GRADA

Vytváříme hry ve Flashi

Zdeněk Schneider

Lukáš Jirka

PODROBNÝ PRŮVODCE
ZAČÍNAJÍCÍHO UŽIVATELE

Základy programování ve Flashi

Marketingové využití Flash her

Tvorba jednotlivých her

Příklady na www.grada.cz

Upozornění pro čtenáře a uživatele této knihy

GRADA®

Obsah

Úvod

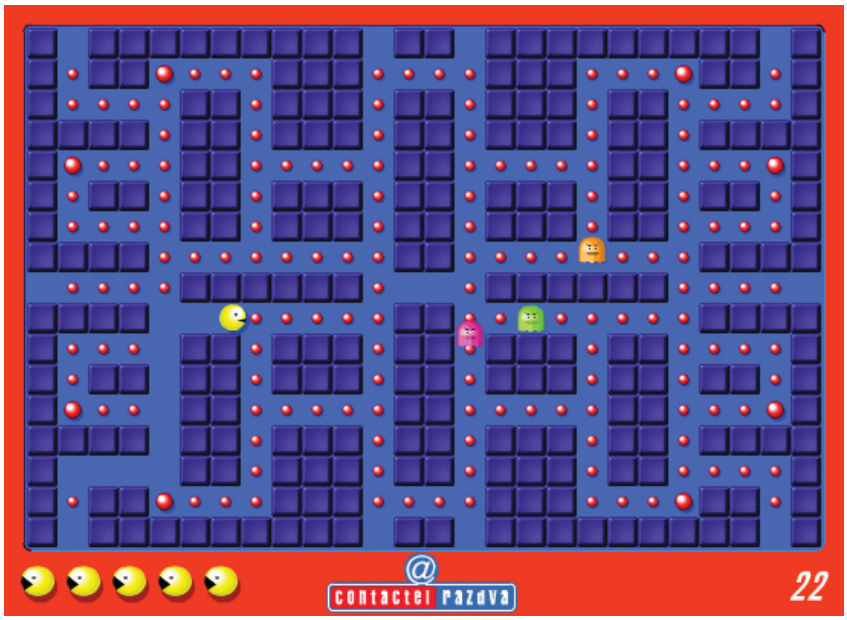
Dalo by se říci, že jsme se dlouhou dobu rozhodovali o tom, jestli tuto knihu máme vůbec napsat a jak ji přesně pojmout tak, aby zaujala co nejvíce čtenářů. Flash hry jsou velmi oblíbené, což dokazuje návštěvnost webových stránek, které takové hry obsahují. Opačným případem je ale právě jejich tvorba. Ne každý je natolik pokročilý programátor, aby byl schopen takové hry vytvářet a ne každý je natolik dobrý grafik, aby jeho hry byly úspěšné pro široké hráčské publikum. V této knize se sice dozvíte něco málo o tom, jak grafiku do her vytvářet, ale je samozřejmé, že grafickému cítění se nedá naučit z knihy a ne každý má podobné nadání. Naproti tomu programovat se může naučit téměř každý a tato kniha vás provede jednotlivými postupy, zádrhely a nástrahami při tvorbě Flash her. Naučíte se díky ní vytvářet hry snad všech myslitelných žánrů, které si dokážete představit. To vše v přehledných postupech, kdy nebudete jen slepě opisovat desítky řádků příkazů, ale postupně dospějete k tomu, k čemu je každý řádek programu potřeba a co se stane, pokud ho nahradíte řádkem jiným. Cílem knihy není naučit se otrocky naprogramovat všechny uvedené hry, ale zažít si jednotlivé postupy, které se pro každou hru mohou podstatně lišit. I z tohoto důvodu jsme se rozhodli knihu napsat jako spoluautoři a každý tak ukázat svůj různorodý přístup k práci, který ale vede ke stejnému výsledku – funkční hře, která bude jistě ozdobou vašich stránek.

Ačkoliv měla být kniha původně určena spíše pokročilým uživatelům programu, tuto myšlenku jsme poměrně rychle opustili, protože cílová skupina knihy by byla natolik malá, že by bylo obtížné podobné dílo obhájit. Zkušení uživatelé se ale nemusí bát, že by vzniklé dílo bylo nějakým kompromisem. Úvod knihy je věnován připomenutí základů programu, (zároveň ale doufáme, že se bude jednat opravdu o připomenutí, protože pokud bychom knihu věnovali jen základům, nezbylo by už na samotné hry místo) a dalšími kapitolami už začíná nefalšované programovaní od jednoduchých hříček až po velmi složité hry (i hry pro více hráčů – multiplayerové hry), které dají bez naší knihy zabrat i zkušeným programátorům. Zároveň jsme se snažili zahrnout do knihy takové hry, které jsou originální především svým netradičním řešením a nejedná se pouze o používání základních funkcí programu.

