події pictureBox1\_Paint:
     + Малювання прапора України за допомогою графічних примітивів (FillRectangle).
   * У події pictureBoxJapan\_Paint:
     + Малювання прапора Японії з білим фоном та червоним колом у центрі (FillEllipse).
3. **Обробка курсора**:
   * Подія MouseEnter для pictureBox1:
     + Увімкнення видимості pictureBoxJapan.
   * Подія MouseLeave для pictureBox1:
     + Вимкнення видимості pictureBoxJapan.

**Функціональність:**

1. **Малювання прапорів**:
   * Динамічне малювання прапора України у pictureBox1.
   * Малювання прапора Японії у pictureBoxJapan.
2. **Інтерактивність**:
   * При наведенні курсора на pictureBox1 з’являється прапор Японії у pictureBoxJapan.
   * Прапор Японії зникає, коли курсор залишає область pictureBox1.
3. **Дизайн**:
   * Узгоджений стиль із завданням 1.
   * Використання графічних примітивів для створення прапорів.

**Висновок:**

Програма успішно реалізує інтерактивне відображення прапорів за допомогою графічних примітивів. Код організований і легко розширюється. Використання подій MouseEnter та MouseLeave забезпечує динамічність взаємодії. Реалізація відповідає вимогам завдання, є адаптивною до різних розмірів компонентів.

**Завдання 3**

Модифікувати минуле завдання, замінивши подію наведения мишачого курсору на натискання лівої кнопки миші на компоненті, а подію видалення курсору на відпускання кнопки миші. Локацію ліворуч чи праворуч треба замінити на зверху чи знизу (в залежності від локації похідного пpaпopa) від пpaпopa України.

**Текст програми:**

***Task\_3.cs:***

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Task\_3

{

public partial class Task\_3 : Form

{

public Task\_3()

{

InitializeComponent();

}

private void pictureBox1\_Paint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics; // Отримання графічного об'єкта

// Малювання прапора України

int flagWidth = pictureBox1.Width;

int flagHeight = pictureBox1.Height;

g.FillRectangle(Brushes.Blue, new Rectangle(0, 0, flagWidth, flagHeight / 2)); // Верхня синя частина

g.FillRectangle(Brushes.Yellow, new Rectangle(0, flagHeight / 2, flagWidth, flagHeight / 2)); // Нижня жовта частина

}

private void pictureBoxJapan\_Paint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics; // Отримання графічного об'єкта

// Малювання прапора Японії

int flagWidth = pictureBoxJapan.Width;

int flagHeight = pictureBoxJapan.Height;

g.FillRectangle(Brushes.White, new Rectangle(0, 0, flagWidth, flagHeight)); // Білий фон

// Червоне коло в центрі

int circleDiameter = Math.Min(flagWidth, flagHeight) / 2;

int x = (flagWidth - circleDiameter) / 2;

int y = (flagHeight - circleDiameter) / 2;

g.FillEllipse(Brushes.Red, x, y, circleDiameter, circleDiameter); // Червоне коло

}

private void pictureBox1\_MouseDown(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

pictureBoxJapan.Visible = true; // Показати прапор Японії

}

}

private void pictureBox1\_MouseUp(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

pictureBoxJapan.Visible = false; // Сховати прапор Японії

}

}

}

}

**Результат роботи програми:**

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Изображение выглядит как снимок экрана, текст, диаграмма, дизайн

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

**Алгоритм побудови коду:**

1. **Ініціалізація форми**:
   * Додавання двох компонентів PictureBox:
     + pictureBox1 для прапора України.
     + pictureBoxJapan для прапора Японії (початково невидимий).
   * Завантаження форми через InitializeComponent().
2. **Малювання прапорів**:
   * У pictureBox1\_Paint:
     + Малювання прапора України (синя верхня частина та жовта нижня).
   * У pictureBoxJapan\_Paint:
     + Малювання прапора Японії (білий фон і червоне коло в центрі).
3. **Обробка подій миші**:
   * MouseDown для pictureBox1:
     + При натисканні лівої кнопки миші прапор Японії стає видимим (Visible = true).
   * MouseUp для pictureBox1:
     + При відпусканні кнопки прапор Японії зникає (Visible = false).
4. **Розташування прапора**:
   * pictureBoxJapan розташований над або під pictureBox1 залежно від завдання.

**Функціональність:**

1. **Малювання прапорів**:
   * Динамічне створення прапора України та Японії за допомогою графічних примітивів.
2. **Інтерактивність**:
   * Натискання лівої кнопки миші на pictureBox1 показує прапор Японії.
   * Відпускання кнопки ховає прапор.
3. **Розташування**:
   * pictureBoxJapan розташовується вертикально (над або під прапором України).
4. **Дизайн**:
   * Узгоджений стиль із попередніми завданнями.

**Висновок:**

Програма успішно реалізує інтерактивне управління відображенням прапорів. Події MouseDown і MouseUp забезпечують плавний і зрозумілий механізм взаємодії. Використання графічних примітивів дозволяє уникнути зовнішніх зображень, роблячи програму універсальною. Реалізація відповідає вимогам завдання та легко адаптується до нових умов.

**Завдання 4**

Створити застосунок, який комбінував би роботу трьох попередніх завдань, змінюючи локацію пpaпopy іншої країни відносно пpaпopy України згідно з локаціями цих завдань під час спрацьовування таймеру, який треба використати в процесі розробки застосунку. Тобто застосунок мас перемикати вміст **PictureBox** між станами: пpaпop України в однині, праворуч чи ліворуч пpaпopy України з’являеться пpaпop іншої країни, пpaпop іншої країни переміщусться до гори чи до низу. Інтервал спрацьовування таймеру можна вибрати за бажанням.

**Текст програми:**

***Task\_4.cs:***

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Task\_4

{

public partial class Task\_4 : Form

{

private int state = 0; // Позиція прапора Японії

public Task\_4()

{

InitializeComponent();

timer.Start(); // Запуск таймера

}

private void PictureBoxUkraine\_Paint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics; // Отримання графічного об'єкта

// Малювання прапора України

int flagWidth = pictureBoxUkraine.Width;

int flagHeight = pictureBoxUkraine.Height;

g.FillRectangle(Brushes.Blue, new Rectangle(0, 0, flagWidth, flagHeight / 2)); // Синя частина

g.FillRectangle(Brushes.Yellow, new Rectangle(0, flagHeight / 2, flagWidth, flagHeight / 2)); // Жовта частина

}

private void PictureBoxJapan\_Paint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics; // Отримання графічного об'єкта

