Програмування графічного інтерфейсу в C#

Основні теми

- Базові засади організації графічного інтерфейсу.
- Найважливіші інтерфейсні компоненти.
- Динамічне додавання компонентів.
- Використання GDI+.
- Діалоги.
- Створення власних компонентів.

Графічний інтерфейс: початок

- Зручні можливості для створення в середовищі Visual Studio, але графічні програми можна писати і в режимі створення консольних застосувань, і **лише засобами командного рядка.**

Найпростіше вікно: створення в “ручному” режимі

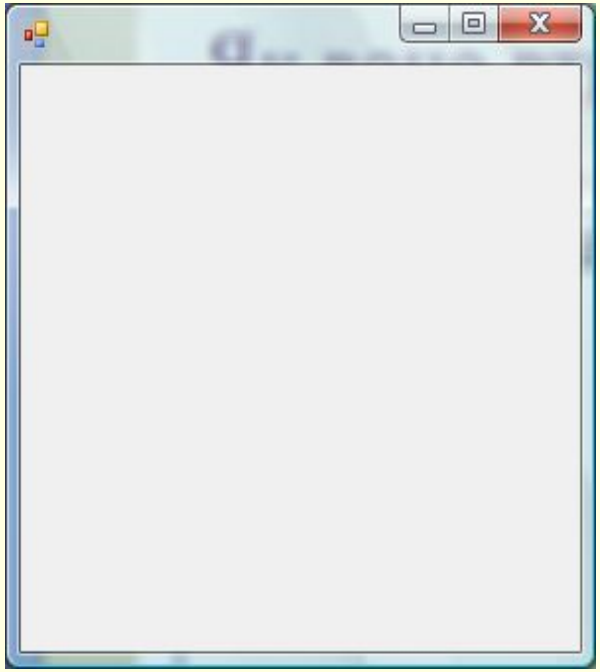
```
using System;
using System.Windows.Forms; //імпорт бібліотеки

namespace MyApp {
public class MainWindow:Form
{//підклас класу Form
public static void Main() {
Application.Run(new MainWindow()); //запуск
}
}
}
```

Режими компіляції

- Це застосування можна відкомпілювати і запустити способом, характерним для консольних застосувань, але тоді спочатку запуститься консоль, а потім відкриється вікно.
- Натомість можна відкомпілювати так:

csc /target:winexe /out:mainwindow.exe mainwindow.cs

Як воно виглядає



Зробимо щось цікавіше…

```
public class MainWindow:Form {

public MainWindow() {
//Встановлення властивостей
this.Text="Крута прога";

//Додавання компонента
this.Controls.Add(new MonthCalendar());
}
public static void Main() {
Application.Run(new MainWindow());
}
}
```

Клас Application

- Важливі методи для роботи з графічними застосуваннями.
- **Run(форма)** – запуск Windows-застосування з головною формою, вказаною як аргумент.
- **Exit()** – завершення застосування.

Завершення застосування

- Подія **ApplicationExit** тісно пов’язана з делегатом **EventHandler**.
- Опис:

delegate void EventHandler (object sender,

EventArgs e);

- Для інших подій використовуються подібні делегати.

Приклад коду

```
public MainWindow() {
this.Text="Крута прога";
this.Controls.Add(new MonthCalendar());
Application.ApplicationExit+=new
    EventHandler(MainWindow_OnExit);
}

private void MainWindow_OnExit(object sender,
    EventArgs args) {
MessageBox.Show("Message","Кінець Вашому
    застосуванню");
}
```

Клас Control

- Клас, базовий для графічних компонент (в тому числі і для класу Form).

Клас Form

- Форми, які створюються для того чи іншого графічного застосування, як правило, мають бути похідними від класу **Form**.
- Одне застосування може використовувати декілька форм.

Життєвий цикл форми

- Load.
- Activated.
- Deactivate.
- Closing.
- Closed.

Компоненти

- Основні характеристики: ***клас компонента***, ***властивості, події***.
- Visual Studio надає зручні засоби для роботи і з властивостями, і з подіями.
- Динамічне створення компонент під час виконання програми: власне додавання компонента (**Controls.Add(…)**), додавання обробників подій (на основі механізму обробки подій).