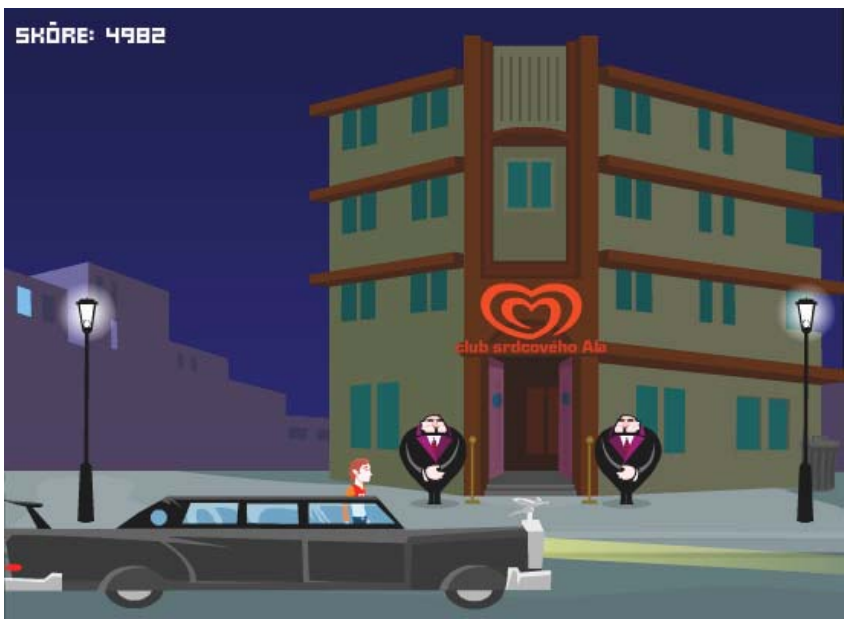
K naší knize rozhodně nelze přistupovat jako k románu, který si na jeden „zátah“ přečtete, ale vyžadujeme vaši aktivní spolupráci a počítáme s tím, že budete nad každým řádkem přemýšlet a sami si vymýšlet, jak hry ještě více zdokonalit a vylepšit. Opravdovým cílem této knihy ale je, aby se vám zde popsané postupy natolik vžily, že budete schopni sami bez problému vytvořit libovolnou hru i úplně jiného žánru.

Kupodivu tvorba her ale není jediná věc, kterou se v této knize naučíte. Kromě různých dalších vylepšení, jako jsou herní tabulky, kam se mohou hráči zapisovat, ukládání herního skóre či pozice v rozehrané hře do cookies, se naučíte také pracovat ve *Flashi* s 3D grafikou, se zvuky (například si vytvoříme MP3 přehrávač, který umí používat skiny z programu *WinAmp*), optimalizovat hry (a nejenom je) na co největší plynulost a rychlost načítání a mnoho dalšího.

Zdrojové kódy všech her a aplikací, které jsou v knize popsané, zároveň naleznete na webových stránkách nakladatelství Grada Publishing www.grada.cz. Doufáme, že se nespokojíte pouze s jejich stažením, ale že vám budou sloužit spíše ke kontrole postupů popsaných v této knize. Tímto vám přejeme příjemné čtení naší knihy a pevně doufáme, že vám přinese mnoho nových poznatků při práci s programem *Macromedia Flash MX*.

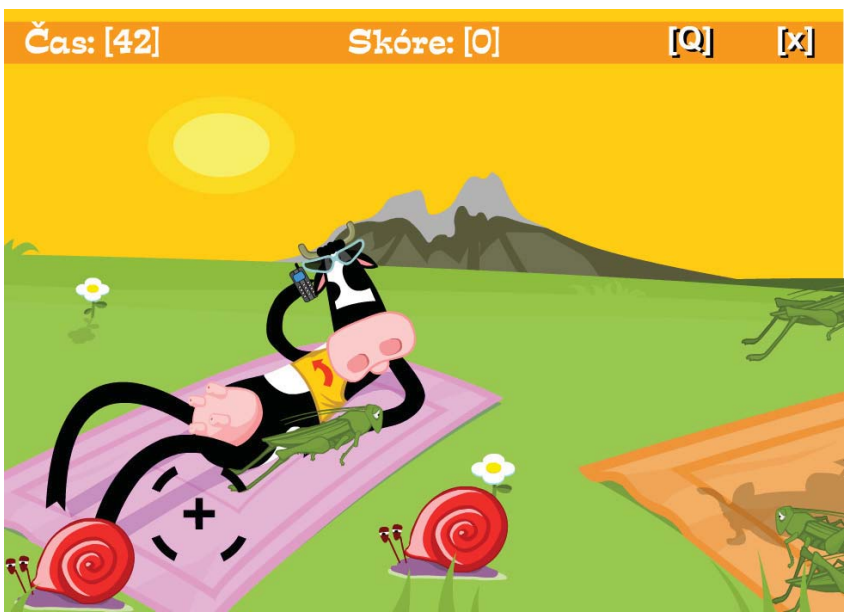


Akční hra Paeman společnosti Contactel



Adventure hra společnosti Algida

Logická hra společnosti Contactel



Akční střílečka

Akční hra Western

Použité konvence

V celé knize jsou použity následující typografické konvence, které usnadňují orientaci v textu:

- **Tučným písmem** jsou zdůrazněny důležité pojmy.
- ***Tučnou kurzivou*** jsou označeny příkazy nabídek, popisky tlačítek, názvy dialogových oken, „citace z obrazovky".
- ***Modify → Document*** – posloupnost příkazů (zadávaná v nabídkách a následně otevíraných dialogových oknech) oddělená šipkou.
- *Kurzivní písmo* je vyhrazeno pro názvy programů a firem vyrábějících softwarové produkty.
- KAPITÁLKY slouží k popisu kláves a klávesových zkratek.

