Okna

269. Minimální velikost okna

Běžně může uživatel měnit velikost okna, pokud má nastaven typ okraje, který to umožňuje. Abyste omezili volnost, s jakou je možné okno zmenšit, můžete nastavit vlastnost formuláře `MinimumSize`, a ta již zajistí vše potřebné.

```
Me.MinimumSize = New Size(100, 100)
```

270. Maximální velikost okna

Zamezit uživateli změnu velikosti byste mohli například pomocí události `Resize`, ale to by vedlo ke skokovému návratu velikosti až po ukončení změny. Mnohem jednodušeji a bez nežádoucího vizuálního efektu můžete nastavit maximální velikost do vlastnosti, která se už při průběhu zvětšování postará, aby uživatel nepřekročil požadovanou mez.

```
Me.MaximumSize = New Size(300, 400)
```

271. Implicitní potvrzovací tlačítko

Běžným zvykem u dialogů je, že pokud uživatel zmáčkne klávesu Enter, potvrdí tím akci, jako by zmáčkl příslušné tlačítko. Nastavením vlastnosti formuláře `AcceptButton` na ono potvrzovací tlačítko se vyhnete zachytávání mačkání této klávesy a celá akce proběhne automaticky.

```
'butAccpet - tlačítko na formuláři
Me.AcceptButton = butAccpet
```

272. Implicitní rušící tlačítko

Tak jako funguje Enter pro potvrzení akce, slouží klávesa ESC pro zrušení akce. Bez zachytávání zmáčknutých kláves obslouží tuto akci tlačítko, jež nastavíte formuláři do vlastnosti `CancelButton`.

```
'butCancel - tlačítko na formuláři
Me.CancelButton = butCancel
```

273. Návratový stav dialogu

Pokud zobrazíte dialog pomocí metody `ShowDialog` a chcete po jeho uzavření zpracovat výsledek akce, vrací tato metoda návratovou hodnotu dialogu, jejíž možné hodnoty jsou dány výčtovým typem `DialogResult`.

```
'vytvoří instanci dialogu
Dim dialog As New Form2
```

```
'Otevře jej modálně a zkontroluje návratový stav
If dialog.ShowDialog() = Windows.Forms.DialogResult.OK Then
  MessageBox.Show("Návratový stav je OK")
End If
```

Návratový stav, který je vrácen, nastavíte tak, že například po stisku tlačítka OK nastavíte vlastnost `DialogResult` na některou z hodnot, kterou nabízí zmíněný výčtový typ.

```
Me.DialogResult = Windows.Forms.DialogResult.OK
```

274. Automatický dialog

Ačkoliv návratový stav dialogu, který vrací metoda `ShowDialog`, si můžete nastavit sami, je možné nastavit jednotlivým tlačítkům dialogu stav, který se automaticky nastaví po jejich zmáčknutí. Tato akce nejen způsobí samotné nastavení, ale také uzavření okna, takže pokud nepotřebujete provádět žádnou další akci a stačí vám vyhodnotit, co uživatel udělal, skončí tím pro vás celá práce bez napsání řádku kódu.

```
'Můžete nastavit v návrhu formuláře v okně vlastností
Form2.butAccept.DialogResult = Windows.Forms.DialogResult.OK
Form2.butCancel.DialogResult = Windows.Forms.DialogResult.Cancel

'vytvoří instanci dialogu
Dim dialog As New Form2
'Otevře jej modálně a zkontroluje návratový stav
If dialog.ShowDialog() = Windows.Forms.DialogResult.OK Then
  MessageBox.Show("Zmáčknuto tlačítko butAccept")
End If
```

275. Dialog se zprávou

Sdělit uživateli nějakou stručnou informaci můžete různými způsoby, ale asi nejrychlejší a nejjednodušší je použít již vytvořený dialog se zprávou, jež se skrývá ve třídě `MessageBox`. Jeho metodě `Show` předáte text zprávy a ta zobrazí okno s tlačítkem OK.

```
MessageBox.Show("Ahoj, Světe!")
```

276. Dialog se zprávou a tlačítky

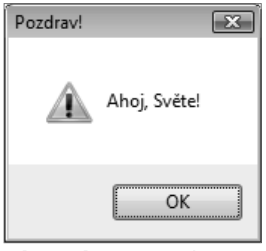
Zobrazit dialog se zprávou, který má uživatel jen možnost potvrdit tlačítkem OK, mnohdy nestačí. Metoda `Show` vám umožní vybrat z několika předdefinovaných variant kombinací tlačítek a po jeho zavření vám vrátí hodnotu reprezentující stisknuté tlačítko.

```
If MessageBox.Show("Ahoj, Světe!", "Pozdrav!", _
  MessageBoxButtons.YesNoCancel) = Windows.Forms.DialogResult.Yes Then
  '... Kód reagující na potvrzení
End If
```

277. Dialog se zprávou a ikonou

Implicitně se dialog se zprávou zobrazí jen s textem a tlačítky bez jakékoliv grafiky. Metoda `Show` vám ovšem pro zdůraznění povahy informace, kterou uživateli zobrazíte, umožní zadat jednu z předdefinovaných ikon z výčtového typu `MessageBoxIcon`.

```
MessageBox.Show("Ahoj, Světe!", "Pozdrav!", _
  MessageBoxButtons.OK,_
MessageBoxIcon.Exclamation)
```



Obrázek 73: Dialog se zprávou a ikonou

278. Dialog se zprávou a implicitním tlačítkem

Pokud zavoláte dialog se zprávou, který má více než jedno tlačítko, zjistíte, že vždy to první je nastaveno jako implicitní. Pokud vám toto chování nevyhovuje, máte možnost jej ovlivnit nastavením jedné z položek výčtového typu `MessageBoxDefaultButton`. Ta obsahuje tři možnosti pro první, druhé a třetí tlačítko, a jelikož není možné zobrazit více než tři tlačítka, je tento výčet dostačující.

```
MessageBox.Show("Ahoj, Světe!", "Pozdrav!", _
  MessageBoxButtons.YesNoCancel, MessageBoxIcon.None, _
  MessageBoxDefaultButton.Button3)
```

279. Automatické uzavření dialogu se zprávou

Jistě jste již v některých aplikacích zaznamenali, že dialog se zprávou, na kterou uživatel za nějaký čas nereagoval, se sám uzavře. Zobrazit jej není problém, ale jak jej uzavřít, když po zavolání metody Show nad ním ztrácíte kontrolu? Trik spočívá v tom, že před zavoláním metody `Show` si vytvoříte `Timer` s délkou intervalu odpovídající času, který uživateli dáte na reakci.

```
'Deklarace časovače na úrovni třídy
Private WithEvents mTimer As Timers.Timer

'Spustí časovač
mTimer = New Timers.Timer(5000)
mTimer.Start()
'Otevře okno se zprávou
If MessageBox.Show("Ahoj, Světe!", "Pozdrav!", _
  MessageBoxButtons.YesNoCancel) = Windows.Forms.DialogResult.OK Then
  '... reakce na potvrzení dialogu
End If
'Zastaví časovač
mTimer.Stop()
```

Po uplynutí zadaného intervalu je vyvolána událost `Elapsed`, ve které vyvoláte klávesu Escape, jež dialog uzavře. Poté už jen časovač zastavíte.

```
Private Sub mTimer_Elapsed(ByVal sender As Object, _
ByVal e As System.Timers.ElapsedEventArgs) Handles mTimer.Elapsed
  'Pošle klávesu pro ukončení dialogu
  SendKeys.SendWait("{ESC}")
  'Zastaví časovač
  mTimer.Stop()
End Sub
```

280. Kontextové menu okna 1

Po klepnutí pravým tlačítkem na formulář je možné použitím vlastnosti `ContextMenu` zobrazit kontextovou nabídku. Nemusíte se tedy starat o obsluhu tlačítka myši, ale vše již funguje automaticky.

```
'Příklad vytvoření menu
Dim cm As New ContextMenu
cm.MenuItems.Add("Položka 1")
cm.MenuItems.Add("Položka 2")
'Přiřazením menu jako kontextové nabídky okna
Me.ContextMenu = cm
```

281. Kontextová nabídka okna 2

Pokud budete v implicitním nastavení Visual Studia hledat v nástrojové liště s prvky `Menu` nebo `ContextMenu`, zjistíte, že tam nejsou. Místo nich ale najdete `MenuStrip` a `ContextMenuStrip`. Původní varianty jdou sice přidat a s jejich použitím nebudete mít žádný problém, ale ty byly vystřídány novou, lepší generací menu, které má daleko více možností. Proto má formulář také vlastnost `ContextMenuStrip`, do které můžete nastavit tuto novou generaci kontextové nabídky.

```
'Příklad vytvoření menu
Dim cms As New ContextMenuStrip
cms.Items.Add("Položka 1X")
cms.Items.Add("Položka 2X")
'Přiřazením menu jako kontextové nabídky okna
Me.ContextMenuStrip = cms
```

Jen pro zajímavost, pokud nastavíte obě vlastnosti, `ContextMenuStrip` i `ContextMenu`, tak přestože první z nich je Microsoftem preferovaná, zobrazí se stará dobrá klasická varianta.

