DUM č. 3 v sadě

28. Inf-4 Jednoduchá hra Had ve Flashi (ActionScript)

Autor: Robert Havlásek

Datum: 05.03.2014

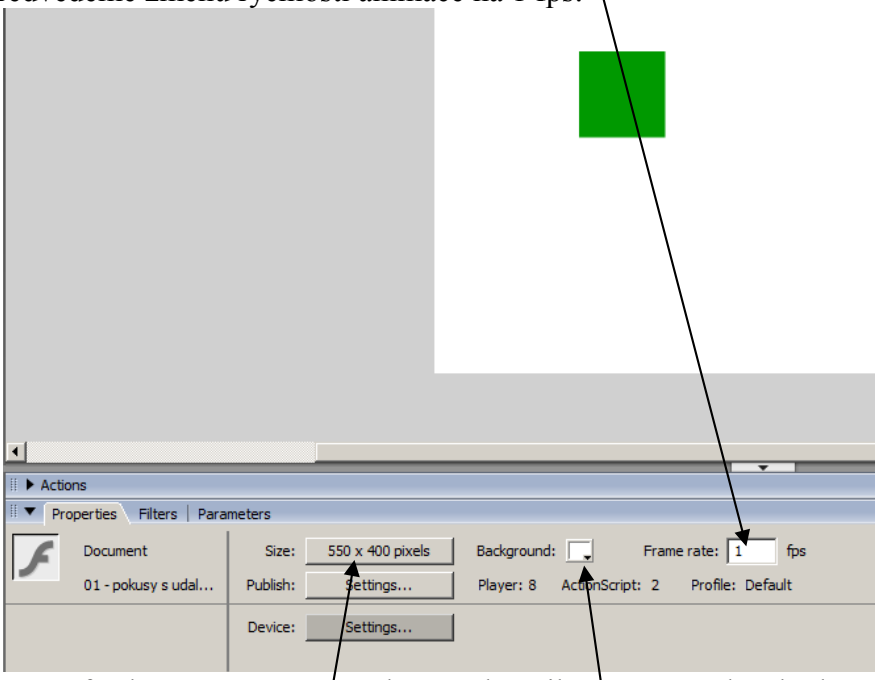
Ročník: 5AV

Anotace DUMu: Flash - teorie: Nastavení vlastností animace. Důležité vlastnosti objektů (pozice, rozměry, viditelnost) a jak se oslovují skriptem. Tvorba jednoduché proměnné a její platnost.

evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Vlastnosti animace

Vlastnosti animace lze nastavit v panelu Properties, pokud není označen žádný objekt (např. po kliknutí na plochu animace).
Studentům předvedeme změnu rychlosti animace na 1 fps:

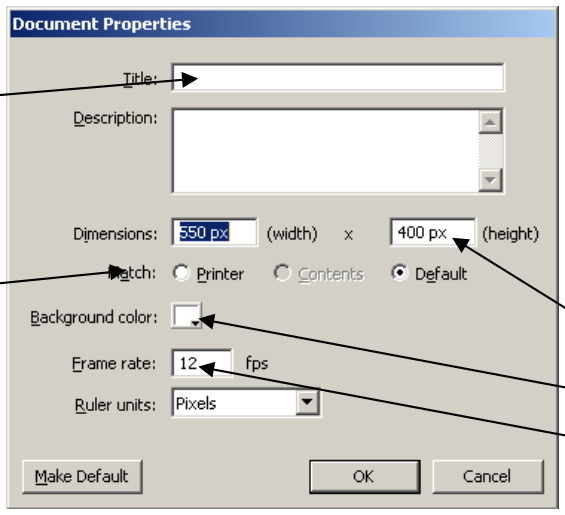


Pro ukázku změny fps lze použít program, kterým končil 1. DUM, tedy s kódem
`onClipEvent(enterFrame) {trace ("hop");}`
Slovně lze též zmínit, že Frame rate může být i menší, například 0.5, v praxi se ale obvykle setkáváme s vyššími Frame rate.

Tlačítko pro změnu barvy pozadí obvykle jen zmíním, že existuje.