V textu se budete také často setkávat se zvláštními odstavci označenými ikonou, která bude charakterizovat druh informace v daném odstavci:

*V **poznámkách** jsou umístěny informace týkající se tématu a prozrazující další souvislosti.*

*Pokud uvidíte takto označený odstavec, znamená to, že je nablízku nějaký **tip** nebo **trik**, s jehož pomocí si můžete usnadnit práci, případně snadno dosáhnout efektních výsledků.*

*Takový odstavec by se měl stát varovně vztyčeným prstem, který vás upozorňuje na něco, na co byste si měli dát **pozor**, co vás může nepříjemně překvapit, nebo co by vám mohlo způsobit problémy.*

1.

Základy tvorby v programu Flash

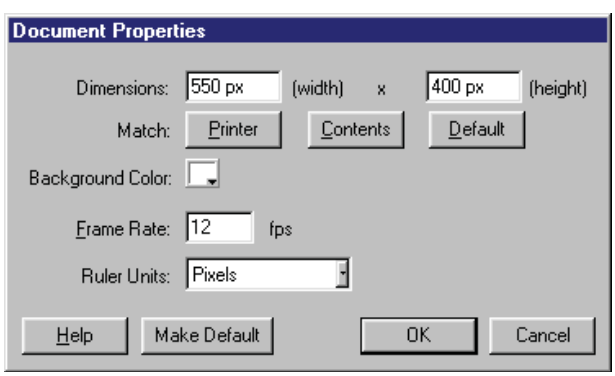
1. Základy tvorby v programu Flash

Pokud se nepovažujete za začátečníky v programu *Flash*, můžete následující kapitoly, věnované jeho základům, směle přeskočit. Stejně tak je ale pravděpodobné, že jste si tuto knihu zakoupili a s *Flashem* teprve začínáte. Ačkoliv není zrovna ideální začít hned programováním her, vysvětlíme si úvodem základy tvorby v tomto programu. Naučíme se, jak si nastavit scénu před tvorbou hry, jak vytvořit jednotlivé typy symbolů, se kterými program pracuje, vysvětlíme si, jak a kam umístit skripty popsané u tvorby všech her, povíme si, jak naprogramovat preloader, který zajistí, aby uživatel byl informován o tom, jak dlouho se bude hra načítat, či jaké existují typy textových polí. Tyto kapitoly rozhodně nepovažujeme za kompletní kurz o programu, ale spíš připomenutí některých jeho základů.

1.1 Nastavení scény

Tvorbu každé hry začínáme přizpůsobením scény našim požadavkům. K tomu slouží panel ***Modify → Document*** (obrázek 1.1.1). Jako první tu nalezneme položky ***Dimension***, které určují rozměr scény. Standardní rozměr scény je 550 x 400 bodů, což bude pro většinu her vyhovovat. Doporučujeme počítat s uživateli, kteří doposud používají rozlišení 800 x 600 (těch je k dnešnímu dni zhruba 30–40 %), a nedělat hry na větší rozlišení než 770 x 430, což odpovídá zhruba velikosti stránky po odečtení všech nabídek, ikon či posuvníků. Navíc, čím menší rozlišení hra má, tím běží plynuleji i na pomalejších počítačích. Naštěstí je *Flash* vektorový program, takže v mnoha případech stačí rozlišení hry upravit, až když je hra hotová, pouze v HTML souboru. Při práci s bitmapovými obrázky ale doporučujeme zachovat přesné rozměry, následným zvětšením či zmenšením celé scény trpí především kvalita jejich zobrazení.

Background Color je barva pozadí celé scény. Tu je možno měnit i pomocí HTML souboru, ve kterém SWF zobrazujeme bez nutnosti zásahu do *Flashe*. Poslední důležitou položkou je ***Frame Rate***. Na rozdíl od rozlišení hry je standardní volba 12 fps (snímků za sekundu) pro 90 % her absolutně nedostatečná. Proto zkuste nastavit hodnoty na 20–25 fps, jinak nebude většina her dostatečně dynamická a rychlá. Zbytečně vysoký Frame Rate (maximem je 120) zase klade vysoké nároky na výkon počítače, navíc málokdy této hodnoty reálně dosáhnete a takováto plynulost nalezne využití jen výjimečně – vhodná je například v případě přesných hitTestů, ale tím se nebudeme hned ze začátku knihy zabývat.



