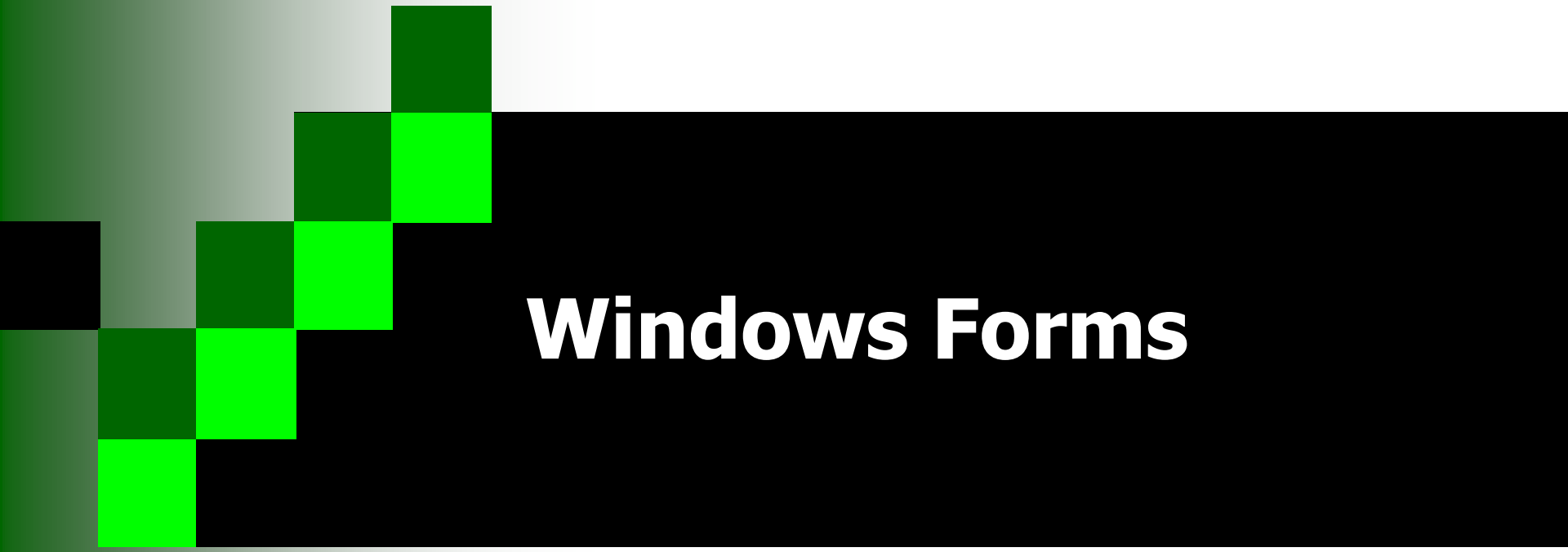
Windows Forms

.NET Framework

- Technologia firmy Microsoft wprowadzona w 2002 roku
  - 1.1 – 2003, 2.0 – 2005, 3.0 – 2006, 3.5 – 2007
- Integracja z wersjami systemu Windows:
  - 1.0, 1.1, 2.0
    - dostępne dla: 98, NT 4.0, 2000+
    - wersja 1.1 w instalacji Windows Server 2003
    - Microsoft .NET Framework Version 2.0 Redistributable Package (x86) – 22.4 MB instalator, 280 MB miejsca na dysku
  - 3.0, 3.5
    - dostępne dla: XP SP2, 2003, Vista, 2008
    - wersja 3.0 w instalacji Windows Vista i Windows Server 2008
    - Microsoft .NET Framework 3.5 – 197 MB pełny instalator, do 500 MB miejsca na dysku

Windows Forms (WinForms)

- Klasy należące do .NET Framework używane do tworzenia aplikacji z graficznym interfejsem użytkownika dla systemów Windows
  - przestrzeń nazw **`System.Windows.Forms`**
- Kategorie klas:
  - podstawowe (np. **`Application`**, **`Form`**)
  - kontrolki – dziedziczące z klasy **`Control`** (np. **`Button`**)
  - komponenty – niedziedziczące z klasy **`Control`** (np. **`Timer`**, **`ToolTip`**)
  - standardowe okna dialogowe (np. **`OpenFileDialog`**, **`PrintDialog`**)

