UNIVERZITET U NOVOM SADU

PRIRODNO-MATEMATIČKI FAKULTET

INSTITUT ZA MATEMATIKU I INFORMATIKU

Dejan Katašić

Flash i ActionScript

Diplomski rad

Mentor: dr Zoran Budimac

Novi Sad, 2002

SADRŽAJ

PREDGOVOR

Pojava servisa World Wide Web proteklih godina uvela je revoluciju u korišćenju Interneta i kompjutera uopšte. Od tih momenata kompjuter sve više prestaje da bude alatka koju koriste visokostručni profesionalci i postaje još jedan zabavan i koristan aparat u domaćinstvu. Masovnim korišćenjem povećali su se i zahtevi za mogućnostima koje servis treba da obezbedi. Najveći nedostatak Interneta, protok podataka, uslovio je, s jedne strane, velika ulaganja u razvoj mrežne infrastrukture i povećanje propusnog kapaciteta same mreže i, s druge strane, razvoj i primenu raznih tehnika komprimovanja podataka i steaming formata, kako bi postojeći resursi mogli da obezbede što veću iskoristivost, pre svega, multimedijalnih mogućnosti Interneta.

Flash, proizvod kompanije Macromedia, je jedna tehnologija posebno prilagođena sukobljenim zahtevima Interneta. Grafički je zasnovan na vektorskom opisu objekata sa animacijom vođenom opisom promena na samim objektima, uz mogućnost steaminga podataka. U ovom trenutku predstavlja najefikasnije rešenje na webu iz domena animacije. Mogućnost dizajniranja korisničkog interfejsa i programiranja njegovih elemenata za interakciju sa korisnikom čini ga takođe i liderom interaktivnosti na mreži.

Cilj ovog rada jeste upoznavanje sa svetom koji Flash može da predstavi, sa akcentom na mogućnosti programiranja u ActionScriptu, Flashovom internom skript jeziku.

Uvod čitaoca upoznaje sa problematikom sadržaja na webu i predstavlja na koji način Flash odgovara na postavljene izazove.

Prvo poglavlje predstavlja program Flash i njegovo radno okruženje, način rada i mogućnosti. U drugom poglavlju nalazi se pregled programskog jezika ActionScript, dok se u narednim poglavljima kroz primere bliže osvetljavaju neki aspekti ActionScripta i Flasha. Svi primeri imaju malu veličinu izlaznog fajla, uz potpunu funkcionalnost programa.

Treće poglavlje predstavlja Pasijans, varijaciju na dobro poznatu igru Solitaire sa Windows platformi. U ovom primeru posebno je istaknuta manipulacija grafičkim elementima u Flashu, kao i detalji pri radu s objektima u jeziku. U četvrtom poglavlju, Slagalica, testiraju se perfomanse Flasha i ActionScripta za pronalaženje optimalnog rešenja zadatog problema i daje jednan način animacije nađenog rešenja. Takođe se ispituje brzina izračunavanja u različitim Flash Playerima. Peto poglavlje predstavlja mogućnost komunikacije Flasha sa spoljnim okruženjem kroz mrežnu igru za dva igrača Web memorija.

Zaključak podseća na glavne osobine Flasha i mogućnosti primene.

Zahvaljujem se porodici, profesorima i prijateljima na podršci, znanju i pomoći tokom studija i izrade ovog rada.

UVOD

Protok podataka jeste razlog zašto medijalne mogućnosti prezentacije na Internetu nisu na nivou one koja se skladišti na CD-u. Dok CD može slobodno da sadrži kompleksne programe i nekomprimovane video i audio zapise, slike i velike količine tekstualnih podataka, na web prezentaciji (dalje: prezentaciji), uz uvažavanje ograničenog protoka, je "dozvoljeno" mnogo manje – toliko da posetilac može da primi informaciju koja mu se nudi u nekom, dovoljno kratkom, roku. Sa druge strane, raznovrsnost sadržaja koji se može naći na globalnoj mreži postavlja zahtev autoru prezentacije da pružena informacija omogući korisniku da prati tok informacija koji mu se nudi putem prezentacije. Korisnik takođe ima svoje zahteve: želi "što više – to bolje" u roku "što brže – to bolje"; a ukoliko mu se po tom kriterijumu ne izađe u susret, jednostavno će krenuti da zadovoljava svoje potrebe na nekoj drugoj lokaciji, na nekoj drugoj prezentaciji.

PROBLEMATIKA KREIRANJA HTML PREZENTACIJE

Autor mora da ima na umu da sadržaj web stranice (osnovne celine u prezentaciji) bude čitljiv na očekivani način od strane web-browsera koji posetilac koristi. Međutim, ovo je osnovni kamen spoticanja pri kreiranju i dizajniranju stranice putem klasičnog HTML pristupa jer sam prikaz stranice zavisi od mnogo faktora: koji tip browsera se koristi i koja verzija na kom operativnom sistemu, da li su fontovi koji se specificiraju na stranici instalirani na računaru posetioca, koju ekransku rezoluciju koristi posetilac...

HTML – standardizacija i konkurencija

Jezik HTML predstavlja jezik za opis podataka za prikaz sadržaja stranica web prezentacije. Cilj jezika je da omogući predstavljanje različitih sadržaja posetiocu prezentacije na način koji osmisli autor. Da bi se tome izašlo u susret, oktobra 1994. osnovan je web konzorcijum (www.w3c.org), čiji zadatak je da se na svetskom nivou stara o standardizaciji protokola, jezika i tehnologija od značaja za razvoj svetske mreže. Tako je, na primer, jedan od njihovih dokumenata i specifikacija jezika HTML, zvanično u nazivu "preporuka" (umesto specifikacija). Dokument sadrži opis elemenata jezika i načine upotrebe.

Upoznavanjem sa specifikacijom, autor bi trebalo da bude u stanju da, prateći preporuku, sprovede svoje ideje u projekat prezentacije i omogući na taj način da posetilac prezentacije može da vidi stranice projekta koje izgledaju tačno onako kako je autor osmislio.

U praksi, čitače HTML strana, koji su poznatiji pod nazivom web-browseri, proizvode različite kompanije čiji cilj je, pre svega, osvajanje tržišta i, kao posledica toga, ostvarenje profita. Stoga kompanije nude tržištu verzije programa koje zadovoljavaju

one aspekte jezika, koje sama kompanija smatra za bitne ili koje jeste uspela da realizuje, istovremeno nudeći neka druga rešenja čime se stiče prednost u odnosu na konkurentske kompanije i njihove programe.

Ovakva stvarnost autoru prezentacije stvara samo probleme. Autor mora poznavati tačno šta koja verzija kog programa može da pruži, ukoliko i dalje želi da obezbedi posetiocu svoje prezentacije prezentovanje sadržaja na željeni način. Iako savremeni programi za kreiranje web prezentacija poseduju određenju bazu znanja o ovoj problematici, krajnji rezultat je: ili povećanje količine podataka za opis sadržaja (zbog raznih ispitivanja o browseru klijenta i kreiranja različitih rešenja za isti prikaz), ili optimizacija sadržaja za pojedine browsere (čime se oštećuju posetioci koji koriste ostale), ili korištenje najjednostavnijih elemenata jezika, podržanih od svih browsera (gubi se atraktivnost naprednih elemenata).

Dobra ilustracija autorske muke jeste mogućnost korištenja klijentskih skript jezika u okviru HTML stranica. Ovi jezici omogućavaju dodavanje različitih efektnih rešenja za proširenje funkcionalnosti stranice kao, na primer, poboljšanje navigacionog sistema. I opet novi problemi: jezik VBScript prepoznaće samo Internet Explorer, a mogućnosti JavaScripta (u Internet Exploreru JScript) zavise od modela dokumenta koji poznaje posmatrana verzija posmatranog browsera. Znači, skriptuje se u JavaScriptu a "univerzalnost" se obezbeđuje sledećom konstrukcijom (pseudokod):

```
ako je Netscape
      ako je verzija manja od 4
            //    radi ovo
      inače
            ako je verzija 4.x
                  //    ovo za 4.x
            inače //    verzija je 6
                  //    i tada radi ovo
inače //    nije Netscape, znači IE
      ako je verzija....
```

Rezultat je različita realizacija istih zadataka u različitim uslovnim granama. Umanjuje se preglednost koda, povećava se veličina fajla, produžava se vreme učitavanja stranice, povećava se količina podataka koja klijentu nije od koristi...

Bitmapirana grafika

Grafički fajlovi – fotografije, crteži, navigacione slike i drugo, na HTML stranici najšire su prisutni u dva formata - .jpg i .gif. To nije slučajno, jer najbolje odgovaraju zahtevu da pruže što više informacija u što manjem fajlu. Jer, stranica je učitana tek onda kada su učitani svi njeni elementi. Znači da ukupna količina podataka, koja treba da stigne do klijenta da bi browser mogao ispravno da prikaže stranicu, nije manja od zbira veličina fajlova svih elemenata stranice. Takođe, svaki element putuje po posebnom zahtevu, što uslovljava određena kašnjenja u komunikaciji klijenta i servera i produžava vreme učitavanja. Keširanje elemenata koji se ponavljaju na različitim stranicama jeste od pomoći, ali ne i pri prvom pristupu prezentaciji (prva stranica jeste najbitnija u odluci posetioca hoće li ostati ili odustati od posete).

Animacija

Klasični video formati neprimenljivi su za primenu na Internetu zbog veličine zapisa fajlova. Iako je sve raširenija upotreba tipova zapisa koji koriste velike stepene kompresije podataka, ona ne obezbeđuje prikaz u realnom vremenu preko globalne mreže. Pojava streaming formata zapisa dozvoljava prikaz pre skidanja kompletnog fajla, mada je realno primenljiva samo pri korištenju zapisa manjih dimenzija (u pikselima) i na delovima mreže koji mogu da obezbede potreban protok za prikaz.

Interakcija na HTML strani ograničena je na elemente formulara i klijentsko skriptovanje sa ograničenom grafičkom kontrolom nad elementima stranice, a konkretna promena sadržaja stranice jeste odlazak na drugu stranicu (novi zahtev serveru)

ODGOVOR: FLASH

Flash je proizvod kompanije Macromedia čiji se fajlovi (filmovi) sa ekstenzijom .swf prikazuju u web-browseru putem plug-in dodatka. Flash plug-in Player je razvijen i razvija se za sve vodeće platforme (procesor + operativni sistem + browser). Format .swf je u formi open-source što omogućava njegovo brže i raširenije prisustvo u svetskim razmerama. Na naslovnoj stranici prezentacije firme Macromedia najsvežiji podatak je da 98.3% korisnika Interneta poseduje ovaj dodatak. Ovo praktično govori da Flash jeste postao standard u svetu globalne mreže i, uz činjenicu da većina novijih browsera pri samoj instalaciji postavljaju ovaj plug-in u svoje okruženje, da onaj "ostatak" uglavnom predstavljaju korisnisnici zastarelih verzija programa koji ionako ne mogu da udovolje zahtevima savremenih prezentacija.

Nezavisnost od platforme

Znači, svaki posetilac koji poseduje Flash Player ima mogućnost da pogleda .swf film jer ne zavisi od platforme sa koje pristupa fajlu. Dalje, postoji mogućnost da se svi fontovi potrebni za prezentaciju u Flashu uključe u dokument, tako da izgled ne zavisi ni od fontova instaliranih na računaru. Svi grafički elementi u Flashu, pa i fontovi, su vektorski opisani (ako se izuzme mogućnost uvoza bitmapiranih slika) tako da je moguće skaliranje celog filma prema prozoru browsera odnosno – nema zavisnosti od ekranske rezolucije.

Kako Flash ne zavisi od platforme, konstrukcije ispitivanja okruženja zbog prilagođenja prikaza nisu potrebne – svaka realizacija prikaza radi se tačno jednom.

Vektorska grafika

Opis vektorske grafike je jednostavan i nema gubitka podataka jer nema ni kompresije. Film je u jednom fajlu tako da server obrađuje samo jedan zahtev pri slanju filma klijentu. Ukupna veličina .swf fajla je manja od analogno tome rađene HTML stranice. I sledeće: Flash film može da počne da se prikazuje odmah po učitavanju prvih kadrova filma, pre nego što se kompletno učita kod klijenta.

Animacija i interakcija

Flash fajl naziva se film zbog mogućnosti animacije, osnovne prednosti na Internetu u odnosu na sva druga ponuđena rešenja.

Primer: animacija kvadrata koji se rotira po nekoj putanji i pritom menja svoju veličinu, pamti se kao objekat opisan pozicijom nekoliko tačaka, početnom i krajnjom pozicijom objekta, faktorom slaliranja objekta i parametrom broja okretaja tokom animacije. Ceo taj opis može biti smešten u desetak bajtova, bez obzira na veličinu kvadrata i broj slika koje učestvuju u celoj animaciji. Realizacija iste ove animacije u bitmapiranom svetu je niz potrebnog broja slika (dužina animacije), od kojih svaka slika kompletno opisuje sadržaj boje svojih tačaka (broj tačaka zavisi od dimenzija – širina puta visina) ili promenu sadržaja u relaciji na prethodnu sliku. Znači, dimenzije animacije i njena dužina značajno utiču na veličinu zapisa animacije.

Jedan od elemenata Flasha su objekati koji mogu da reaguju na određene događaje, među kojima i su korisnikove akcije unosa putem tastature ili akcije miša, koji dalje mogu da uslovljavaju dalje ponašanje u filmu. Kompletan sadržaj filma u Flashu može se menjati, kao odgovor na korisničke akcije, bez potrebe za dovlačenjem sadržaja sa servera, jer se celokupni sadržaj može se nalaziti u okviru samog filma. U okviru Flasha se za upravljanje kontrolom filma koristi jezik ActionScript, kome je u okviru ovog rada posvećena posebna pažnja.