Tlačítko pro změnu velikosti animace v sobě skrývá celý dialog, v němž je možné provést více změn:

Změnu názvu animace
(v některých přehrávačích se název objeví v horní liště)



Též lze zde znovu změnit:

rozměry

barvu pozadí

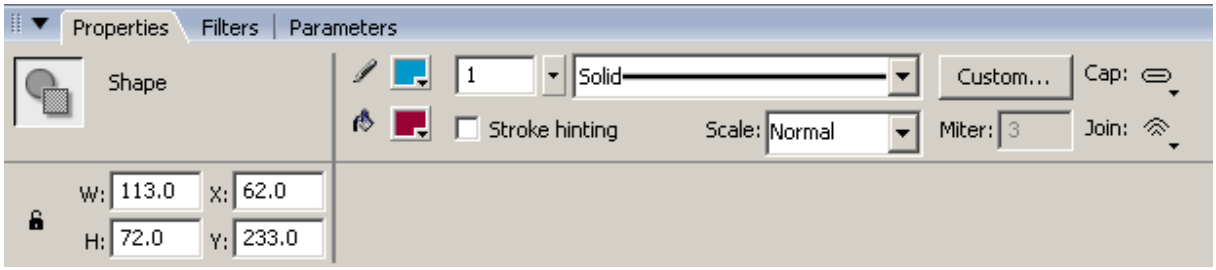
Frame rate

Přizpůsobení rozměrů:
Default znamená, že je nepřizpůsobujeme.
Printer znamená přizpůsobení na poměry papíru ($1 : \sqrt{2}$).
Contents znamená přizpůsobení rozměrů podle obsahu; plocha se nastaví tak široká a vysoká, aby byl obsah (skupina objektů umístěných v ploše) „vystředěn“ – měl od pravého i levého okraje stejnou vzdálenost a též od horního i dolního okraje stejnou vzdálenost.

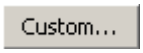
Vlastnosti objektů

Vlastnosti kresby

Kresbě lze při návrhu nastavit následující parametry v panelu Properties:
Pedagogická poznámka: Chceme-li mít všechny vlastnosti kresby aktivní, nakreslíme a vybereme vhodný objekt (mající plochu, mající rohy), například čtverec.




… barva okrajové čáry a barva plochy. V barevném dialogu studenty upozorníme na možnost zvolit percentuální průhlednost (v dialogu Alpha) a též možnost zvolit barvu  , tedy neviditelnou (objekt existuje, ale jeho barva není vidět).

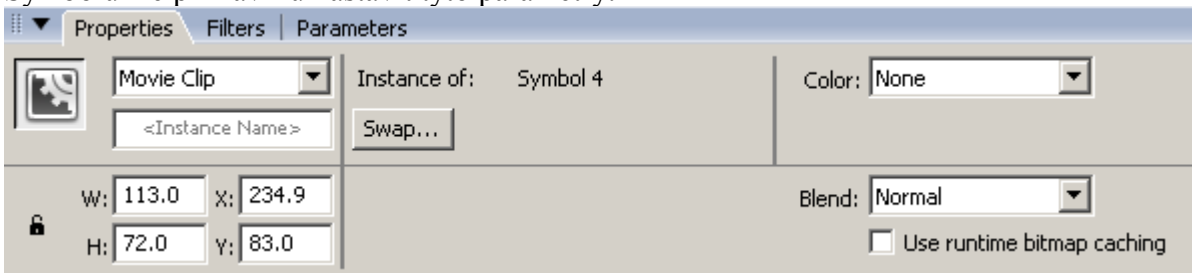
… šířka a styl čárkování/tečkování, příp. lze tlačítkem 

nastavit detaily, jak dlouhými úseky budeme čárkovat/tečkovat.
Hairline (vlasová tloušťka) má tu zvláštní vlastnost, že je v libovolném přiblížení vidět jako jednopixelová čára.

Programátorsky zajímavé jsou pak už jen rozměry (W: a H:) a pozice (souřadnice X, Y), která se váže k „levému hornímu rohu objektu“, resp. znamená vzdálenost objektu od levého či horního okraje plochy.

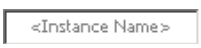
Vlastnosti symbolu

Symbolu lze při návrhu nastavit tyto parametry:



Panel Properties se mírně liší podle toho, jde-li o Movie Clip, Button nebo Graphic – změnit symbol na jiný druh lze přímo zde, výběrem 

vlevo nahoře.

Z hlediska programátora je nejzajímavější vlastností symbolu Instance Name 

. Jde o pojmenování konkrétní instance, kterou pak lze tímto jménem oslovovat v programech psaných v ActionScriptu.