Przykładowy program

```
using System;
using System.Windows.Forms;

namespace MyWindowsApp
{
    public class MainWindow : Form
    {
        static void Main(string[] args)
        {
            Application.Run(new MainWindow());

            MessageBox.Show("That's all",
                            "Sample program",
                            MessageBoxButtons.OK,
                            MessageBoxIcon.Stop);
        }
    }
}
```

Obsługa zdarzeń

- GUI jest oparty na zdarzeniach wynikających z interakcji z użytkownikiem
- *Event handlers* – metody wywoływane do obsługi zdarzeń
- Dla każdego zdarzenia istnieje definicja sygnatury dla metody obsługi tego zdarzenia (*delegate*)
  - dla każdego zdarzenia trzymana jest lista referencji do metod zapisanych na to zdarzenie
  - w momencie wystąpienia zdarzenia wszystkie metody zapisane na tej liście są wywoływane (kolejność ich wywoływania nie jest zdefiniowana)

Zdarzenia - przykład

```
namespace MyApplication {
  public class MyForm : Form {
    public MyForm() {
      FormClosing += new FormClosingEventHandler(OnClosing);
    }

    private void OnClosing(Object sender,
                           FormClosingEventArgs e) {
      if (MessageBox.Show("Sure to close?", "Question",
                MessageBoxButtons.YesNo) == DialogResult.No) {
        e.Cancel = true;
      }
    }
  }
}
```

```
namespace MyApplication {
  static class Program {
    static void Main() {
      Application.ApplicationExit += new
          EventHandler(Program_ApplicationExit);
      Application.Run(new MyForm());
    }

    private static void Program_ApplicationExit(
                Object sender, EventArgs e) {
      MessageBox.Show("That's all");
    }
  }
}
```

Klasa `Application`

- Statyczne metody i właściwości do zarządzania aplikacją
  - **`Run()`**
  - **`Exit()`, `ExitThread()`**
  - **`DoEvents()`** – przydatne podczas długotrwałych obliczeń (umożliwia obsługę oczekujących zdarzeń)
  - **`EnableVisualStyles()`** – obsługa stylów Windows XP
  - zdarzenie **`Idle`** – wywoływane w momencie rozpoczęcia okresu bezczynności aplikacji
  - zdarzenie **`ApplicationExit`** – powiadomienie o zakończeniu pracy aplikacji
  - właściwości: **`ExecutablePath`, `StartupPath`**

Klasa `Form`

- Reprezentuje główne okna, okna dialogowe oraz okna robocze w aplikacji MDI
- Hierarchia dziedziczenia:

```
Object
  MarshalByRefObject
    Component
      Control
        ScrollableControl
          ContainerControl
            Form
```

Schemat funkcjonowania formularza

1. **`Constructor`**
   - **`InitializeComponent()`** – utworzenie i zainicjowanie wszystkich kontrolek potomnych dodanych do formularza przy pomocy Form Designer'a z Visual Studio
2. **`Form.Load`**
3. **`Form.Activated`**
4. **`//...`**
5. **`Form.Deactivate`**
6. **`Form.FormClosing`**
7. **`Form.FormClosed`**
8. **`Dispose`** (interface **`IDisposable`**)
   - miejsce zwalniania wszystkich zasobów używanych przez formularz
9. **`Destructor`** (wywoływany przez *Garbage Collector*)

Rozmiar i pozycja formularza

- Widoczność
  - `Show()`
  - `Visible`
  - `Shown, VisibleChanged`
- Właściwości:
  - `Region, Bounds, DesktopBounds, ClientRectangle`
  - `Left, Top, Right, Bottom`
  - `StartPosition,`
  - `Location, DesktopLocation`
  - `Width, Height,`
  - `Size, ClientSize,`
  - `MinimumSize, MaximumSize`
  - `AutoSize, AutoSizeMode`
  - `WindowState, TopMost`

Rozmiar i pozycja formularza c.d.