282. Menu se seznamem MDI potomků 1

Většina aplikací, které používají rozhraní MDI okna, nabízí také možnost navigace mezi otevřenými potomky pomocí menu. Vždy jedna z položek, typicky s názvem Okno nebo Window, nabízí položky, na něž když klepnete, provede se aktivace svázaného

okna. Třída `MenuItem` podporuje tuto funkčnost automaticky, a pokud použijete návrháře menu, nemusíte napsat jediný řádek kódu.

```
'Vytvoří novou položku menu
Dim mdiList As New MenuItem("Okna")
'Nastaví menu jako seznam oken
mdiList.MdiList = True
'Přidá položku do menu
Me.Menu.MenuItems.Add(mdiList)
```

283. Menu se seznamem MDI potomků 2

S novou generací menu `MenuStrip` přicházejí nejen nové možnosti, ale i nové způsoby implementace některých funkčností. Informaci o nositeli seznamu MDI potomků už nenese samotná položka, ale menu jako celek s odkazem na zvolenou položku.

```
'Vytvoří novou položku menu
Dim childList As New ToolStripMenuItem("Okna")
'Nastaví menu jako seznam oken
Me.MainMenuStrip.MdiWindowListItem = childList
'Přidá položku do menu
Me.MainMenuStrip.Items.Add(childList)
```

Opět platí, že když použijete návrhář menu, vytvoříte tuto funkčnost bez napsání jediného řádku kódu.

284. Přidání potomka do MDI okna

Hlavním účelem okna MDI je zastřešovat a ovládat podřízená okna. Pokud vytvoříte nový formulář a chcete jej nastavit jako potomka MDI, stačí nastavit vlastnost `MdiParent` na instanci vámi zvoleného hlavního MDI okna.

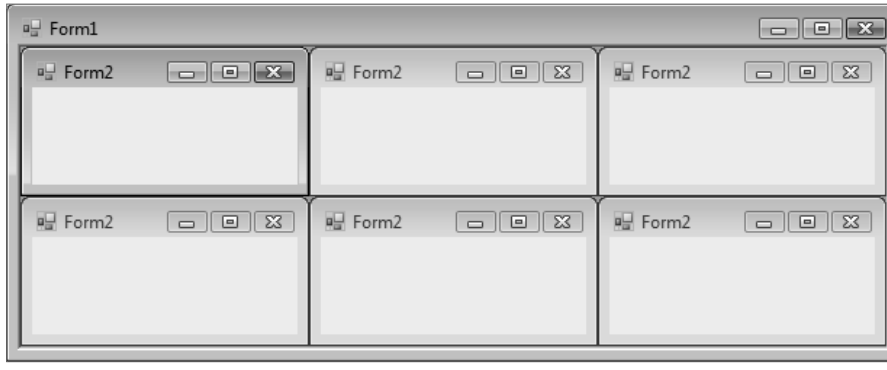
```
'Vytvoří instanci nějakého formuláře
Dim mdiChild As New Form2
'Nastaví odkaz na MDI okno
mdiChild.MdiParent = Me
'Zobrazí Okno uvnitř MDI okna
mdiChild.Show()
```

285. Uspořádání potomků v MDI Oknu

Součástí menu MDI okna bývá často i volba, která seřadí okna podle daného schématu do kaskády nebo vertikálně či horizontálně seřazené vedle sebe. Pro dosažení stejné funkčnosti nemusíte rozmístění řešit sami, ale vše za vás obstará metoda `LayoutMdi`, které jako parametr předáte způsob uspořádání.

```
'Přidá potomky
Dim frm As Form2
For i As Integer = 1 To 6
```

```
  frm = New Form2
  frm.MdiParent = Me
  frm.Show()
Next

'Uspořádá je vedle sebe vertikálně
Me.LayoutMdi(MdiLayout.TileVertical)
```



Obrázek 74: Uspořádání potomků v MDI Oknu

286. Seznam MDI potomků

Každý potomek MDI formuláře má nastavenu vlastnost `MdiParent` na instanci formuláře hlavního MDI okna. Pokud potřebujete zjistit seznam všech těchto potomků, nemusíte kontrolovat tuto vlastnost u všech otevřených oken, stačí jej získat z vlastnosti `MdiChildren`.

```
Array.ForEach(Me.MdiChildren, AddressOf Debug.WriteLine)
```

287. Maximalizace a minimalizace okna

Pokud při otevření okna nebo při jakékoliv vámi zvolené akci chcete, aby se okno maximalizovalo nebo minimalizovalo, nastavte vlastnost `WindowState`. Zde pomocí možností výčtového typu `FormWindowState` určíte, co se má s oknem provést. Pro návrat do běžné velikosti stačí nastavit tuto vlastnost na hodnotu `Normal`.

```
'Maximalizace okna
Me.WindowState = FormWindowState.Maximized
```

288. Poměrné zvětšení formuláře

Ke změně velikosti formuláře můžete použít vlastností `Width` a `Height` nebo `Size`. Pokud chcete ovšem změnit jeho velikost poměrně, nemusíte násobit jejich hodnoty, ale zavolat metodu `Scale`. Té jako parametr předáte sice objekt `SizeF`, ale ten nebude obsahovat rozměr, nýbrž hodnoty, kterými bude každý rozměr automaticky vynásoben.

```
'Poloviční šířka, dvojnásobná výška
Me.Scale(New SizeF(0.5, 2))
```

289. Nastavení velikosti okna podle pracovní oblasti

Pokud nastavujete velikost formuláře, zřejmě to provádíte buď pomocí vlastnosti `Size` nebo dvojice vlastností `Width` a `Height`. To je ovšem velikost vnějšího okraje a nic vám neřekne o velikosti uživatelské oblasti, jejíž rozměry jsou pro práci s prvky pro vás jistě důležitější. Vlastnost `ClientSize`, ze které zjistíte velikost uživatelského prostoru, lze i nastavovat.

```
'Nastavení velikosti pracovní oblasti
Me.ClientSize = New Size(400, 200)
'Velikost celého okna
Debug.WriteLine(Me.Size)
```

290. Přepočet souřadnic okna na obrazovku

Umístění všech prvků a akcí je dáno vnitřním prostorem okna, které je ohraničeno titulkem a okraji. Přepočíst souřadnice z tohoto vztažného prostoru do souřadnic obrazovky od vás vyžaduje znát jejich velikost a umístění okna. Naštěstí to lze mnohem snadněji provést metodou `PointToScreen`.

```
Dim p As Point
'Přepočte souřadnice počátečního bodu
p = Me.PointToScreen(New Point(0, 0))
'Vypíše souřadnice
Debug.WriteLine(p)
```

291. Přepočet souřadnic obrazovky na okno

Aniž byste se museli zabývat umístěním okna a velikostí jeho okrajů a titulku, lze přepočítat bod ze souřadnic obrazovky do vnitřního prostoru okna pouhým voláním metody `PointToClient`.

```
Dim p As Point
'přepočte souřadnice formuláře
p = Me.PointToClient(Me.Location)
'Vypíše souřadnice
Debug.WriteLine(p)
```

292. Zjištění pozice myši

Pro zjištění pozice kurzoru myši nejste odkázáni jen na parametry procedur obsluhujících události myši, ale můžete použít statickou vlastnost `MousePosition` třídy `Form`. Ta vrátí pozici v souřadnicích obrazovky, a tu je pak tedy nezbytné přepočíst na souřadnice uživatelského prostoru formuláře.

```
'Zjistí pozici myši v souřadnicích obrazovky
```

```
Dim mp As Point = Form.MousePosition
'Přepočte do souřadnic okna
mp = Me.PointToClient(mp)
Debug.WriteLine(mp)
```

293. Obrazovka, na které se okno nachází

Dnes už není nic výjimečného, když je k počítači připojen více než jeden monitor, a tak i každý formulář vaší aplikace se může nacházet na jiné obrazovce. Metoda `FromControl` vám vrátí podle zadaného formuláře obrazovku, na které se nachází jeho největší část.

```
Debug.WriteLine(Windows.Forms.Screen.FromControl(Me).ToString())
```

294. Zobrazení posuvných lišt v okně

Bez toho, abyste přidávali na formulář posuvné lišty a starali se o jejich obsluhu, umí formulář po nastavení vlastnosti `AutoScroll` zobrazit posuvníky automaticky. Když žádný z prvků nevybočuje z rámce formuláře, nejsou vidět žádné, ale po přidání či posunutí některého z prvků mimo rámec velikosti okna je ihned zobrazen vertikální nebo horizontální, podle toho, kam je zapotřebí se posouvat.

```
'Automatické zobrazení posuvných lišt v okně
Me.AutoScroll = True

'Přidá prvek mimo viditelný rozsah
Dim lbl As New Label()
lbl.Size = New Size(100, 30)
lbl.Location = New Point(500, 500)
lbl.Text = "Ahoj, Světe!"
Me.Controls.Add(lbl)
```

Stejného chování dosáhnete i v návrhovém módu okna po nastavení příslušné vlastnosti.