// Малювання прапора Японії

int flagWidth = pictureBoxJapan.Width;

int flagHeight = pictureBoxJapan.Height;

g.FillRectangle(Brushes.White, new Rectangle(0, 0, flagWidth, flagHeight)); // Білий фон

int circleDiameter = Math.Min(flagWidth, flagHeight) / 2; // Діаметр кола

int x = (flagWidth - circleDiameter) / 2; // Координата X центру

int y = (flagHeight - circleDiameter) / 2; // Координата Y центру

g.FillEllipse(Brushes.Red, x, y, circleDiameter, circleDiameter); // Червоне коло

}

private void Timer\_Tick(object sender, EventArgs e)

{

state = (state + 1) % 2; // Перемикання стану

switch (state)

{

case 0: // Прапор Японії ліворуч від України

pictureBoxJapan.Location = new Point(

pictureBoxUkraine.Left - pictureBoxJapan.Width - 10,

pictureBoxUkraine.Top

);

break;

case 1: // Прапор Японії зверху від України

pictureBoxJapan.Location = new Point(

pictureBoxUkraine.Left,

pictureBoxUkraine.Top - pictureBoxJapan.Height - 10

);

break;

}

pictureBoxJapan.Refresh(); // Оновлення прапора Японії

}

}

}

**Результат роботи програми:**

Изображение выглядит как снимок экрана, текст, диаграмма, дизайн

Автоматически созданное описание

Изображение выглядит как снимок экрана, текст, диаграмма, дизайн

Автоматически созданное описание

**Алгоритм побудови коду:**

1. **Ініціалізація форми**:
   * Додавання двох компонентів PictureBox:
     + pictureBoxUkraine для прапора України.
     + pictureBoxJapan для прапора Японії (початково розташований поза межами форми).
   * Запуск таймера через timer.Start().
2. **Малювання прапорів**:
   * У pictureBoxUkraine\_Paint:
     + Малювання прапора України з використанням графічних примітивів (синя верхня та жовта нижня частини).
   * У pictureBoxJapan\_Paint:
     + Малювання прапора Японії (білий фон і червоне коло в центрі).
3. **Таймер**:
   * Подія Timer\_Tick:
     + Змінна state перемикає стан між двома значеннями (0, 1).
     + У кожному стані прапор Японії змінює своє положення:
       - state = 0: прапор Японії розташовується ліворуч від прапора України.
       - state = 1: прапор Японії розташовується зверху від прапора України.
     + Оновлення прапора Японії через Refresh().
4. **Розташування прапорів**:
   * Координати прапора Японії обчислюються динамічно відносно позиції прапора України з урахуванням відступу.

**Функціональність:**

1. **Динамічне малювання**:
   * Прапори України та Японії створюються за допомогою графічних примітивів.
2. **Циклічне перемикання станів**:
   * Таймер змінює розташування прапора Японії між лівою та верхньою позицією відносно прапора України.
3. **Автоматичне оновлення**:
   * Програма автоматично перемикає стан прапорів із заданим інтервалом.
4. **Дизайн**:
   * Узгоджений із попередніми завданнями, адаптивний до зміни розмірів.

**Висновок:**

Програма успішно комбінує функціональність трьох попередніх завдань. Таймер забезпечує динамічне перемикання положення прапора Японії відносно прапора України. Реалізація з використанням графічних примітивів дозволяє уникнути зовнішніх ресурсів, роблячи програму гнучкою та легкою для розширення. Інтерфейс є інтуїтивним і відповідає вимогам завдання.

**Завдання 5**

Розробити програму, форма якої має фонове зображення електричного умовного позначення елемента за наступним зразком

Изображение выглядит как снимок экрана, текст, Шрифт, линия

Автоматически созданное описание

та власним варіантом

|  |  |  |  |
| --- | --- | --- | --- |
| № варіанту | Елемент | Опис | Колір |
| 1 | 2 | 3 | 4 |
| 12. |  | гучномовець | Сірий |

**Зауваження:** рисунки мають бути створені за допомогою графічних примітивів, використання зовнішніх зображень чи фото ЗАБОРОНЯЄТЬСЯ

**Текст програми:**

***Task\_5.cs:***

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Task\_5

{

public partial class Task\_5 : Form

{

public Task\_5()

{

InitializeComponent();

}

private void Form5\_Paint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics;

// Рамка форми

Pen turquoisePen = new Pen(Color.Turquoise, 5);

g.DrawRectangle(turquoisePen, 10, 10, this.ClientSize.Width - 20, this.ClientSize.Height - 20);

// Білий прямокутник

Brush whiteBrush = new SolidBrush(Color.White);

Rectangle whiteRect = new Rectangle(30, 80, this.ClientSize.Width - 60, this.ClientSize.Height - 130);

g.FillRectangle(whiteBrush, whiteRect);

// Текст "Гучномовець"

string title = "Гучномовець";

Font titleFont = new Font("Arial", 14, FontStyle.Bold);

Brush titleBrush = new SolidBrush(Color.Black);

SizeF titleSize = g.MeasureString(title, titleFont);

float titleX = (this.ClientSize.Width - titleSize.Width) / 2; // Центр тексту

float titleY = whiteRect.Y - titleSize.Height - 30; // Позиція тексту

g.DrawString(title, titleFont, titleBrush, titleX, titleY);

// Гучномовець

Pen grayPen = new Pen(Color.Gray, 5);

// Розміри

int squareWidth = 60; // Квадрат ширина

int squareHeight = 60; // Квадрат висота

int lineLength = 40; // Довжина проводів

int trapezoidOffset = 40; // Трапеція зміщення

// Центрування

int squareX = whiteRect.X + (whiteRect.Width - squareWidth - lineLength - trapezoidOffset) / 2 + lineLength;

int squareY = whiteRect.Y + (whiteRect.Height - squareHeight) / 2;

// Квадрат

g.DrawRectangle(grayPen, squareX, squareY, squareWidth, squareHeight);

// Проводка

g.DrawLine(grayPen, squareX - lineLength, squareY + 15, squareX, squareY + 15); // Верхня

g.DrawLine(grayPen, squareX - lineLength, squareY + squareHeight - 15, squareX, squareY + squareHeight - 15); // Нижня

// Трапеція

Point[] trapezoidPoints = {

new Point(squareX + squareWidth, squareY), // Верхній правий

new Point(squareX + squareWidth + trapezoidOffset, squareY - 30), // Верхній трапеції

new Point(squareX + squareWidth + trapezoidOffset, squareY + squareHeight + 30), // Нижній трапеції

new Point(squareX + squareWidth, squareY + squareHeight) // Нижній правий

};

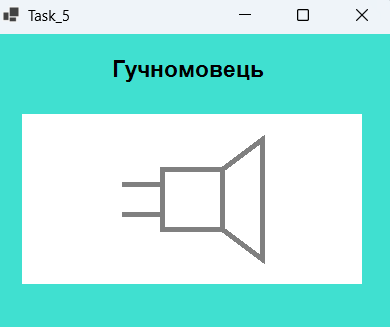
g.DrawPolygon(grayPen, trapezoidPoints);