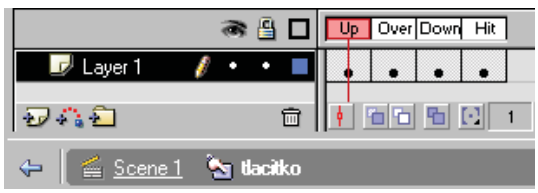
Obr. 1.1.1: Panel se základním nastavením vlastnosti scény

1.2 Tvorba tlačítka

Tlačítko je prvkem, který umožňuje zjistit, zda uživatel klepnul na nějaký objekt, či jen přes něj přejel myší, nebo stiskl nějaké tlačítko na klávesnici a podobně.

Nové tlačítko vytvoříme nakreslením libovolného objektu, jeho označením a volbou ***Insert → Convert to Symbol*** a volbou ***Button***, což je zmíněné tlačítko. Z objektu je nyní symbol tlačítka, do kterého se přepneme klepnutím na něj a zjistíme, že má časovou osu pouze na čtyřech snímcích (obrázek 1.2.1). Jsou to tyto:

- **Up** – obrázek Buttonu v klidovém stavu.
- **Over** – Button ve chvíli, kdy přes něj uživatel přejede kurzorem myši.
- **Down** – Button ve chvíli, kdy ho uživatel stiskne.
- **Hit** – aktivní plocha Buttonu (není viditelná).



Obr. 1.2.1: Časová osa tlačítka má pouze čtyři využitelné snímky

Pokud nechceme některé snímky využít, necháme je prázdné.

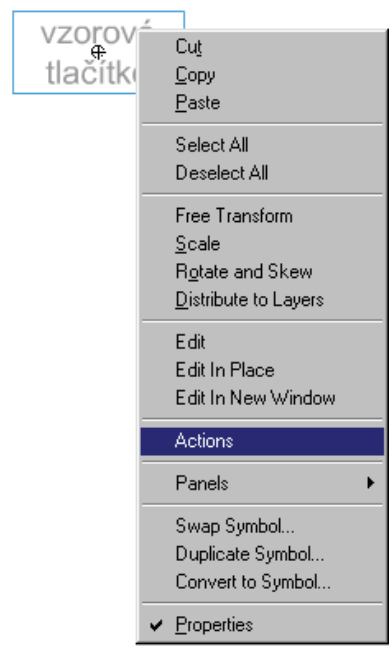
Na každý Button můžeme umístit libovolné programové akce, a to tak, že na něm stiskneme pravé tlačítko myši a zvolíme ***Actions*** (obrázek 1.2.2). Otevře se nám okno programovacího jazyka ActionScript, kde se programují akce, které Button bude provádět.

Ať zvolíme jakýkoliv příkaz ActionScriptu, objeví se nám uvozen mezi závorky příkazu:

```
on (release) {}
```

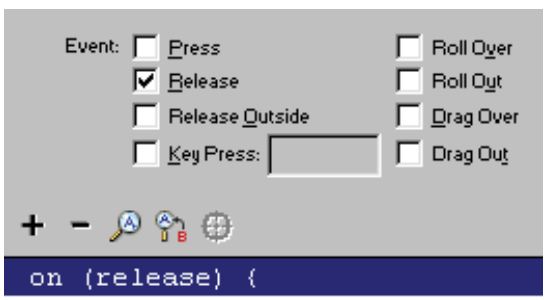
To je základní příkaz, který ověřuje, jestli uživatel stiskl a uvolnil tlačítko myši na ploše Buttonu (tj. stiskl tlačítko). Těchto příkazů ale existuje víc, stačí se přepnout na první řádek se zmíněnou akcí a *Flash* nabídne tyto další možnosti:

- **Press** – podobné jako základní příkaz, pouze program provede akci hned po stisknutí tlačítka myši a nečeká na jeho uvolnění.



Obr. 1.2.2: K volbě Actions se dostaneme stisknutím pravého tlačítka myši na objektu

- **Release Outside** – příkaz se provede po uvolnění tlačítka myši mimo aktivní plochu Buttonu. Tj. uživatel najede myší na Button a stiskne tlačítko myši, to drží a odjede kurzorem mimo Button a tlačítko myši uvolní.
- **Key Press** – akce se provede po stisku určité klávesy. V tomto případě se ani nemusí Button nacházet na scéně, ale i mimo její aktivní plochu, protože na poloze kurzoru myši nezáleží.
- **Roll Over** – akce na Buttonu se provede, jakmile přes něj uživatel jen přejede myší.
- **Roll Out** – provedení akce je podmíněno tím, že uživatel přejede přes Button myší, a pak z něj sjede pryč. V tu chvíli se spustí akce.
- **Drag Over** – akce se spustí, když uživatel stiskne tlačítko myši mimo aktivní plochu Buttonu, stále ho drží a najede na Button.
- **Drag Out** – akce je spuštěna stisknutím tlačítka myši na Buttonu, jeho držením a následným opuštěním aktivní plochy Buttonu.



