Metodický list k didaktickému materiálu

Číslo a název šablony	III/2 Inovace a zkvalitnění výuky prostřednictvím ICT
Číslo didaktického materiálu	EU-OPVK-VT-III/2-ŠR-309
Druh didaktického materiálu	DUM
Autor	RNDr. Václava Šrůtková
Jazyk	čeština
Téma sady didaktických materiálů	Programování v C# v příkladech III
Téma didaktického materiálu	Animace
Vyučovací předmět	Seminář z informatiky
Cílová skupina (ročník)	Žáci ve věku 17–18 let
Úroveň žáků	Středně pokročilí
Časový rozsah	1–2 vyučovací hodiny
Klíčová slova	Animace, časovač
Anotace	Studenti programují pohyb ovládacích prvků na obrazovce pomocí časovače a metody Thread.Sleep
Použité zdroje	VYSTAVĚL, Radek. *Moderní programování: sbírka úloh k učebnici pro středně pokročilé*. 1. vyd. Ondřejov: moderníProgramování, 2008-2009, 2 sv. ISBN 978-80-903951-3-8. VYSTAVĚL, Radek. *Moderní programování: učebnice pro středně pokročilé*. Ondřejov: moderníProgramování s.r.o, 2008. ISBN 978-80-903951-2-1.
Typy k metodickému postupu učitele, doporučené výukové metody, způsob hodnocení, typy k individualizované výuce apod.	Text je možno využít ke společné práci, samostatné přípravě studentů, domácímu studiu apod. Stručná prezentace umožňuje na začátku hodiny seznámení s problémem. Při společné práci je vhodné nejprve obtížnější úlohy rozebrat, potom společně se studenty implementovat na počítači. (Rozbor nejlépe na tabuli, synchronní řešení s promítáním) V pracovním listu je zadání cvičení – většinou se jedná o úlohy, které by měli studenti naprogramovat samostatně. Není nutné, aby všichni zpracovali všechno, vhodné je diferencovat podle jejich zájmu a schopností. Obtížnější úlohy jsou označeny hvězdičkou. Součástí materiálu je zdrojový kód těchto příkladů. Návrh způsobu hodnocení: ohodnocení samostatné práce během hodiny např. podle volby a počtu úloh a elaborace řešení (efektivnost, komentáře…).

Prohlášení autora

Tento materiál je originálním autorským dílem. K vytvoření tohoto didaktického materiálu nebyly použity žádné externí zdroje s výjimkou zdrojů citovaných v metodickém listu.

309. Animace

Základní princip je jednoduchý: Objekt vykreslíme, chvilku počkáme, posuneme o kousek dál atd. K posunutí „o kousek dál“ budeme potřebovat vypočítat nové souřadnice a tady se nám časem budou hodit znalosti z analytické geometrie, zatím vyzkoušíme pohyb ve vodorovném a svislém směru. Pro pozdržení běhu programu – čekání – můžeme využít metodu **Sleep** třídy **Thread** nebo časovač. (komponenta **Timer**) Naprogramujeme si obě možnosti.

Příklad 1.

Naprogramujte pohyb tlačítka ve vodorovném směru.

```
using System.Threading;
//je třeba připojit, abychom mohli pracovat s třídou Thread
namespace Animace1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void buttonJed_Click(object sender, EventArgs e)
        {
            int a = buttonJede.Left;
            //výchozí poloha běžícího tlačítka
            int b = ClientSize.Width - buttonJede.Width - 10;
            //koncová poloha
            //b-a počet bodů
            for (int i = a; i < b-a; i++)
            {
```

```
                buttonJede.Left++;
                Thread.Sleep(10);//10 milisekund čekání
            }

        }
    }
}
```

Poznámka: Pozici není nutno střídat po jednom bodu.

Hlavní nevýhodou metody Sleep je skutečnost, že pokud tento program běží, nedá se v systému dělat nic jiného.

Příklad 2.

Naprogramujte pohyb tlačítka ve svislém směru pomocí časovače, začátek a konec pohybu zajistí stisknutí tlačítka.

Časovač – Timer je komponenta, která generuje pravidelně událost popsanou ve své implicitní metodě **Tick** v časových intervalech zadaných ve vlastnosti **Interval.** Zapíná se a vypíná tak, že její vlastnost **Enabled** nastavíme na **True** nebo **False**.