}

}

}

**Результат роботи програми:**



**Алгоритм побудови коду:**

1. **Ініціалізація форми**:
   * Завантаження форми через InitializeComponent().
   * Обробка події Form\_Paint для малювання фонових елементів.
2. **Малювання рамки форми**:
   * Використання Pen для створення рамки бірюзового кольору навколо області клієнта форми.
3. **Малювання базових елементів**:
   * **Білий прямокутник**:
     + Фон для гучномовця.
     + Центрування в межах форми.
   * **Текст "Гучномовець"**:
     + Розташування тексту над білим прямокутником у центрі форми.
4. **Малювання гучномовця**:
   * **Сірий квадрат**:
     + Центральний елемент, що символізує корпус гучномовця.
   * **Проводка**:
     + Лінії зліва від квадрата.
   * **Трапеція**:
     + Зображує звуковий випромінювач, розташований праворуч від квадрата.
     + Створюється за допомогою масиву точок Point[] і методу DrawPolygon.
5. **Центрування елементів**:
   * Розміщення квадрата, проводки та трапеції по центру білого прямокутника.

**Функціональність:**

1. **Малювання елементів**:
   * Зображення гучномовця, що складається з квадрата, проводки та трапеції.
   * Використання виключно графічних примітивів для створення схеми.
2. **Центрування та масштабування**:
   * Елементи динамічно центруються відносно розмірів форми.
3. **Дизайн**:
   * Рамка бірюзового кольору.
   * Текст "Гучномовець" розташований над елементом.
   * Сірий колір використовується для основного елемента.

**Висновок:**

Програма успішно реалізує умовне позначення гучномовця за допомогою графічних примітивів. Код структурований та адаптований для роботи з різними розмірами форми. Використання простих геометричних фігур робить програму гнучкою та універсальною. Реалізація відповідає вимогам завдання й забезпечує естетичний вигляд.

**Завдання 6**

Розробити програму, форма якої містить графік коливання курсу національної валюти за наступним зразком

Изображение выглядит как текст, снимок экрана, линия, Шрифт

Автоматически созданное описание

Виміри курсу повинні мати випадковий характер. Кількість валютних вимірів (обумовлює крок графіку) за бажанням, але не менш ніж l0. Стиль та колір лінії графіку визначаеться власним варіантом

|  |  |  |
| --- | --- | --- |
| № варіанту | Стиль лінії графіку | Колір лінії графіку |
| 1 | 2 | 3 |
| 12. | Dash | Сірий |

Додати до форми кнопку із написом оновити, яка перебудовує графік за випадковими даними.

**Текст програми:**

***Task\_6.cs:***

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Task\_6

{

public partial class Task\_6 : Form

{

private Random random = new Random(); // Генератор випадкових чисел

private double[] currencyValues; // Массив валютних значень

public Task\_6()

{

InitializeComponent();

GenerateRandomValues();

}

private void GenerateRandomValues()

{

// Генерація 15-20 випадкових значень [30.00, 32.00]

int count = random.Next(15, 21);

currencyValues = new double[count];

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

currencyValues[i] = Math.Round(30.00 + random.NextDouble() \* 2.00, 2);

}

private void PanelChart\_Paint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics; // Графічний об'єкт

g.Clear(Color.White); // Очищення

// Знаходимо min/max значення

double max = Math.Ceiling(currencyValues[0]);

double min = Math.Floor(currencyValues[0]);

foreach (var value in currencyValues)

{

if (value > max) max = Math.Ceiling(value);

if (value < min) min = Math.Floor(value);

}

// Висота і ширина графіка

int chartHeight = panelChart.Height - 100;

int chartWidth = panelChart.Width - 100;

// Малюємо осі

g.DrawLine(Pens.Black, 50, 50, 50, panelChart.Height - 50); // Вертикальна

g.DrawLine(Pens.Black, 50, panelChart.Height - 50, panelChart.Width - 50, panelChart.Height - 50); // Горизонтальна

// Розмітка осі Y

int stepY = chartHeight / 10;

for (int i = 0; i <= 10; i++)

{

double value = min + (max - min) \* i / 10.0;

int y = (int)(panelChart.Height - 50 - i \* stepY);

g.DrawLine(Pens.Gray, 45, y, 50, y);

g.DrawString(value.ToString("F2"), new Font("Arial", 8), Brushes.Black, 5, y - 8);

}

// Обчислення точок графіка

Point[] points = new Point[currencyValues.Length];

int stepX = chartWidth / (currencyValues.Length - 1);

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

{

int x = 50 + i \* stepX;

int y = (int)(panelChart.Height - 50 - (currencyValues[i] - min) / (max - min) \* chartHeight);

points[i] = new Point(x, y);

g.DrawString(currencyValues[i].ToString("F2"), new Font("Arial", 8), Brushes.Black, x - 15, y - 20);

}

// Лінія графіка

Pen pen = new Pen(Color.Gray, 2) { DashStyle = System.Drawing.Drawing2D.DashStyle.Dash };

g.DrawLines(pen, points);

// Точки графіка

foreach (var point in points)

g.FillRectangle(Brushes.Black, point.X - 2, point.Y - 2, 4, 4);

}

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

{

// Оновлення даних і графіка

GenerateRandomValues();

panelChart.Invalidate();

}

}

}

**Результат роботи програми:**

Изображение выглядит как текст, диаграмма, линия, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, линия, диаграмма, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, диаграмма, линия, снимок экрана

Автоматически созданное описание

**Алгоритм побудови коду:**

1. **Ініціалізація форми**:
   * Завантаження компонентів форми через InitializeComponent().
   * Додавання панелі для графіка (panelChart) і кнопки "Оновити" (btnUpdate).
2. **Генерація даних**:
   * Метод GenerateRandomValues() генерує 15–20 випадкових значень у діапазоні [30.00, 32.00], округлених до 2 знаків після коми.
3. **Побудова графіка**:
   * У події PanelChart\_Paint:
     + Очищення панелі та малювання осей (X та Y).
     + Визначення мінімального та максимального значення для масштабування графіка.
     + Розрахунок координат точок для кожного значення курсу.
     + Малювання:
       - Лінії графіка зі стилем Dash.
       - Точок графіка у вигляді квадратів.
       - Розмітки осей із підписами.
4. **Оновлення графіка**:
   * Подія BtnUpdate\_Click:
     + Генерація нових даних методом GenerateRandomValues().
     + Перемальовування графіка через panelChart.Invalidate().