1 FLASH

Flash Player je program za prikazivanje .swf filmova. Postoji u obliku plug-in dodatka za web-browsere, a takođe kao i samostalni program za Windows i MacIntosh platforme. Program za kreiranje filmova zove se Flash i takođe je realizovan za ove dve platforme. Najnovija verzija programa nosi naziv Flash MX (MX – predstavlja obaveštenje o 10 godina rada kompanije Macromedia) i to šesta verzija ovog programa.

1.1 ISTORIJAT FLASHA

1995. godine kompanija Future Wave objavljuje jednostavan vektorski-orijentisan program za ilustrovanje pod nazivom SmartSketch. Iste godine kreiran je i plug-in Future Splash Player za pregled ilustracija iz SmartSketcha. Naredne godine izlazi naredna verzija programa pod novim nazivom Cel Animator, sa mogućnošću kreiranja animacije – i ubrzo se program preimenovao u Future Splash Animator. Tada Macromedia uviđa u ovom programskom paketu (Animator + Player) ogroman potencijal i to je razlog kupovine kompanije Future Wave od strane Macromedije.

Prvu verziju Flasha Macromedia je objavila 1997. godine. Narednih godina izlazile su nove verzije programa uvodeći kvalitativne novine u domenu animacije i interakcije s korisnikom. Godine 2000. pojavljuje se Flash 5 uvodeći novi koncept u programerske mogućnosti paketa – skript jezik ActionScript. 15. marta 2002. Flash MX dobija zadatak da opravda renome svojih predhodnika i predstavi nove mogućnosti korišćenja u svetu web-aplikacija.

Izlaganje ovog rada odnosi se na program Flash 5, dok svi izuzeci sadrže napomenu.

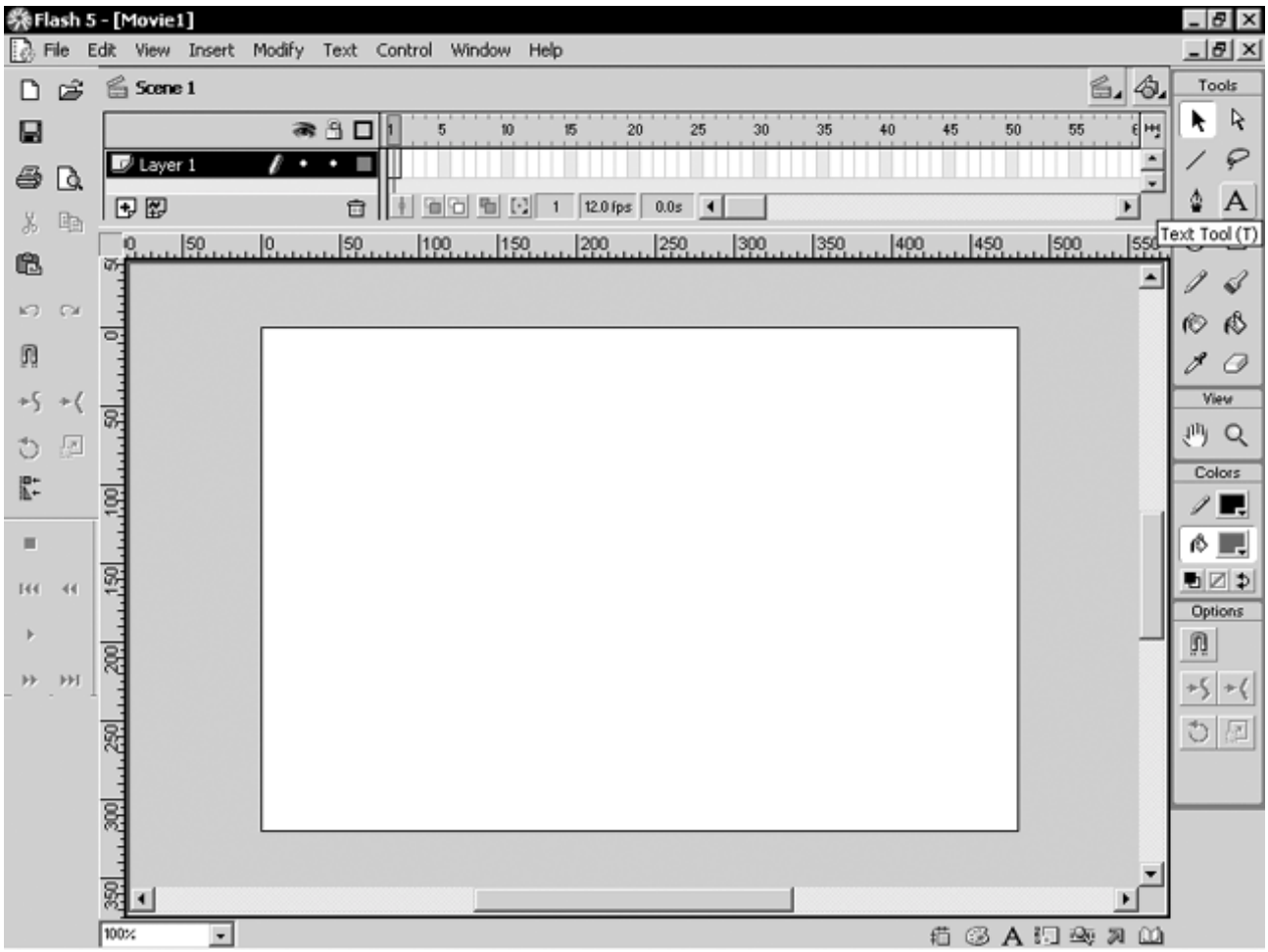
1.2 PROGRAMSKO OKRUŽENJE

Flash film koji se pravi u Flashu snima se kao .fla dokument. Radno okruženje je podešeno za kreiranje grafičkih elemenata i njihovu animaciju u vremenu. Elementi okruženja su:

- Pozornica
- Vremenska osa
- Kutija alata
- Biblioteka simbola
- Paneli

1.2.1 Pozornica

Centralni deo radnog okruženja dodeljen je pozornici (stage). Ona odslikava "svet" dvodimenzionalnog koordinatnog sistema sa početkom u gornjem levom uglu, čije se dimenzije (u pikselima) određuju kao globalne za ceo film. Na slici dole, pozornica je centralni beli pravougaonik.



Sam film može da se organizuje po scenama. Tako je pri kreiranju praznog dokumenta kreirana i aktivna prva scena (Scene 1). Na slici se ovo vidi pri gornjem levom uglu, ispod menija (slika filmske "klape" i natpis "Scene 1").

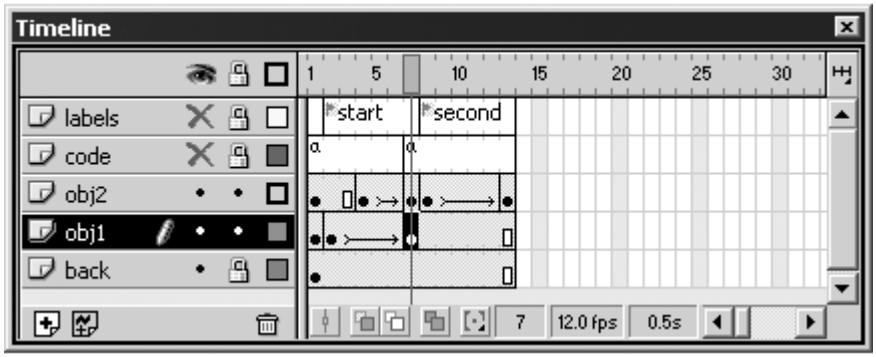
1.2.2 Vremenska osa

Svakoj sceni pridružuje se vremenska osa (timeline). Ona dodaje treću prostornu dimenziju kroz koncept lejera (leyers) i vremensku komponentu, preko frejmova.

Lejeri se mogu posmatrati kao providne folije naslagane jedna na drugu, određujući tako šta se nalazi "ispod" a šta "iznad". Primer: ako imamo dva lejera sa nekim sadržajem koji se na nekom delu scene preklapa, u preklopljenom delu biće vidljiv sadržaj gornjeg lejera jer on je "iznad" donjeg.

Pojam frejma poznat je iz filmske terminologije, predstavlja stanje u nekom posmatranom trenutku. Vremenska komponenta je diskretizovana na frekvenciju frejmova po sekundi, što se takođe globalno određuje na nivou filma.

Ključni frejm je frejm sa nekim definisanim stanjem elemenata određenog lejera. Između dva ključna frejma u lejeru može se zadati animacija elemenata lejera, odnosno distribucija promena po frejmovima između uočenih ključnih frejmova. Ovaj pojam koristi se pri izradi animiranih filmova, gde glavni crtači crtaju ključne slike (frejmove), a animatori dovršavaju posao crtanjem međuslika.



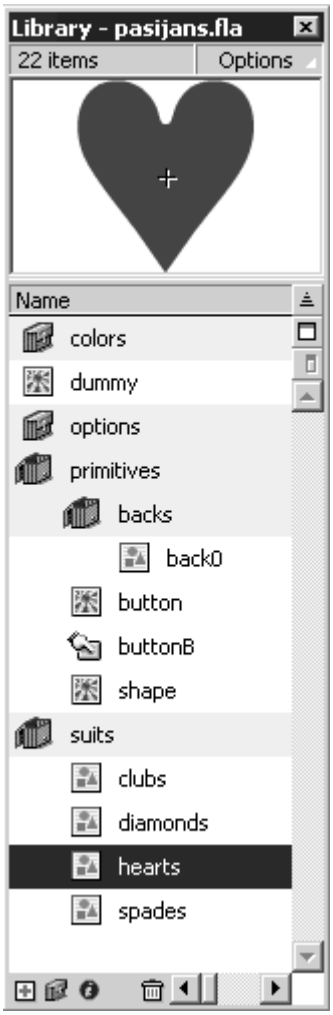
Za ilustraciju ove priče primer je slika iznad. Postoji pet lejera, nazvani `back`, `obj1`, `obj2`, `code` i `labels`. Lejer `back` je najdonji, iznad njega je `obj1`, i tako redom do lejera `labels`, koji je najgornji. Selektovan je lejer `obj1`. Lejeri `labels` i `code` nisu vidljivi (oznaka "x" ispod "oka") i zaključani su (katanac), kao i lejer `back`. Kako se lejer `obj2` nalazi iznad `obj1`, čiji elementi trenutno dizajniraju, to se kod lejera `obj2` prikazuju samo obrisi oblika elemenata (kvadratić isključuje vidljivost popune) kako bi elementi lejera `obj1` mogli biti vidljivi i pored eventualnih preklapanja. Film zauzima 13 frejmova, a u donjem redu očitava se da je aktivan sedmi frejm, da se film podešen za prikazivanje 12 frejmova u sekundi (fps) i da trenutna pozicija predstavlja 0.5 sekundi od početka filma. Na samom sedmom frejmu prisutna je vertikalna crta koja označava selektovani frejm. U preseku selektovanog lejera i frejma je selektovan ključni frejm (popunjen kružić) lejera `obj1`. Između nekih ključnih frejmova nalazi se strelica, koja predstavlja animaciju elemenata lejera između uočenih frejmova (program u frejmovima strelice izračunava promenu elemenata lejera u odnosu na ključne frejmove). U lejeru `code` vide se ključni frejmovi sa oznakom "a", što ukazuje da ti frejmovi sadrže neke akcije ActionScripta. U lejeru `labels` dva ključna frejma imaju zastavice i tekst koji predstavlja labelu (oznaku) frejma, koja olakšava prepoznavanje delova filma i čije ime se može koristiti pri zadavanju komandi ActionScripta za kontrolu toka filma (na primer, može se narediti skok na poziciju imenovanog frejma).

1.2.3 Kutija alata

Za rad u filmu, sceni, lejeru i frejmu koriste se razne alatke iz kutije alata (slika desno), koja je podeljena na četiri dela: alati za crtanje, vizuelno pozicioniranje, selektori boja i opcije selektovane alatke.

Poput drugih programa za rad sa vektorskom grafikom i u Flashu su "standardne" alatke za crtanje: selektor, linija, pravougaonik, elipsa, slobodno crtanje, gumica... Postoje dva selektora za boje: za boju linije (konture) i za boju popune konture. Za popunu se, pored osnovnih boja, mogu odabrati i radijalno ili linearno nijansirani prelazi boja. Boja pozadine određuje se globalno na nivou filma.

1.2.4 Biblioteka simbola



Crtanje grafičkih elemenata može se izvoditi direktno na sceni i tom prilikom kreiraju se oblici (shapes). Drugi prisup jeste kreiranje simbola. Simboli su elementi koji se mogu više puta koristiti. Poseduju sopstvenu vremensku osu. Opis simaola čuva se i organizuje u okviru biblioteke simbola (Library).

Na slici levo prikazana je biblioteka simbola koja se koristi u okviru prvog primera, program pasijans. Sa slike se vidi da svaki simbol ima svoje ime, kao i da se simboli mogu organizovati po folderima simbola (na slici su neki folderi zatvoreni, npr. `colors`, dok su neki otvoreni – `suits`). Različite sličice levo od imena simbola označavaju tip simbola.

Postoji tri osnovna editabilna tipa simbola: filmski klip (Movie Clip), dugme (Button) i grafike (Graphic). Na slici je simbol `dummy` tipa Movie Clip, `buttonB` tipa Button, a `back` tipa Graphic. Postoje još i tipovi uvoznih simbola – slike, zvučni i video zapisi.

