DUM č. 11 v sadě

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

Autor: Robert Havlásek

Datum: 06.04.2013

Ročník: 5AV

Anotace DUMu: Flash - teorie: Softwarové kreslení. Kolize hloubek při vyrábění nových objektů i při klonování. Tvorba čar a křivek.

esf evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
2007-13 OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

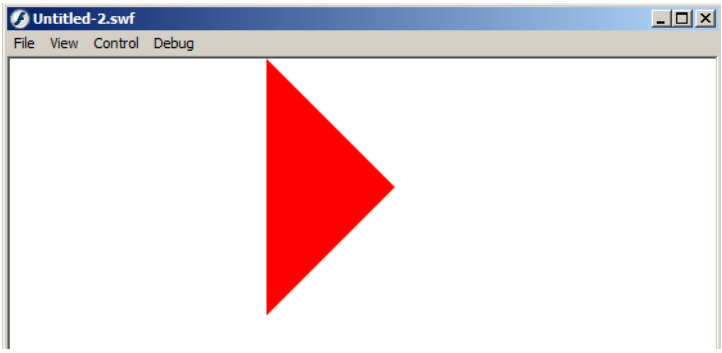
Vyrobení nového objektu, softwarové kreslení

V programu není vůbec nutné mít všechny objekty připraveny předem. Naopak lze většinu objektů vyrobit a nakreslit v kódu až v dané chvíli, dynamicky. Musíme ale počítat s časem, který uplyne, než objekt vznikne.
Vyzkoušíme vyrobit nový objekt dynamicky. Začneme s prázdnou plochou, do jejího Actions napíšeme:

```
this.createEmptyMovieClip("sipka", 1);
```

Tento řádek vyrobí nový prázdný MovieClip, jeho souřadnice (vlastnosti _x a _y) zůstanou defaultně [0,0]. Do něj můžeme kreslit pomocí čar (funkcí lineTo), příp. přesouvat perem (funkcí moveTo). Všechny souřadnice jsou při kreslení zadávány relativně, vzhledem k počátku MovieClipu (tedy ne absolutně vzhledem k ploše).
Chceme-li vyrobit uzavřený vyplněný celek, na jeho začátku zavoláme beginFill (s parametrem kód barvy, nejlépe hexadecimálně) a na konci endFill.
Studentům napíšeme tento jednoduchý příklad:

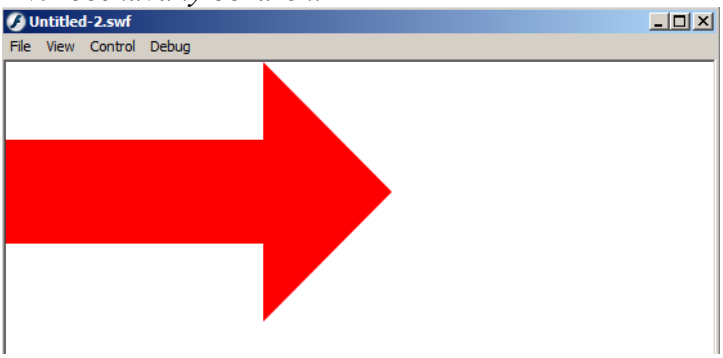
```
this.createEmptyMovieClip("sipka", 1);
sipka.beginFill(0xFF0000);
sipka.moveTo(200,0);
sipka.lineTo(200,200);
sipka.lineTo(300,100);
sipka.lineTo(200,0);
sipka.endFill();
```



Pedagogická poznámka: Neumějí-li studenti kódovat barvy v RGB modelu, buď jim systém vysvětlíme, nebo je odkážeme na nějaký generátor, např. na http://www.colorpicker.com/.

Pedagogická poznámka: Vzniklý trojúhelník nakreslíme na tabuli a rozebereme souřadnice, odkud kam jsme se pohybovali.

Praktický úkol: Dokreslete (buď jako samostatnou výplň nebo doplněním dalších příkazů do již hotového příkladu) body, aby vznikla šipka s vodorovnou částí o výšce 80 a šířce 200 px – viz očekávaný obrázek:



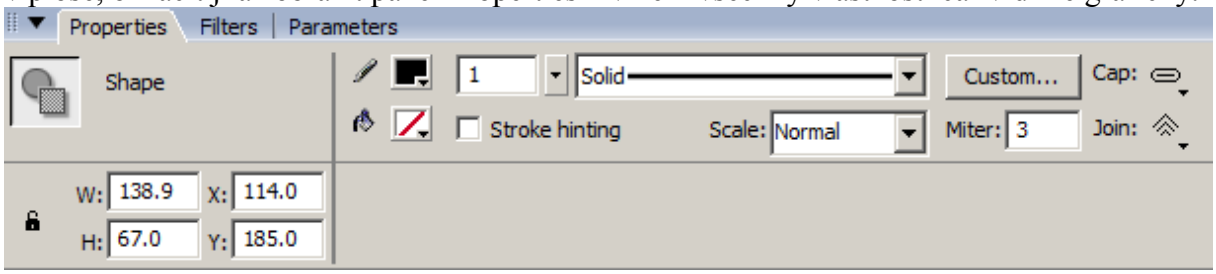
Řešení:

```
sipka.beginFill(0xFF0000);
sipka.moveTo(200,60);
sipka.lineTo(000,60);
sipka.lineTo(000,140);
sipka.lineTo(200,140);
sipka.endFill();
```

Studenti, povšimněte si, že není nezbytně nutné uzavírat mnohoúhelník zpět do úvodního bodu, Flash jej uzavře „nejkratší cestou“.