**Функціональність:**

1. **Графік курсу валюти**:
   * Будує графік коливання курсу на основі випадкових значень.
   * Підписи осей та точок відображають точні значення.
2. **Оновлення даних**:
   * Кнопка "Оновити" дозволяє перебудувати графік з новими даними.
3. **Дизайн графіка**:
   * Лінія графіка сірого кольору зі стилем Dash.
   * Точки графіка позначені чорними квадратами.
   * Осі мають розмітку та підписи.

**Висновок:**

Програма успішно реалізує відображення графіка коливання курсу валюти з використанням випадкових даних. Стиль Dash і колір Gray забезпечують відповідність вимогам завдання. Кнопка "Оновити" додає інтерактивність. Код структурований, що полегшує його модифікацію та розширення. Реалізація відповідає вимогам і забезпечує приємний користувацький досвід.

**Завдання 7**

Розробити програму, форма якої містить графік завдання 9 лабораторної роботи № 1 з курсу «Основи мови С#».

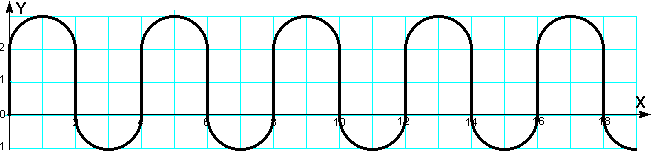
**Завдання 9 лабораторної роботи №1 з курсу «Основи мови С#».**

Використовуючи завдання власного варіанта до лабораторної роботи 7 з електронних методичних вказівок з основ програмування та алгоритмічних мов створити проект з реалізації цих завдань засобами мови C#.

***Лабораторна робота №7***

Для функції ***y = f***(***x***), графік якої наведений у виданому індивідуальному завданні, вивести на екран значення ***y*** для ***x =*** 0, 0.25, 0.5 ... 19.75. Необов’язкова для виконання додаткова вимога – отримати на екрані графік функції з використанням засобів текстового режиму.

Варіант № 12:



**Текст програми:**

***Task\_7.cs:***

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Task\_7

{

public partial class Task\_7 : Form

{

public Task\_7()

{

InitializeComponent();

GenerateTextGraph(); // Генерація текстового графіка

}

private void GraphPanel\_Paint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics; // Графічний об'єкт

g.Clear(Color.White); // Очищення

// Параметри осей

double xMin = 0, xMax = 20;

double yMin = -2, yMax = 2;

double scaleX = (graphPanel.Width - 150) / (xMax - xMin); // Дод. простір для осі X

double scaleY = (graphPanel.Height - 60) / (yMax - yMin); // Враховуємо відступи

int xOffset = 50; // Відступ зліва

int yOffset = (int)(yMax \* scaleY + 30); // Зміщення для осі Y

// Малюємо осі

Pen axisPen = new Pen(Color.Black, 2);

g.DrawLine(axisPen, xOffset, yOffset, graphPanel.Width - 70, yOffset); // Подовжена вісь X

g.DrawLine(axisPen, xOffset, 10, xOffset, graphPanel.Height - 25); // Вісь Y (з відступом зверху)

// Підписи для осей

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

Brush textBrush = Brushes.Black;

// Підпис осі X

g.DrawString("X", axisFont, textBrush, graphPanel.Width - 80, yOffset + 10);

// Підпис осі Y

g.DrawString("Y", axisFont, textBrush, xOffset - 25, 10);

// Малюємо мітки на осі X

for (double x = xMin; x <= xMax; x += 2)

{

int screenX = xOffset + (int)((x - xMin) \* scaleX);

g.DrawLine(Pens.Black, screenX, yOffset - 5, screenX, yOffset + 5); // Маленька риска

g.DrawString(x.ToString("0"), axisFont, textBrush, screenX - 10, yOffset + 10); // Число

}

// Малюємо мітки на осі Y

for (double y = yMin; y <= yMax; y += 1)

{

int screenY = (int)(yOffset - y \* scaleY);

g.DrawLine(Pens.Black, xOffset - 5, screenY, xOffset + 5, screenY); // Маленька риска

g.DrawString(y.ToString("0"), axisFont, textBrush, xOffset - 30, screenY - 8); // Число

}

// Малюємо графік

Pen graphPen = new Pen(Color.Blue, 2);

for (double x = xMin; x < xMax; x += 0.01)

{

double y1 = CalculateY(x);

double y2 = CalculateY(x + 0.01);

int screenX1 = xOffset + (int)((x - xMin) \* scaleX);

int screenY1 = (int)(yOffset - y1 \* scaleY);

int screenX2 = xOffset + (int)(((x + 0.01) - xMin) \* scaleX);

int screenY2 = (int)(yOffset - y2 \* scaleY);

g.DrawLine(graphPen, screenX1, screenY1, screenX2, screenY2);

}

}

private void GenerateTextGraph()

{

textBoxOutput.Clear();

textBoxOutput.AppendText("| x | y |\n");

textBoxOutput.AppendText("|-------------|------------------|\n");

for (double x = 0; x <= 20; x += 0.25)

{

double y = CalculateY(x);

textBoxOutput.AppendText($"| {x,7:0.00} | {y,15:0.0000} |\n");

}

}

private double CalculateY(double x)

{

// Функція для обчислення Y

if (x % 4 < 1)

return 2 \* (x % 4) - 1; // Перший сегмент

else if (x % 4 < 3)

return 1 - Math.Sqrt(1 - Math.Pow((x % 4) - 2, 2)); // Другий сегмент

else

return 7 - 2 \* (x % 4); // Третій сегмент

}

}

}

**Результат роботи програми:**

Изображение выглядит как текст, снимок экрана, линия, График

Автоматически созданное описание

**Алгоритм побудови коду:**

1. **Ініціалізація форми**:
   * Додавання панелі для графіка (graphPanel) та текстового поля (textBoxOutput) для таблиці значень.
   * Виклик GenerateTextGraph() для обчислення таблиці значень.
2. **Обчислення значень**:
   * Метод CalculateY(double x) визначає значення y для кожного сегмента функції.
3. **Побудова текстового графіка**:
   * У методі GenerateTextGraph():
     + Обчислення значень y для x із кроком 0.25.
     + Виведення значень у таблицю у форматі | x | y |.
4. **Малювання графіка**:
   * У події GraphPanel\_Paint:
     + Малювання осей X та Y із мітками.
     + Розрахунок координат точок графіка на основі масштабів осей.
     + З'єднання точок лініями для побудови графіка.
5. **Відображення графіка**:
   * Кожен сегмент графіка будується з використанням ліній між точками, розрахованими для невеликого кроку Δx = 0.01.