Simbol hearts je selektovan, a u gornjem prozorčetu prikazan je njegov izgled.

Postavljanje simbola na scenu u stvari je njegovo instanciranje. Instanci simbola moguće je menjati razne osobine (pozicija, skaliranje, rotacija, transparencija,...) bez uticaja na definiciju simbola. Menjanje osobina u definiciji simbola odražava se na svim njegovim instancama.

Simbol tipa dugme reaguje na događaje miša – kada se pokazivač miša nalazi iznad simbola i kada je pritisnuto dugme miša dok je pokazivač iznad simbola. Zato vremenska osa simbola tipa dugme sadrži četiri specijalna frejma:

- `up` – stanje simbola kada se pokazivač miša ne nalazi iznad simbola
- `over` – pokazivač je iznad simbola
- `down` – pokazivač je iznad i pritisnuto je levo dugme miša
- `hit` – ovaj frejm određuje površinu na osnovu koje simbol određuje svoje stanje

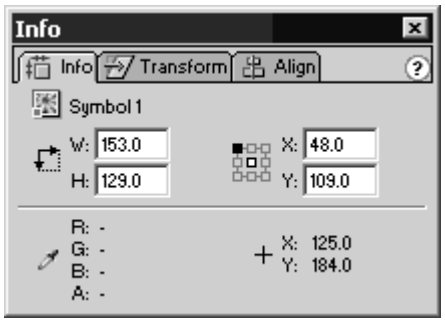
Grafički simbol može biti animiran, ali se njegova vremenska osa vezuje za scenu (ili filmski klip, ako se u njemu instancira), tako da sve promene toka (po frejmovima) roditeljskog elementa utiču i na instancu grafičkog simbola. Tako, na primer, ako se u

nekom trenutku zaustavi animacija na sceni, zaustavlja se i animacija svih instanci simbola tipa grafika.

Drugačije ponašanje u odnosu na ovo realizuju simboli tipa filmski klip. Gledano na istom primeru, zaustavljanje animacije na sceni ne utiče na ponašanje instanci filmskih klipova postavljenih na scenu već naprotiv, one nastavljaju tok svoje vremenske ose. Pored ovoga, svakoj instanci simbola filmskog klipa može se dodeliti jedinstveno ime instance, kojim se potom referencira instanca u akcijama ActionScripta.

1.2.5 Paneli

Za pregled, organizaciju i modifikaciju elemenata Flash filma koriste se paneli sa komandama i opcijama vezanim za posmatrani tip elementa. Mogu se modifikovati simboli, instance, boje, tekst, frejmovi i drugi elementi filma.



Slika levo predstavlja jedan panel. Paneli se mogu organizovati u grupe, tako da se na ovoj slici u stvari nalaze tri panela, od kojih je aktivan (jezičak gore) panel `Info`. Panel svoj prikaz vezuje za selektovani element scene, frejm i slično, i nudi opcije koje se mogu vršiti nad selekcijom.

* * *

Pored pomenutih elemenata okruženja, postoje meniji, toolbari (standardni i kontroler toka filma), kao i dodatni alati za testiranje filma, dibager i izlazni prozor.

1.3 RAD U FLASHU

Da bi autor ostvario svoj projekat filma u Flashu, koristi se različitim tehnikama editovanja sadržaja svog rada. U nastavku su predstavljeni osnovni koncepti rada koje nudi okruženje.

1.3.1 Početak rada

Otvaranjem programa, ili zadavanjem komande za novi projekat, kreira se inicijalni dokument u kome je definisana i aktivna prva scena (Scene 1), koji sadrži jedan lejer (Layer 1) sa jednim praznim frejmom. Dimenzije scene, broj slika u sekundi i boja pozadine uzimaju vrednosti koje su u programu postavljene za podrazumevane. Ovi parametri mogu se menjati u bilo kom trenutku rada, a od trenutka promene postaju globalni za ceo film.

1.3.2 Crtanje

Grafički elemenati mogu biti kreirani korišćenjem alata za crtanje.

Crtanje se odvija u okviru selektovanog ključnog frejma selektovanog lejera, u okviru elementa filma koji se edituje. Element filma može biti scena ili simbol.

Odabirom alatke (linija, elipsa, pravouganik, slobodno crtanje...) određuje se šta će se crtati, selekcijom boja određuju se boje linije i popune elementa koji se crta, a u panelima moguće je odrediti tip i debljinu linije, način popune (čista boja, radijalni ili linearni gradijent, bitmapa) i procenat transparencije boja. Sve osobine moguće je i naknadno menjati selekcijom željenih kontura ili popuna i promenom vrednosti osobine u određenom panelu ili selektoru boje.

Primer: na slici levo je elipsa sa podebljanom isprekidanom linijom i radijalnom popunom

Za pomoć pri crtanju koriste se rešetka (grid), vođice (guidelines), koje se vuku sa lenjira (ruler), i već postojeći objekti na sceni (ili u definiciji simbola). Postoji mogućnost vezivanja (snap) ključnih tačaka elementa koji se crta za neki od pomenutih elemenata, a takođe se određuje i stepen vezivanja (koliko pokazivač miša može biti udaljen od elementa da bi došlo do vezivanja). Parametri rešetke, horizontalni i vertikalni razmak između linija rešetke, menjaju se po potrebi.

Pozicioniranje elementa moguće izvodi se poravnavanjem u odnosu na druge elemente ili u odnosu na pozornicu, kao i distribucija elemenata, levo, desno ili centrirano.

Pozicija elementa može se i numerički odrediti, kao i dimenzije, a moguće je izvoditi i relativnu transformaciju dimenzija (procentualno), rotacije i zakošenja (skew).

1.3.3 Uvozni elementi

Vektorski elementi pozornice mogu se dobiti i uvozom vektorske grafike urađene u nekom drugom programu. Ovo je naročito korisno pri potrebi za zahtevnijim grafikama, koje je ipak lakše realizovati u specijalizovanim programima (Adobe Illustrator, Macromedia Freehand, Corel Draw i slično). Uvežen vektorski element može se dalje obrađivati na isti način kao i grafika crtana u Flashu.

Pored uvoza vektorskih slika, mogu se uvoziti bitmapirane slike koje se, kao bitmapirani simboli, smeštaju pri uvozu u biblioteku, dok se instanca postavlja na pozornicu. Instanci se mogu menjati pozicija, veličina, rotacija i zakošenje.

Kao i kod bitmapiranih slika, zvučni i video zapisi takođe uvozom postaju elementi biblioteke simbola.

1.3.4 Animacija

Pri dizajniranju filma mogu se koristi dve vrste animacije: animacija oblika (shape tween) i instanci simbola ili grupisanih objekata (motion tween). Zajedno za obe vrste animacije je da se definiše između dva ključna frejma (početnog i krajnjeg frejma animacije) istog lejera i definisanje dinamike animacije (easing – pozitivne vrednosti daju bržu promenu u na početku animacije, negativne obratno).

Animacija oblika predstavlja jednostavno pretapanje oblika između uočenih ključnih frejmova. Ne može se primenjivati na instance simbola i grupisane objekte, već samo na oblike. Primer bi bio transformacija pravougaonika u elipsu: u prvom ključnom frejmu crta se pravougaonik, u drugom elipsa, i na kraju se kod prvog ključnog frejma zadaje animacija (tweening) postavljanjem na "shape". O frejmovima između stara se Flash i izračunava međuoblike.

Motion tween, slično, zahteva samo jednu (istu) instancu simbola ili grupu simbola prisutnu na oba ključna frejma lejera. Sve promene vrednosti osobina elementa ključnih frejmovima se pri zadavanju animacije (tweening na "motion") distribuiraju po međufrejmovima. Pri zadavanju može se odrediti i dodatna rotacija elementa (smer i broj okretaja) tokom animacije. Naprednija varijanta ove animacije postiže se tako što se lejeru pridruži vodeći lejer (motion guide). Na vodećom lejeru iscrtava se samo putanja elementa. Element se u ključnim frejmovima vezuje za početnu i krajnju tačku putanje. Pri ovakvom zadavanju animacije određuje se da li se element tokom animacije orijentiše i po putanji, ili ne. Sadržaj vodećeg lejera se ne prikazuje pri testiranju filma.

1.3.5 Maskiranje

Sadržaj nekog lejera može maskirati oblikom u lejeru iznad, ako se gornji lejer proglasi za maskirni.

Primer: na slici levo nalaze se dva potpuno ista pravougana oblika, na različitim lejerima, od kojih je jedan maskiran lejerom koji sadrži elipsu.

1.3.6 Tekst

Na pozornicu se mogu dodavati i tekstualni elementi. Dodaju se kao tekstualna polja koja se definišu kao:

- statička,
- dinamička ili
- ulazna

Statičko polje određuje svoj sadržaj kucanjem potrebnog teksta tokom editovanja u samo polje.

Dinamičko polje vezuje svoj sadržaj za neku promenljivu koja se definiše u kodu ActionScripta, ovo polje prikazuje vrednost promenljive. Polje je definisano kada se odredi ime promenljive kojoj se polje pridružuje.

Slično je i sa ulaznim poljem, sadržaj se vezuje za promenljivu. Razlika što se korisniku dozvoljava da menja sadržaj polja, time i vrednost promenljive.

Tekstualna polja mogu zauzimati jednu liniju ili se prostirati na više linija. Slično kao i u programu Word, moguća su razna podešavanja sadržaja polja po osobinama paragrafa (centriranje, margine, uvlačenje, razmak između redova) i karaktera (tip fonta, veličina, boja, stil, definisanje hiperlinka).

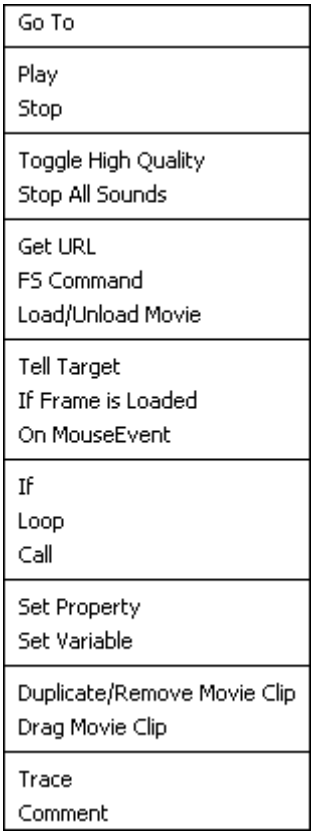
1.3.7 Kodiranje

Više puta pomenuti jezik ActionScript koristi se u Flashu za programiranu kontrolu toka filma. Elementi jezika nazivaju se akcije. Kod ActionScripta može se pridružiti kjučnom frejmu i instancama simbola tipa dugme i filmski klip.

ActionScript je predstavljen u narednom poglavlju.

2 ACTIONSCRIPT

Slika desno predstavlja spisak akcija u Flashu 4 kojima je bilo moguće uticati na tok filma. Akcije su se pridruživale određenom frejmu ili događaju nad simbolom tipa dugme. Iz spiska se vidi da su mogućnosti programiranja pre svega bile orijentisane na upravljanje tokom filma (zaustavljanje, kretanje, skokovi na određene frejmove), uz skromno prisustvo komandnih struktura naredbe grananja i loop petlje. Naredba Call imala je za zadatak poziv bloka komandi pridruženih nekom frejmu, simulirajući na taj način korisničku funkciju. Ovakav pristup sigurno da nije omogućavao komotan rad s programerske tačke gledišta, ali je doveo do pomeranja sa akcenta programa sa animacije ka interakciji, što je za rezultat imalo pojavu šireg spektra aplikacija rađenih u Flashu.



Peta verzija programa predstavlja kvalitativni pomak u pravcu programiranja, odlučeno je da Flash postane ozbiljan alat za programiranje. U Flashu se pojavljuje ActionScript.

ActionScript je skript jezik čiji je cilj kontrola objekata u Flash filmovima, kreiranje navigacionih i interaktivnih elementa i za kreiranje filmova visokog stepena interakcije i web aplikacija.

2.1 SKRIPT JEZIK

Skript jezik je programski jezik koji se koristi za manipulaciju, prilagođenje i automatizaciju postojećeg sistema. U takvim sistemima, funkcionalnost je već obezbeđena putem korisničkog interfejsa, a skript jezik predstavlja mehanizam za proširenje funkcionasti programskom kontrolom. Za postojeći sistem se kaže da obezbeđuje domaćinsko okruženje za objekte i mogućnosti koje kontroliše skript jezik.

Asocijacija evropskih proizvođača kompjutera (ECMA – the European Computers Manufacturers Association, www.ecma.ch) sastavila je dokument pod nazivom ECMA-262 čiji je cilj standardizacija jezika JavaScript. ActionScript zasniva se na specifikaciji ovog dokumenta.
-:- http://www.ecma.ch/ecma1/STAND/ECMA-262.HTM

2.2 ELEMENTI JEZIKA

Sintaksa ActionScripta podseća na Javu. Razlika je u tome što jezik upućuje na jednostavnost u korišćenju. Na primer, promenjiva ne mora da ima deklarisan tip niti se tipovi povezuju sa osobinama objekta, a definisane funkcije ne moraju biti deklarisane u tekstu pre njihovog poziva.

ActionScript se zasniva na objektima: osnovni jezik i okruženje Flasha obezbeđuju se putem objekata, a sam program u ActionScriptu je put komunikaciji među objektima. Sam objekat je kolekcija osobina (properties) koje mogu imati atribute koji određuju način korišćenja osobine (na primer – ReadOnly). Osobine su kontejneri koji sadržavaju druge objekte, primitivne vrednosti ili metode. Objekat u ovom smislu predstavlja referencu na objekat, primitivna vrednost je vrednost nekog primitivnog tipa podataka, a metod je funkcija koja se s objektom povezuje putem osobine.

