Interaktivna multimedija, prvi sklop vaj

Naloga 2 – sovražniki

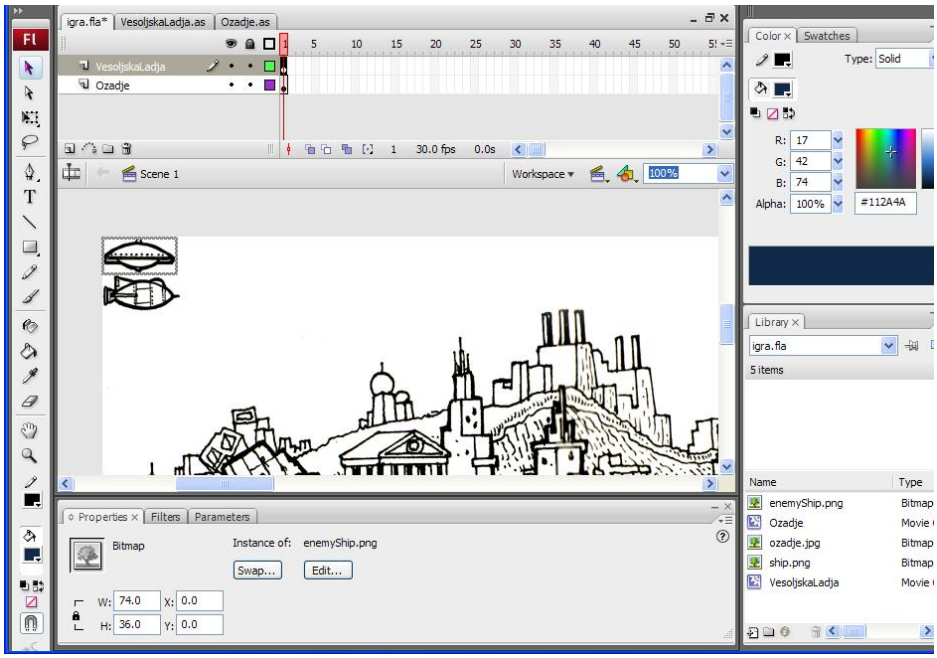
Na voljo imamo precej izboljšano igro iz prvega dela vaje – vesoljska ladja, ki leti nad mestom in ki jo lahko upravljamo s pomočjo smernih tipk.

Da bo igra bolj zanimiva, bomo v njo dodali še sovražnike.

Zapremo odprte datoteke v Flash-u in odpremo .fla datoteko ter obe .as datoteki iz direktorija vaja1-2.

