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-310
Druh didaktického materiálu	DUM
Autor	RNDr. Václava Šrůtková
Jazyk	čeština
Téma sady didaktických materiálů	Programování v C# v příkladech III
Téma didaktického materiálu	Animace II
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	Rovnoměrný pohyb po přímce, po kružnici
Anotace	Studenti spojují poznatky z matematiky a fyziky a modelují rovnoměrný přímočarý a kruhový pohyb a dále procvičují princip animace
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. Při společné práci je vhodné nejprve obtížnější úlohy rozebrat, potom společně se studenty implementovat na počítači. (Rozbor nejlépe na tabuli, synchronní řešení s promítáním) Prezentace obsahuje stručné shrnutí poznatků potřebných pro řešení příkladů. 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.

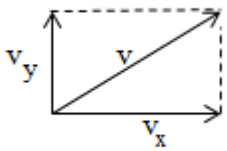
Obrázky (schémata a snímky obrazovek) pocházejí od autora.

310. Animace II

Modelování rovnoměrného přímočarého pohybu

Dosud jsme pohybovali objekty ve svislém nebo vodorovném směru. Asi je zřejmé, že při pohybu šikmo se budou muset měnit obě souřadnice.

Připomeňme si rovnoměrný přímočarý pohyb ve fyzice:
Poloha (dráha) hmotného bodu $s = s_0 + v\,t$
Rychlost obvykle rozkládáme na složky rovnoběžné s osami x a y, pak platí:



$x = x_0 + v_x t$

$y = y_0 + v_y t$

(Všimli jste si, že úplně stejně vypadá parametrická rovnice přímky v rovině, kde $[x_0, y_0]$ jsou souřadnice výchozího bodu a (v_x, v_y) souřadnice směrového vektoru přímky?)

Analogicky budeme postupovat při programování, budeme pohybovat vyplněným kolečkem – kuličkou.

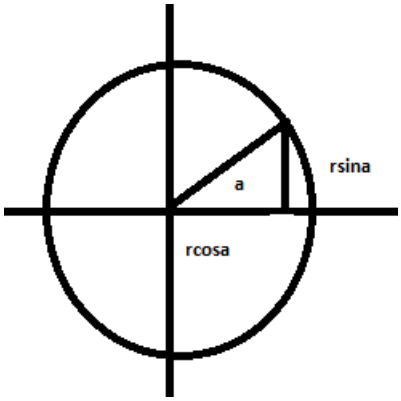
```
int xs = 100;int ys = 100;//výchozí souřadnice středu kuličky
int rk = 10; //poloměr kuličky
int vx=10;//složky rychlosti
int vy=5;
Hodit se nám bude vlastní metoda pro kreslení kuličky ze středu, protože pohyb
většinou vztahujeme k těžišti objektů.
private void puntik(Graphics p,Color b, int x, int y, int r)
{
 //vyplní kolečko se středem o souřadnicích x, y a poloměrem r barvou b
 SolidBrush stetec=new SolidBrush(b);
 p.FillEllipse(stetec, x - r, y - r, 2 * r, 2 * r);
 }
 private void Form1_Paint(object sender, PaintEventArgs e)
        {
            Graphics kp=e.Graphics;
            puntik(kp, Color.Blue, xs, ys, rk);
         }
```

Výpočet posunutých souřadnic můžeme umístit také do implicitní události časovače:

```
private void timer1_Tick(object sender, EventArgs e)
        {
            int t = 1; ;
            xs += t* vx;
            ys += t* vy;
            Refresh();
}
```

Modelování rovnoměrného kruhového pohybu

Opět využijeme svých poznatků z matematiky a fyziky. Rovnoměrný pohyb po kružnici má konstantní úhlovou rychlost – ω – tedy úhel/čas. Souřadnice, na kterých se nachází kroužící bod, pak můžeme vypočítat podle následujícího obrázku:



r – poloměr kružnice se středem v počátku

(Vzpomeňte na definici goniometrických funkcí, kdy byl poloměr kružnice roven 1)

a – úhel = ωt

x = r cosa

y = r sina

Pokud má střed kružnice souřadnice S[xo, yo] x = xo + r cos a, y = yo + r sina

Výpočet novýh souřadnic můžeme umístit opět do implicitní události časovače nebo do události Paint okna.

```
public partial class Form1 : Form
    {
       double xs = 300; double ys = 200;//výchozí souřadnice středu kuličky
        int xo = 200; int yo = 200; //střed otáčení
        int ro = 100; //poloměr otáčení
        int rk = 10; //poloměr kuličky
        double omega=60; //úhlová rychlost
        double a = 0; //úhel
       …
private void Form1_Paint(object sender, PaintEventArgs e)
        {
            Graphics kp=e.Graphics;
            puntik(kp, Color.Black, xo, yo, 5); //Černě vyznačíme střed otáčení
            xs= xo + Math.Cos(a) * ro; //Černě vyznačíme střed otáčení
```