295. Zobrazení prvku v okně s posuvníky

Při nastavení vlastnosti `AutoScroll` na `True` vám formulář umožní pracovat s větší plochou, než je jeho rozměr. Pomocí posuvníků se můžete posunout na hledané místo, ale pokud chcete vyobrazit konkrétní prvek, je mnohem jednodušší zavolat metodu `ScrollControlIntoView`, jejímž parametrem bude prvek, který chcete vyobrazit.

```
'Automatické zobrazení posuvných lišt v okně
Me.AutoScroll = True

'Přidá prvek mimo viditelný rozsah
Dim lbl As New Label()
lbl.Size = New Size(100, 30)
lbl.Location = New Point(500, 500)
```

```
lbl.Text = "Ahoj, Světe!"
Me.Controls.Add(lbl)

'Nastaví posuvníky tak, aby byl prvek vidět
Me.ScrollControlIntoView(lbl)
```

296. Velikost virtuálního prostoru formuláře

Pokud máte okno s posuvníky, které umožní uživateli pracovat s prvky ležícími mimo viditelný rozsah okna, máte dvě velikosti. První určuje velikost formuláře samotného a druhá velikost prostoru, kterou obsluhují posuvníky a na kterou se formulář automaticky zvětšuje.

```
'Automatické zobrazení posuvných lišt v okně
Me.AutoScroll = True

'Přidá prvek mimo viditelný rozsah
Dim lbl As New Label()
lbl.Size = New Size(100, 30)
lbl.Location = New Point(500, 500)
lbl.Text = "Ahoj, Světe!"
Me.Controls.Add(lbl)

'Vypíše virtuální velikost
Debug.WriteLine(Me.DisplayRectangle)
```

297. Automatické zvětšování a zmenšování okna

Potřebná velikost okna je určena na základě prvků, které jsou na něm rozmístěny. S libovolným zvětšením, zmenšením či posunem prvku je vyhodnocena nová velikost a té je okno přizpůsobeno. Toto chování je dodrženo i v případě, že je prvek přidán nebo ubrán.

```
'Automatické zvětšování a zmenšování formuláře
Me.AutoSize = True
Me.AutoSizeMode = Windows.Forms.AutoSizeMode.GrowAndShrink

'Přidá prvek mimo rozsah stavající velikosti
Dim lbl As New Label()
lbl.Size = New Size(100, 30)
lbl.Location = New Point(500, 500)
lbl.Text = "Ahoj, Světe!"
Me.Controls.Add(lbl)

'Obnoví formulář
Me.Refresh()
```

298. Počáteční pozice okna uprostřed obrazovky

Přestože oknu nastavíte souřadnice do vlastnosti `Location`, tak je tato pozice ignorována, pokud máte nastavenu vlastnost `StartPosition`. Ta ovlivňuje, kam bude formulář po zobrazení umístěn, podle možností výčtového typu `FormStartPosition`.

```
Me.StartPosition = FormStartPosition.CenterScreen
```

299. Vlastní nastavení počáteční pozice okna

Pokud po přidání nového formuláře necháte původní nastavení, nastaví se pozice okna tak, jak uzná Windows za vhodné. To samozřejmě není vždy žádoucí a ani automatizované možnosti vlastnosti `StartPosition` nemusí stačit. Proto je nutné na počátku nastavit tuto vlastnost na `FormStartPosition.Manual`, která způsobí, že jsou brány do úvahy nastavené souřadnice okna.

```
Dim f As New Form2()
'Vypne automatizované nastavení
f.StartPosition = FormStartPosition.Manual
'Nastaví pozici
f.Location = New Point(100, 100)
'Zobrazí formulář
f.ShowDialog()
```

300. Vycentrování formuláře na obrazovce

Pokud chcete kdykoliv vystředit formulář a nechcete se spoléhat na jeho startovní pozici, máte možnost použít metodu `CenterToScreen`. Ta vystředí formulář na té obrazovce, na které se nachází jeho největší část.

```
Me.CenterToScreen()
```

301. Vycentrování formuláře v rodičovském formuláři

Pokud je formulář potomek, a tedy je obsažen uvnitř svého rodičovského formuláře, nemá smysl jej vystředit vůči obrazovce, což by mohlo vést i k tomu, že potomek zmizí mimo viditelný rámec svého rodiče. V takovou chvíli je nejlépe vystřeďovat vůči rodiči, a to metodou `CenterToParent`.

```
Me.CenterToParent()
```

302. Pohyb okna v ose Z

Ačkoliv je uživatelské rozhraní Windows dvourozměrné, a má tedy viditelné jen osy X a Y má i rozměr třetí. Ten určuje pořadí oken vůči sobě, tedy v jakém pořadí jsou vykreslena a jak se vzájemně překrývají.

```
'Pošle okno dozadu
Me.SendToBack()
```

```
'Pošle okno dopředu
Me.BringToFront()
```

303. Plovoucí okno

Typickým použitím tohoto typu okna jsou nejrůznější nástrojové lišty. Ty se vždy vznášejí nad pracovní plochou a vyobrazení dalších oken je nepřekryje, tedy uživateli nezmizí z pracovní plochy. To zajistíte jednoduchým nastavením vlastnosti `TopMost`.

```
Me.TopMost = True
```

304. Okno plovoucí nad jiným oknem

Zajistit, aby okno plavalo jen vůči jednomu konkrétnímu, lze zajistit nastavením tzv. vlastníka. V případě plovoucího okna to znamená, že mu nastavíte vlastnost `Owner` na instanci okna pod ním. Zjednodušit si to můžete tak, že při volání metody `Show` jí jako parametr předáte vlastníka, a ta se již postará nejen o zobrazení, ale i o nastavení vlastnosti `Owner`.

```
Dim frmChild As New Form2
'Zobrazí okno s nastavením vlastníka
frmChild.Show(Me)
```

305. Seznam vlastněných oken

Pokud oknu nastavíte vlastníka, snadno jej získáme z vlastnosti `Owner`. Co ale v případě, že u vlastníka chcete zjistit, která všechna okna vlastní? Jistě by bylo možné projít všechna otevřená okna a u nich ověřovat nastavení této vlastnosti, ale to je poněkud kostrbaté. Formulář má totiž i vlastnost `OwnedForms`, která vrací pole vlastněných formulářů.

```
Array.ForEach(Me.OwnedForms, AddressOf Debug.WriteLine)
```

306. Zobrazení okna uvnitř jiného

Aniž by okno muselo být nutně MDI, je možné uvnitř něj zobrazit okno jiné. To se pak chová úplně stejně, jako by bylo uvnitř MDI okna, a můžete s ním zacházet jako s jakýmkoliv jiným prvkem na formuláři. Aby to bylo možné provést, musí být nastavena vlastnost `TopLevel` na `False`, tedy že rodičem není plocha, ale nějaké jiné okno.

```
Dim frmChild As New Form2
'Okno není nejvyšší úrovně
frmChild.TopLevel = False
'Nastavení rodičovského okna
frmChild.Parent = Me
'Zobrazení okna uvnitř jiného
frmChild.Show()
```

307. Seznam otevřených oken

Každé okno, které ve vaší aplikaci vytvoříte, je uloženo v jedné centrální kolekci, jejíž prvky si kdykoliv můžete procházet. Jelikož jde o informaci vztahující se k celé aplikaci, najdete ji ve třídě `Application` v metodě `OpenForms`.

```
For Each f As Form In Application.OpenForms()
  Debug.WriteLine(f.ToString())
Next
```

308. Aktivní okno v aplikaci

Každá aplikace, která pracuje s formuláři, může mít v jednu chvíli jedno jediné okno označené jako aktivní, tedy takové, které je v danou chvíli přístupné uživateli k bezprostřední práci. Tento formulář získáte z třídní vlastnosti `ActiveForm` třídy `Form`.

```
Debug.WriteLine(Form.ActiveForm.Text)
```

309. Skrytí okna z lišty programů

Implicitně je každé okno, které přidáte do projektu, viditelné v liště programů. To ovšem není vždy žádoucí, takže pokud chcete zajistit, aby program reprezentovalo pouze jedno okno a zbývající se tam nevyskytovala, nastavte jim vlastnost `ShowInTaskbar` na `False`.

```
Me.ShowInTaskbar = False
```

310. Skrytí ikony formuláře

V levém horním rohu formuláře má většina formulářů ikonu, za kterou se schovává ovládací menu samotného formuláře. V něm se nalézají základní funkce pro změnu pozice, zvětšení apod. Tuto ikonu můžete schovat, aniž byste přišli o ovládací menu, a to tak, že nastavíte vlastnost `ShowIcon` na `False`.

```
Me.ShowIcon = False
```