Obr. 1.2.3: Výběr z možností, při jakých se tlačítko aktivuje

Pozor na jednu drobnost. Na HTML stránce pozná Flash stisk klávesy až poté, co je SWF soubor aktivní, tj. předtím už do něj uživatel někam klepnul (i naprázdno) myší. Je to z bezpečnostních důvodů, aby například reklamní Flash banner umístěný na stránce vstupu do nějakého freemailu nebyl schopen zachytávat stisknuté klávesy (zadávání hesla) atd.

Tím by se dalo toto stručné pojednání o tlačítkách uzavřít. Závěrem si jen řekneme, že na jednom Buttonu se dají všechny tyto možnosti vzájemně kombinovat. Pokud chcete volat symbol z jiného skriptu, lze ho ve chvíli, kdy je označený, pojmenovat v panelu ***Properties → <Instance Name>***. Jméno symbolu pak můžete volat odkudkoliv. Dáte-li na hlavní scénu například příkaz:

```
_root.nazevtlacitka._visible=0
```

tak Button zmizí ze scény.

1.3 Tvorba Movie Clipu

Movie Clip se dá považovat za nejvíce používaný symbol v programu *Flash*. Uvnitř tohoto symbolu může být libovolný počet vrstev s libovolným počtem snímků v každé z nich. V podstatě se tak Movie Clip stává scénou ve scéně. Animace uvnitř Movie Clipu běží nezávisle na hlavní scéně. Pokud umístíte na hlavní scénu, která má pouze jeden snímek, Movie Clip, uvnitř kterého se odehrává animace, tak ta poběží i přesto, že na hlavní scéně

se na první pohled nic neanimuje. Stejně jako na Button, tak i na Movie Clip se dají umisťovat akce programovacího jazyka ActionScript. Postup je naprosto shodný – tj. na symbolu stiskneme opět pravé tlačítko myši a zvolíme položku ***Actions***.

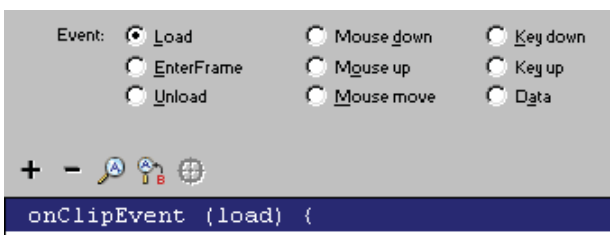
Jakákoliv akce se, stejně jako u Buttonu, objeví uvozena mezi závorky tohoto příkazu:

```
onClipEvent (load) {}
```

Tento příkaz znamená, že následující akce se provedou pouze při prvním objevení symbolu na scéně. Tj. když se symbol bude nacházet například na pátém snímku časové osy, tak se akce spustí, až *Flash* dojde na tento snímek. Najedeme-li myší na tento příkaz, nabídnou se nám opět další volby:

- **EnterFrame** – akce uvozené tímto příkazem se provádějí stále dokola, dokud je Movie Clip na scéně.
- **Unload** – akce za tímto příkazem se provedou pouze jednou, a to ve chvíli, kdy Movie Clip zmizí ze scény.
- **Mouse down** – tento příkaz se provede pokaždé, když uživatel stiskne levé tlačítko myši (s pravým a prostředním program *Flash* neumí pracovat). Pokud uživatel tlačítko myši drží, příkaz už se dále neprovádí.
- **Mouse up** – následující akce se provedou pouze ve chvíli, kdy uživatel uvolní předtím stisknuté tlačítko myši.
- **Mouse move** – podobné jako základní příkaz, pouze program provede akci hned po stisknutí tlačítka myši a nečeká na jeho uvolnění.
- **Key down** – akce se provede, jakmile uživatel stiskne libovolnou klávesu.
- **Key up** – zde se naopak akce provede až po uvolnění stisknuté klávesy na klávesnici.
- **Data** – akce se provede ve chvíli, kdy je Movie Clip zavolán odjinud externím příkazem. Například:

```
_root.jmenoklipu.promenna = 1
```



Obr. 1.3.1: Výběr podmínky, při jaké se akce na Movie Clipu provede

Všechny příkazy se dají opět kombinovat, takže v rámci jednoho Movie Clipu můžete používat všechny podmínky najednou. Možnosti těchto příkazů můžou svádět k používání programových akcí už jenom na Movie Clipech. Pokud umístíte ten samý příkaz přímo na časovou osu, je vždy o něco rychleji proveden, což je znatelné, především pokud máte na scéně například dvacet Movie Clipů s akcemi ***EnterFrame***.

Movie Clip se dá pojmenovat a později volat stejným způsobem jako Button.