2.2.1 Tipovi podataka i promenljive

Tip podataka predstavlja vrstu informacije koju promenljiva ili element ActionScripta može da sadrži. Tipovi podataka dele se na primitivne i referentne tipove, a postoje i specijalni tipovi podataka.

- Primitivni tipovi su `string`, `number` i `boolean`
- Referentni tipovi su `object`, `function` i `movieClip`
- Specijalni tipovi podataka: `null` i `Undefined`

Referentni tipovi kao vrednost imaju adresu svog sadržaja. Tip `movieClip` je grafička reprezentacija elemenata Flasha.

```
var myVar;
trace (“typeof (myVar): ” + typeof (myVar));    //    undefined
myVar = true;
trace (“typeof (myVar): ” + typeof (myVar));    //    boolean
myVar = 0;
trace (“typeof (myVar): ” + typeof (myVar));    //    number
myVar = “my String”;
trace (“typeof (myVar): ” + typeof (myVar));    //    string
myVar = new Array ();
trace (“typeof (myVar): ” + typeof (myVar));    //    object
myVar = trace;
trace (“typeof (myVar): ” + typeof (myVar));    //    function
myVar (“typeof (myVar): ” + typeof (myVar));    //    function
```

Promenljiva se može deklarisati sa `var`. Deklaracija određuje njeno ime i lokalnost, odnosno promenljiva tada postaje lokalna u bloku u kome je deklarisana. U navedenom primeru korištene su funkcije `trace`, koja služi za ispis string vrednosti izraza svog argumenta u posebni izlazni prozor (korisno pri testiranju programa), i `typeof`, koja vraća naziv tipa izraza svog argumenta. Dvostruka kosa crta (`//`) predstavlja početak linijskog komentara, pri interpretaciji se ignorišu svi karakteri u liniji nakon ove oznake (komentar u više linija postavlja se između oznaka `/*` i `*/`). Prvi poziv funkcije `trace` u izlazni prozor ispisuje “typeof (myVar): undefined” jer promenljivoj nije dodeljena vrednost, tako da joj nije određen ni tip. Dodela vrednosti određuje tip promenljive. To znači da promenljiva može da menja svoj tip. U primeru, poslednjom dodelom (`myVar = trace`) je promenljivoj dodeljena funkcija `trace`,

znači ona je tipa `function`, ona postaje funkcija koja se ponaša na isti način kao i `trace`, tako da poslednja linija poziva promenljivu `myVar` kao funkciju i dobija se takođe odziv u izlaznom prozoru.

Promenljiva se ne mora deklarisati. U tom slučaju smatra se za globalnu.

ActionScript vrši automatsku konverziju tipova pri izvođenju operacija, tako da nedefinisane promenljive mogu učestvovati u operacijama, gde se prevode kao neutralni element sabiranja (logičko – `false`, numerički – `0`, string – `""`).

2.2.2 Operatori

ActionScript definiše skup ugrađenih operatora koji se mogu podeliti u sledeće grupe: numerički operatori, operatori poređenja, String operatori, logički operatori, bit-orijentisani operatori, operatori jednakosti i operatori dodele. Kao i u drugim programskim jezicima, operatori su hijerarhijski određeni po prioritetu izvršavanja operacija.

Zanimljivi operatori su, recimo, operatori dodele, što ilustruje sledeći primer:

```
var myString = "my";
myString += " String";
trace ("myString: " + myString);    //     "myString: my String"
```

Prva linija deklariše promenljivu `myString` i dodeljuje mu ("obična" dodela) String vrednost "my". U drugoj linija koda takođe je dodela, ali sa operatorom `+=`, što semantički predstavlja sledeće: saberi vrednost operanda s leve strane operatora (promenljivu `myString`) sa operandom s desne strane (" String") i rezultat sabiranja dodeli promenljivoj s leve strane.

Drugi primer ilustruje operatore inkrementacije (`++`):

```
var myVar = 0;
trace ("myVar: " + myVar);          //    "myVar: 0"
trace ("myVar++: " + myVar++);      //    "myVar++: 0"
trace ("myVar: " + myVar);          //    "myVar: 1"
trace ("++myVar: " + ++myVar);      //    "++myVar: 2"
trace ("myVar: " + myVar);          //    "myVar: 2"
```

Post-inkrementacija povećava vrednost promenljive nakon njenog učestvovanja u izrazu, a pre-inkrementacija prvo povećava vrednost promenljive koja daje svoju novu vrednost za izračunavanje izraza.

Treći primer, uslovna dodela:

```
var myBool = true;
var notBool = !myBool;
trace ("myBool: " + ((myBool) ? myBool : notBool));
trace ("notBool: " + ((notBool) ? myBool : notBool));
```

Uslovna dodela je ternarni operator. Prvi operand je logički izraz koji se izračunava. Ako je izračunao `true`, operator računa vrednost drugog operanda (iza znaka pitanja) i to vraća kao rezultat, inače (izračunato `false`) se računa vrednost trećeg operanda (iza dvotačke) i to postaje rezultat. U ovom primeru prvi poziv trace ispisuje "myBool: true", jer se ispituje uslov `myBool` (`true`) i vraća vrednost drugog operanda (`myBool`

ima vrednost `true`), dok drugi poziv ispisuje "notBool: false" jer je uslov netačan (`notBool`, pri dodeli je `!myBool`, negacija od `true`) pa se vraća vrednost trećeg operanda (`notBool` je `false`).

2.2.3 Grananje i petlje

Naredba `if` u ActionScriptu ima sledeću sintaksu:

```
if (condition) {
      //    statements
} else {
      //    statements
}
```

Ovde je `condition` logički izraz, obavezno naveden u zagradama, na osnovu kojeg se donosi odluka da li se izvršava kod bloka naredbi unutar vitričastih zagrada. Blok naredbi se uvek navodi unutar vitričastih zagrada, a pojedine naredbe u bloku razdvajaju se tačka-zarezom (`;`). U slučaju samo jedne naredbe nije neophodno navođenje vitričastih zagrada. Druga grana (`else`) takođe nije obavezna. U ActionScriptu kod Flasha 5 ne postoji `switch` naredba za višestruko grananje, a to je jedna od novina koje donosi nova verzija programa, Flash MX.

Ponavljanje dela koda obezbeđeno je petljama `for`, `while`, `do-while` i `for-in`. U svim petljama mogu se koristiti komande `continue`, za skok na sledeću iteraciju, i `break`, za napuštanje petlje.

```
for (init; condition; next) {
      //    statements
}
```

Slično kao i u jeziku Java, `for` petlja zadaje se s tri parametra: inicijalna vrednost kontrolne promenljive, izlazni uslov i izraz za izračunavanje pre naredne iteracije. Po definiciji je dozvoljeno izostavljanje bilo kog od ova tri dela (ostaje tačka-zarez). Izraz `init` se izračunava samo jednom, pre početka prve iteracije, izraz `condition` mora biti logički i izračunava se pre svake iteracije i odlučuje da li se nastavlja sa iteracijom (izračunato `true`) ili ne (`false`), izraz `next` izračunava se na kraju svake iteracije. Ovakva konstrukcija dozvoljava mnogo veću slobodu pri baratanju petljom u odnosu na, recimo, paskaloliku konstrukciju. Ipak, uobičajeno se koristi baš simulacija te osnovne konstrukcije, postavljanjem inicijalne vrednosti kontrolne promenljive u prvom delu, poređenje kontrolne promenljive s nekom izlaznom vrednošću u drugom delu i inkrementiranje ili dekrementiranje kontrolne promenljive u trećem delu.

```
while (condition) {
      //    statements
}
```

Dogod je uslov `condition` ispunjen izvršava se blok naredbi petlje `while`. Ovo ispitivanje može biti i numeričko, u kom slučaju se ispituje različitost od vrednosti `0`.

```
do {
      //    statements
} while (condition)
```

Slično je i kod petlje `do-while`, s razlikom što se prvo izvršava blok naredbi, pa tek onda proverava uslov (što znači, kod bloka izvršava se bar jednom). Ovo podseća na

paskalsko `repeat-until`, a razlika je kod uslova – `do-while` radi dok je ispunjen uslov, a `repeat-until` prekida po ispunjenju uslova.

```
for (variableiterant in object){
      //    statements
}
```

Petlja `for-in` koristi se osobinama objekta. Primer:

```
myObject = {name:'Tara', age:27, city:'San Francisco'};
for (name in myObject) {
      trace ("myObject." + name + " = " + myObject [name]);
}
```

Primer je iz rečnika ActionScripta 5. Izlaz izgleda ovako:

```
myObject.name = Tara
myObject.age = 27
myObject.city = San Francisco
```

Prva linija koda daje jedan način zadavanja objekta. Objekat `myObject` ima 3 osobine, svakoj je dodeljena određena vrednost. Promenljivoj name u `for-in` petlji pridružuju se imena osobina objekta `myObject`, redom kroz iteracije. Primer takođe prikazuje dva načina pristupa vrednostima osobina objekta. U izlaznom prozoru prikazan je prikaz referenciranja "tačkom", odnosno navodi se ime objekta, tačka i sledi ime osobine. To i jeste uobičajen način referenciranja, ali nije bilo moguće unutar `for-in` petlje pozvati `myObject.name`, jer bi se to odnosilo na osobinu name objekta, a ne promenljivu petlje. Stoga je korišćena referenca `myObject [name]`, koja izračunava izraz u uglastim zagradama (vrednost promenljive `name`) i potom traži vrednost odgovarajuće osobine objekta.

2.2.4 Funkcije

Definicija funkcije:

```
function myFunc ([arg0, arg1,...argN]) {
      //    statements
}
//    ili
function ([arg0, arg1,...argN]) {
      //    statements
}
```

Funkcija je blok koda koji se može više puta koristiti pozivom funkcije. Primer poziva bio bi `myFunc ([par1, par2,...parN])`. U zagradi ovde nalaze se parametri funkcije, koje ona pri izvršavanju ona prima kao argumente. Uglaste zagrade su u primeru navedene u smislu BNF notacije, odnosno ukazuju na opcionu pojavu parametara i argumenata. Funkcija ove parametre vidi kao vrednosne parametre, nema varijabilnih parametara, te se ne mogu menjati vrednosti ovih parametara van lokalnosti funkcije, uz dužnu pažnju referentnim podacima, gde se može direktno pristupati osobinama objekta, na primer.

Ulazne parametre funkcija vidi kao niz `arguments`, broj deklarisanih imenovanih parametara funkcije ne mora da se poklapa sa brojem argumenata koji se prosleđuju funkciji. Pristup imenovanim parametrima u funkciji moguć je direktno navođenjem

imena parametra, a pristup proizvoljnom parametru ostvaruje se kao pristup elementu niza `arguments` na određenom indeksu. Primer:

```
function myFunc () {
      var myVar;
      for (var i = 0; i < arguments.length; i++) {
            myVar += arguments [i];
      }
      trace (“myVar: ” + myVar);
}
myFunc (3, 4, “to”, 1, 1);    //    “myVar: 7to11”
```

U primeru se lokalna promenjiva `myVar` samo deklariše, inicijalno nema vrednost, znači tipa je `undefined`. Prvo sabiranje je sa argumentom tipa `number`, pa se promenljiva konvertuje u vrednost `0`, sabira sa `3` i dobija vrednost `3`. Sabiranje sa drugim argumentom je sabiranje dva broja, pa `myVar` dobija vrednost `7`. Treći argument je tipa `string`, što povlači konverziju `myVar` u `string`. Četvrti i peti argument su brojevi, ali se konvertuju u `string`. Tako `myVar` na kraju dobija vrednost “7to11”.

Funkcija može da vrati vrednost korišćenjem naredbe `return returnValue`, ovde `returnValue` predstavlja izraz čiju vrednost funkcija vraća. Korišćenjem same naredbe `return` (bez izraza) samo se napušta funkcija. Naredba `return` je opciona u funkciji, a može se pojavljivati na više mesta u kodu funkcije (obično nakon ispunjenja nekog uslova), vraćajući različite vrednosti, čak i različitog tipa. Primer:

```
function returner (inputValue) {
      if (inputValue < 0) return;
      if (inputValue == 0) return false;
      if (inputValue == 1) return 0;
      if (inputValue == 2) return “”;
      if (inputValue == 3) return new Array ();
      if (inputValue == 4) return trace;
}
for (var i = -1; i < 6; i++) {
      trace (i + “: ” + typeof (returner (i)));
}
/*    output:
-1: undefined
0: boolean
1: number
2: string
3: object
4: function
5: undefined
*/
```

Ovde je izbegnuto duboko ugnježđavanje `if-else`, jer se iz funkcije izlazi po prvom ispunjenju nekog uslova.

U ActionScriptu postoje predefinisane funkcije, a mogu se definisati i nove. Često se funkcija pridružuje nekom objektu, tada ona postaje metod objekta.