- Metody:
  - `SetBounds(), SetDesktopBounds(), SetDesktopLocation()`
  - `CenterToParent(), CenterToScreen()`
  - `BringToFront()`
  - `PointToClient(), PointToScreen(), RectangleToClient(), RectangleToScreen()`
  - `SizeFromClientSize()`
- Zdarzenia:
  - `ClientSizeChanged, SizeChanged`
  - `LocationChanged`
  - `MaximumSizeChanged, MinimumSizeChanged`
  - `Resize, ResizeBegin, ResizeEnd`

Wygląd formularza

- Właściwości:
  - `ForeColor, BackColor`
  - `BackgroundImage, BackgroundImageLayout (Center, None, Stretch, Tile, Zoom)`
  - `Cursor, Icon`
  - `Font`
  - `FormBorderStyle, ShowIcon, ControlBox, MinimizeBox, MaximizeBox, HelpButton, SizeGripStyle`
  - `ShowInTaskbar, TopLevel`
  - `Opacity`
  - `TransparencyKey`

Wygląd formularza c.d.

- Metody:
  - `ResetBackColor(), ResetForeColor(),`
  - `ResetFont(), ResetCursor()`
- Zdarzenia:
  - `ForeColorChanged, BackColorChanged`
  - `BackgroundImageChanged, BackgroundImageLayoutChanged`
  - `FontChanged`

Mysz

- Zdarzenia myszy
  - **`MouseMove, MouseDown, MouseUp, MouseClick, MouseDoubleClick`**
    - **`MouseEventArgs {Button, Clicks, Delta, Location, X, Y}`**
  - **`MouseEnter, MouseHover, MouseLeave, MouseWheel`**
  - **`Click, DoubleClick`**
- Klasa **`Cursor`**
  - konstruktor wczytuje kursor ze strumienia, pliku, zasobów programu lub systemu
  - **`Cursor`** – właściwość formularza
  - standardowe kursory: klasa **`Cursors`**, np. **`Cursors.WaitCursor`**

Klawiatura

- Fokus
  - `CanFocus`, `Focused`, `ContainsFocus` (z potomnymi)
  - `Focus()`
  - `GotFocus`, `LostFocus` (zdarzenia niskiego poziomu)
  - `Enter`, `Leave` (dla kontrolek), `Activated`, `Deactivate` (dla formularzy)
- Zdarzenia
  1. `KeyDown (KeyEventArgs: KeyCode, KeyData, Modifiers, Alt, Control, Shift)`
  2. `KeyPress (KeyPressEventArgs: KeyChar)`
  3. `KeyUp (KeyEventArgs)`
- Wyliczenie `Keys` (np. `Keys.Q`, `Keys.F5`, `Keys.LShiftKey`)

Formularze modalne i niemodalne

MODALNE

- Blokują dostęp do innych okien aplikacji
- `ShowModal()`
- `Close()`
  - `FormClosing`
  - `FormClosed`
- `CancelButton, AcceptButton`
- `DialogResult`

NIEMODALNE

- `Show()`
- `Close()`
  - `FormClosing`
  - `FormClosed`

Obsługa komunikatów Win32

```
private const int HTCAPTION = 0x0002;
private const int WM_NCHITTEST = 0x0084;


protected override void WndProc(ref Message m)
{
  switch (m.Msg)
  {
    case WM_NCHITTEST:
      m.Result = (IntPtr)HTCAPTION;
      break;
    default:
      base.WndProc (ref m);
      break;
  }
}
```

- `Message: {HWnd, LParam, Msg, Result, WParam}`
- `Form.Handle` – `HWND` jako typ `IntPtr`

Ustawienia aplikacji

- Pliki .config
  - do odczytu i zapisu z poziomu aplikacji
  - XML
- Dynamiczne właściwości

```
<configuration>
  <appSettings>
    <add key="welcome" value="Hello my user"/>
    <add key="Form1.Opacity" value="1"/>
  </appSettings>
</configuration>
```

```
AppSettingsReader reader = new AppSettingsReader();
string s = (string)reader.GetValue("welcome",
                                   typeof(string));
```

Zasoby

- *Assembly* jest zbiorem typów i zasobów
  - dane binarne, pliki tekstowe, dźwiękowe, video, tablice ciągów znaków, ikony, obrazki, pliki XML
- Wielojęzyczne aplikacje
  - problem z wielojęzycznym interfejsem użytkownika
  - każdy zasób dodany do assembly może mieć określony swój język i kraj (np. "pl-PL", "en-US", "de-De", "de-AT")
  - *satellite assemblies*