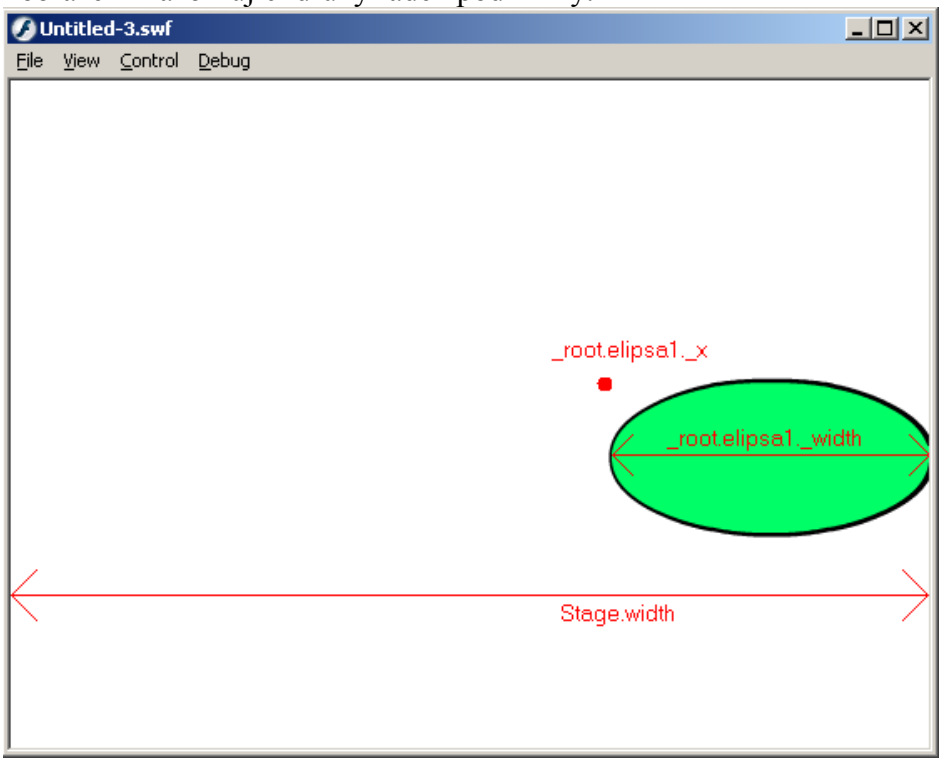
Vlastnosti symbolů, které lze používat v kódu ActionScriptu

`_height` … výška symbolu
`_width` … šířka symbolu
`_x` … xová souřadnice symbolu; jaká je vzdálenost objektu od levého okraje plochy
`_y` … yová souřadnice symbolu, tedy vzdálenost objektu od horního okraje plochy
(Pozor, osa y je v počítačích naopak než v matematice – její kladné hodnoty jdou směrem dolů, souřadný systém počítá s bodem [0,0] umístěným v levém horním rohu.)
`_rotation` … o jaký úhel bude objekt rotován oproti stavu těsně po vytvoření
`_visible` … bude-li objekt viditelný (má hodnotu `true` nebo `false`)
`_alpha` … číselná hodnota udávající průhlednost
`_xmouse` … pozice xové souř. myši (relativně vzhledem ke značce křížku symbolu)
`_ymouse` … pozice yové souřadnice myši (též relativně vzhledem k pozici objektu)
`_name` … obsahuje Instance name symbolu

Má-li tedy symbol umístěný na ploše jméno (Instance Name) `elipsa1`, oslovíme jeho xovou souřadnici jako `_root.elipsa1._x`. Celou plochu pak lze oslovit jako `Stage`.
Typicky, chceme-li zjistit, jestli je celá elipsa1 v ploše animace (nikde „nevyčnívá ven“), napíšeme souhrn podmínek:

```
    _root.elipsa1._x>=0
and _root.elipsa1._x<=Stage.width-_root.elipsa1._width
and _root.elipsa1._y>=0
and _root.elipsa1._y<=Stage.height-_root.elipsa1._height
```

Pro ilustraci obrázek znázorňující druhý řádek podmínky:



Celý zápis podmínky se všemi formalitami je součástí 5. DUMu.

Budeme-li psát kód jako reakci na událost objektu elipsa1, nemusíme ji oslovovat složitě `_root.elipsa1`, ale lze napsat pouze `this` (ve smyslu: „tento objekt; neboli objekt, který událost vyrobil“).

Objekty se navíc spojují do složitějších celků (řekněme „dědičných stromů“), lze tak oslovovat např. všechny potomky vzniknuvší na ploše nebo všechny objekty, které jsou uvnitř nějakého symbolu.

Tvorba jednoduché proměnné a její platnost

Formálně vzato, měli bychom před prvním použitím nějaké proměnné tuto proměnnou deklarovat. Děje se tak pomocí příkazu `var jmeno_promenne`
nebo `var jmeno_promenne : typ_promenne`
nebo `var jmeno_promenne : typ_promenne = pocatecni_hodnota`
či jen `var jmeno_promenne = pocatecni_hodnota`
Typem proměnné může být (pro úplnost): `Array`, `Attribute`, `Boolean`, `Custom` (můžeme tedy zadefinovat vlastní typ), `DataProvider`, `Date`, `DeltaPacket`, `Number`, `Object`, `PhoneNumber`, `SocialSecurity`, `String`, `XML`, `ZipCode`.
Studentům zmíníme prozatím jen `Boolean` (obsahuje logickou hodnotu false nebo true), `Number` (číslo) a `String` (řetězec znaků). Pokud nezadáme hodnotu, defaultně jí Flash (podle typu) přiřadí `false`, `NaN` a `null` (tedy „nepravda“, „není číslo“ a „odkaz nikam“).
Pokud proměnné nezadáme ani typ ani hodnotu a rovnou ji použijeme, defaultně jí Flash přiřadí hodnotu `undefined`.