2.2.5 Objekti

Objekat je skup osobina koje mogu imati vrednosti nekog tipa. Svaki objekat jedinstveno se identifikuje svojim imenom. Već pomenuti primer:

```
myObject = {name:'Tara', age:27, city:'San Francisco'};
```

Objekat `myObject` ima tri osobine i svakoj osobini dodeljena je vrednost. Pristup osobini `age`, na primer, može se ostvariti sa `myObject.age` ili `myObject ["age"]`. Dodavanje nove osobine moguće je: `myObject.state = 'California'`, kao i modifikacija postojeće osobine: `myObject.age = 28`.

```
function person (name, age, city) {
      this.name = name;
      this.age = age;
      this.city = city;
}
myObject = new person ('Tara', 27, 'San Francisco');
```

Ovaj kod kao krajnji rezultat daje takođe objekat `myObject` sa potpuno istim osobinama i vrednostima osobina kao i prethodni primer. Razlika postoji. Sada je `myObject` instanca klase `person`, a funkcija `person` naziva se konstruktorom klase `person`. Objekat se instancira pozivanjem ključne reči `new` i konstruktora. Ključna reč `this` u konstruktoru referencira na objekat koji se kreira. Sledeći primer dodaje metod klasi person:

```
function showPerson () {
      var myString = this.name + ", " + this.age + ", " + this.city;
      trace (myString);
}
person.prototype.show = showPerson;
myObject.show (); //    "Tara, 27, San Francisco"
```

Sve funkcije poseduju osobinu `prototype` koja se automatski kreira pri definiciji funkcije. Pri kreiranju novog objekta konstruktorom, sve osobine i metode konstruktorske osobine `prototype` postaju osobine i metode osobine `__proto__` novokreiranog objekta. Kada se referencira neka osobina ili metoda objekta, ActionScript traži postoji li takva osobina ili metoda, ako ne postoji, traži se u okviru osobina i metoda njegovog `__proto__` objekta (i dalje `__proto__.__proto__`, ...). Ovo je nasleđivanje u ActionScriptu. Uobičajena praksa je dodeljivanje metoda osobini `prototype` konstruktora, kao što i stoji u navedenom primeru.

ActionScript već dolazi sa dobrim skupom predefinisanih objekata.

Postoji skup ugrađenih objekata preuzetih iz ECMA specifikacije:

- `Object` – najopštija klasa objekata
- `Array` – niz, sa metodama manipulaciju nizom, dodavanje i uzimanje elementa niza sa početka ili kraja, sortiranje itd, osobina `length` je dužina niza...
- `String` – metode za manipulaciju stringovima, podniz, konkatenacija...
- `Boolean`
- `Number` – konstante, najmanja i najveća vrednost, beskonačnost...
- `Math` – aritmetičke i trigonometrijske funkcije, matematičke konstante...
- `Date` – datum i vreme

Pored ovih, ugrađeni su i bitni objekti za Flashovo okruženje:

- `MovieClip` – osobine od značaja za vizuelnu reprezentaciju objekta, širina, visina, horizontalna i vertikalna pozicija, vidljivost, providnost, rotacija...; metode za kontrolu osobina, za kretanje po vremenskoj osi, kreiranje i uništavanje instanci, prevlačenje objekata...
- `Selection` – kontrola tekstualnih polja
- `Mouse` – skrivanje i pokazivanje pokazivača miša
- `Key` – konstante specijalnih tastera tastature, kodovi karaktera...
- `Color` – transformacije boje MovieClip objekata...
- `Sound` – kontrola zvuka i zvučnih efekata
- `XML` – učitavanje, slanje, parsiranje, izgradnja XML dokumenata...
- `XMLSocket` – stalna veza sa serverom, uspostavljanje i prekid veze...

U prvom primeru, pasijans, predstavljen je deo ponašanja koje se realizuje objektom `movieClip`, a treći primer, web memorija, predstavlja objekte `XML` i `XMLSocket`.

2.3 PROGRAMIRANJE U ACTIONSCRIPTU

Programski elementi, akcije, u Flashovom filmu mogu se nalaziti na više mesta. Kako se film odvija po vremenskoj osi podeljenoj na frejmove, delovi koda mogu biti dodeljeni frejmovima. Taj kod izvršava se kada se film nađe na određenom frejmu. Pored toga, instance simbola tipa dugme takođe se mogu programirati za izvođenje određenih akcija pri pojavi nekog događaja, poput pritiska na dugme. Instance simbola tipa filmski klip takođe mogu reagovati na neke događaje ako im se pridruže potrebne akcije. Uz to, simbol tipa filmski klip je takođe film po svojoj definiciji, poseduje svoju vremensku osu, a može sadržavati i druge instance simbola tipa dugme ili filmski klip, pa se cela priča može ponoviti u kontekstu definicije simbola. Ovo daje mogućnost velike zbrke i raštrkavanja koda na sve moguće strane, što s druge strane onemogućava korektno praćenje toka filma, nalaženje grešaka, i slično. Stoga je preporuka držanje koda na manje mesta, pogotovo definicije promenljivih, funkcija i objekata, recimo u prvom frejmu filma ili definicije simbola, dok se za obradu događaja može samo navesti poziv već definisane funkcije ili metoda nekog objekta.

2.3.1 Događaji instanci simbola tipa dugme

Kod koji se pridružuje instanci simbola tipa dugme nalazi se u hendleru `on`:

```
on (mouseEvent) {
      //    statements
}
```

Za jedno instancu može se postaviti više hendlera, zavisno od događaja mouseEvent. Jedan hendler može opsluživati više događaja razdvojenih zarezom u zaglavlju hendlera. Kod pridružen hendleru izvršava se kada se desi neki od događaja navedenih u zaglavlju hendlera. Sledi opis događaja koje prepoznaje objekat tipa button:

- `press` – dugme miša je pritisnuto dok je pokazivač iznad dugmeta
- `release` – dugme miša je otpušteno dok je pokazivač iznad dugmeta
- `releaseOutside` – dugme miša je otpušteno dok je pokazivač van dugmeta

- `rollOver` – pokazivač miša kreće se iznad dugmeta
- `rollOut` – pokazivač miša je napustio površinu dugmeta
- `dragOver` – dok je pokazivač iznad dugmeta, dugme miša je pritisnuto, pokazivač pomeren van površine dugmeta i potom vraćen iznad
- `dragOut` – dok je pokazivač iznad dugmeta, dugme miša je pritisnuto i potom pokazivač pomeren van površine dugmeta
- `keyPress ("key")` – taster `key` je pritisnut

2.3.2 Događaji instance simbola tipa filmski klip

Instanca simbola tipa filmski klip takođe reaguje na događaje.

```
onClipEvent (movieEvent) {
      //    statements
}
```

Ovde je `movieEvent` neki od sledećih događaja:

- `load` – akcija se inicira instanciranjem objekta njegovom pojavom u vremenskoj osi
- `unload` – akcija se inicira u prvom frejmu nakon uklanjanja objekta sa vremenske ose, a izvodi se pre bilo koje akcije vezane za posmatrani frejm
- `enterFrame` – akcija se inicira pri svakom frejmu, a izvodi se nakon svih akcija vezanih za posmatrani frejm
- `mouseMove` – akcija se inicira pri svakom pomeranju miša
- `mouseDown` – akcija se inicira pritiskom na levo dugme miša
- `mouseUp` – akcija se inicira otpuštanjem levog dugmeta miša
- `keyDown` – akcija se inicira pritiskom na neki taster
- `keyUp` – akcija se inicira otpuštanjem tastera
- `data` – akcija se inicira prijemom podataka pri učitavanju

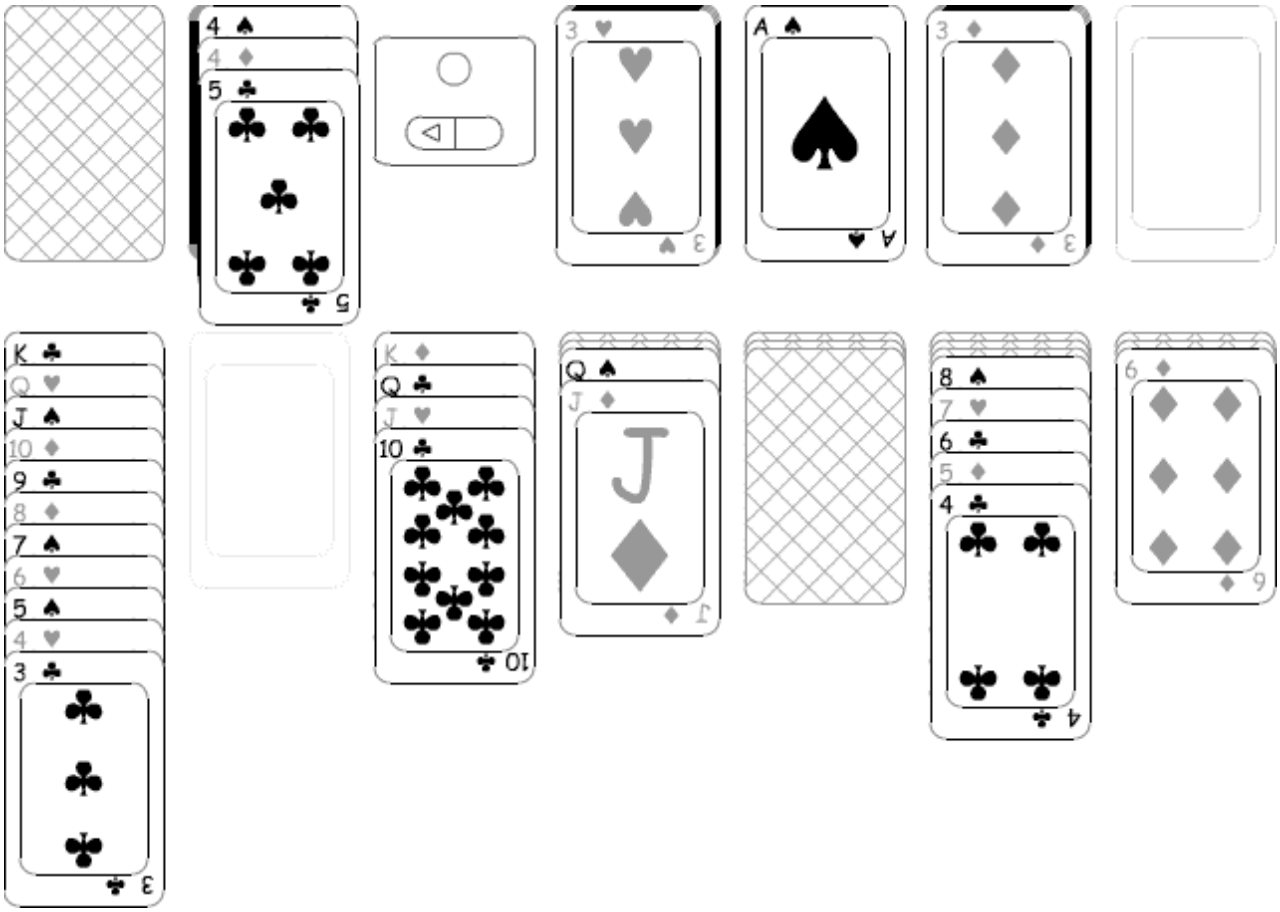
3 PRVI PRIMER: PASIJANS

Jedna od najzaslužnijih aplikacija u računarskom opismenjavanju svakako jeste čuvena igra kartama Solitaire iz standardnog paketa Windows. Pasijans je poznatiji naziv igre. Program Pasijans ima za cilj ilustraciju korišćenja objekta `movieClip` i realizaciju interakcije u ActionScriptu, u Flashu.

3.1 ELEMENTI PASIJANSA

Pasijans je igra sa kartama. Radna površina podeljena je na:

- 7 kolona – celom širinom, centralni i donji deo površine, sa kartama slaganim jedna na drugu i određenim smicanjem ka dole
- dek – sadrži ostatak špila, karte nepodeljene na kolone, gornji levi ugao
- 4 steka – za slaganje karata po bojama i veličini, gore desno
- kontrolni deo – nova partija i undo-redo, između deka i stekova



3.1.1 Karte

Karte su grafički objekti sa kojima korisnik vrši interakciju, tj. igra igru.

Na karti su uočena tri dela:

- prednji
- zadnji
- dugme

Svaki deo može biti vidljiv ili nevidljiv, u zavisnosti od stanja u kome se neka posmatrana karta nalazi. Stanje "otvorena" predstavlja kartu kojoj se vidi lice, a naličje ne vidi, dok stanje "zatvorena" vidi naličje i ne vidi lice. Dugme na karti svojom "vidljivošću" (dugme je providno) dozvoljava ili ne korisničku interakciju.

Karte su u potpunosti vektorski realizovane, tako da se, na primer, lice karte se, pri prvom zahtevu za prikaz lica, inicijalizuje preko vrednosti simbola i veličine karte postavljanjem odgovarajućih simbola na predefinisane pozicije za odgovarajuću veličinu.

3.1.2 Kolone

Kolone zauzimaju centralnu površinu aplikacije. Ima ih sedam i mogu sadržavati karte. Karte u koloni mogu biti otvorene ili zatvorene.

Prilikom deljenja se postavlja određeni broj zatvorenih karata (čiji broj tačno odgovara rednom broju kolone) i na vrh jedna otvorena karta. Tokom igre se karte se prebacuju i sa kolona i na kolone, a zatvorene karte se otvaraju.

3.1.3 Dek

Dek je "ostatak" karata iz špila koji pri deljenju nije stao na kolone. Nalazi u se u gornjem levom uglu. Podeljen je na deo u kome se nalaze zatvorene karte i drugi s otvorenim kartama. Pri deljenju se preostale karte smeštaju u zatvoreni deo. Karta na vrhu zatvorenog dela reaguje na klik prebacivanjem (okretanjem) određenog broja karata u otvoreni deo, a karta na vrhu otvorenog dela može se prebacivati na kolone ili stekove.

3.1.4 Stekovi

Cilj igre je prebaciti sve karte na stekove. Postoji četiri steka, a sve karte u nekom steku moraju biti istog znaka iređaju se od najmanje ka najvećoj. Stekovi se nalaze gore, desno.