311. Schování titulku okna 1

Titulek okna včetně nápisu a ovládacích prvků pro zavření, minimalizaci a maximalizaci je součástí okraje okna. Pokud schováte okraj, zmizí s ním i titulek okna. Tím samozřejmě přijdete o možnost uživatelsky měnit velikost.

```
'Skryje okraj formuláře
Me.FormBorderStyle = Windows.Forms.FormBorderStyle.None
```

312. Schování titulku okna 2

Zachovat okraj a formuláře a přitom schovat titulek formuláře můžete tak, že postupně schováte všechny ovládací prvky a text titulku nastavíte na prázdný řetězec.

```
Me.MinimizeBox = False
Me.MaximizeBox = False
Me.ControlBox = False
Me.Text = ""
```

313. Změna kurzoru na formuláři

Možnosti, jak může vypadat kurzor myši, nejsou omezeny jen na šipku a textový kurzor. Výčtový typ `Cursors` jich nabízí širokou škálu a jednu z těchto možností můžete nastavit vlastností formuláře `Cursor`. Pokud pak přes něj přejíždíte myší, změní se jeho kurzor. To platí i pro všechny podřízené prvky, které nemají tuto vlastnost nastavenu, respektive ji mají nastavenu na hodnotu `Cursors.Default`.

314. Aktivace dalšího prvku na formuláři

Pořadí, v jakém se uživatel přepíná mezi jednotlivými prvky na formuláři, je dáno vlastností `Tabindex`. Tou jsou očíslovány veškeré prvky v daném kontejneru a prvky v každém kontejneru mají svá vlastní číslování. Pokud chcete simulovat přepínání mezi nimi, můžete to sice provést vyvoláním klávesy `Tab`, ale mnohem čistší je použití metody `SelectNextControl`.

Výhodou této metody je to, že jste schopni detailně ovlivnit způsob přepnutí, a to nejen směr, ale i způsob vnoření u kontejnerů nebo zohlednění prvků, které jsou běžně z pořadí vyřazeny vlastností `TabStop`.

```
'Přepne se na další prvek pouze v rámci kontejneru
Me.SelectNextControl(Me.ActiveControl, True, True, False, True)

'Přepne se na předchozí prvek - zohlední podřízené prvky
Me.SelectNextControl(Me.ActiveControl, False, True, True, True)
```

315. Získání následujícího prvku v pořadí

Pokud chcete zjistit, jaký prvek bude následovat poté, co uživatel zmáčkne klávesu `Tab`, nemusíte se na něj přepínat a teprve poté provést ověření, ale zavoláte metodu `GetNextControl`, která vám jej vrátí.

```
'Získá další prvek
Dim ctl As Control
ctl = Me.GetNextControl(Me.ActiveControl, True)

'Vypíše jeho jméno
Debug.WriteLine(ctl.Name)
```

Pokud použijete v druhém parametru `False`, získáte prvek předchozí.

316. Zobrazení přesýpacích hodin

Při jakékoliv delší akci, která není na první pohled uživateli patrná, je vhodné zobrazit známé přesýpací hodiny. Ty jasně dají najevo, že se něco děje a že formulář tak říkajíc nezamrzl. Toho lze dosáhnout pouhým nastavováním vlastnosti `UseWaitCursor`.

```
'Zobrazí čekající kurzor
Me.UseWaitCursor = True

'Provedení dlouho trvající operace
'...

'Schová čekající kurzor
Me.UseWaitCursor = False
```

317. Zamezení uzavření okna

Pokud chcete zabránit tomu, aby byl formulář uzavřen, použijte událost `FormClosing`. Ta ve svém parametru typu `FormClosingEventArgs` nese vlastnost `Cancel`, po jejímž nastavení na `True` informujete okno, že se zavírání ruší.

```
Private Sub Form2_FormClosing(ByVal sender As Object, _
ByVal e As System.Windows.Forms.FormClosingEventArgs) _
Handles Me.FormClosing
  e.Cancel = True
End Sub
```

318. Schování okna místo zavření

Ať už uživatel, metoda `Close` či jiné akce uzavřou formulář, nemáte poté již žádnou možnost přistupovat k prvkům ve stavu před uzavřením. Abyste toto chování obešli, nezbývá vám než si odchytit událost zavření formuláře, zavírání přerušit a nahradit jej skrytím.

```
Private Sub Form2_FormClosing(ByVal sender As Object, _
ByVal e As System.Windows.Forms.FormClosingEventArgs) _
Handles Me.FormClosing
  'Zruší akci zavření
  e.Cancel = True
  'Formulář jen skryje
  Me.Hide()
End Sub
```

319. Důvod uzavření okna

Ačkoliv je uzavření formuláře jasnou známkou toho, co se děje, nebývá někdy na škodu vědět, co jej způsobilo, zdali uživatel nebo třeba ukončení celé aplikace. Tuto informaci získáte ve dvou událostech, a to `FormClosed` anebo `FormClosing`. Obě sice mají jiný typ

parametru `e`, ale oba obsahují vlastnost `CloseReason`. Ta vrací jednu z hodnot výčtového typu `CloseReason`, což jsou možné důvody uzavření formuláře.

```
Private Sub Form2_FormClosing(ByVal sender As Object, _
ByVal e As System.Windows.Forms.FormClosingEventArgs) _
Handles Me.FormClosing
  If e.CloseReason = CloseReason.UserClosing Then
    MessageBox.Show("Okno uzavřel uživatel")
  End If
End Sub
```

320. Okno, jehož velikost nelze měnit

To, jestli uživatel může nebo nemůže měnit velikost okna, je při standardním postupu dáno typem okraje. Ten buď je zvětšovatelný (`Sizable`) nebo není (`Fixed`). Nastavením vlastnosti `FormBorderStyle` na jeden z nabízených typů tak určíte, zdali velikost bude možné měnit.

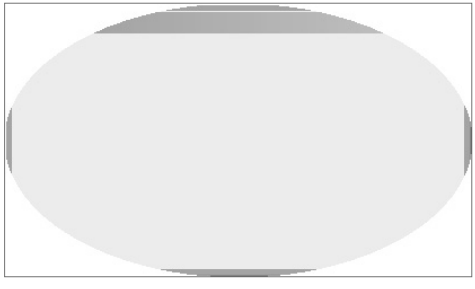
```
Me.FormBorderStyle = FormBorderStyle.Fixed3D
```

321. Eliptický tvar okna

Tvar většiny oken je obdélníkový, ale pokud chcete uživatele ohromit netuctovým vzhledem svého formuláře, můžete si jeho tvar nastavit. Toto nastavení má ten drobný háček, že budete ořezávat klasický obdélníkový tvar, což se projeví například jen částečnou viditelností titulku okna.

```
'Vytvoření elipsy vymezené velikostí formuláře
Dim gp As New Drawing2D.GraphicsPath()
gp.AddEllipse(New Rectangle(0, 0, Me.Width, Me.Height))

'Vytvoření regionu a přiřazení formuláři
Dim reg As New Region(gp)
Me.Region = reg
```

Vlastnost, která určuje tvar, je `Region`, kterou naplníte proměnnou stejnojmenného typu.



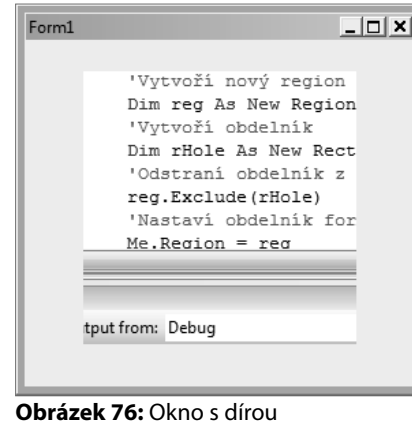
Obrázek 75: Eliptický tvar okna

322. Okno s dírou

Tak jako nastavíte vnější tvar formuláře, je možné i jeho části odstraňovat. Třída `Region` pro tento účel disponuje metodou `Exclude`, které předáte požadovaný tvar otvoru v podobě obdélníku či univerzálně tvarovatelného grafického prvku `GraphicsPath`.

```
'Vytvoří nový region - implicitně nekonečný
Dim reg As New Region()
'Vytvoří obdélník
Dim rHole As New Rectangle(50, 50, 200, 200)
'Odstraní obdélník z regionu
reg.Exclude(rHole)
'Nastaví obdélník formuláři
Me.Region = reg
```

Ačkoliv je třída `Region` po své bezparametrické inicializaci nekonečná, neznamená to, že by se tím stal i formulář nekonečně velký. Ten je totiž limitován i svou vlastní velikostí.