Vlastnosti čáry

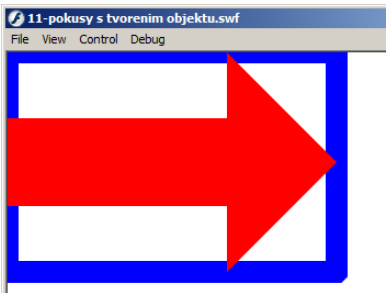
Chceme-li kreslit objekty nikoliv výplní, ale čarou, je potřeba před kreslením zvolit vlastnosti čáry pomocí funkce `lineStyle`. Ta má velké množství parametrů:
`thickness:Number` … tloušťku čáry (v pixelech),
`color:Number` … barvu čáry (hexadecimální číslo v RGB systému),
`alpha:Number` … průhlednost (číslo 0 znamená nejvíc průhledný, číslo 100 neprůhledný),
`pixelHinting:Boolean` … při true se Flash snaží čáry zarovnat do pixelů (není důležité),
`noScale:String` … zda se čára při zvětšení objektu taky zvětší (hodnoty jsou "normal", "none", "vertical" a "horizontal" – poslední dvě znamenají „čáru zvětši jen pokud objekt zvětšujeme v daném směru),
`capsStyle:String` … jak se Flash chová k okrajům ("none" je uřízne, "round" je zaobli půlkruhem, "square" je ukončí čtvercem se středem v koncovém bodě) ,
`jointStyle:String` … jak se Flash chová k úhlům ("mitter" rohy ponechá do špičky, "round" je zaobli, "bevel" je usekne),
`miterLimit:Number` … číslo od 1 do 255, udává, jaký je limit, kdy roh řízneme
Než precizně popisovat všechny detaily všech parametrů funkcí, je lepší čáru nakreslit v ploše, označit ji a zobrazit panel Properties – v něm všechny vlastnosti čar vidíme graficky:



Do stejného projektu vyrobíme okrajový modrý obdélník:

```
this.createEmptyMovieClip("obdelnik", 0);
obdelnik.lineStyle(20, 0x0000FF, 100, false, "none", "none", "miter", 1);
obdelnik.lineTo(300, 0);
obdelnik.lineTo(300, 200);
obdelnik.lineTo(0, 200);
obdelnik.lineTo(0, 0);
```

Neřekneme-li jinak, obdélník se vyrobí v nulových souřadnicích, v hloubce 0, tedy hlouběji, než je šipka. Evidentně je jeho modrá čára pod šipkou:



Vyzkoušíme přepsat jeho hloubku při inicializaci z 0 na číslo vyšší než 1, například:


```
this.createEmptyMovieClip("obdelnik", 5);
…
```

Obdélník pak bude nad šipkou:

Podobnou situaci docílíme, vyrobíme-li obdélník v hloubce 0 a do jiné hloubky jej pak přesuneme pomocí funkce swapDepths:

```
this.createEmptyMovieClip("obdelnik", 0);
obdelnik.lineStyle(20, 0x0000FF, 100, false, "none", "none", "miter", 1);
obdelnik.lineTo(300, 0);
obdelnik.lineTo(300, 200);
obdelnik.lineTo(0, 200);
obdelnik.lineTo(0, 0);
obdelnik.swapDepths(5);
```

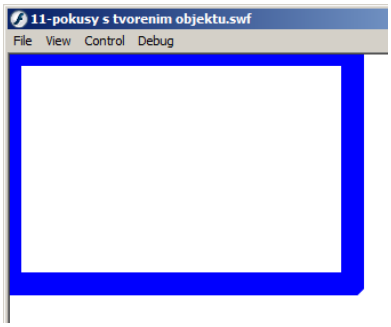
Namísto čísla 5 lze použít i funkci `_root.getNextHighestDepth()`, která vrátí nejnižší hloubku takovou, aby byl objekt nad ostatními.

Stejná hloubka ⇒ přepíšeme objekt!