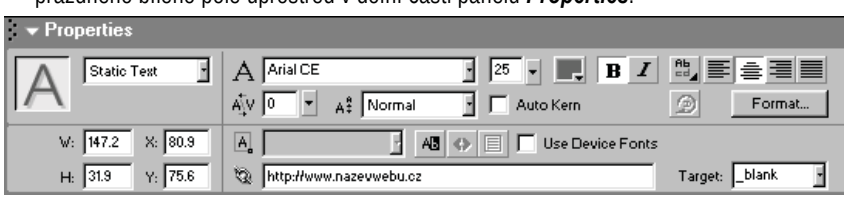
1.4 ActionScript

Jak jsme si již řekli v předchozích odstavcích, tak ActionScript je programovací jazyk aplikace *Flash*, který slouží k provádění většiny složitějších akcí. Příkazy mohou být umístěny jak na tlačítka (Buttony), kde se provedou například po stisknutí tlačítka, přejetím kurzoru myši přes něj, či stisknutím libovolného tlačítka na klávesnici, tak na Movie Clipy – zde se provedou například při načtení objektu, nebo se provádějí nonstop. Poslední možností je umístění přímo na časovou osu. V tomto případě nemají žádné další volby a příkaz se provede vždy, když program dojde na požadovaný snímek, kde se akce nachází. Okno ActionScriptu se objeví, pokud na zvoleném objektu či snímku časové osy stisknete pravé tlačítko myši a zvolíte ***Actions***. ActionScript má v současné verzi programu již několik set příkazů, z nichž si velkou část probereme až přímo při programovaní her v dalších kapitolách. Příkazy ActionScriptu lze buď vybírat přímo z jejich seznamu, a pak dopisujete jen některé jejich parametry, nebo je můžete v případě, že je znáte nazpaměť, psát přímo. Pomocí CTRL+SHIFT+E se přepnete do takzvaného *Expert* módu a pomocí CTRL+SHIFT+N zase zpět do módu *Normal*.

1.5 Textová pole

Flash rozlišuje celkem tři typy textových polí, podle jejich použití:

- **Static Text** – běžný, pevně daný text, jehož obsah nelze měnit pomocí ActionScriptu. Část textu může odkazovat na jinou HTML stránku. Odkaz se zadává do velkého prázdného bílého pole uprostřed v dolní části panelu ***Properties***.



Obr. 1.5.1: Odkaz na webovou stránku v textu, která se otevře v novém okně

- **Dynamic Text** – text, jehož obsah je shodný s obsahem libovolné proměnné na scéně. Může tak zobrazovat například herní skóre, či různé texty, které se průběžně mění. ***Instance Name*** v panelu ***Properties*** je název tohoto pole (stejně jako název například Movie Clipu či Buttonu), který umožňuje s textovým polem dále pracovat. Do položky ***Var:*** se zadává, jakou proměnnou textové pole zobrazuje.

Obr. 1.5.2: Název pole a proměnné

- **Input Text** – textové pole, do kterého zadává obsah uživatel za běhu aplikace. Například své jméno před začátkem hry apod. Toto pole má, stejně jako Dynamic Text, položku ***Instance Name*** a ***Var:***. Číslem v ***Maximum Characters*** můžeme omezit

maximální počet napsaných znaků do textového pole. Uživatel pak nebude moci zadat delší jméno než například na 15 znaků. Číslo 0 znamená v tomto případě neomezený počet. U toho typu pole můžete také změnit volbu ***Single Line*** na ***Password*** a všechny vložené znaky budou vypadat jako hvězdičky (z důvodu bezpečnosti při zadávání hesla).

Textová pole obsahují další možnosti nastavení, jako zalamování řádků, zapnutí bílého podkladu či zapnutí/vypnutí možnosti si obsah textového pole označit a mnoho dalších – cílem této knihy ale není vás naučit základy programu, proto podrobný popis vynecháme.

1.6 Tvorba preloaderu

Preloader je většinou krátký program, který pozastaví spuštění hry (programu) na dobu, než se celá hra (program) nebo její část načte z internetu do paměti počítače. Může vypadat například jako procentuální ukazatel toho, jak se hra načítá. Preloader přinutí hráče počkat, než se vše načte. (Při absenci preloaderu může dojít k nedostatečnému načtení hry, kterou se pak nemusí povést spustit, nebo bude fungovat jen její část.)

Podoba preloaderu může být libovolná. Většinou postačí běžný ukazatel v procentech, také to může být linka, která se prodlužuje podle toho, jak se hra načítá, či obrázek, který se například vyplňuje barvou. V případě složitějších her, které se načítají dlouho, může preloader vypadat jako malá jednoduchá hra, která návštěvníka zabaví i na několik minut, dokud se nenačte celá větší hra.

Nyní si už ukážeme několik různých typů preloaderů:

Začneme tím nejsnadnějším, který bude ukazovat, kolik procent ze hry už je načteno. Vytvoříme si textové pole *Dynamic* a do pole ***Var:*** napíšeme „*info*“, což bude název proměnné, kterou bude pole zobrazovat. Z hotového textového pole uděláme Movie Clip a na něj dáme tyto akce:

```
1  onClipEvent (enterFrame) {
2     info = Math.round(100*_root.getBytesLoaded()/
    ↳_root.getBytesTotal())+" %";
3     if (info == "100 %") {
4       _root.gotoAndStop(2);
5     }
6  }
```