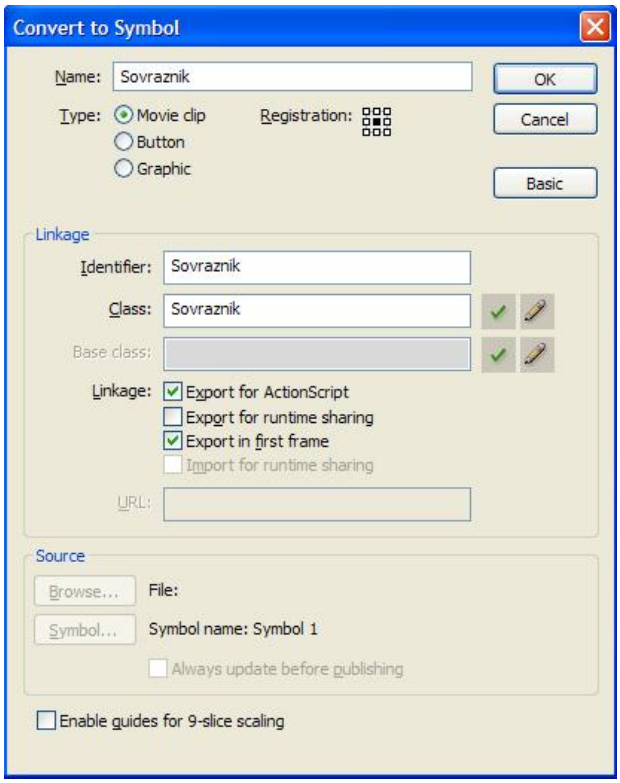
Dodajanje sovražnikov v igro

Najprej potrebujemo grafično predstavitev sovražnika, oziroma njegovo sliko. Vsi sovražniki so videti isto, zato potrebujemo samo eno sliko sovražnika. Najdemo jo v datoteki enemyShip.png.
Sliko sovražnika najprej uvozimo v *stage* Flash aplikacije: File -> Import -> Import to stage.
Sovražnik se je pojavil v zgornjem levem kotu stage-a. Vendar pa je to za enkrat še samo slika sovražnika in slikam ne moremo direktno programirati obnašanja. Da lahko programiramo obnašanje sovražnikov, moramo najprej sliko sovražnika pretvoriti v *movie clip.*



MovieClip je poseben vgrajen razred v ActionScript 2, ki ima že veliko definiranih funkcij. Med ostalimi so tudi funkciji *onEnterFrame()* in *onLoad()* funkciji tega razreda. Da lahko sovražnikom programiramo obnašanje na podoben način kot smo to naredili za našo vesoljsko ladjo in za ozadje, bomo tudi sovražnike pretvorili v razred MovieClip.

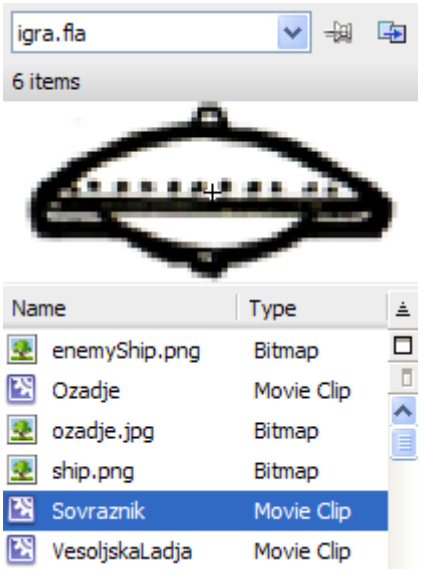
Najprej mora biti sovražnik izbran - okrog sovražnika je siv rob, kot na zgornji sliki. **Kliknemo na menu Modify -> Convert to symbol. Odpre se nam novo okno kot na spodnji sliki:**



V njem definiramo kako bomo izbrani grafični element pretvorili v simbol:

- V polju **Type** izberemo **MovieClip**.
- Označimo izbiro **Export for ActionScript**
- Označimo izbiro **Export in first frame**
- V polje **Identifier** vpišemo **Sovraznik** (z veliko začetnico, brez šumnikov)
- V polje **Class** vpišemo **Sovraznik** (z veliko začetnico, brez šumnikov)

Sedaj lahko opazimo, da se je v knjižnici (Library) pojavil nov element - Sovražnik tipa MovieClip. Če ga označimo, se v zgornjem oknu v knjižnici izriše tudi predogled sovražnika.



Sedaj v igri imamo grafično komponento sovražnika v obliki MovieClipa, nimamo pa še nobene kode, ki bi opisala lastnosti in obnašanje sovražnika. Igro lahko tudi preizkusimo s pritiskom CTRL + ENTER. Igra se bo odprla v novem oknu, lahko jo preizkusimo s pomočjo smernih tipk na tipkovnici. V zgornjem levem kotu vidimo našega sovražnika, ki pa zaenkrat ne počne nič, razen da stoji v kotu. Še ena posebnost v igri je ta, da smo tokrat dobili še sporočilo o napaki: "The class or interface 'Sovraznik' could not be loaded." Flash se namreč pritožuje, ker za sovražnika ni našel nobenih datotek s kodo. To nas zaenkrat še ne skrbi.