Vůbec nejhorší situace nastane, pokud vyrobíme nový objekt nebo zduplikujeme objekt do hloubky, v níž již nějaký objekt existuje. Starý objekt se prostě přepíše objektem novým! Tedy pří kódu:

```
this.createEmptyMovieClip("sipka", 1);
sipka.beginFill(0xFF0000);
sipka.moveTo(200,0);
sipka.lineTo(200,200);
sipka.lineTo(300,100);
sipka.lineTo(200,0);
sipka.endFill();
sipka.beginFill(0xFF0000);
sipka.moveTo(200,60);
sipka.lineTo(000,60);
sipka.lineTo(000,140);
sipka.lineTo(200,140);
sipka.endFill();

this.createEmptyMovieClip("obdelnik", 1);
obdelnik.lineStyle(20, 0x0000FF, 100, false, "none",
"none", "miter", 1);
obdelnik.lineTo(300, 0);
obdelnik.lineTo(300, 200);
obdelnik.lineTo(0, 200);
obdelnik.lineTo(0, 0);
```



vidíme pouze obdélník, šipka už neexistuje:

Jak tedy mít jistotu, že vytvořením nového objektu do nějaké hloubky nepřepíšeme objekt, který už v dané hloubce je? Řešením je optat se, zda v hloubce nějaký objekt existuje, funkcí `getInstanceAtDepth`, která pro argument „číslo hloubky“ vrátí odkaz na instanci objektu, jenž v ní je, nebo vrátí undefined, pokud tam žádný objekt není. Typicky:

```
if (this.getInstanceAtDepth(1)==undefined) this.createEmptyMovieClip("obdelnik",1);
```

Nebo můžeme postupně zjišťovat volné hloubky, a když nějakou najdeme, objekt do ní dát:

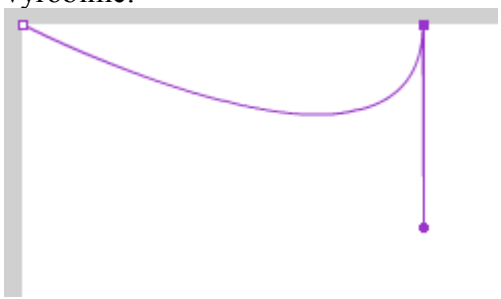
```
var i=0; while (this.getInstanceAtDepth(i)!=undefined) i++;
this.createEmptyMovieClip("obdelnik", i);
```

Křivky

Pro tvorbu křivek z více segmentů použijeme opakovaně příkaz `curveTo` – pro každý segment jeden. Jeho parametry jsou čtyři, jejich význam je mírně složitější: první dvě jsou souřadnice control-pointu (kontrolního bodu, který určuje, ke které tečně se křivka přibližuje a jakou silou), druhé dvě jsou souřadnice cílového uzlu, kam křivka jde.
Při zadání

```
this.createEmptyMovieClip("krivka", 0);
krivka.lineStyle(1, 0x0000FF, 100, false, "none", "none", "miter", 1);
krivka.curveTo(200,100,200,0);
```

vyrobíme:



Neboli: vlevo nahoře je výchozí bod ([0,0]), vpravo nahoře je cílový uzel [200,0], dole pod ním kontrolní bod [200,100], který říká, že těsně před cílovým bodem půjde křivka prakticky svisle nahoru (a jakou silou).

Praktický úkol: V manuálu Flashe najděte příklad, který pomocí několika segmentů křivek vyrábí kruh, a pomocí něj vyrobte kruh bez čáry s červenou výplní a rozměry 100×100 bodů.

Řešení:

```
this.createEmptyMovieClip("circle2_mc", 2);
circle2_mc.beginFill(0xFF0000);
drawCircle(circle2_mc, 100, 100, 100);
circle2_mc.endFill();
function drawCircle(mc:MovieClip, x:Number, y:Number, r:Number):Void {
    mc.moveTo(x+r, y);
    mc.curveTo(r+x, Math.tan(Math.PI/8)*r+y, Math.sin(Math.PI/4)*r+x,
Math.sin(Math.PI/4)*r+y);
    mc.curveTo(Math.tan(Math.PI/8)*r+x, r+y, x, r+y);
    mc.curveTo(-Math.tan(Math.PI/8)*r+x, r+y, -Math.sin(Math.PI/4)*r+x,
Math.sin(Math.PI/4)*r+y);
    mc.curveTo(-r+x, Math.tan(Math.PI/8)*r+y, -r+x, y);
    mc.curveTo(-r+x, -Math.tan(Math.PI/8)*r+y, -Math.sin(Math.PI/4)*r+x,
-Math.sin(Math.PI/4)*r+y);
    mc.curveTo(-Math.tan(Math.PI/8)*r+x, -r+y, x, -r+y);
    mc.curveTo(Math.tan(Math.PI/8)*r+x, -r+y, Math.sin(Math.PI/4)*r+x,
-Math.sin(Math.PI/4)*r+y);
    mc.curveTo(r+x, -Math.tan(Math.PI/8)*r+y, r+x, y);
}
```

Pedagogická poznámka: Tento úkol testuje jinou dovednost než většina předchozích – dovednost najít v nápovědě cizí kód, vyznat se v něm a lehce jej poupravit. V tomto případě se očekává, že namísto původního `circle2_mc.lineStyle(0, 0x000000);`, které by kreslilo kruh čarou bez výplně, napíšeme `circle2_mc.beginFill(0xFF0000);` a za zavoláním drawCircle napíšeme `circle2_mc.endFill();`
Je přitom nepříjemné, že kód z nápovědy využívá funkce, k nimž teorii budeme rozebírat teprve v DUMu č. 13. Obvykle nevědí, kam `circle2_mc.endFill()` dát.
Též je nepříjemné, že je objekt pojmenovaný `circle2_mc`, což je pro studenty nezvyklé.