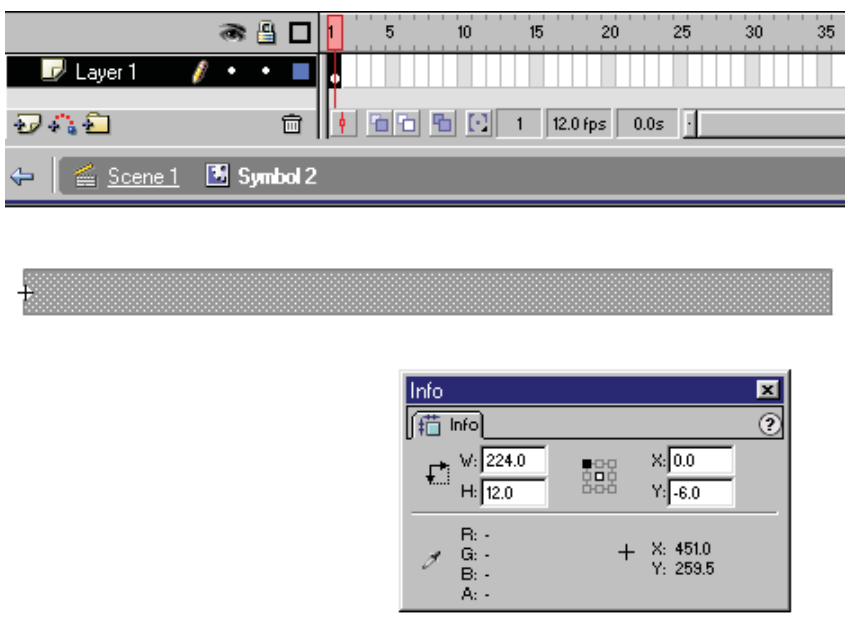
První řádek skriptu provádí všechny následující akce na Movie Clipu neustále. Druhý řádek je nejpodstatnější. Proměnná „info“, kterou zobrazujeme v textovém poli, vznikne tím, že pomocí příkazu `_root.getBytesLoaded()` zjistíme, jak velká část souboru se už načetla do paměti.

Pokud tuto hodnotu dělíme celkovou velikostí souboru `_root.getBytesTotal()`, násobíme stem a zaokrouhlíme na celé číslo, tak získáme hodnotu 0 až 100 odpovídající skutečnému stavu načtení souboru. Za ní stačí pouze dopsat znak % a je hotovo. Teď už jen na třetím řádku ověříme, jestli se hra načetla, a pokud ano, skript provede příkaz na řádku čtyři – tj. skočí na snímek dvě na hlavní scéně. Tam už se nachází hra. Celý preloader zakončíme umístěním akce `stop();` na první snímek časové osy na hlavní scéně. Pokud bychom tam tento příkaz nedali, *Flash* by preloader přeskočil jako každý jiný snímek, čímž by ztratil svou funkčnost.

O něco málo složitější preloader vypadá jako linka, která se prodlužuje podle toho, jak se hra načítá. Je dobré nějak naznačit, kam má linka dosahovat (například udělat kolem ní obrys), aby bylo na první pohled vidět, kdy bude plně načtená.

Nakreslíme si obdélník, představující čáru, který bude mít zapnutou výplň i obrys. Obrys necháme tak, jak je, a z výplně uděláme Movie Clip. Nyní se přepneme do vnitřku tohoto Movie Clipu a posuneme obdélník tak, aby se střed objektu nacházel v jeho levém kraji (obrázek č. 1.6.1). Pokud by se nacházel uprostřed, tak se linka bude prodlužovat ze středu do stran. Teď se přepneme zpět na hlavní scénu a Movie Clip přesuneme, aby se nacházel uprostřed obrysu, tak jako ze začátku. Poté na něj umístíme tyto příkazy:

```
1  onClipEvent (enterFrame) {
2     _xscale = 100*_root.getBytesLoaded()/
      _root.getBytesTotal();
3     if (_xscale == 100) {
4       _root.gotoAndStop(2);
5     }
6  }
```



Obr. 1.6.1: Umístění linky (obdélníku) uvnitř Movie Clipu

Jak vidíte, tak se programový kód moc neliší od předchozího preloaderu. Druhý řádek, místo nastavení proměnné, nastaví rovnou velikost objektu podle toho, jak velká část hry je už načtená. Další řádek ověřuje, jestli nemá linka už stoprocentní velikost. Nezapomene opět na příkaz `stop();` na prvním snímku časové osy.

Posledním preloaderem, který si ukážeme, bude vyplňování respektive vybarvování obrázku. V našem případě jsme použili dva obrázky – jeden černobílý a druhý barevný (což asi

v této knize neuvidíte). Preloader bude fungovat tak, že se zobrazí černobílý obrázek, který se postupně začne vybarvovat. Až se vybarví celý, hra je načtena.