3.2 INTERAKCIJA

Korisnik sa kartama može da čini sledeće, u zavisnosti od toga gde se karta nalazi:

3.2.1 Prebacivanje grupe karata sa kolone na drugu kolonu

Ako je ciljna kolona prazna, moguće je prebaciti grupu otvorenih karata predvođenu kraljem (K).

Ako ciljna kolona nije prazna i ako je karta na vrhu kolone otvorena, može se prebaciti grupa karata kod koje je boja vodeće karte različita od boje ciljne karte i veličina vodeće karte prethodnik ciljne.

3.2.2 Prebacivanje karte sa kolone na stek

Ako je karta kec (A) može se prebaciti na prazan stek.

Karta se može prebaciti na stek ako su karte na steku istog znaka kao i karta i ako je karta na vrhu steka po veličini prethodnik karte koja se prebacuje.

3.2.3 Okretanje karte u koloni

Ukoliko prilikom prebacivanja karata sa kolone na vrhu kolone ostane zatvorena karta, njoj postaje vidljivo dugme i klikom na kartu se karta otvara.

3.2.4 Okretanje karata u deku

Klikom na kartu na vrhu zatvorenog dela deka prebacuju se karte na otvoreni deo i tamo otvaraju. Okreću se po tri karte, ukoliko na zatvorenom delu deka ima dovoljan broj karata, inače koliko može.

3.2.5 Okretanje deka

Kada su sve karte deka u otvorenom delu, klik na zatvoreni deo prebacuje sve karte na zatvoreni deo i pri tom ih zatvara.

3.2.6 Prebacivanje karte sa deka na kolonu ili stek

Karta na vrhu otvorenog dela deka može se prebaciti na kolonu ili stek pod istim uslovima kao i pri prebacivanju karata sa kolona.

3.2.7 Prebacivanje karte sa steka na kolonu

Karta se sa steka može prebaciti na kolonu ako zadovoljava uslov prebacivanja na kolonu.

3.3 KREIRANJE KARATA

Objekat `movieClip` može sadržavati druge objekte `movieClip`. Svaki od objekata koje sadrži jedinstveno se identifikuje svojim imenom i dubinom. Jedan od metoda za

dodavanje novog objekta je `attachMovie`. Ovaj metod zahteva tri parametra. Prvi parametar jeste naziv objekta iz biblioteke koji se želi instancirati. Sam naziv je izvozno ime bibliotečkog elementa koji je proglašen kao izvozni, što omogućava instanciranje ovim metodom. Drugi parametar metoda je jedinstveni naziv (u odgovarajućoj lokalnosti) elementa koji se kreira. Korišćenje postojećeg imena za posledicu proizvodi zamenu objekta. Treći parametar jeste dubina. To je vrednost veća ili jednaka nuli. Na određenoj dubini može se nalaziti najviše jedan `movieClip` objekat. Dodela zauzete dubine novom objektu briše postojeći objekat na istoj dubini. Broj dubine je bitan zbog vizuelizacije, jer se "dublji" objekti (manji broj) nalaze ispod...

Već je napomenuto da se karte sastoje iz tri dela čija vidljivost prikazuje stanje karte. Karta je grafički predstavljena objektom tipa `movieClip` i ona sadrži tri objekta istog tipa koji predstavljaju lice i pozadinu karte, i dugme. Karti su dodeljeni metodi za prikazivanje i sakrivanje ovih elemenata. Inicijalno se metodama za prikazivanje dodeljuju funkcije za kreiranje samih elemenata i metodama za sakrivanje prazna funkcija, dok se nakon prvog poziva metoda za prikazivanje, nakon inicijacije elementa karte, metodama se pridružuju funkcije za prikazivanje i sakrivanje elementa, respektivno. Na primeru pozadine to izgleda ovako:

3.3.1 Kreiranje pozadine karte

```
function showBack () {this.myMovie.myBack.show ();}
function hideBack () {this.myMovie.myBack.hide ();}
function initBack () {
      this.myMovie.attachMovie (_root.back, "myBack",
_root.backLevel);
      this.hideBack = _root.hideBack;
      this.showBack = _root.showBack;
}
```

Ključna reč `this` referencira kartu kojoj su pridruženi metodi, pri pozivu metoda. Pri inicijaciji same karte doljejuje joj se prazan `movieClip` nazvan `myMovie` i postavlja metod `showBack = _root.initBack`. `_root` je korenski objekat, `movieClip` u kom se nalazi kompletan kod ovog primera (to ne mora tako da se realizuje). Pri prvom pozivu `showBack` izvršiće se `initBack ()`, odnosno biće dodat `movieClip` objekat u `myMovie` pod nazivom `myBack` na dubinu `backLevel` (svaki deo karte ima svoju dubinu), a metodama se dodeljuju nove funkcije.

3.3.2 Dugme

Sličnu inicijaciju ima i dugme. Dugemetu se dodeljuje osobina `myCard` koja referencira na kartu. `myButton` je tipa `movieClip`, a sadrži providno dugme koje reaguje na akcije miša. Ovo iz razloga da se omogući uklanjanje dugmeta po potrebi. Dugme sadrži sledeće akcije:

```
on (press) {myCard.onPress ();}
on (release, releaseOutside) {myCard.onRelease ();}
```

Referencira na metode svoje karte. Ova realizacija predstavlja hendlere. Dodelom odgovarajućih funkcija ovim metodama karte obezbeđuje se odgovarajuće ponašanje karte u posmatranom stanju.

3.3.3 Kreiranje lica karte

Karte su realizovane u potpunosti kao vektorski objekti. Da bi se izbeglo kreiranje 52 različita vektorska crteža lica karata, korišćeno je kreiranje na osnovu osobina karte pomoću nekoliko primitiva. Kartu određuje znak i veličina. U gornjem levom i donjem desnom uglu karte nalaze se mali simbol znaka i oznaka veličine karte, dole su obrnuti. U centralnom delu je raspoređen potreban broj simbola znaka, zavisno od veličine. Neke karte imaju slike, ali je to ovom prilikom zanemareno i ponuđeno alternativno rešenje.

Potrebni primitivi su simbol znaka i tekstualno polje boje koja odgovara znaku. Instancira se simbol znaka, umanjuje i postavlja na predefinisanu poziciju u gornjem levom uglu karte. Instancira se simbol tekstualnog polja, postavlja na predefinisano mesto gore levo, dodeljuje vrednost veličine karte. Instancira simbol znaka, umanjuje, okreće za 180 stepeni, postavlja dole desno. Instancira simbol teksta, okreće, postavlja dole desno, dodeljuje vrednost. Ako veličina karte ne odgovara slici, iz predefinisanog niza čitaju se pozicije i rotacije instanci simbola znaka koji zauzimaju centralni deo karte. Slike instanciraju simbol teksta, uvećavaju i uvećavaju, postavljaju ga više centra karte i dodeljuju mu vrednost veličine karte, i instanciraju simbol znaka, uvećavaju i postavljaju niže centra karte. Asovi instanciraju simbol znaka, uvećavaju i postavljaju u centar.

3.4 DELJENJE KARATA

Kolone, dek i stekovi su objekti koji vode računa o kartama čiji su domaćini. Određuju poziciju, dubinu i stanje karata, čuvaju ih u povezanim listama i vode računa o posebnim slučajevima korišćenja. Deljenje:

```
cards.scramble ();
var newCards = cards.slice (0);
for (var i = 0; i < numberOfColumns; i++) {
      columns [i].dealToMe (newCards.pop (), true);
      for (var j = i + 1; j < numberOfColumns; j++) {
            columns [j].dealToMe (newCards.pop (), false);
      }
}
while (newCards.length > 1) {
      deck.dealToMe (newCards.shift (), false);
}
deck.dealToMe (newCards.shift (), true);
```

Karte se mešaju (metod `scramble`) i kopiraju (reference) u privremeni niz `newCards` koji se potom koristi za dodelu karata kolonama i deku. Drugi argument metoda `dealToMe` kod kolona predstavlja da li se karta postavlja licem (`true`) ili naličjem (`false`), dok kod deka vrednost true predstavlja kartu sa vidljivim dugmetom. Deli se otvorena karta na prvu kolonu, pa zatvorene na ostale, dalje otvorena na sledeću kolonu i zatvorene na ostale i tako redom dok se na poslednju kolonu ne postavi otvorena karta. Ove karte uzimaju se "odozgo" metodom `pop`, a preostale se redom spuštaju na dek, "odozdo" metodom `shift`.

3.5 PREBACIVANJE KARATA

Ceo film u Flashu posmatra se takođe i kao `movieClip` objekat, tako da je u primeru pasijansa u svakom trenutku potrebno voditi računa o dubinama na kojima se nalaze pojedine karte kako bi vizuelizacija bila zadovoljavajuća. Kako elementi scene, dek, kolone i stekovi, tokom igre "razmenjuju" karte, njihovo kreiranje kao grafičkih elemenata zahtevalo bi brisanje, s jedne strane, i instanciranje, s druge, karata pri svakom prebacivanju s jednog elementa na drugi. Stoga ovi elementi nisu realizovani kao `movieClip` objekti, već samo objekti koji pomažu kartama i igraču da se igra igra. Zato se karte nalaze u korenskom `movieClip` objektu (`_root`), a svakom elementu je dodeljen dopustivi opseg dubina vodeći računa o maksimalnom broju karata koje element može da primi, i element vodi računa o dubini svojih karata i njihovoj poziciji.

Sada je jasno da karta pri promeni svog kontejnera, menja i svoju dubinu. Tom prilikom koristi se metod `swapDepths`, koji za parametar uzima novu dubinu objekta ili ciljni objekat za zamenu dubine.

3.5.1 Početak prevlačenja

Metod `startDrag` omogućava objektu `movieClip` kretanje po sceni indukovano promenom pozicije pokazivača miša. Kažemo da se objekat "vuče" (drag). Ono što je bitno je da se u jednom trenutku može vući samo jedan `movieClip` objekat. U primeru pasijansa dozvoljeno je sa kolone vući više karata. Da bi to bilo izvodivo, potrebno je staviti sve karte koje se žele vući u neki `movieClip` objektat koji se potom vuče i cela stvar funkcioniše. Ovaj objekat mora biti postavljen na dubinu veću od dubine na kojima se mogu nalaziti pojedine karte kako bi grupa karata koja se vuče vizuelno bila iznad svih elemenata scene. Objekat `movieClip` ne može da menja svog roditelja tako da je rešenje u kreiranju kopije lica karata u objektu koji se vuče i ukidanju vidljivosti originalnih karata dok traje prevlačenje. Kako se jedna karta može više puta premeštati u toku partije, u objektu koji se vuče prave se trajne kopije karata i regulacijom vidljivosti kopija određuje se koje karte se vuku a koje ne. Takođe, taj objekat vodi računa o poziciji kopija i njihovoj dubini zbog adekvatnog prikaza.

3.5.2 Kraj prevlačenja

Stanje prevlačenja objekata prekida se pozivom metoda `stopDrag`.

3.5.3 Odluka o prebacivanju

Rezultat prevlačenja je da karta ili grupa karata menja svoj element, odnosno poziciju i dubinu, ili vraćanje na prethodno stanje. Ova odluka donosi se kroz proveru metodom `hitTest`. Pri početku prevlačenja uočavaju se vrednosti vodeće karte koja se prevlači, boja, znak i veličina, i uočavaju se kolone i stekovi koji zadovoljavaju uslov za primanje vučene karte ili grupe karata. Na kraju prevlačenja se poziva `hitTest`, sa parametrom karte na vrhu uočenog elementa scene. Metod vraća `true` ako se objekti preklapaju, inače vraća `false`. Znači, ako se pri prolasku kroz niz uočenih objekata `hitTest` vrati `true` kod nekog elementa, na taj element se prebacuje karta ili grupa

karata, a ako `hitTest` svuda vraća `false`, jednostavno se karte ponovo postavljaju kao vidljive i nikakva promena se ne dešava u odnosu na situaciju pre prevlačenja.

3.6 UNDO I REDO MEHANIZAM

Jedna prednost programa iz ovog primera u odnosu na poznatiju verziju za Windows jeste mogućnost za neograničeni Undo i Redo. Ovo omogućava igraču da slobodnije greši i ispita više načina na putu rešenja igre.

Suština rešenja je u korišćenju nizova čiji elementi prate akcije promene stanja igre.

U okviru objekta `options` sadrži nizove `undoArray` i `redoArray`. Na početku svake partije ovi nizovi se prazne i onemogućava se pristup grafičkim elementima (dugmad) za iniciranje Undo/Redo akcija.

Kada se desi događaj koji menja stanje sistema (menja stanje bar jedne karte), to se beleži u ovim nizovima za potrebu vraćanja na prethodno stanje. To radi `addUndo`:

```
function addUndo (myUndo, myRedo) {
      var undoAction = new Object ();
      undoAction.myUndo = myUndo;
      undoAction.myRedo = myRedo;
      this.showUndo ();
      this.undoArray.push (undoAction);
      this.redoArray.splice (0);
      this.hideRedo ();
}
```

Funkcija `addUndo` je metod objekta `options`, tako da se `this` odnosi na njega. Metodi `showUndo`, `hideUndo`, `showRedo` i `hideRedo` pale i gase dugmiće za undo i redo (ako je neki niz prazan, ne može se pozvati akcija). Nizovi sadržavaju objekte (`undoAction`) sa delovima za undo (`myUndo`) i redo (`myRedo`). Metod `addUndo` dodaje objekat na niz `undoArray` i prazni niz `redoArray`. Metod `doAction` pokazuje šta su `myUndo` i `myRedo`:

```
function doAction (myAction) {
      var action;
      for (i = myAction.length - 1; i >= 0; i--) {
            action = myAction [i];
            action [0].doThis = action [1];
            if (action.length == 2) {action [0].doThis ();}
            else {action [0].doThis (action [2]);}
      }
}
```

Parametar u `doAction`, `myAction`, je ili `myUndo` ili `myRedo`, a metod se poziva putem metoda `doUndo` ili `doRedo`, iniciranih klikom na odgovarajuće dugme. Parametar `myAction` je niz akcija koje definišu promenu u sistemu. Akcije se pamte kao nizovi, gde je prvi član referenca na objekat, drugi član referenca na funkciju objekta i treći opcioni član je referenca parametra funkcije. Metodu `doThis` objekta (`action [0]`) dodeljuje se funkcija (`action [1]`). Potom se poziva metod `doThis`, ako ima parametra (`action.length != 2`) sa parametrom (`action [2]`), inače bez parametra.