Výhodou použití časovače je, že se program po dobu trvání animace nazablokuje, jak tomu bylo v minulém případě.

```
private void buttonCV_Click(object sender, EventArgs e)
        {
            timer1.Enabled = !timer1.Enabled;
            //střídavě zapínáme a vypíname timer
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            buttonSvisle.Top += 5;
        }
```

Příklad 3.

Umožníme uživateli nastavit rychlost pohybu tlačítka.

Pro malé změny můžeme zvětšit nebo zmenšit přírůstek polohy, jinak je vhodnější použít nastavení intervalu timeru. Rychlost můžeme nastavit např. pomocí komponenty **trackbar.** Minimum zvolíme 1, Maximum 10 a při implicitní události – **trackBar1_Scroll**, tedy pohyb jezdce, nastavíme interval Timeru.

```
        private void trackBar1_Scroll(object sender, EventArgs e)

        {

            timer1.Interval = 500/trackBar1.Value;

     //hodnota ve jmenovateli, aby rychlost pohybu rostla s tažením
    jezdce, jak bude uživatel patrně přepokládat.

        }
```

Příklad 4.

Když tlačítko doputuje k dolní části okna, začne stoupat, nahoře opět změní směr…až do vypnutí časovače.

Zde budeme potřebovat členskou proměnnou, do které uložíme posuv a která bude měnit znaménko v případě, že svisle se pohybující tlačítko dojede k některému okraji formuláře.

```
private void timer1_Tick(object sender, EventArgs e)

        {

            buttonSvisle.Top += y;

                 if ((buttonSvisle.Top > ClientSize.Height -
    buttonSvisle.Width+10)  ||

                (buttonSvisle.Top < 10))

                y = -y;


        }
```

Pokud bychom se chtěli zbavit blikání tlačítka při překreslování, můžeme použít zdvojení obrazové paměti – nastavíme vlastnost okna **DoubleBuffere**d okna na **True**.

Důležité

Pozastavení běhu programu na k milisekund: **Thread.Sleep(k),** nutno přidat jmenný prostor **Systém.Threading.**

Objekt **Timer** generuje událost popsanou v metodě **Tick** v čase zadaném ve vlastnosti **Interval.** Zapíná se a vypíná vlastností **Enabled**.

Pracovní list

Cvičení

1. (*) Naprogramujte tlačítko, které bude objíždět okno po obdélníku podél jeho okraje.

2. Naprogramujte postupnou změnu barvy tlačítka od bílé po černou a zase zpět.

3. Vymyslete a naprogramujte vlastní animaci nějakého objektu na formuláři.

Řešení

```
public partial class FormKolem : Form
    {

        bool doprava = true; //rozjíždíme se doprava z výchozí pozice
        bool doleva = false;
        bool dolu = false;
        bool nahoru = false;
        int x = 1; //přírůstek barevné složky
        int b = 122; //složka barvy r, g, b
        public FormKolem()
        {
            InitializeComponent();
        }

        private void FormKolem_Load(object sender, EventArgs e)
        {
            timerPohyb.Enabled=true;
            timerBarva.Enabled = true;

        }

        private void timer1_Tick(object sender, EventArgs e)
```

```
        {
            if (doprava)
            {
              buttonKolem.Left += 5;
              if(( buttonKolem.Left >500)&&(buttonKolem.Top<20))
                  //pravý horní roh
              {doleva=false;dolu=true;doprava=false;nahoru=false;}
            }
            else
                if (dolu)
                {
                 buttonKolem.Top += 5;
                 if(( buttonKolem.Top >220)&&(buttonKolem.Left>450))
                     //pravý dolní roh
                {doleva=true;dolu=false;doprava=false;nahoru=false;}
                }
                else
                    if(doleva)
                    {
                        buttonKolem.Left -= 5;
              if(( buttonKolem.Left <20)&&(buttonKolem.Top>220))
                  //levý dolní roh
               {doleva=false;dolu=false;doprava=false;nahoru=true;}
                    }
                    else
                        if (nahoru)
                            {
                 buttonKolem.Top -= 5;
                 if(( buttonKolem.Top <20)&&(buttonKolem.Left<20))
                     //levý horní roh
                 { doprava = true; dolu = false; doleva = false; nahoru =
false; }
```

```
            }


        }


    private void timerBarva_Tick(object sender, EventArgs e)
    {
        buttonBarva.BackColor = Color.FromArgb(b, b, b);
        b += x;
        if ((b >= 255) || (b <= 0))
        {
            x = -x;
        }
    }
```

3. Individuální řešení