Основні компоненти

- **Label.**
- **LinkLabel –** гіпертекст.
- **TextBox**. Важливі властивості: **Text**, **ReadOnly**.
- **Button**. Важлива подія – **Click**.
- Вибір: **CheckBox**, **RadioButton**, **ComboBox** і т.п.
- Меню та toolbars.
- Панелі.
- Діалоги.
- **TrackBar**, **UpDown** і т.п.
- Підказки **(HelpProvider)**.
- **ErrorProvider**.

Приклад 1

- Форма з двома текстовими полями та кнопкою. В одному полі вводиться ім’я користувача, після натискання кнопки виводиться привітання цьому користувачеві.
- Подія **Click**.

Обробка

```
private void button1_Click(object sender,
   EventArgs e)
      {
         tout.Text = "Hello, "+tin.Text;



      }
```

Приклад 1: продовження

- Як добитися, щоб аналогічна реакція відбувалася після натискання на Enter в полі введення? Можна обробляти подію **KeyPress** і аналізувати код натисненої клавіші, але є більш простий спосіб – властивість **AcceptButton** форми.

Приклад 2: годинник

- Ідея – використати компонент **Timer**. Налаштувати його так, щоб він з певною періодичністю генерував подію **Tick**; а в обробнику передбачити виведення поточного часу.

Код обробника

```
private void timer1_Tick(object sender,
    EventArgs e)
        {
            tl2.Text =
    DateTime.Now.ToLongTimeString();
        }
```

Приклад 3: перехоплення закриття форми; діалогове вікно

```
private void Form1_FormClosing(object
   sender, FormClosingEventArgs e)
        {
            if (MessageBox.Show( “Do you want to
   exit?", "Closing",
   MessageBoxButtons.YesNo) !=
DialogResult.Yes)
                    e.Cancel = true;
        }
```

Приклад 4: створення заготовки для текстового редактора

- Компонента **RichTextBox**, яка дозволяє працювати з RTF-форматом.
- **ToolStrip**.
- **MenuStrip**.
- Команда **InsertStandardItems**.
- Програмування відповідних обробників.

Основні можливості для малювання

- Бібліотека GDI+.
- Ключовий клас – **System.Drawing.Graphics**. Зокрема, для малювання потрібно отримати об’єкт цього класу.

Простий приклад – малювання кіл при клацанні мишею

```
private void Form1_MouseClick(object sender,
    MouseEventArgs e)
        {
            Graphics g = this.CreateGraphics();
            int cr = r.Next(255);
            int cg = r.Next(255);
            int cb = r.Next(255);
            Pen p = new
    Pen(Color.FromArgb(cr,cg,cb),10);
            g.DrawEllipse(p, e.X, e.Y, 50, 50);

        }
```

Проблема попереднього прикладу

- При зміні розмірів вікна зображення затирається і не перемальовується.

Деякі методи малювання

- Перевизначення методу **OnPaint**(PaintEventArgs e).
- **Обробка події Paint** за допомогою **делегата PaintEventHandler** (object sender, PaintEventArgs e).
- Для примусового перемальовування – **Invalidate()** або **Refresh().**
- Використання класу **Image**, точніше – його підкласу **Bitmap** (т.зв. малювання в пам’яті).

Приклад коду

```
Image im = new Bitmap(file);
  Graphics g = Graphics.FromImage(im);
//Малювання
g.DrawEllipse(new Pen(Color.Red, 20), 100,
   100, 50, 50);
        Font fnt = new Font("Times New
   Roman",12,FontStyle.Italic);
        Brush brsh = Brushes.Chocolate;
        g.DrawString("The image", fnt,
   brsh,110,10);
```

Заповнення PictureBox

```
pictureBox1.Image = im;
```

Збереження зображення на диску

```
im.Save(@"C:\csharp-training\im.jpg");
```

Створення власних компонентів

- C# - **компонентно-орієнтована мова**.
- Класи, похідні від **Component**, **UserControl** або **Control**.
- Visual Studio має досить розвинені візуальні засоби для роботи з такими компонентами.
- Демонстрація.