Začneme nakreslením či načtením barevného obrázku. Ten převedeme opět na Movie Clip, na němž budou tentokrát tyto akce:

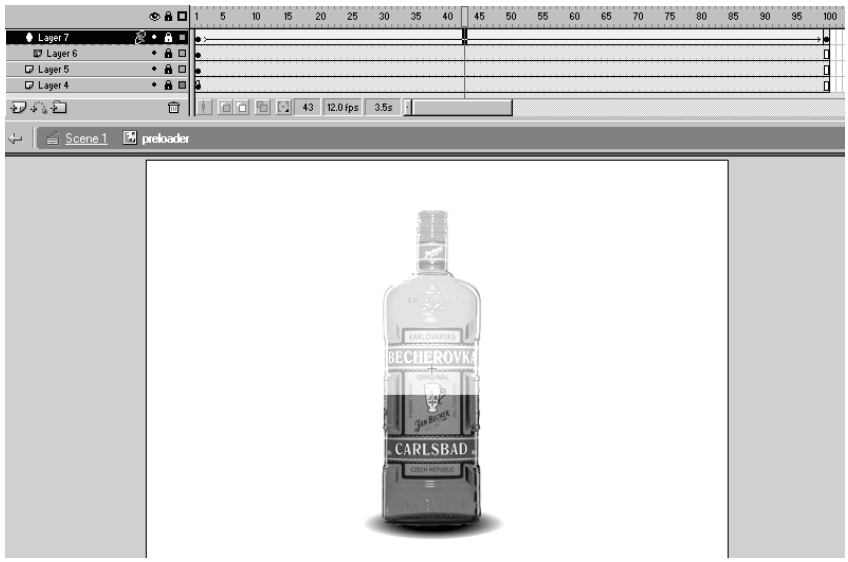
```
1  onClipEvent (enterFrame) {
2     gotoAndStop(Math.round(100*_root.getBytesLoaded()/
    ↳_root.getBytesTotal()));
3     if (_currentframe==100) {
4       _root.gotoAndStop(2);
5     }
6  }
```

Od předchozích případů se téměř neliší, akorát nyní nepočítáme číselně, kolik SWF souboru je již načteno, ani nenastavujeme velikost symbolu, ale skočíme na snímek v Movie Clipu odpovídající stupni načtení. V našem případě těch snímků bude 100, (co snímek, to zhruba jedno procento). Na první snímek hlavní časové osy dáme opět příkaz `stop();` a přepneme se konečně do Movie Clipu s obrázkem.

Na prvním snímku uvnitř Movie Clipu dáme také akci `stop();`, aby se animace vyplňování obrázku nespustila sama od sebe. Barevný obrázek prodloužíme na časové ose až na snímek 100 a vytvoříme novou vrstvu nad ním. Do ní dáme černobílou verzi obrázku, kterou prodloužíme opět na snímek 100 a vytvoříme ještě poslední vrstvu, která bude nejvýš. Do ní nakreslíme obdélník či čtverec, který bude tak velký, že celý obrázek zakryje. Nyní zapneme u objektu animaci typu ***Shape*** a na posledním stém snímku ho přesuneme nad obrázek tak, aby ho zase odkryl. Mezi prvním a posledním snímkem se tak vytvořila animace, jak obdélník či čtverec jede směrem nahoru. Poslední, co nám zbývá, je klepnout pravým tlačítkem myši na název této vrstvy a zvolit ***Mask***. Tím se z ní stane maska vrstvy, která se nachází pod ní. Výsledkem je postupné vyplňování objektu (pozor, přímo ve *Flashi* bude vidět, jen pokud vrstvu s maskou zamknete). Tím je preloader hotov.

Ty nejdůležitější a nejčastěji používané preloadery tedy již umíme, a tak si ještě povíme něco o tom, jak a v jakém pořadí *Flash* celý SWF soubor načítá. Začíná samozřejmě prvním snímkem, který by měl být co nejmenší, aby se nestalo, že uživatel nevidí delší dobu vůbec nic, což by ho mohlo odradit. Snímek se nenačítá najednou, ale postupně po vrstvách. Pořadí jejich načtení lze dokonce ovlivnit ve ***File → Publish Settings → Flash → Load Order***. Při standardním nastavení se nejdříve načte to, co je v horní vrstvě, pak se načítá vrstva pod ní a tak dále, až k poslední. Až se načte i ta, pokračuje *Flash* na další snímek scény. Načte-li se poslední vrstva, pokračuje se další scénou (pokud existuje). K čemu je to ale dobré vědět? Pokud máte například velmi rozsáhlou hru o desítkách úrovní, bylo by zbytečné obtěžovat hráče načtením celé hry, když se pak nebude schopen ani do dalších úrovní dostat. Stačí načíst jen část SWF po první úroveň hry (tj. třeba jen 20 % hry) a ty další se už budou načítat na pozadí, zatímco uživatel bude hrát. Proto je dobré nedávat například úvodní nabídku hry, obrazovku s nápovědou, nebo obrazovku s koncem hry do snímku na konci scény nebo dokonce do dalších scén, protože pak se bude muset zbytečně čekat, než se načte všechno před tím (úrovně hry atd.).

Dobrým řešením je také některé části hry načítat z dalších externích SWF souborů podle toho, jak jsou zrovna zapotřebí, čímž se vyhneme načítání zbytečných dat.



Obr. 1.6.2: Ukázka hotového Movie Clipu s vyplňováním obrázku

2.

Využití internetových Flash her v praxi