Razlika med sovražnikom na eni strani in našo vesoljsko ladjo ter ozadjem na drugi je v tem, da sovražnik ne sme biti prisoten na sceni dokler ne vstopi v igro iz desne strani. **Zato v Flash aplikaciji izbrišemo sovražnika iz stage-a**. Naj nas pri tem ne skrbi, da je bilo dosedanje delo zaman - sovražnik je še vedno prisoten v knjižnici (*library)*, le v *stageu* ni več prisoten, saj ga bomo v igro dodajali programsko.

Pobrišemo sovražnika, shranimo igro in jo testiramo še enkrat - tokrat sovražnika ne vidimo več.

Programiranje razreda Sovražnik

Sedaj se bomo lotili programiranja obnašanja sovražnikov, ki ga bomo najprej na kratko opisali z besedami:

- sovražniki na začetku niso prisotni v igri, dokler v njo ne vstopijo iz desnega roba igre
- sovražniki se v desnem robu igre pojavljajo na različnih višinah
- sovražniki letijo vodoravno iz desne proti levi z različnimi hitrostmi

Najprej bomo naredili vstopanje sovražnikov v igro. Kodo, ki bo sovražnike dodajala v igro, bomo napisali kar v razredu VesoljskaLadja, saj je vesoljska ladja že od samega začetka prisotna v igri. Kodo bi lahko dodali tudi v ozadje.

Najprej potrebujemo eno spremenljivko razreda, ki bo kontrolirala pogostost pojavljanja sovražnikov v igri:

```
var sovraznikTimer;
```

Tej spremenljivki na začetku postavimo vrednost na 0:

```
function onLoad()
{
    hitrost = 10;
    sovraznikTimer = 0;
}
```

Nato ji vrednost v vsakem frame-u (torej v funkciji onEnterFrame()) večamo za ena:

```
sovraznikTimer = sovraznikTimer + 1;
```

Ko doseže ta vrednost 60, jo postavimo spet na 0 in pošljemo še enega sovražnika na sceno. To naredimo v funkciji onEnterFrame():

```
1 if(sovraznikTimer > 60)
  {
   2 sovraznikTimer = 0;
     _root.attachMovie("Sovraznik", "Sovraznik" + _root.getNextHighestDepth(), _root.getNextHighestDepth());
  }
        3               4                 5                                      6
```

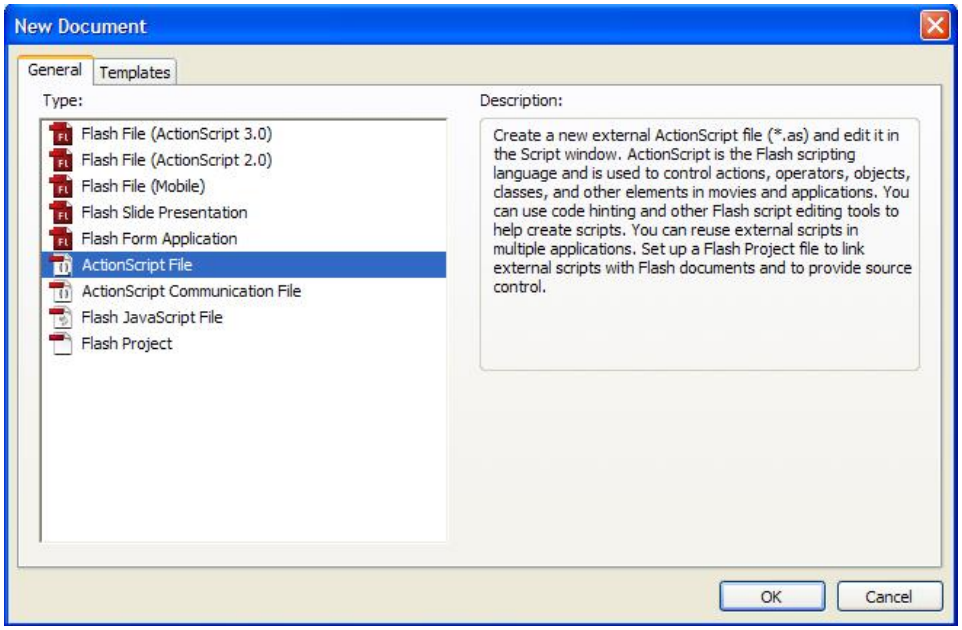
Zgornji kos kode je precej enostaven, čeprav na prvi pogled ni videti tako. Ko doseže spremenljivka sovraznikTimer vrednost 60, jo resetiramo na 0 in dodamo novega sovražnika v igro.