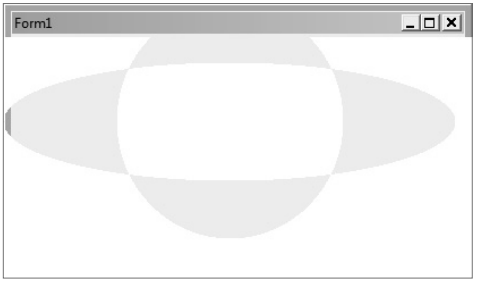
Obrázek 76: Okno s dírou

323. Složitý tvar okna

Třída `Region` nabízí velkou škálu možností, jak tvar formuláře poskládat. Nad nadefinovanými plochami v podobě tříd `GraphicsPath`, `Rectangle` či `Region` můžete provádět operace spojení (`Union`), průnik (`Intersect`) atd. Výsledek na závěr nastavíte vlastností `Region`.

```
'Složitý tvar okna
Dim r1, r2 As Rectangle
With Me.ClientRectangle
  r1 = New Rectangle(.Width / 4, 0, .Width / 2, .Height)
  r2 = New Rectangle(0, .Height / 4, .Width, .Height / 2)
End With

'Vytvoří první elipsu
Dim gp1 As New Drawing2D.GraphicsPath()
gp1.AddEllipse(r1)

'Vytvoří druhou elipsu
Dim gp2 As New Drawing2D.GraphicsPath()
gp2.AddEllipse(r2)

'Vytvoří nový region - implicitně nekonečný
Dim reg As New Region(gp1)
'Provede xor nad oběma elipsami
```

```
reg.Xor(gp2)
'Přidá obdélník horní části okna
reg.Union(New Rectangle(0, 0, Me.Width, 30))

'Nastaví obdélník oknu
Me.Region = reg
```

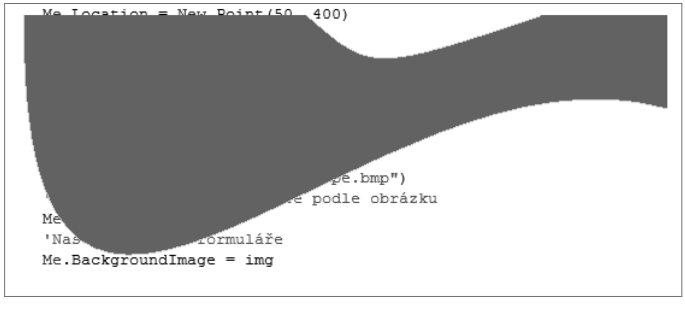


Obrázek 77: Složitý tvar okna

324. Tvar okna podle obrázku

Tvar formuláře nemusíte konstruovat jen za chodu programu, ale můžete si jej připravit pomocí obrázku v libovolném grafickém editoru. Celý trik pak spočívá v tom, že jednu z barev zvolíte jako průhlednou a zbytek obrázku pak určí tvar formuláře. Před tím je ovšem nezbytné zrušit oknu okraje a titulek.

```
'Schová titulek a okraje
Me.FormBorderStyle = Windows.Forms.FormBorderStyle.None
'Bílá barva z obrázku bude průhledná
Me.TransparencyKey = Color.White
'Načte obrázek
Dim img As New Bitmap("FormShape.bmp")
'Nastaví velikost formuláře podle obrázku
Me.Size = img.Size
'Nastaví pozadí formuláře
Me.BackgroundImage = img
```



Obrázek 78: Tvar okna podle obrázku

325. Skleněný efekt okna 1

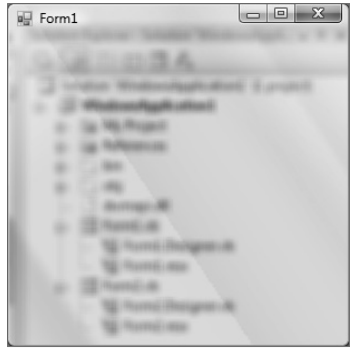
Ve Windows Vista bylo uvedeno nové uživatelské rozhraní Aero. V tom mají okna poloprůhledný okraj, za kterým je vidět rozmazané pozadí. Tento efekt je možné rozšířit i na zbývající plochu formuláře. Pro jeho aktivaci si nevystačíte s knihovnami .NET a budete muset použít Windows API.

```
Public Structure MARGINS
```

```
  Public Left As Integer
  Public Right As Integer
  Public Top As Integer
  Public Buttom As Integer
End Structure

'Nastaví skleněný efekt
Private Declare Function DwmExtendFrameIntoClientArea Lib _
  "dwmapi.dll" (ByVal hWnd As IntPtr, ByRef pMarinset As MARGINS)_
As Integer
'Ověří, zdali je možné efekt použít
Private Declare Function DwmIsCompositionEnabled Lib "dwmapi.dll" _
  (ByRef pfEnabled As Integer) As Integer
'Aero je zapnuto
Public mAeroEnabled As Boolean
```

Skleněný efekt bude aplikován na místa v černé barvě, takže součástí následujícího kódu, který umístěte do události `Load` formuláře, je i nastavení pozadí na černou.

```
'Ověří zdali je zapnuto Aero
Dim pfEnabled As Integer
DwmIsCompositionEnabled(pfEnabled)
mAeroEnabled = (pfEnabled = 1)

'Zkontroluje zapnutí Aero
If mAeroEnabled Then
  'Nastaví černou barvu na celý formulář
  Me.BackColor = Color.Black

  'Velikost efektu bez omezení
  Dim margins As MARGINS = New MARGINS() _
    With {.left = -1, .top = -1, .right = -1, .Buttom = -1}

  'Nastaví skleněný efekt
  DwmExtendFrameIntoClientArea(Me.Handle, margins)
End If
```



Obrázek 79: Skleněný efekt okna 1

326. Skleněný efekt okna 2

Na rozdíl od předchozího tipu nebude efektu dosaženo rozšířením stylu okraje do klientské oblasti, ale o samostatnou aplikaci skleněného efektu. Viditelný rozdíl bude v tom, že odstín klientské oblasti bude jiný a okraj bude ohraničen linkou.

```
Private Structure DwmBlurBehind
  Public Flags As Integer
  Public Enable As Boolean
  Public RgnBlur As System.IntPtr
  Public TransitionOnMaximized As Boolean
End Structure
'Ověří, zdali je možné efekt použít
Private Declare Function DwmIsCompositionEnabled Lib "dwmapi.dll" _
  (ByRef pfEnabled As Integer) As Integer
'Nastaví skleněný efekt
Private Declare Sub DwmEnableBlurBehindWindow Lib "dwmapi.dll" _
  (ByVal hWnd As IntPtr, ByRef BlurBehind As DwmBlurBehind)
'Aero je zapnuto
Public mAeroEnabled As Boolean
```

Efekt je aplikován jednak na černou barvu pozadí a také na zvolený region, který můžete nastavit na libovolný tvar. Tak jste schopni velmi detailně určovat, kde se průhlednost projeví.

```
'Ověří, zdali je zapnuto Aero
Dim pfEnabled As Integer
DwmIsCompositionEnabled(pfEnabled)
mAeroEnabled = (pfEnabled = 1)
'Zkontorluje zapnutí Aero
If mAeroEnabled Then
  'Nastaví černou barvu na celý formulář
  Me.BackColor = Color.Black

  'Nastavení parametrů efektu
  Dim dbb As New DwmBlurBehind()
  dbb.Flags = DwmBlurBehindFlags.DwmBbEnable Or _
    DwmBlurBehindFlags.DwmBbBlurRegion
  dbb.Enable = True
  dbb.TransitionOnMaximized = False

  'Region pro aplikace - implicitně nekonečný
  Dim reg As New Region
  dbb.RgnBlur = reg.GetHrgn(Me.CreateGraphics())

  'Nastavení efektu
  DwmEnableBlurBehindWindow(Me.Handle, dbb)
End If
```

Obrázek 80: Skleněný efekt okna 2

327. Skleněný efekt výřezu okna

Jak bylo ukázáno v předchozích dvou tipech, je jednou z podmínek pro aplikaci skleněného efektu černá barva. Tu pokud chcete aplikovat jen na vybrané oblasti, můžete toho dosáhnout tak, že touto barvou pokreslíte jen potřebné části. Jelikož obě metody pracují na stejném principu, stačí do obou předchozích tipů doplnit následující metodu a odstranit z události `Load` nastavení pozadí na černou barvu.

```
Protected Overrides Sub OnPaintBackground_
  (ByVal e As System.Windows.Forms.PaintEventArgs)
  'Zavolá metodu základní třídy
  MyBase.OnPaintBackground(e)

  If mAeroEnabled Then
    With Me.ClientRectangle
      'Vytvoří obdélník uprostřed formuláře
      Dim r As New Rectangle_
        (.Width / 4, .Height / 4, .Width / 2, .Height / 2)
      'Vykreslí černý obdélník
      e.Graphics.FillRectangle(Brushes.Black, r)
    End With
  End If
End Sub
```