Pliki zasobów

```
Language=Polish
Next=Następna strona
Prev=Poprzednia strona
```

- .txt
  - tekstowy format nazwa/wartość
  - prosty w użyciu dla zasobów tekstowych
- .resx
  - format XML
  - użyteczny dla tekstów i innych typów (np. obrazków)
- .resources
  - format binarny
  - binary odpowiednik pliku XML
  - tylko w tym formacie zasoby mogą być dołączone do *assembly*, pozostałe formaty muszą być konwertowane

Tworzenie plików .resx

```
ResXResourceWriter w =
    new ResXResourceWriter(@"C:\myRes.resx");

Image img = new Bitmap("pattern.bmp");
w.AddResource("background", img);

w.AddResource("Next", "Następny");

w.Generate();
w.Close();
```

```
<?xml version="1.0" encoding="utf-8"?>
<root>
  (...)
  <data name="background"
        type="System.Drawing.Bitmap, System.Drawing"
        mimetype="application/x-microsoft.net.object.bytearray.base64">
    <value>
        Qk32BAAAAAAAAHYAAAAoAAAAMAAAADAAAAABAAQ
        AAAAAAAAAAADDDgAAww4AABAAAAAQAAAAAAA/w
        ...
    </value>
```

Tworzenie plików .resources

- Konwersja z plików .resx i .txt

```
resgen myRes.resx myRes.resources
resgen polish.txt polish.resources
```

- Z poziomu kodu źródłowego

```
ResourceWriter rw = new
        ResourceWriter(@"C:\myRes.resources");

Image img = new Bitmap("pattern.bmp");
w.AddResource("background", img);

w.AddResource("HelloWorld", "Witaj świecie");

w.Generate();
w.Close();
```

Dodawanie plików .resources do *assembly*

- Zasoby muszą być fizycznie włączone do *assembly*
  - każde *assembly* zawiera manifest opisujący jego zawartość

```
csc /t:exe /resource:polish.resources
           /resource:myRes.resources myApp.cs
```

Odczyt zasobów z poziomu kodu źródłowego

```
ResourceManager rm = new ResourceManager(
    "myApp.myRes",
    Assembly.GetExecutingAssembly());

MessageBox.Show(rm.GetString("HelloWorld"));

pictureBox.Image =
    (Bitmap)rm.GetObject("background");

rm.ReleaseAllResources();
```

Użycie Visual Studio 2005 lub 2008

- Dodawanie zasobów
  - menu: Project / Add New Item / Resources File
  - dwa sposoby dodawania zasobów: *linked*, *embedded*
  - zostanie wygenerowana specjalna klasa z właściwościami odpowiadającymi poszczególnym zasobom (Resources1.Designer.cs)
- Edycja zasobów
  - wbudowane edytory Visual Studio
    - binarny, graficzny
  - zewnętrzne edytory (np. Paint)
    - można powiązać uruchamiane aplikacje z typami zasobów
- Kompilacja zasobów do *assembly*
  - użycie resgen.exe

Typy zasobów: *Linked, Embedded*

- *Linked* (domyślny typ dla VS)
  - trzymane w zewnętrznych plikach w projekcie
  - plik .resx zawiera tylko względną ścieżkę lub odnośnik do pliku na dysku
  - podczas kompilacji zawartość plików jest kopiowana do tworzonego *assembly*
- *Embedded*
  - dane zasobu są trzymane w pliku .resx (w tekstowej reprezentacji danych binarnych przy użyciu kodowania base64)
- W obu przypadkach zasoby są w całości włączane do wynikowego pliku (.exe lub .dll)

Użycie klasy wygenerowanej przez VS

```
internal class Resource1 {
  internal static string String1 {
    get {
      return ResourceManager.GetString(
                        "String1", resourceCulture);
    }
  }
  internal static System.Drawing.Bitmap FeatherTexture {
    get {
      object obj = ResourceManager.GetObject(
                         "FeatherTexture", resourceCulture);
      return ((System.Drawing.Bitmap)(obj));
    }
  }
}
```

```
pictureBox1.Image = Resource1.FeatherTexture;
label1.Text = Resource1.String1;
```