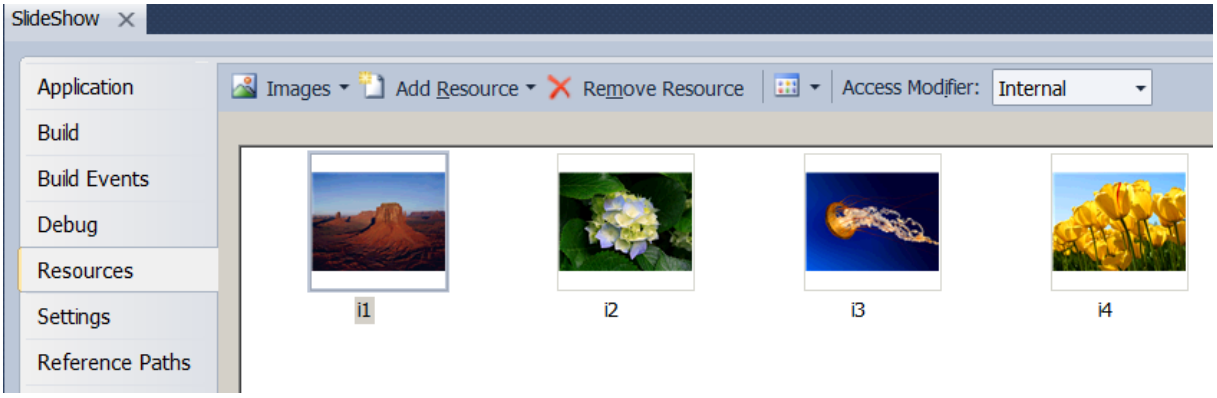
```
            ys= yo + Math.Sin(a) * ro;
            int xsi = Convert.ToInt32(xs); //souřadnice objektů jsou celočíselné
            int ysi = Convert.ToInt32(ys);
            puntik(kp, Color.Red, xsi, ysi, rk);
                   }

        private void timer1_Tick(object sender, EventArgs e)
        {
            double t = 0.01;
            a += omega * t * Math.PI / 180;//zvětšíme úhel a převedeme na radiány
            Refresh();
         }
```

Pracovní list

Cvičení

1. Doplňte příklad s přímočarým pohybem tak, aby se kulička po nárazu na okraje okna pružně odrážela
2. (*)Zkuste pomocí časovače připravit Slideshow (Cyklické promítání několika obrázků) – můžete použít vestavěné obrázky (Project/Properties/Resources/Add Resource/Add Existing File)



3. Naprogramujte odpočítávání – v okně se objeví velká desítka, po sekundě devítka, po objevení jedničky se okno zavře.
4. Vymyslete si vlastní animaci obrázku nebo objektu.

Řešení

1.

(Stačí si uvědomit, že při nárazu na boční okraje se mění x-ová složka rychlosti na opačnou, při nárazu na dolní a horní okraj pak y-ová složka.)

```
private void timer1_Tick(object sender, EventArgs e)
        {
            int t = 1; ;
```

```
            xs += t* vx;
            ys += t* vy;
            Refresh();
            if ((xs>=Width-2*rk)||(xs<=0)) vx=-vx;//odraz na okrajích
            if ((ys >= Height-4*rk) || (ys <= 0)) vy = -vy;
        }
```

2.

1. řešení s přepínačem switch – u většího počtu obrázků poněkud nepohodlné

```
int p = 1; //členská proměnná pro větvení
…

private void button1_Click(object sender, EventArgs e)
{
   //toto tlačítko střídavě spouští a zastavuje promítání
    if (buttonSpust.Text == "Stop")
    {
        timer1.Enabled = false;
        buttonSpust.Text = "Go";
    }
    else
        {
        timer1.Enabled = true;
        buttonSpust.Text = "Stop";
    }
}

private void timer1_Tick(object sender, EventArgs e)
{
    Image obr=null;
    switch (p)
    {
        case (1):obr=Properties.Resources.i1;
            break;
        case (2): obr = Properties.Resources.i2;
            break;
        case (3): obr = Properties.Resources.i3;
            break;
        case (4): obr = Properties.Resources.i4;
            break;
    }
    p++;
    if (p > 4) p -= 4;
```

```
            BackgroundImage=obr;
        }
```

2. Řešení s polem obrázků

```
        Image[] obry = new Image[] {Properties.Resources.i1,Properties.Resources.i2,
           Properties.Resources.i3,Properties.Resources.i4};

        int p = 0;           //index výchozího obrázku


        private void timer1_Tick(object sender, EventArgs e)
        {

            BackgroundImage=obry[p];
            p++;
            if (p >= 4) p -= 4;

        }
```

3.

```
        int x = 10;


        private void buttonOdpocet_Click(object sender, EventArgs e)
        {
            timer1.Enabled = true;
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            label1.Text = x.ToString();
            x--;
            if (x < 0) Close();
        }
```

4.

Individuální řešení