Jelikož se výřez dynamicky mění se změnou velikosti formuláře, tak je nezbytné pro hladký chod přidat následující řádek na konec události `Load`, ten totiž zajistí korektní překreslení při změně velikosti.

```
Me.SetStyle(ControlStyles.ResizeRedraw, True)
```

328. Průhledné okno

Aby byla zajištěna úplná průhlednost okna, je nutné stanovit barvu, která bude nahrazena průhledností. Tuto barvu nastavíte jako barvu pozadí a poté jí přiřadíte vlastnosti

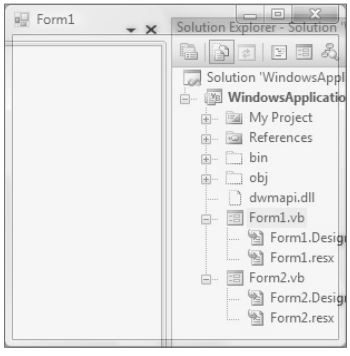
`TransparencyColor`. Samozřejmě vám nic nebrání použít stávající barvu pozadí, jak je vidět v příkladu. Průhlednost se týká jen vnitřního prostoru formuláře, tedy bez titulku a okrajů.

```
'Průhledné okno
Me.TransparencyKey = Me.BackColor
```

329. Poloprůhledné okno

Míru průhlednosti formuláře, a to nejen vnitřního prostoru, ale i titulku a okrajů, vám umožní nastavit vlastnost `Opacity`. Ta umožní zvolit libovolnou hodnotu mezi 0 (úplná průhlednost) a 1 (neprůhlednost).

```
Me.Opacity = 0.5
```



Obrázek 81: Poloprůhledné okno

330. Efekt roztažení a stažení okna 1

V tomto příkladu uvidíte, jak vytvořit okno, které se po zavolání této akce plynule roztáhne anebo stáhne do požadované velikosti a vytvoří tak zajímavý efekt.

```
'Roztažení a stažení okna
Private Sub ExpandWindow(ByVal destSize As Size)
  Dim sizeStep As Size
  'Získá rozdíl mezi stávající a původní velikostí
  sizeStep = Size.Subtract(destSize, Me.Size)

  'Vytvoří zlomkovou velikost
  sizeStep = New Size(sizeStep.Width / 100, sizeStep.Height / 100)

  'Postupně mění velikost
  For i = 1 To 100
    Me.Size = Size.Add(Me.Size, sizeStep)
    'Průběžně překresluje obsah okna
    Me.Refresh()
    'Chvíli počká
```

```
    Threading.Thread.Sleep(5)
  Next i
End Sub
```

Pokud chcete změnit rychlost změny velikosti, můžete upravit parametr metody `Sleep`. Samotné volání pak vypadá následovně, parametr je cílová velikost:

```
ExpandWindow(New Size(500, 500))
```

331. Efekt roztažení a stažení okna 1

Animaci okna v podobě roztažení umožňuje provést funkce `AnimateWindow` z Windows API. Její použití je velmi jednoduché, kromě `Handle` okna jí předáte dobu trvání celé animace a příznaky, jak má být animace provedena. První `AW_CENTER` příznak určuje, že půjde o animaci od středu, a druhý určí, zdali bude okno zobrazeno (`AW_ACTIVATE`) nebo schováno (`AW_HIDE`).

```
<Runtime.InteropServices.DllImport("user32.dll")> _
Shared Function AnimateWindow(ByVal hwnd As IntPtr, _
  ByVal time As Integer, ByVal flags As Integer) As Integer
End Function

Const AW_CENTER As Integer = &H10
Const AW_HIDE As Integer = &H10000
Const AW_ACTIVATE As Integer = &H20000

Private Sub ShowWindowRollCentral(ByVal show As Boolean)
  Dim flag As Integer = AW_CENTER
  If Not show Then
    flag += AW_HIDE
  Else
    flag += AW_ACTIVATE
  End If

  AnimateWindow(Me.Handle, 500, flag)
End Sub
```

332. Efekt stmívání a rozsvěcení okna 1

Tento efekt můžete znát například z menu, pokud je má uživatelské prostředí Windows nastaveno. Okno se postupně objeví nebo naopak zmizí za použití vlastnosti `Opacity`, která řídí průhlednost okna.

```
Private Sub OpacityWindow(ByVal destOpacity As Double)
  Dim opStep As Double
  'Nastaví krok změny průhlednosti
  opStep = (destOpacity - Me.Opacity) / 10

  'Postupně mění průhlednost
```

```
  For i As Integer = 1 To 10
    Me.Opacity += opStep
    'Průběžně překresluje obsah okna
    Me.Refresh()
    'Chvíli počká
    Threading.Thread.Sleep(50)
  Next
End Sub
```

Posledním problémem je správné umístění volání této metody. Nejvhodnějšími událostmi jsou `Shown` a `Closing`.

```
'ztmavení po zavření okna
Private Sub Form1_FormClosing(ByVal sender As Object, _
ByVal e As System.Windows.Forms.FormClosingEventArgs) _
Handles Me.FormClosing
  OpacityWindow(0)
End Sub

'Rozsvícení po zobrazení okna
Private Sub Form1_Shown(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles Me.Shown
  Me.Opacity = 0
  OpacityWindow(1)
End Sub
```

333. Efekt stmívání a rozsvěcení okna 2

Tento efekt můžete provést také pomocí Windows API. Stačí zavolat funkci `AnimateWindow` a ta se postará o vše potřebné. Té musíte předat typ efektu `AW_BLEND` a nezbytný směr animace, tedy `AW_ACTIVATE` pro zobrazení a `AW_HIDE` pro skrytí okna.

```
Const AW_HIDE As Integer = &H10000
Const AW_ACTIVATE As Integer = &H20000
Const AW_BLEND As Integer = &H80000

<Runtime.InteropServices.DllImport("user32.dll")> _
Shared Function AnimateWindow(ByVal hwnd As IntPtr, _
  ByVal time As Integer,   ByVal flags As Integer) As Integer
End Function

Private Sub ShowWindowBlend(ByVal show As Boolean)
  Dim flag As Integer = AW_BLEND
  If Not show Then
    flag += AW_HIDE
  Else
    flag += AW_ACTIVATE
  End If
```

```
  AnimateWindow(Me.Handle, 1000, flag)
End Sub
```

334. Efekt vyklouznutí okna

Na rozdíl od roztažení okna je efekt vyklouznutí proveden tak, jako by se do levého horního rohu okna zobrazil pravý dolní roh formuláře a ten se postupně posouval dál ve zvětšujícím se rámci viditelného výřezu. Tento efekt provedete pomocí funkce `AnimateWindow` z Windows API. Při zobrazování (`AW_ACTIVATE`) a skrývání (`AW_HIDE`) je nutné obrátit i směr animace, aby okno zaklouzlo zpátky tam, odkud vyklouzlo.

```
Const AW_HOR_POSITIVE As Integer = &H1
Const AW_HOR_NEGATIVE As Integer = &H2
Const AW_VER_POSITIVE As Integer = &H4
Const AW_VER_NEGATIVE As Integer = &H8
Const AW_HIDE As Integer = &H10000
Const AW_ACTIVATE As Integer = &H20000
Const AW_SLIDE As Integer = &H40000

<Runtime.InteropServices.DllImport("user32.dll")> _
Shared Function AnimateWindow(ByVal hwnd As IntPtr, _
  ByVal time As Integer, ByVal flags As Integer) As Integer
End Function

  Private Sub ShowWindowSlide(ByVal show As Boolean)
  Dim flag As Integer = AW_SLIDE
  If Not show Then
    flag += AW_HIDE + AW_VER_NEGATIVE + AW_HOR_NEGATIVE
  Else
    flag += AW_ACTIVATE + AW_VER_POSITIVE + AW_HOR_POSITIVE
  End If

  AnimateWindow(Me.Handle, 500, flag)
End Sub
```

335. Zablikání okna

Upozornit uživatele na to, že v aplikaci nastala nějaká významná událost, můžete například zablikáním titulku okna a lišty úloh. Bohužel .NET Framework nemá pro tuto funkčnost přímou podporu, a tak to můžete realizovat pomocí Windows API.

```
Private Structure FLASHWINFO
  Public cbSize As Integer
  Public hwnd As IntPtr
  Public dwFlags As Integer
  Public uCount As Integer
  Public dwTimeout As Integer
End Structure
```

```
Private Declare Function FlashWindowEx Lib "user32.dll" _
  (ByRef pfwi As FLASHWINFO) As Integer

Private Const FLASHW_ALL = &H3
```