**Функціональність:**

1. **Графік функції**:
   * Побудова графіка y = f(x) у панелі graphPanel.
   * Автоматичне масштабування осей для відображення функції.
2. **Таблиця значень**:
   * Виведення значень x та відповідних y у текстовому полі textBoxOutput.
3. **Дизайн**:
   * Чіткі осі X та Y із мітками.
   * Графік із синьою лінією.

**Висновок:**

Програма успішно реалізує побудову графіка функції та виведення таблиці значень. Код забезпечує адаптивність до змін параметрів форми та функції. Використання графічних примітивів для малювання осей і графіка робить програму гнучкою та ефективною. Реалізація відповідає вимогам завдання, забезпечуючи точність і наочність.

**Завдання 8**

Розробити програму, форма якої містить кругову діаграму, використовуючи наступний зразок

Изображение выглядит как текст, снимок экрана, Операционная система, программное обеспечение

Автоматически созданное описание

згідно свого варіанту

|  |  |
| --- | --- |
| № варіанту | Завдання на побудову діаграми |
| 1 | 2 |
| 12. | Горні піки України: Говерлу — 2061 м (найвища точка України),  Бребенескул — 2032 м, Піп Іван — 2022 м, Петрос — 2020 м, Гутин- Томнатик — 2016 м i Ребра — 2001 м. |

**Текст програми:**

***Task\_8.cs:***

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Task\_8

{

public partial class Task\_8 : Form

{

public Task\_8()

{

InitializeComponent();

}

private void Form8\_Paint(object sender, PaintEventArgs e)

{

// Дані для діаграми

string[] peaks = { "Говерла", "Бребенескул", "Піп Іван", "Петрос", "Гутин-Томнатик", "Ребра" };

float[] heights = { 2061, 2032, 2022, 2020, 2016, 2001 };

float totalHeight = 0;

foreach (var height in heights)

totalHeight += height; // Загальна висота

float startAngle = 0; // Початковий кут

Color[] colors = { Color.Green, Color.Yellow, Color.Blue, Color.Red, Color.Purple, Color.Brown };

// Малюємо сектори

Graphics g = e.Graphics;

Rectangle chartArea = new Rectangle(150, 100, 300, 300);

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

{

float sweepAngle = heights[i] / totalHeight \* 360; // Кут сектора

using (Brush brush = new SolidBrush(colors[i]))

{

g.FillPie(brush, chartArea, startAngle, sweepAngle);

}

startAngle += sweepAngle;

}

// Легенда

int legendX = 500, legendY = 120;

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

{

float percentage = (heights[i] / totalHeight) \* 100; // Відсоток

using (Brush brush = new SolidBrush(colors[i]))

{

g.FillRectangle(brush, legendX, legendY + i \* 30, 30, 20); // Кольоровий прямокутник

}

g.DrawRectangle(Pens.Black, legendX, legendY + i \* 30, 30, 20);

g.DrawString($"{peaks[i]} - {percentage:F2}% ({heights[i]} м)", new Font("Arial", 10), Brushes.Black, legendX + 40, legendY + i \* 30);

}

// Заголовок

g.DrawString("Розподіл висот Горних вершин України", new Font("Arial", 14, FontStyle.Bold), Brushes.Black, new Point(180, 50));

}

}

}

**Результат роботи програми:**

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

**Алгоритм побудови коду:**

1. **Ініціалізація форми**:
   * Використання події Form\_Paint для малювання кругової діаграми.
   * Визначення даних для вершин (назви та висоти).
2. **Розрахунок діаграми**:
   * Визначення загальної суми висот вершин.
   * Розрахунок кута кожного сектора (пропорційно до висоти вершини).
3. **Малювання кругової діаграми**:
   * Початковий кут сектора встановлюється на 0.
   * Для кожної вершини:
     + Визначається кут сектора на основі пропорції.
     + Сектор малюється методом FillPie.
4. **Додавання легенди**:
   * Прямокутники кольорів для кожної вершини.
   * Підписи з назвою вершини, її висотою та відсотковим значенням.
5. **Заголовок діаграми**:
   * Додається текст у верхній частині форми для пояснення змісту діаграми.

**Функціональність:**

1. **Графічне представлення даних**:
   * Кругова діаграма показує пропорційний розподіл висот основних вершин України.
2. **Легенда**:
   * Легенда поруч із діаграмою містить кольорові прямокутники, назви вершин, висоти та їх відсотковий внесок.
3. **Дизайн**:
   * Використовуються яскраві кольори для наочності.
   * Заголовок чітко пояснює зміст діаграми.

**Висновок:**

Програма успішно реалізує відображення кругової діаграми розподілу висот гірських вершин України. Використання графічних примітивів забезпечує точність та естетичний вигляд. Легенда покращує інформативність, а структура коду дозволяє легко змінювати або доповнювати дані. Реалізація відповідає завданню та є зручною для користувача.

**Завдання 9**

Використовуючи засоби графіки та компонент timer створити динамічний застосунок, який реалізуе pyx двовимірних кульок різного діаметру та кольору у прямокутній області. Під час ружу мас бути реалізована імітація стикання кульок між собою та із сторонами прямокутної області. Форма реалізації застосунку може бути створена, наприклад, за таким зразком

Изображение выглядит как снимок экрана, Операционная система, мультимедиа, программное обеспечение

Автоматически созданное описание

Параметри кульок визначаються власним варіантом

|  |  |  |  |
| --- | --- | --- | --- |
| № варіанту | Кількість кульок | Особливості діаметрів кульок | Кольорові параметри |
| 1 | 2 | 3 | 4 |
| 12. | 5 | Діаметри однакові | Усі кольори різні |

**Текст програми:**

**Task\_9.cs:**

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Task\_9

{

public partial class Task\_9 : Form

{

private System.Windows.Forms.Timer timer;

private Ball[] balls;

private Random random = new Random();

private const int BallCount = 5; // Кількість кульок

public Task\_9()

{

InitializeComponent();

InitializeBalls();

InitializeTimer();

}

private void InitializeBalls()

{

balls = new Ball[BallCount]; // Масив кульок

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

{

int diameter = 40; // Однаковий діаметр кульок

balls[i] = new Ball(

x: random.Next(50, ClientSize.Width - 50),

y: random.Next(50, ClientSize.Height - 50),

diameter: diameter,

color: Color.FromArgb(random.Next(256), random.Next(256), random.Next(256)),

dx: random.Next(-10, 11),

dy: random.Next(-10, 11)

);

}

}

private void InitializeTimer()

{

timer = new System.Windows.Forms.Timer

{

Interval = 20 // Інтервал таймера

};

timer.Tick += UpdateBalls; // Обробка тіку таймера

timer.Start(); // Запуск таймера

}

private void UpdateBalls(object sender, EventArgs e)

{

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

{

balls[i].Move(ClientSize); // Рух кульок

for (int j = i + 1; j < balls.Length; j++) // Перевірка зіткнень

{

balls[i].CheckCollision(balls[j]);