Podroben opis te kode je v spodnjem okvirju:

Oglejmo si to bolj podrobno:

1. `if(sovraznikTimer > 60)` – preverjamo, če je vrednost spremenljivke sovraznikTimer več kot 60.
2. `sovraznikTimer = 0;` – vrednost spremenljivke sovraznikTimer spet postavimo na 0.
3. `_root.attachMovie` – funkcija attachMovie bo dodala instanco razreda Sovrazik v stage. _root pomeni, da bomo sovraznika dodali v osnovni (_root) timeline igre. Funkcija attachMovie zahteva tri argumente znotraj oklepajev: **identifikator filmčka**, ki ga dodajamo, **ime inštance filmčka** in **globino** na katero filmček dodajamo.
4. `"Sovraznik"` – to je identifikator movie clipa. To polje smo izpolnili, ko smo sovraznika konvertirali v movie clip (Identifier).
5. `"Sovraznik" + _root.getNextHighestDepth()` – to je ime inštance, ki ga bomo dali Sovrazniku, ki ga v tem trenutku dodajamo v igro. Ker bomo dodali novega sovražnika vsakič, ko doseže vrednost spremenljivke sovraznikTimer 60, potrebujemo za vsakega sovražnika edinstveno ime. Zato uporabimo vgrajeno funkcijo `_root.getNextHighestDepth()`, ki poišče prvi prost globinski nivo na stage-u. Nivoji so označeni s številkami, tako da bodo imena instanc imela obliko: Sovraznik1, Sovraznik 2, Sovraznik 3 itd.
6. `_root.getNextHighestDepth()` – vsak filmček (movie clip) v stage-u se mora nahajati na svojem nivoju globine, tako da poleg edinstvenega imena inštance potrebujemo še edinstveno globino. Le-to nam najde funkcija `getNextHighestDepth()`.

Sedaj lahko spremenjene datoteke shranimo in testiramo igro s pritiskom na CTRL + ENTER. Po krajšem času se bo na zaslonu pojavil sovražnik, ampak v zgornjem levem kotu in ne na desnem robu igre. To je zaradi tega, ker smo zaenkrat napisali samo kodo, ki dodaja sovražnike, ne pa tudi kode, ki pove kje se bo sovražnik pojavil in kaj bo delal. Koda, ki opisuje kako se nek objekt v igri obnaša, sodi v kodo razreda tega objekta. **Zato bomo za sovražnika naredili nov razred, ki ni nič drugega kot tekstovna datoteka z nastavkom *.as*: File -> New -> ActionScript file**.



Ko kliknemo OK, se v *stage*-u odpre nov tab z novo, prazno, tekstovno datoteko. To bo naš razred Sovraznik. **V prazno datoteko napišemo naslednji kos kode:**

```
class Sovraznik extends MovieClip
{


}
```

Datoteko shranimo kot Sovraznik.as, z veliko začetnico! (File –> Save as).