Všechny informace o tom, co a jak má blikat, předáte pomocí struktury `FLASHWINFO`. Ta nese informaci o `Handle` okna (`hwnd`), o počtu bliknutí (`uCount`) či rychlosti blikání (`dwTimeout`).

```
'Zablikání okna
Dim fi As New FLASHWINFO
fi.cbSize = System.Runtime.InteropServices.Marshal.SizeOf(fi)
'Handle okna
fi.hwnd = Me.Handle
'Bude blikat jak okno, tak i lišta úloh
fi.dwFlags = FLASHW_ALL
'Počet bliknutí
fi.uCount = 5
'Rychlost blikání - 0 implicitní
fi.dwTimeout = 0
'Vyvolání blikání
FlashWindowEx(fi)
```

336. Blikání okna do jeho aktivace

Blikání okna se dá funkcí API `FlashWindowEx` na neomezeně dlouhou dobu. Aby to mělo význam, je třeba ve vhodném momentu blikání zastavit. Pro upozornění uživatele bude dostačující, když blikání skončí ve chvíli, kdy je okno aktivováno.

```
Private Structure FLASHWINFO
  Public cbSize As Integer
  Public hwnd As IntPtr
  Public dwFlags As Integer
  Public uCount As Integer
  Public dwTimeout As Integer
End Structure

Private Declare Function FlashWindowEx Lib "user32.dll" _
  (ByRef pfwi As FLASHWINFO) As Integer

Private Const FLASHW_ALL = &H3
Private Const FLASHW_STOP = &H0
Private Const FLASHW_TIMER = &H4

Private Sub Form3_Activated(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles Me.Activated
  Dim fi As New FLASHWINFO
  fi.cbSize = System.Runtime.InteropServices.Marshal.SizeOf(fi)
  'Handle okna
```

```
  fi.hwnd = Me.Owner.Handle
  'Zastaví blikání
  fi.dwFlags = FLASHW_STOP
  FlashWindowEx(fi)
End Sub

Private Sub Form2_Shown(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles Me.Shown
  'Blikání okna do jeho aktivace
  Dim fi As New FLASHWINFO
  fi.cbSize = System.Runtime.InteropServices.Marshal.SizeOf(fi)
  'Handle okna
  fi.hwnd = Me.Owner.Handle
  'Bude blikat jak okno, tak i lišta úloh
  fi.dwFlags = FLASHW_ALL Or FLASHW_TIMER
  'Vyvolání blikání
  FlashWindowEx(fi)
End Sub
```

337. Získání Handle okna

Pro použití funkcí Windows API ve spojitosti s formuláři je velmi důležitý jeden parametr, který je označován jako `Handle` okna (často zkracován jako `hWnd`). Jde o jednoznačný identifikátor okna ve Windows. Pokud tedy předáváte funkci API odkaz na okno, není to samozřejmě v podobě instance třídy `Form`, ale právě tento identifikátor.

```
Debug.WriteLine(Me.Handle)
```

338. Zvětšení formuláře kolečkem myši

Dnes už je běžným vybavením každé myši rolovací kolečko, které jednoduchým způsobem umožňuje pohybovat se stránkou nebo měnit měřítko apod. Pracovat s ním vám umožní událost `MouseWheel`, jež má z tohoto pohledu jediný důležitý parametr, a tím je `e.Delta`. Ten obsahuje informaci, o kolik bylo kolečko pootočeno.

```
Private Sub Form1_MouseWheel(ByVal sender As Object, _
ByVal e As System.Windows.Forms.MouseEventArgs) Handles Me.MouseWheel
  Me.Size = New Size_
    (Me.Size.Width + e.Delta / 10, Me.Size.Width + e.Delta / 10)
End Sub
```

339. Test Control, Alt a Shift při klepnutí myši

Pokud zachytáváte některou z událostí myši, nemáte možnost přímo z parametrů události zjistit, jaké klávesy jsou stisknuty. To zjistíte z jiných událostí, což je vám v tuto chvíli k ničemu. Můžete si sice ukládat stav kláves a ten poté použít při obsluze událos-

tí myši, ale to není úplně elegantní. Tento problém vyřeší třída `Form` svou statickou metodou `ModifierKeys`, která umí indikovat stisknutí kláves Control, Alt a Shift.

```
Private Sub Form1_MouseClick(ByVal sender As Object, ByVal e_
As System.Windows.Forms.MouseEventArgs) Handles Me.MouseClick
  Dim modKeys As Keys
  'Klávesy pro test
  modKeys = Keys.Alt + Keys.Shift + Keys.Control
  'Test na stisknuté Control, Alt a Shift
  If (Form.ModifierKeys And modKeys) = modKeys Then
    MessageBox.Show _
      ("Při klepnutí bylo stisknuto Control, Alt a Shift")
  End If
End Sub
```

340. Test tlačítek myši při stisknutí klávesy

Podobný problém při testování zmáčknutých kláves v událostech myši platí i v obráceném gardu. Jak zjistíte stav tlačítek myši, když jste v události vyvolaných klávesnicí? Třída `Form` má statickou metodu `MouseButtons`, která vrací jejich stav platný v danou chvíli.

```
Private Sub Form1_KeyPress(ByVal sender As Object, _
ByVal e As System.Windows.Forms.KeyPressEventArgs) Handles Me.KeyPress
  'Test na zmáčknuté prostřední tlačítko myši
  If (Form.MouseButtons And Windows.Forms.MouseButtons.Middle) _
    > 0 Then
    MessageBox.Show _
      ("Při zmáčknutí klávesy bylo drženo prostřední tlačítko myši")
  End If
End Sub
```

341. Zachycení kláves na celém formuláři

V běžném případě dostane informaci o zmáčknuté klávese vždy ten prvek, na kterém tato událost nastane. To je samozřejmě limitující ve chvíli, kdy potřebujete zpracovávat některé klávesové zkratky v rámci celého formuláře. Pro tento případ slouží vlastnost `KeyPreview`, která zajistí, že nejdříve bude událost vyvolána nad formulářem a teprve poté nad podřízenými prvky.

```
Me.KeyPreview = True
```

Jak to funguje si můžete vyzkoušet například tak, že si na formulář umístíte prvek `TextBox` a budete zachytávat události `KeyDown` nad textovým polem i formulářem.

```
Private Sub TextBox1_KeyDown(ByVal sender As Object, _
ByVal e As System.Windows.Forms.KeyEventArgs) Handles TextBox1.KeyDown
  Dim ctl As Control
  ctl = CType(sender, Control)
  'Vypíše klávesu a prvek
```

```
  Debug.WriteLine("TextBox1_KeyDown:" & ctl.Name & "," & e.KeyData)
End Sub

Private Sub Form_KeyDown(ByVal sender As Object, _
ByVal e As System.Windows.Forms.KeyEventArgs) Handles Me.KeyDown
  Dim ctl As Control
  ctl = CType(sender, Control)
  'Vypíše klávesu a prvek
  Debug.WriteLine("Form_KeyDown:" & ctl.Name & "," & e.KeyCode)
End Sub
```

342. Filtrování kláves na úrovni formuláře

Prvním krokem, který musíte zajistit, aby bylo možné zpracovat zmáčknutí klávesy, je nastavit vlastnost `KeyPreview` na `True`. Ta způsobí, že všechny klávesy nejdříve projdou formulářem a teprve poté podřízenými prvky. A o ty v tomto tipu půjde především. Pokud totiž parametru `e` události `KeyDown` nastavíte do vlastnosti `SuppressKeyPress`, podřízené prvky se už o ní nedoví, a tak můžete filtrovat, které klávesy dostanou a které ne.

```
Private Sub Form_KeyDown(ByVal sender As Object, ByVal e_
As System.Windows.Forms.KeyEventArgs) Handles Me.KeyDown
  'Pravou a levou šipku podřízené prvky nedostanou
  If e.KeyCode = Keys.Right Or e.KeyCode = Keys.Left Then
    e.SuppressKeyPress = True
  End If
End Sub
```

343. Klávesa Enter jako tabulátor

Aby klávesa Enter fungovala stejně jako tabulátor pro všechny prvky, musíte nejdříve nastavit vlastnost formuláře `KeyPreview` na `False` a poté odchytávat událost `KeyDown` formuláře. V tu chvíli před vámi leží tři úkoly. Přepnout se na následující prvek, zamezit tomu, aby se klávesa dostala na podřízené prvky, a hlavně formuláři říct, že jste si ji obsloužili sami.

```
Private Sub Form_KeyDown(ByVal sender As Object, _
ByVal e As System.Windows.Forms.KeyEventArgs) Handles Me.KeyDown
  'Enter jako tabulátor
  If e.KeyCode = Keys.Enter Then
    'Přepne na další prvek
    Me.SelectNextControl(Me.ActiveControl, True, True, True, True)
    'Klávesa se nedostane na podřízené prvky
    e.SuppressKeyPress = True
    'Klávesu jste si obsloužili sami
    e.Handled = True
  End If
End Sub
```