}

}

Invalidate(); // Оновлення форми

}

protected override void OnPaint(PaintEventArgs e)

{

base.OnPaint(e); // Малювання базового стану

foreach (var ball in balls)

{

ball.Draw(e.Graphics); // Малювання кульки

}

}

}

public class Ball

{

public double X { get; set; } // Координата X

public double Y { get; set; } // Координата Y

public int Diameter { get; } // Діаметр кульки

public Color Color { get; } // Колір кульки

public double Dx { get; set; } // Швидкість X

public double Dy { get; set; } // Швидкість Y

private static Random random = new Random();

public Ball(double x, double y, int diameter, Color color, double dx, double dy)

{

X = x; // Початковий X

Y = y; // Початковий Y

Diameter = diameter; // Діаметр

Color = color; // Колір

Dx = dx; // Початковий Dx

Dy = dy; // Початковий Dy

}

public void Move(Size clientSize)

{

X += Dx; // Переміщення X

Y += Dy; // Переміщення Y

// Відбивання від стінок

if (X <= 0)

{

Dx = Math.Abs(Dx); // Відбивання від лівої

X = 0;

}

else if (X + Diameter >= clientSize.Width)

{

Dx = -Math.Abs(Dx); // Відбивання від правої

X = clientSize.Width - Diameter;

}

if (Y <= 0)

{

Dy = Math.Abs(Dy); // Відбивання від верху

Y = 0;

}

else if (Y + Diameter >= clientSize.Height)

{

Dy = -Math.Abs(Dy); // Відбивання від низу

Y = clientSize.Height - Diameter;

}

}

public void CheckCollision(Ball other)

{

double dx = (X + Diameter / 2.0) - (other.X + other.Diameter / 2.0);

double dy = (Y + Diameter / 2.0) - (other.Y + other.Diameter / 2.0);

double distance = Math.Sqrt(dx \* dx + dy \* dy);

if (distance < Diameter)

{

// Уникнення залипання

double overlap = Diameter - distance;

double correctionX = (overlap \* dx) / distance / 2.0;

double correctionY = (overlap \* dy) / distance / 2.0;

X += correctionX; // Корекція X

Y += correctionY; // Корекція Y

other.X -= correctionX;

other.Y -= correctionY;

// Зміна напрямку

double nx = dx / distance;

double ny = dy / distance;

// Швидк. в напрямку нормалі

double p = 2 \* (Dx \* nx + Dy \* ny - other.Dx \* nx - other.Dy \* ny) / 2; // Масова рівновага припускає однакову масу

Dx = Dx - p \* nx; // Швидкість після удару

Dy = Dy - p \* ny;

other.Dx = other.Dx + p \* nx;

other.Dy = other.Dy + p \* ny;

// Додавання випадковості

Dx += (random.NextDouble() - 0.5) \* 0.1;

Dy += (random.NextDouble() - 0.5) \* 0.1;

other.Dx += (random.NextDouble() - 0.5) \* 0.1;

other.Dy += (random.NextDouble() - 0.5) \* 0.1;

}

}

public void Draw(Graphics g)

{

using (Brush brush = new SolidBrush(Color))

{

g.FillEllipse(brush, (float)X, (float)Y, Diameter, Diameter);

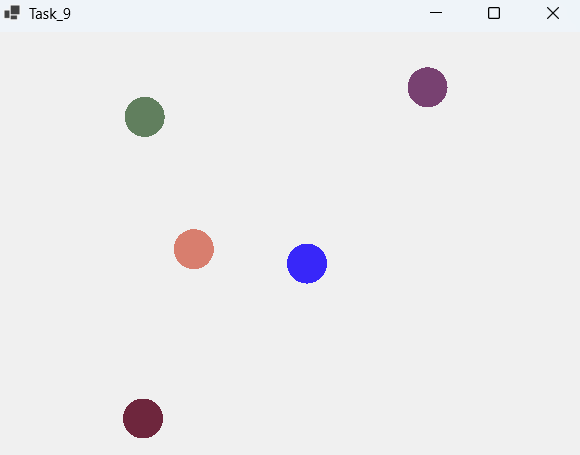
}

}

}

}

**Результат роботи програми:**



Изображение выглядит как снимок экрана, Красочность

Автоматически созданное описание

Изображение выглядит как снимок экрана, Красочность

Автоматически созданное описание

**Алгоритм побудови коду:**

1. **Ініціалізація форми**:
   * Завантаження компонентів через InitializeComponent.
   * Додавання таймера з інтервалом 20 мс.
2. **Ініціалізація кульок**:
   * Масив із п’яти кульок однакового діаметра.
   * Кожна кулька має:
     + Випадковий початковий колір.
     + Випадкові координати в межах області.
     + Випадкові початкові швидкості по осях XXX і YYY.
3. **Рух кульок**:
   * У методі UpdateBalls:
     + Кульки переміщуються відповідно до швидкості.
     + Відбивання від стінок реалізується зміною напрямку швидкості при досягненні меж області.
4. **Стикання кульок**:
   * Перевірка відстані між центрами двох кульок.
   * Якщо кульки стикаються:
     + Корекція позицій для уникнення залипання.
     + Обчислення нових швидкостей із урахуванням еластичності та випадковості.
5. **Малювання кульок**:
   * У методі OnPaint кожна кулька малюється як заповнена еліпса (FillEllipse) із використанням її кольору.
6. **Оновлення форми**:
   * Виклик Invalidate() у події Tick таймера для перерисовки форми.

**Функціональність:**

1. **Динамічний рух**:
   * Кульки постійно переміщуються у межах прямокутної області.
   * Реалізовано відбивання від стінок.
2. **Імітація стикання**:
   * При зіткненні кульок змінюється їхній напрямок і швидкість.
   * Додано випадкові зміни для природнішої поведінки.
3. **Візуалізація**:
   * Кульки мають однаковий діаметр, але різні кольори.
   * Рух кульок плавний завдяки високій частоті оновлення.

**Висновок:**

Програма успішно реалізує анімацію руху кульок із використанням засобів графіки та таймера. Код дозволяє легко змінювати параметри кульок (кількість, діаметр, швидкість). Імітація фізики стикання та відбивання створює реалістичну поведінку об’єктів. Реалізація відповідає вимогам завдання, забезпечуючи ефективність і візуальну привабливість.