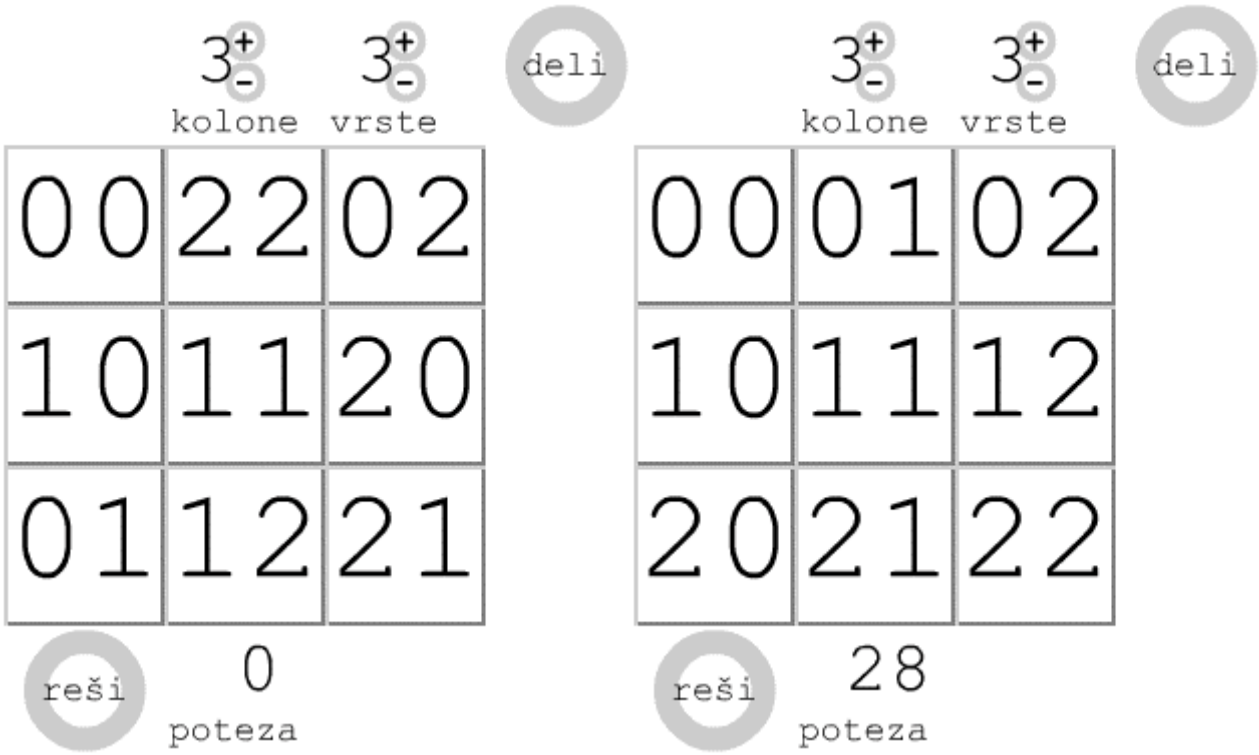
Ovakav pristup je zadovoljavajuć jer primer nije zahtevao akcije koje sadrže više od jednog parametra, mada je moguće i uopštenje za proizvoljan broj parametara.

Ovakva realizacija zahteva tačno definisanje svih akcija koje se izvode pri jednoj promeni stanja kao i inverznih akcija.

4 DRUGI PRIMER: SLAGALICA

Slagalica predstavlja sliku podeljenu na m*n pravougaonika, koji se izmešaju tako da slika ne može da se prepozna. Cilj igrača jeste da sliku ponovo složi pomeranjem njenih delova. Delovi se nalaze u ravni, a postoji jedno polje viška (van slike, recimo levo iznad) koje obezbeđuje kretanje pravougaonika prema "rupi", pri čemu se rupa pomera omogućavajući na taj način pomeranje drugih delova.

Druga varijanta sadrži delove sa upisanim brojevima koji se takođe izmešaju i zadatak postaje slaganje delova u redosledu po svojim brojnim oznakama. Varijacija ovoga je obeležavanje delova sa rednim brojem vrste i kolone u kojima delovi na kraju treba da se nađu.



Slika prikazuje izgled slagalice formata 3*3 pre slaganja (levo) i nakon slaganja (desno). Postoji interaktivna verzija slagalice gde igrač pomera delove slagalice kliktanjem na njih – u ovom slučaju je posao prepušten Flashu i ActionScriptu. Korisnik može da utiče na broj vrsti i kolona, odnosno dimenziju slagalice, da zada komandu za generisanje početnog problema klikom na dugme deli i nakon generisanja da zada komandu za traženje optimalnog rešenja (najmanji broj poteza) na dugme reši. Po nalaženju rešenja delovi se pomeraju kroz generisanu animaciju, brojač poteza beleži pomeranja delova, a kraj svega je po završetku animacije – svi delovi se nalaze

na svojoj ciljnoj poziciji a vrednost brojača poteza je broj poteza u minimalnom rešenju.

Za realizaciju ovako predstavljenog problema potrebno je obezbediti generisanje početnog problema na osnovu zadatih dimenzija, pogodnu strukturu i algoritam za traženje rešenja i odgovarajuću grafičku reprezentaciju rešenja.

4.1 GENERISANJE POČETNOG PROBLEMA

Kako postoji m*n delova, pravi se niz brojeva odgovarajuće dimenzije koji se potom izmeša.

```
var myArray = new Array ();
for (var i = 0; i < myNumber; i++) {myArray.push (i);}
```

Ovde je `myNumber` jednak baš m*n, tako da se nakon for petlje dobija niz brojeva od nule do `myNumber - 1`. Sada se niz `myArray` izmeša na neki dopustiv način. U tu svrhu dodeljuje mu se metod `scramble`:

```
myArray.scramble = function () {
      var myIndex, temp;
      var odd = ((myNumber % 2) != 0);
      this.push (this.shift ());
      for (i = this.length - 1; i > 2; i--) {
            myIndex = Math.floor (Math.random () * i);
            odd = (myIndex % 2 != 0) ? odd : (!odd);
            temp = this [myIndex];
            this.splice (myIndex, 1);
            this.push (temp);
      }
      if (odd) {
            this.push (this.shift ());
            this.push (this.shift ());
      } else {
            temp = this.shift ();
            this.push (this.shift ());
            this.push (temp);
      }
}
```

Za validnost mešanja odgovorna je promenjiva `odd`, jer rešivost zavisi od parnosti permutacije elemenata niza. Ideja je da se slučajno vade elementi niza i stavljaju na kraj, pri čemu se vodi računa o tome da se bira od preostalih elemenata niza. Prvo se na kraj prebacuje vodeći element, jer se njemu odgovarajući deo nalazi pored rupe i pomeranjem tog dela se uopšte omogućava kretanje ostalih delova. Potom se kroz `for` petlju obezbeđuje mešanje ostalih elemenata niza i na kraju, kada preostanu samo dva neizmešana, u zavisnosti od vrednosti `odd`, određuje se u kom redosledu će oni biti postavljeni na kraj niza. Preostaje samo pozivanje metoda:

```
myArray.scramble ();
```

Ovakav niz spreman je za generisanje odgovarajuće strukture za početni problem.

4.2 STRUKTURA

Prirodna struktura po opisu problema je matrica odgovarajućeg formata koja sadrži elemente, recimo objekte, koji vode računa o potrebnim informacijama. Matrica je promenljiva `matrix` kojoj su pridružena dva metoda za pretragu nad njenim elementima: `search` i `newSearch`.

4.2.1 Element matrice

Element matrice je objekat kreiran konstruktorom kome se prosleđuju tri parametra: vrsta i kolona elementa u matrici, i inicijalni `movieClip` objekat (deo slagalice). Konstruktor povezuje prosleđeni `movieClip` objekat pod nazivom `mov`, postavlja njegovu početnu vizuelnu poziciju, izračunava, na osnovu prosleđene vrste i kolone, poziciju u nizu, te vrednost elementa niza na nađenoj poziciji dodeljuje `mov` kao vrednost osobine `value` i na osnovu ove vrednosti izračunava ciljnu vrstu i kolonu, vrednosti osobina `row` i `column` objekta `mov` i konačno postavlja vrednost tekstualnog polja u objekta `mov`, koja predstavlja ciljnu vrstu i kolonu. Konstruktor takođe uvodi i osobinu `tree` koju postavlja na `null`, o čemu će biti više napomena dalje u tekstu.

```
Klasa tMatrixElement:
- osobine:
      myIndex: integer
      tree: tNode
      mov: movieClip
            value, row, column: integer
            text: String
- metod:
      findNode: tNode
```

4.2.2 Stablo pretraživanja

Prilikom pretraživanja se u stvari nad elementima matrice vrši zamena `mov` objekata čime se simulira kretanje elemenata slagalice u cilju traženja rešenja. Tako se u stvari menja sadržaj matrice i dolazi se do novih stanja matrice. Stanja matrice mogu se reprezentovati vrednostima `value` `mov` objekata, po redosledu kako se oni javljaju u elementima matrice. Ove vrednosti mogu se poređati u niz. Na ovaj način, niz `myArray` odslikava početno stanje matrice. Broj mogućih pozicija u igri je velik, iznosi (`myNumber! / 2`), tako da pogodna struktura za izvođenje pretraživanja mora biti u najmanju ruku binarno stablo. Čvorovi stabla imaju jedinstveni identifikator čvora, a zbog potrebe poređenja se niz `myArray` na početku prevodi u string promenljivu `IDnode`, koja predstavlja vrednost tekućeg čvora tokom pretraživanja. Svaki element matrice ima osobinu `tree`, što predstavlja drvo svih posećenih čvorova kod kojih se rupa nalazi na poziciji tog elementa matrice. Na ovaj način urađena je distribucija svih čvorova po elementima matrice. Čvorovi, pored svog identifikatora i pokazivača na elemente levog i desnog podstabla, poseduju još nekoliko osobina od kojih su najvažnije, za čvorove koji se nađu na putu konačnog rešenja, smer (`direction`) kretanja rupe ka stanju koje predstavlja čvor i pokazivač na sledeći čvor puta (`next`). Po uspešnom završetku pretrage tako je moguće odrediti putanju rupe koja početno stanje prevodi u krajnje.

4.3 ALGORITAM PRETRAŽIVANJA

Sama pretraga funkcioniše po algoritmu backtrackinga. Inicijalno se pretpostavlja da rešenje mora biti kraće od neke predefinisane vrednosti, uzima se da je to vrednost dužine rešenja i pri nalaženju svakog kraćeg rešenja se ova vrednost postavlja na dužinu nađenog rešenja. Promenljivoj `matrix` pridruženi su metodi `search` i `newSearch`. Parametri ovih metoda su vrsta i kolona tekućeg elementa matrice, broj poteza i smer pretrage. Metod `search` traži čvor nad stablom elementa matrice, vodi računa o određenim odsecanjima (tekući broj poteza ne može biti veći od vrednosti nađenog rešenja, posećeni čvor koji nije na putu rešenja nije od interesa, posećeni čvor koji se poseti u potezu većem od prethodne posete nije na najboljem putu...), detektuje prvo rešenje, ažurira čvorove pri nalaženju kraćih rešenja, upućuje na dalju pretragu (traži naredni čvor) u dozvoljenim smerovima (dozvoljen smer je mogući smer, onaj smer koji neće izaći van dimenzija matrice) pozivom metoda `newSearch` i, u pogodnom trenutku, vraća čvor ili `null` u slučaju neuspele pretrage. Metod `newSeach` se bavi administrativnim stvarima, poput pripreme nove vrednosti promenljive `IDnode`, zamene `mov` objekata elemenata matrice. Po obradi svih pripremnih radnji inicira novu pretragu (poziva `search`), nakon čega restaurira prethodno stanje i vraća rezultat.