344. Zvýraznění formuláře při pohybu myši přes něj

Aby bylo možné něco takového obsloužit, je zapotřebí mít informaci o tom, kdy kurzor myši vstoupí do rámce formuláře a kdy jej naopak opustí. Události, které vám tyto dvě situace oznámí, jsou `MouseEnter` a `MouseLeave`.

```
'Kurzor myši vstoupil na formulář
Private Sub Form1_MouseEnter(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles Me.MouseEnter
  Me.BackColor = Color.Red
End Sub

'Kurzor myši opustil formulář
Private Sub Form1_MouseLeave(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles Me.MouseLeave
  Me.BackColor = SystemColors.Window
End Sub
```

345. Posun okna bez titulkové lišty 1

Pokud dáte formuláři svůj tvar a zrušíte mu titulkovou lištu, narazíte na problém, jak uživateli i nadále umožnit pohybovat s formulářem. Možností, které nabízí přímo .NET Framework, je použití událostí myši `MouseDown` pro aktivaci a získání vztažného bodu a `MouseMove` pro posun.

```
'Pro uložení pozice ve chvíli zmáčknutí
Private mMousePosition As New Point(-1, -1)

Private Sub Form2_MouseDown(ByVal sender As Object, _
ByVal e As System.Windows.Forms.MouseEventArgs) Handles Me.MouseDown
  'Souřadnice kurzoru vzhledem k obrazovce
  mMousePosition = Me.PointToScreen(e.Location)
  'Odečtení pozice formuláře
  mMousePosition = Point.Subtract(mMousePosition, Me.Location)
End Sub

Private Sub Form2_MouseMove(ByVal sender As Object, _
ByVal e As System.Windows.Forms.MouseEventArgs) Handles Me.MouseMove
  If mMousePosition.X >= 0 Then
    'Aktuální pozice myši vůči obrazovce
    Dim tmpPosition As Point
    tmpPosition = Me.PointToScreen(e.Location)
    'Odečte původní pozici myši
    Me.Location = Point.Subtract(tmpPosition, mMousePosition)
  End If
End Sub

Private Sub Form2_MouseUp(ByVal sender As Object, _
```

```
ByVal e As System.Windows.Forms.MouseEventArgs) Handles Me.MouseUp
  'Deaktivuje posun formuláře
  mMousePosition = New Point(-1, -1)
End Sub
```

346. Posun okna bez titulkové lišty 2

Druhou možností, jak pohybovat oknem bez titulkové lišty, je zachytávat zprávy Windows. Tou zprávou, kterou musíte zachytit, je `WM_NCHITTEST`. Poté co zjistíte, že zachytila klepnutí do klientské oblasti, jí namluvíte, že došlo ke klepnutí na titulek. Okno se pak bude chvat tak jako při podržení a pohybu myši nad titulkovou lištou.

```
Protected Overrides Sub WndProc(ByRef m As Message)
  'Zamezení posunu okna
  Const WM_NCHITTEST As Integer = &H84
  Const HTCLIENT As Integer = &H1
  Const HTCAPTION As Integer = &H2

  'Zavolá základní třídu
  MyBase.WndProc(m)

  'Test klepnutí myši
  If m.Msg = WM_NCHITTEST Then
    'Pokud jde o klepnutí do uživatelského prostoru ...
    If m.Result = New IntPtr(HTCLIENT) Then
      '... potom předstírej klepnutí na titulek
      m.Result = New IntPtr(HTCAPTION)
    End If
  End If
End Sub
```

347. Zamezení posunu okna

Uživatelské operace s oknem jsou dány systémovou nabídkou, kterou naleznete po klepnutí na ikonu formuláře. Tyto příkazy po své aktivaci posílají zprávu `WM_SYSCOMMAND`. Parametrem této zprávy je druh operace, která byla zavolána. V případě, že to byl posun formuláře, vynulujte jej a tím k této akci nedojde.

```
Protected Overrides Sub WndProc(ByRef m As Message)
  Const WM_SYSCOMMAND As Integer = &H112
  Const SC_MOVE As Integer = &HF010

  'Příkaz systémového menu
  If m.Msg = WM_SYSCOMMAND Then
    'Akce posouvání
    If (m.WParam.ToInt32() And &HFFF0) = SC_MOVE Then
      'Zruší akci
      m.WParam = 0
```

```
    End If
  End If

  'Zavolá základní třídu
  MyBase.WndProc(m)
End Sub
```

348. WPF Zamezení změny velikosti okna

Na rozdíl od formulářů Windows Forms je oddělen ve WPF vzhled okraje a jeho funkčnost. Takže nastavení 3D okraje ještě nic nevypovídá o jeho schopnosti uživatelsky měnit rozměry. Tou správnou vlastností je `ResizeMode`.

```
<Window x:Class="Window2"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Window2" Height="300" Width="300"
    ResizeMode="NoResize" WindowStyle="ThreeDBorderWindow">
</Window>
```

349. Minimální a maximální velikost okna

Každé okno, u kterého je možné měnit velikost, není nijak velikostně limitováno. Abyste uživateli vymezili, do jaké míry může okno zvětšit nebo naopak zmenšit, máte k dispozici čtyři vlastnosti: `MinWidth` a `MaxWidth` pro minimální a maximální šířku a `MinHeight` a `MaxHeight` pro minimální a maximální výšku.

```
<Window x:Class="Window2"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Window2" Height="300" Width="300"
    MinWidth="150" MinHeight="150" MaxWidth="500" MaxHeight="500">
</Window>
```

350. WPF Efekt rozsvěcení okna

Rozsvícení okna nám provede třída `DoubleAnimation`. Ta dokáže v zadaném čase provést lineární změnu mezi dvěma hodnotami. Tuto akci navážeme na průhlednost, tedy vlastnost `Opacity`. Celá akce je spuštěna ve spoušti `Load` okna.

```
<Window x:Class="Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Window1" Height="300" Width="300" Name="Window1"
    AllowsTransparency="True" WindowStyle="None" Background="Blue">
  <Window.Triggers>
    <EventTrigger RoutedEvent="Window.Loaded">
      <BeginStoryboard>
```

```
        <Storyboard Name="FormFade">
          <DoubleAnimation Name="FadeAnimation"
              Storyboard.TargetName="Window1"
              Storyboard.TargetProperty="(Window.Opacity)"
              From="0.0" To="1.0" Duration="0:0:3"/>
        </Storyboard>
      </BeginStoryboard>
    </EventTrigger>
  </Window.Triggers>
</Window>
```

351. WPF Efekt roztažení okna 1

Následující efekt nám při zobrazení okna plynule roztáhne okno v obou rozměrech. Ve spoušti `Load` můžete vidět, že `Storyboard` má nastaveny dvě animace pomocí třídy `DoubleAnimation`. Obě mají stejnou délku trvání a tak se vlastnosti, které nastavují, budou měnit současně.

```
<Window x:Class="Window3"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Window1" Height="300" Width="300" Name="Window3">
  <Window.Triggers>
    <EventTrigger RoutedEvent="Window.Loaded">
      <BeginStoryboard>
        <Storyboard Name="FormRoll">
          <DoubleAnimation Name="RollAnimation1"
              Storyboard.TargetName="Window3"
              Storyboard.TargetProperty="(Window.Width)"
              From="10" To="500" Duration="0:0:1" />

          <DoubleAnimation Name="RollAnimation2"
              Storyboard.TargetName="Window3"
              Storyboard.TargetProperty="(Window.Height)"
              From="10" To="500" Duration="0:0:1" />
        </Storyboard>
      </BeginStoryboard>
    </EventTrigger>
  </Window.Triggers>
</Window>
```

352. WPF Efekt roztažení okna 2

V následujícím vizuálním efektu při otevírání okna proběhne nejdříve roztažení titulku a pak se z něj vysune směrem dolů zbytek okna. Aby obě změny probíhaly postupně, má třetí instance `DoubleAnimation` nastaven počátek spuštění až po doběhnutí té druhé,

a to pomocí vlastnosti `BeginTime`. První animace je zde vložena proto, aby na počátku bez prodlení změnila počáteční velikost.

```
<Window x:Class="Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Window1" Height="300" Width="300" Name="Window1">
  <Window.Triggers>
    <EventTrigger RoutedEvent="Window.Loaded">
      <BeginStoryboard>
        <Storyboard Name="FormRoll">
          <DoubleAnimation Name="RollAnimation0"
              Storyboard.TargetName="Window1"
              Storyboard.TargetProperty="(Window.Height)"
              From="10" To="10" Duration="0:0:0" />
          <DoubleAnimation Name="RollAnimation1"
              Storyboard.TargetName="Window1"
              Storyboard.TargetProperty="(Window.Width)"
              From="10" To="500" Duration="0:0:1" />
          <DoubleAnimation Name="RollAnimation2"
              Storyboard.TargetName="Window1"
              Storyboard.TargetProperty="(Window.Height)"
              From="10" To="500" Duration="0:0:1" BeginTime="00:00:01" />
        </Storyboard>
      </BeginStoryboard>
    </EventTrigger>
  </Window.Triggers>
</Window>
```