**Завдання 10**

Модифікувати застосунок минулого завдання таким чином, щоб створити можливість налаштувань в окремій формі такі параметри кульок:

* кількість;
* кольори;
* діаметри;
* швидкість

**Текст програми:**

***Task\_10.cs:***

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

namespace Task\_10

{

public partial class Task\_10 : Form

{

private System.Windows.Forms.Timer timer;

private Ball[] balls;

private Random random = new Random();

// Поля налаштувань

private int ballCount = 5;

private int ballDiameter = 40;

private int speed = 10;

private List<Color> ballColors = new List<Color> { Color.Red, Color.Green, Color.Blue, Color.Yellow, Color.Purple };

public Task\_10()

{

InitializeComponent();

InitializeBalls();

InitializeTimer();

}

private void InitializeBalls()

{

if (ballColors.Count < ballCount)

{

// Додавання кольорів

while (ballColors.Count < ballCount)

{

Color newColor = GetRandomColor();

if (!ballColors.Contains(newColor))

{

ballColors.Add(newColor);

}

}

}

else if (ballColors.Count > ballCount)

{

// Вибір кольорів

ballColors = ballColors.OrderBy(x => random.Next()).Take(ballCount).ToList();

}

balls = new Ball[ballCount];

List<Color> shuffledColors = new List<Color>(ballColors);

shuffledColors = shuffledColors.OrderBy(x => random.Next()).ToList();

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

{

// Створення кульки

int diameter = ballDiameter;

Color color = shuffledColors[i];

double dx = GetRandomSpeed();

double dy = GetRandomSpeed();

if (dx == 0) dx = 1;

if (dy == 0) dy = 1;

balls[i] = new Ball(

x: random.Next(50, this.ClientSize.Width - 50 - ballDiameter),

y: random.Next(50, this.ClientSize.Height - 50 - ballDiameter),

diameter: diameter,

color: color,

dx: dx,

dy: dy,

speed: speed

);

}

}

private double GetRandomSpeed()

{

// Швидкість кульки

double s = 0;

while (s == 0)

{

s = random.NextDouble() \* (speed - 1) + 1;

if (random.Next(2) == 0)

s = -s;

}

return s;

}

private Color GetRandomColor()

{

// Випадковий колір

Color newColor;

do

{

newColor = Color.FromArgb(random.Next(256), random.Next(256), random.Next(256));

}

while (ballColors.Contains(newColor));

return newColor;

}

private void InitializeTimer()

{

// Налаштування таймера

timer = new System.Windows.Forms.Timer

{

Interval = 20

};

timer.Tick += UpdateBalls;

timer.Start();

}

private void UpdateBalls(object sender, EventArgs e)

{

// Оновлення кульок

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

{

balls[i].Move(this.ClientSize);

for (int j = i + 1; j < balls.Length; j++)

{

balls[i].CheckCollision(balls[j]);

}

}

Invalidate();

}

protected override void OnPaint(PaintEventArgs e)

{

// Малювання кульок

base.OnPaint(e);

foreach (var ball in balls)

{

ball.Draw(e.Graphics);

}

}

protected override void OnLoad(EventArgs e)

{

// Завантаження форми

base.OnLoad(e);

Button btnSettings = new Button

{

Text = "Налаштування",

Location = new Point(this.ClientSize.Width - 120, 10),

Size = new Size(100, 30),

Anchor = AnchorStyles.Top | AnchorStyles.Right

};

btnSettings.Click += BtnSettings\_Click;

Controls.Add(btnSettings);

}

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

{

// Обробка кнопки

timer.Stop();

using (SettingsForm settingsForm = new SettingsForm(ballCount, ballDiameter, speed, ballColors))

{

if (settingsForm.ShowDialog() == DialogResult.OK)

{

ballCount = settingsForm.BallCount;

ballDiameter = settingsForm.Diameter;

speed = settingsForm.Speed;

ballColors = settingsForm.Colors.Distinct().ToList();

InitializeBalls();

}

}

timer.Start();

}

private class Ball

{

public int X { get; set; }

public int Y { get; set; }

public int Diameter { get; }

public Color Color { get; }

public double Dx { get; set; }

public double Dy { get; set; }

private static Random random = new Random();

private double speed;

public Ball(int x, int y, int diameter, Color color, double dx, double dy, double speed)

{

// Конструктор

X = x;

Y = y;

Diameter = diameter;

Color = color;

Dx = dx;

Dy = dy;

this.speed = speed;

}

public void Move(Size clientSize)

{

// Рух кульки

X += (int)Math.Round(Dx);

Y += (int)Math.Round(Dy);

if (X <= 0 || X + Diameter >= clientSize.Width)

{

Dx = -Dx;

X = Math.Max(0, Math.Min(X, clientSize.Width - Diameter));

}

if (Y <= 0 || Y + Diameter >= clientSize.Height)

{

Dy = -Dy;

Y = Math.Max(0, Math.Min(Y, clientSize.Height - Diameter));

}

}

public void CheckCollision(Ball other)

{

// Перевірка зіткнення

double dx = (X + Diameter / 2.0) - (other.X + other.Diameter / 2.0);

double dy = (Y + Diameter / 2.0) - (other.Y + other.Diameter / 2.0);

double distance = Math.Sqrt(dx \* dx + dy \* dy);

if (distance < Diameter)

{

double overlap = Diameter - distance;

double correctionX = (overlap \* dx) / distance / 2.0;

double correctionY = (overlap \* dy) / distance / 2.0;

X += (int)Math.Round(correctionX);

Y += (int)Math.Round(correctionY);

other.X -= (int)Math.Round(correctionX);

other.Y -= (int)Math.Round(correctionY);

double nx = dx / distance;

double ny = dy / distance;

double tx = -ny;

double ty = nx;

double v1n = Dx \* nx + Dy \* ny;

double v1t = Dx \* tx + Dy \* ty;

double v2n = other.Dx \* nx + other.Dy \* ny;

double v2t = other.Dx \* tx + other.Dy \* ty;

double v1nAfter = v2n;

double v2nAfter = v1n;

Dx = v1nAfter \* nx + v1t \* tx;

Dy = v1nAfter \* ny + v1t \* ty;

other.Dx = v2nAfter \* nx + v2t \* tx;

other.Dy = v2nAfter \* ny + v2t \* ty;

AdjustSpeed();

other.AdjustSpeed();

}

}

private void AdjustSpeed()

{

// Швидкість після зіткнення

double currentSpeed = Math.Sqrt(Dx \* Dx + Dy \* Dy);

if (currentSpeed < 1.0)

{

if (currentSpeed == 0)

{

double angle = random.NextDouble() \* 2 \* Math.PI;

Dx = 1.0 \* Math.Cos(angle);

Dy = 1.0 \* Math.Sin(angle);

}

else

{

double scale = 1.0 / currentSpeed;

Dx \*= scale;

Dy \*= scale;

}

}

}

public void Draw(Graphics g)

{

// Малювання кульки

using (Brush brush = new SolidBrush(Color))

{

g.FillEllipse(brush, X, Y, Diameter, Diameter);