Datoteke razreda se morajo začeti z rezervirano besedo **class,** ki ji sledi ime razreda, v našem primeru **Sovraznik**. Ime datoteke Sovraznik.as mora biti popolnoma enako imenu razreda Sovraznik! Ker smo sliko sovraznika prej pretvorili v movie clip, napišemo še **extends MovieClip**, kar pomeni, da je razred Sovraznik pravzaprav razred MovieClip, ki mu bomo dodali nove funkcionalnosti. Temu v objektnem programiranju pravimo dedovanje (inheritance).

Sedaj bomo opisali lastnosti in obnašanje sovražnika, ki smo ga na začetku tega podpoglavja že opisali z besedami. Pri tem nam bo pomagalo znanje, ki smo ga pridobili v prvem delu vaje. **Definiramo še spremenljivko `hitrost` in dodamo funkciji `onLoad()` in `onEnterFrame()`, ki smo ju spoznali v prvem delu vaje.**

V onLoad() funkciji razreda Sovraznik bi radi naredili tri stvari:

1. postavili x koordinato sovražnikove ladje na 700 (izven desnega roba igre)
2. postavili y koordinato sovražnikove ladje na slučajno vrednost med 0 in 300
3. postavili hitrost sovražnikove ladje na slučajno vrednost med 5 in 10

To naredimo v naslednjih treh vrsticah kode:

```
_x = 700;
_y = Math.random()*300;
hitrost = Math.random()*5 + 5;
```

Vgrajena funkcija **`Math.random()`** nam vsakič, ko jo kličemo, vrne naključno število med 0 in 1. Če potrebujemo slučajno število med 0 in 300 (kot v našem primeru), moramo to slučajno število množiti s 300. Če pa potrebujemo slučajno število med 5 in 10, najprej množimo slučajno število s 5 (tako dobimo slučajno število med 0 in 5) in mu nato prištejemo 5 (tako dobimo slučajno število med 5 in 10).

Sedaj moramo v vsakem frame-u (torej v funkciji `onEnterFrame()`) premakniti sovražnikovo ladjo v levo. Gre za isti postopek kot pri premikanju ozadja:

```
_x = _x - hitrost;
```

Ko smo prepričani, da je sovražnikova ladja zašla za levi rob zaslona in da je več ne vidimo, lahko jo popolnoma umaknemo iz igre in tako privarčujemo na spominu:

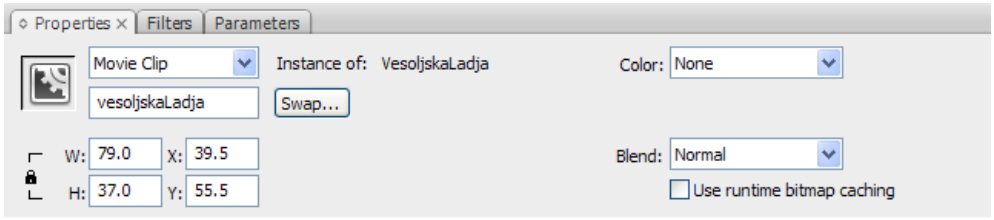
```
if(_x < -100)
{
    this.removeMovieClip();
}
```

Še enkrat vse skupaj shranimo in testiramo igro s pritiskom na CTRL + ENTER. Če je vse v redu, bi sedaj proti naši ladji morali leteti sovražniki.

Zaznavanje trkov med našo ladjo in sovražnikom

Če se poskusimo z našo ladjo zaleteti v sovražnika, se ne bo zgodilo nič. Zato bomo sedaj dodali zaznavanje trkov, tako da bo sovražnike ob stiku z našo ladjo razneslo. Trke bomo preverjali v vsaki sovražnikovi ladji, zato potrebujemo ime inštance za našo ladjo, da se bomo lahko sklicevali nanjo iz razreda Sovraznik.