Proměnná má platnost pouze v bloku, v němž byla používána, a pouze ve skriptech toho objektu, v němž byla používána.

Lokální platnost v bloku

Typicky, pokud ji deklarujeme v nějaké funkci (např. v obsluze události on (press)), platí pouze v této funkci. Chceme-li ji používat i vně, musíme ji vně zadeklarovat.
Studentům obvykle ukazuji (bez nutnosti chápání syntaxe funkce):

```
var vnejsi = "vnejsi";
function pokus()   // jen deklarace funkce, neprovede se
{ vnejsi = vnejsi + "1";
   trace(vnejsi);}  // vypise text vnejsi1
pokus();      // zde je volani funkce, provede tu funkci deklarovanou vyse
trace(vnejsi);    // opet vypise text vnejsi1
```

Pokud bychom ale napsali:

```
var vnejsi = "vnejsi";
function pokus()   // jen deklarace funkce, neprovede se
{ vnejsi = vnejsi + "1";
   trace(vnejsi);}  // vypise text vnejsi1
pokus();      // zde je volani funkce, provede tu funkci deklarovanou vyse
trace(vnejsi);    // vypise jen text vnejsi
```

uvnitř funkce se vyrobí nová (lokální) proměnná stejného jména, dokonce dostane i hodnotu z té globální proměnné; po skončení funkce se ale lokální hodnota zapomene a dál se pracuje s původní globální proměnnou.

Pokud napíšeme:

```
function pokus()   // jen deklarace funkce, neprovede se
{ var vnitrni = "1";
   vnitrni = vnitrni + 1;
   trace(vnitrni);}  // vypise text 11 (spojil dva řetězce)
pokus();      // zde je volani funkce, provede tu funkci deklarovanou vyse
trace(vnitrni);    // vypise undefined
```

poslední řádek vypíše `undefined`, protože proměnná `vnitrni` tou dobou už neexistuje.

Platnost v objektu a mimo objekt

Deklarujeme-li proměnnou jménem `prom` uvnitř skriptu navázaného na nějaký objekt (například v panelu Actions po kliknutí na symbol elipsa1), můžeme ji jako `prom` oslovovat pouze ve skriptech tohoto objektu (elipsa1). V ostatních skriptech ji musíme oslovovat jako `_root.elipsa1.prom`.
Budeme-li chtít používat proměnnou ve všech objektech stejnojmenně, musí být tzv. globální, musíme ji (v ActionScriptu 2.0) deklarovat zápisem:
`var _global.jmenopromenne = hodnota` či `var _global.jmenopromenne : typ = hodnota`
Používat ji pak můžeme ve všech symbolech jako `jmenopromenne` (tedy, pokud stejnojmenná proměnná neexistuje i aktuálním symbolu; jinak by se použila ta, podle pravidla „bližší košile než kabát“) i jako `_global.jmenopromenne`

Pedagogická poznámka: Je to logické. Flash se brání situaci, kdy bychom nesměli importovat „cizí“ objekty, protože netušíme, jaké mají proměnné uvnitř – aby náhodou nekolidovaly, objekty si nezasahovali navzájem do svých proměnných, atd.
Představme si třeba proměnnou i, kterou programátoři s oblibou používají jako řídící prom. v cyklech – pokud by neměl každý objekt svoji, zapouzdřenou, dva paralelně běžící cykly by sahaly po stejné proměnné, různě by si ji jeden druhému měnily…

Například, budeme-li mít situaci:

v objektu elipsa1:

```
on (press) {
_global.zivoty = 10;
lokalni = 77;
}
```

v objektu ctverec1:

```
onClipEvent(enterFrame)
 { trace(_global.zivoty);
   trace(zivoty);
   trace(_root.elipsa1.lokalni);
   var zivoty=3;  // deklarujeme lokalni,
                  // ta bude silnejsi
   }
```

při každém skoku na aktuální snímek (rychlostí dle fps animace) se bude vypisovat:
napoprvé: `undefined`, `undefined`, `77`,
při dalších skocích: `undefined`, `3`, `77`.
Až klikneme myší na elipsu1, změní se výpisy na: `10`, `3`, `77`.

Pokud by se nám povedlo kliknout na elipsu1 dřív, než by nastal první enterFrame, vypadal by pak první výpis takto: `10`, `10`, `77`, (zde se druhá desítka bere z globální proměnné)
všechny další výpisy pak: `10`, `3`, `77`. (a zde už ne, už máme lokální, která je silnější)