}

}

}

}

}

***SettingsForm.cs:***

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

namespace Task\_10

{

public partial class SettingsForm : Form

{

public int BallCount { get; private set; }

public int Diameter { get; private set; }

public int Speed { get; private set; }

public List<Color> Colors { get; private set; }

public SettingsForm(int currentBallCount, int currentDiameter, int currentSpeed, List<Color> currentColors)

{

InitializeComponent();

nudBallCount.Value = currentBallCount;

nudDiameter.Value = currentDiameter;

nudSpeed.Value = currentSpeed;

Colors = new List<Color>(currentColors.Distinct()); // Унікальні кольори

UpdateColorList();

lbColors.DrawMode = DrawMode.OwnerDrawFixed; // Режим малювання списку

lbColors.DrawItem += lbColors\_DrawItem; // Додано обробник подій

}

private void UpdateColorList()

{

lbColors.Items.Clear(); // Очистка списку

foreach (var color in Colors) // Додавання кольорів

{

lbColors.Items.Add(color);

}

}

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

{

using (ColorDialog colorDialog = new ColorDialog()) // Відкриття діалогу кольору

{

if (colorDialog.ShowDialog() == DialogResult.OK) // Перевірка вибору

{

if (!Colors.Contains(colorDialog.Color)) // Унікальність кольору

{

Colors.Add(colorDialog.Color); // Додавання кольору

UpdateColorList(); // Оновлення списку

lbColors.Invalidate(); // Перемальовування

}

else

{

MessageBox.Show("Цей колір вже додано.", "Інформація", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

}

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

{

if (lbColors.SelectedIndex >= 0) // Перевірка вибору

{

Colors.RemoveAt(lbColors.SelectedIndex); // Видалення кольору

UpdateColorList(); // Оновлення списку

lbColors.Invalidate(); // Перемальовування

}

}

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

{

BallCount = (int)nudBallCount.Value; // Отримання кількості кульок

Diameter = (int)nudDiameter.Value; // Отримання діаметру

Speed = (int)nudSpeed.Value; // Отримання швидкості

if (Colors.Count < BallCount) // Перевірка кількості кольорів

{

MessageBox.Show("Кількість кольорів повинна бути не менше кількості кульок.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

DialogResult = DialogResult.OK; // Завершення роботи

Close();

}

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

{

DialogResult = DialogResult.Cancel; // Відміна

Close();

}

private void lbColors\_DrawItem(object sender, DrawItemEventArgs e)

{

if (e.Index < 0) return; // Перевірка індексу

e.DrawBackground(); // Малювання фону

Color color = Colors[e.Index]; // Отримання кольору

using (Brush brush = new SolidBrush(color)) // Кисть для кольору

{

e.Graphics.FillRectangle(brush, e.Bounds.Left + 2, e.Bounds.Top + 2, e.Bounds.Height - 4, e.Bounds.Height - 4); // Малювання кольору

}

using (Brush textBrush = new SolidBrush(Color.Black)) // Кисть для тексту

{

string colorName = color.IsKnownColor ? color.Name : $"#{color.R:X2}{color.G:X2}{color.B:X2}"; // Назва кольору

e.Graphics.DrawString(colorName, e.Font, textBrush, e.Bounds.Left + e.Bounds.Height + 5, e.Bounds.Top); // Малювання назви

}

e.DrawFocusRectangle(); // Малювання фокусу

}

}

}

**Результат роботи програми:**

Изображение выглядит как снимок экрана, текст, Красочность

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Изображение выглядит как снимок экрана, Красочность, круг, Графика

Автоматически созданное описание

**Алгоритм побудови коду:**

1. **Головна форма**:
   * Ініціалізація масиву кульок із початковими параметрами.
   * Створення таймера для оновлення положення кульок кожні 20 мс.
   * Реалізація обробки руху кульок із врахуванням відбивання від стінок та зіткнення між кульками.
2. **Налаштування параметрів**:
   * Кнопка "Налаштування" відкриває форму налаштувань.
   * Передача поточних параметрів у форму налаштувань.
   * Після підтвердження змін:
     + Оновлюється кількість кульок.
     + Змінюються діаметри, кольори та швидкості кульок.
     + Перегенерація масиву кульок.
3. **Форма налаштувань**:
   * Поля для введення:
     + Кількості кульок.
     + Діаметру кульок.
     + Швидкості кульок.
   * Список кольорів кульок із можливістю додавання та видалення кольорів.
   * Перевірка:
     + Кількість кольорів повинна бути не меншою за кількість кульок.
4. **Динамічний рух кульок**:
   * Оновлення положення кульок у головній формі.
   * Врахування зіткнень кульок із відбиванням та змінюваними швидкостями.

**Функціональність:**

1. **Рух кульок**:
   * Кульки рухаються в межах форми, відбиваючись від стінок.
   * Реалізовано зіткнення кульок із корекцією їх положення та швидкості.
2. **Налаштування**:
   * Можливість змінювати:
     + Кількість кульок.
     + Кольори (через діалог вибору кольору).
     + Діаметри кульок.
     + Швидкість руху кульок.
3. **Візуалізація**:
   * Кульки різних кольорів та діаметрів динамічно рухаються та взаємодіють між собою.
4. **Інтерактивність**:
   * Зміни параметрів застосовуються в реальному часі після підтвердження.

**Висновок:**

Програма успішно реалізує динамічний рух кульок із додатковою функціональністю налаштування параметрів у окремій формі. Код дозволяє легко змінювати та розширювати функціонал. Реалізація відповідає вимогам завдання, забезпечуючи інтерактивність та гнучкість у використанні.

**Загальний Висновок**

Під час виконання лабораторної роботи №13 з теми "Використання засобів графіки Windows Forms" було розроблено серію застосунків, що демонструють різні аспекти роботи з графікою у середовищі .NET Framework.

Основним інструментом для візуалізації графічних примітивів був клас Graphics, що надає методи для малювання ліній, прямокутників, еліпсів, заповнення областей, виведення тексту та інші.

Для відображення графіки використовувався компонент PictureBox, а для динамічної зміни графічного вмісту застосовувався компонент Timer.

У ході роботи було реалізовано наступні завдання:

* Малювання прапорів України та інших країн з використанням графічних примітивів.
* Створення інтерактивних елементів, що реагують на події миші.
* Відображення умовних позначень електричних елементів.
* Побудова графіків функцій та динамічних графіків курсу валют.
* Створення кругової діаграми для відображення даних про гірські вершини.
* Реалізація анімації руху кульок з імітацією зіткнень та налаштуванням параметрів.

В цілому, лабораторна робота дозволила набути практичних навичок у використанні графічних можливостей Windows Forms, включаючи малювання різних примітивів, обробку подій миші, використання таймеру та створення динамічних графічних застосунків.