Izberemo našo ladjo, tako da enkrat kliknemo nanjo. V panelu Properties vpišemo ime inštance (Instance Name) za našo ladjo: *vesoljskaLadja* (ena beseda, z malo začetnico in velikim L-jem). Sedaj se lahko iz drugih razredov sklicujemo na našo ladjo z uporabo imena `_root.vesoljskaLadja`, saj je to MovieClip, ki se nahaja na timeline-u **_root.**



Do sedaj smo uporabljali samo funkciji `onLoad()` in `onEnterFrame()`, ki sta že vgrajeni v Flashu. **Sedaj pa bomo napisali svojo funkcijo. Poimenovali jo bomo `bum()` in se bo izvajala, ko se zaletimo v sovražnika. Dodamo jo v razred Sovražnik, takoj za funkcijo `onEnterFrame()`.**

```
function bum()
{
    this.removeMovieClip();
}
```

Ko zadanemo sovražnika, naj ta zaenkrat enostavno izgine iz igre.

Zato, da vemo kdaj moramo klicati funkcijo `bum()`, moramo ugotoviti, če smo zadeli sovražnika z našo vesoljsko ladjo.

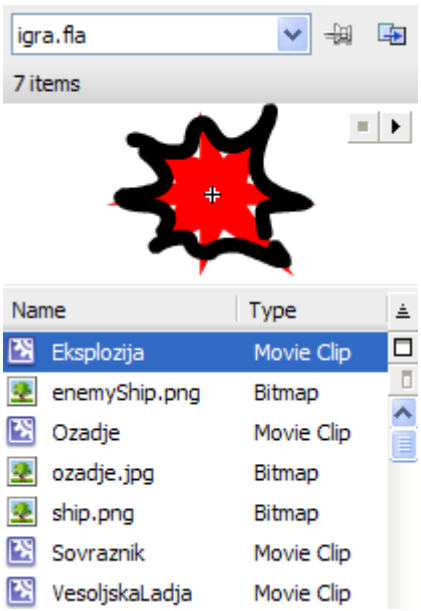
V razredu Sovražnik vpišemo naslednji kos kode v funkcijo `onEnterFrame()`:

```
if(this.hitTest(_root.vesoljskaLadja))
{
    bum();
}
```

Tukaj se sklicujemo na našo vesoljsko ladjo s pomočjo imena inštance, ki smo ga ustvarili na začetku vaje (`_root.vesoljskaLadja`). Uporabili smo vgrajeno funkcijo `hitTest`, ki v tem primeru ugotavlja, če je prišlo do stika med sovražnikom (`this`) in našo ladjo (`_root.vesoljskaLadja`). Če je to res, izvedemo funkcijo `bum()`.

Testiramo našo igro (Ctrl + Enter) in se z ladjo zaletimo v sovražnika, da vidimo kaj se zgodi. Sovražnik bi moral izginiti.

Ko nam prejšnji korak deluje, ga nadgradimo tako, da ob trku dodamo animacijo eksplozije. Z ustvarjanjem eksplozije se tukaj ne bomo ukvarjali, v knjižnici bomo našli že dodan MovieClip z imenom ***Eksplozija***. MovieClip Eksplozija ima tudi svoj pripadajoči razred, ki je že narejen, mi bomo samo na kratko pogledali, kaj se v njem dogaja:



```
class Eksplozija extends MovieClip
{
    function onEnterFrame()
    {
        if(this._currentframe == this._totalframes)
        {
            this.removeMovieClip();
        }
    }
}
```

Zgornja koda samo preverja, če se je animacija eksplozije odvrtela do konca. V tem primeru jo umakne iz scene.

Sedaj bomo nadgradili funkcijo bum() tako, da bomo ob trku dodali še eksplozijo.

```
function bum()
{
    var eksplozija = _root.attachMovie( "Eksplozija" , "Eksplozija" +
                                        _root.getNextHighestDepth(),
                                        _root.getNextHighestDepth() );
    eksplozija._x = _x;
    eksplozija._y = _y;
    this.removeMovieClip();
}
```

Vse shranimo in spet testiramo našo igro (Ctrl + Enter). Ko se sedaj zaletimo v sovražnika, bi moral ta eksplodirati.