Deo koda metoda `search`:

```
myNode.onWay = false;
moveNum++;
var temp;
if (row > 0) {
      temp = this.newSearch (row, column, moveNum, _parent.up);
      if (temp != null) {
            myNode.onWay = true;
            myNode.next = temp;
      }
}
```

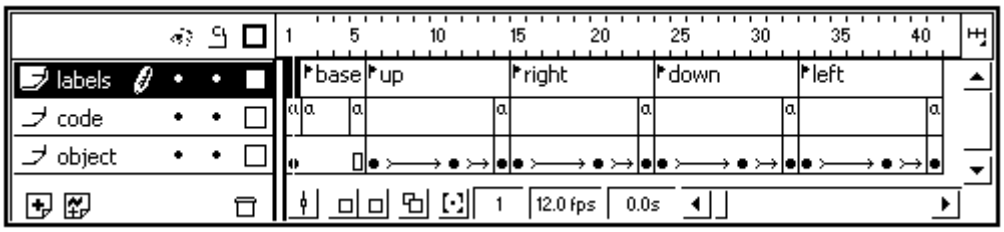
Promenljiva `myNode` predstavlja nađeni čvor koji odslikava poziciju pretrage. Kada se program nađe u ovom delu koda to znači da su preskočena sva odsecanja i da postoji potreba za daljom pretragom. Zato se povećava vrednost parametra `moveNum` koji će se prosleđivati metodu `newSearch`, jer se radi o narednom potezu (ovo se radi samo jednom). Pretpostavlja se da se čvor ne nalazi na putu rešenja, a ako se nađe da jeste, to će biti zabeleženo. Uslov kaže da se pretraga na gore može izvoditi samo u slučaju da se ne nalazimo u najgornjoj vrsti matrice (nulta vrsta). Sledeći uslov ispituje uspešnost pretrage na gore i, ako je ispunjen, zaključuje se da se čvor nalazi na putanji i da je nađen njegov sledeći čvor. Slično se ispituje i za ostale smerove, desno, dole i levo, koje vraćaju čvor samo ako je pronađen kraći put. Na kraju:

```
if (myNode.onWay) {
      myNode.direction = direction;
      return myNode;
}
return null;
```

Ako je čvor na putu, postavlja mu se odgovarajući smer ulaza i vraća se čvor, inače je pretraga neuspešna i vraća se `null`.

4.4 ANIMACIJA

Kada rešenje postoji, koristi se najveća prednost Flasha – animacija. Na ovom mestu od pomoći je slika vremenske ose simbola elementa slagalice:



Vide se tri lejera. Gornji lejer sadrži labele objekta, srednji kod, a na donjem lejeru nalazi se objekat koji se animira. Po instanciranju objekta automatski kreće da se odvija njegova vremenska komponenta. Kako inicijalno nije potrebna nikakva animacija, u lejeru code ubacuje se instrukcija `stop ();` koja prekida dalje kretanje vremenske ose. Takvo stanje zadržava se sve do nalaženja rešenja i do prozivanja objekta da izvrši neku animaciju. Poziv mu se upućuje instrukcijom `gotoAndPlay` koja kao parametar ima naziv labele frejma od koje treba da nastavi sa odvijanjem animacije.

Ukoliko, na primer, element slagalice treba da se kreće gore, dobija instrukciju `gotoAndPlay ("up").` Pomera se na početni frejm labele `up`, gde se u lejeru `object` nalazi objekat na svojoj startnoj poziciji (crna tačka u lejeru označava ključni frejm). Poslednji frejm grupe `up` takođe sadrži ključni frejm u lejeru `object`, gde se nalazi objekat pomeren nagore za visinu objekta. Strelica u lejeru `object` između ključnih frejmova označava animaciju, zadatak Flashu da distribuira promene stanja objekta po frejmovima između dva ključna frejma. Malo slovo "a" u lejeru code predstavlja ključni frejm sa kodom u ActionScriptu. To je sledeći kod:

```
setProperty (this, _visible, false);
setProperty (this, _y, getProperty (this, _y) - 80);
gotoAndPlay ("base");
```

Prva linija koda privremeno "gasi" objekat, druga ga pomera za jednu visinu (80) nadole (zato što je animacijom za toliko otišao gore) i treće pomera objekatat na labelu base, koja "pali" objekat (`setProperty (this, _visible, true);`) i nakon pauze od nekoliko frejmova:

```
_parent.animate ();
stop ();
```

Daje se instrukcija programu da može da nastavi sa animacijom i zaustavlja svoju animaciju. Funkcija animate prati tok rešenja i daje direktive za izvođenje animacije:

```
function animate () {
      if (solution != null) {
            hole.action = actions [solution.direction];
            hole.action ();
            solution = solution.next;
      }
}
```

Na početku animacije `solution` je prvi čvor rešenja. Niz `actions` sadrži funkcije koje vode računa o tome koji element slagalice treba da se pozove na animaciju, vodeći

računa o smeru. Uočena funkcija pridružuje se objektu `hole` kao metod `action`, koji se odmah i poziva i na taj način inicira se animacija potrebnog elementa matrice. Objekat `hole` svojim metodom `action`, odnosno funkcijama iz niza `actions`, vodi računa o poziciji rupe, zadaje komande `mov` elementima za za kretanje u određenom smeru i vrši razmenu `mov` elemenata u okviru matrice pri pomeranju.

4.5 PERFOMANSE

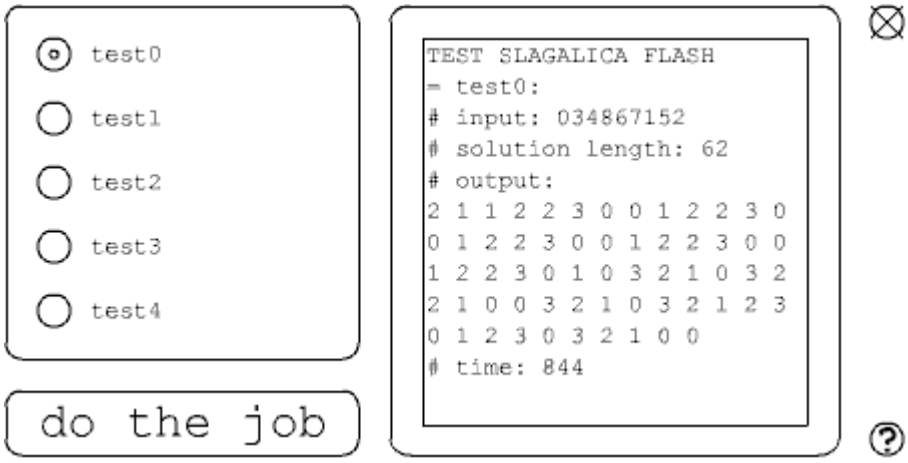
Algoritam backtrackinga je izuzetno nezahvalan u interaktivnim sistemima, povećanjem dimenzije problema višestruko se uvećava broj operacija algoritma. Kod slagalice, pri testiranju programa, uočava se da je dobijanje rešenja na slagalici formata 2*2 gotovo trenutno, slično je i sa formatima 2*3 i 3*2, dok format 3*3 na novijim računarima daje odziv u roku od nekoliko desetina sekundi do minuta, što se može proceniti kao dovoljno za demonstraciju, ali nedovoljno za interaktivnu primenu. Dozvoljen format 4*4 stoga nije ni testiran.

Ovakvi rezultati daju ideju za testiranje brzine Flasha i ActionScripta. Funkcija `getTimer` vraća broj milisekundi proteklih od startovanja programa i može se iskoristiti za merenje vremena nalaženja rešenja na sledeći način:

```
time = - getTimer ();
var temp = search (0, 0, 1, down);
time += getTimer ();
```

Po izvršenju ovog dela koda promenljiva time sadržava broj milisekundi proteklih prilikom nalaženja rešenja. Za samo testiranje je napravljen novi program.

4.5.1 Program za testiranje



Program sadrži niz od pet predefinisanih ulaznih stringova koji odgovaraju različitim početnim problemima. Selekcijom određenog testa i pritiskom na dugme “do the job” program startuje sa izvršavanjem, kreira se struktura i izvršava algoritam pretrage, nakon kojeg se u desnom delu prikazuju rezultati testa: redni broj testa, ulazni niz, dužina nađenog rešenja, rešenje (kao niz smerova koji bi se redom dodeljivali elementima pri animaciji, 0 je gore, 1 desno, 2 dole i 3 levo) i izmereno vreme

pretraživanja. Sam algoritam je uprošten, odnosno traži se samo prvo rešenje, ne i optimalno.

4.5.2 Merenje i rezultati merenja

Testiranje je obavljeno korišćenjem opisanog programa za testiranje, na PC platformi sa procesorom AMD Duron na 700 MHz i 256 MB RAM memorije, sa operativnom sistemom Microsoft Windows XP Professional, Version 2002.

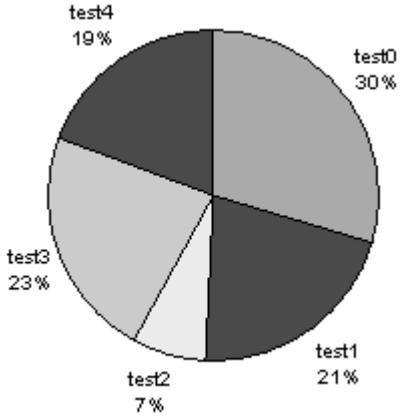
Vršeno je tri tipa testova: testiranje tzv. projektorskih izvršnih fajlova (Flash kreira izvršni fajl tako što upakuje plejer i aplikaciju, tako da se projektor može izvršavati i na sistemima gde nisu instalirani Flash ni Flash plejer), testiranje .swf fajlova iz samostalnih aplikacija Flash plejera, i testiranje u okruženju web browsera.

Takođe je sam program za testiranje rađen u tri verzije: kompilacijom iz Flasha 5 (Test Slagalica Flash 5), iz Flasha MX (Test Slagalica Flash MX), i iz Flasha MX sa opcijom podrške za plejer Flasha 5 (Test Slagalica Flash MX as Flash 5).

Jedno testiranje sastoji se od po pet merenja svakog testa. Jedna tabela za primer:

7918,4	Test Slagalica Flash 5.exe					Prosek
test0	2368	2384	2397	2381	2380	2382
test1	1654	1659	1659	1652	1660	1656,8
test2	561	559	562	560	556	559,6
test3	1805	1808	1805	1817	1811	1809,2
test4	1514	1509	1510	1509	1512	1510,8

Pet kolona sa brojevima predstavlja rezultate (broj milisekundi proteklih pri traženju rešenja) pet merenja testova (po vrstama). Poslednja kolona sadrži prosečnu vrednost svih merenja jednog testa. Tokom svih testova uočeno je da se odstupanje vrednosti pojedinačnih merenja od proseka kreće do granice od 1-2% (u većini testova su ova odstupanja i manja). Slika desno prikazuje relativni odnos prosečnih vremena po testovima. Pokazalo se da je ovaj odnos konstantan pri svim testiranjima, tako da je za dalju analizu dovoljno da se koristi samo suma proseka svih testova (gornje levo polje u tabeli).



Sledeća tabela sadrži sume proseka svih testova (druga kolona tabele).

01	7918,4	Test Slagalica Flash 5.exe
02	886,4	Test Slagalica Flash MX.exe
03	841	Test Slagalica Flash MX as Flash 5.exe
04	8130,6	Player 5, Test Slagalica Flash 5.swf
05	7932	Player 5, Test Slagalica Flash MX as Flash 5.swf
06	845	Player 6, Test Slagalica Flash 5.swf
07	889,8	Player 6, Test Slagalica Flash MX.swf
08	841,6	Player 6, Test Slagalica Flash MX as Flash 5.swf
09	868,4	MS IE, Test Slagalica Flash 5.html

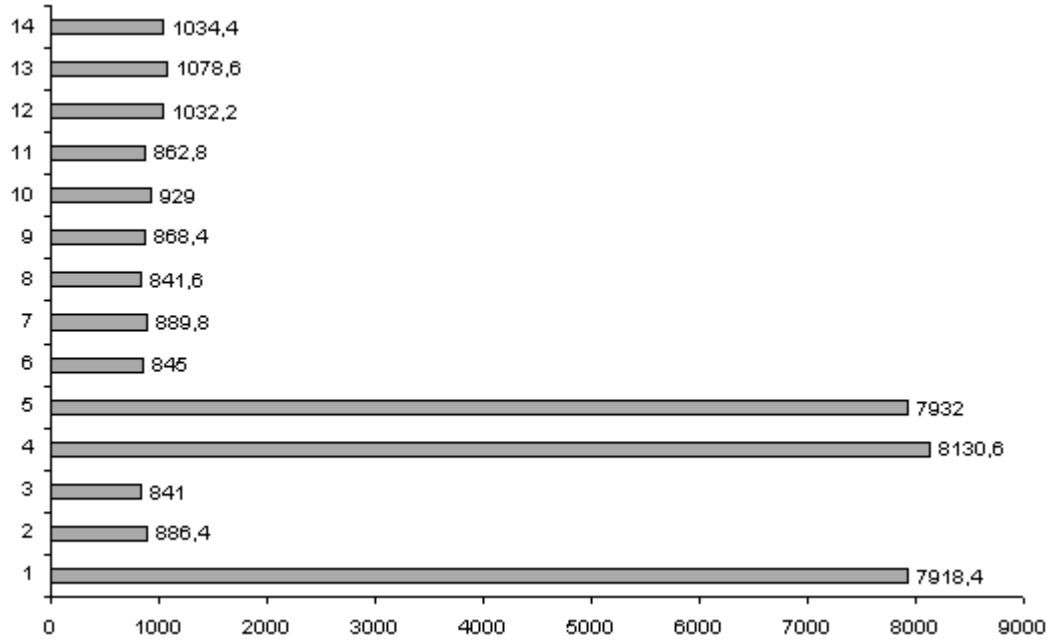
10	929	MS IE, Test Slagalica Flash MX.html
11	862,8	MS IE, Test Slagalica Flash MX as Flash 5.html
12	1032,2	Opera, Test Slagalica Flash 5.html
13	1078,6	Opera, Test Slagalica Flash MX.html
14	1034,4	Opera, Test Slagalica Flash MX as Flash 5.html

U tabeli se koriste sledeće oznake za browsere:

- `MS IE` – Microsoft Internet Explorer 6.0.2600.0000.xpclient.010817-1148
- `Opera` – Opera 5.01 build 840

Oba browsera imaju plug-in za Flash Player 6.

Isti podaci pregledniji su na grafikonu dole:

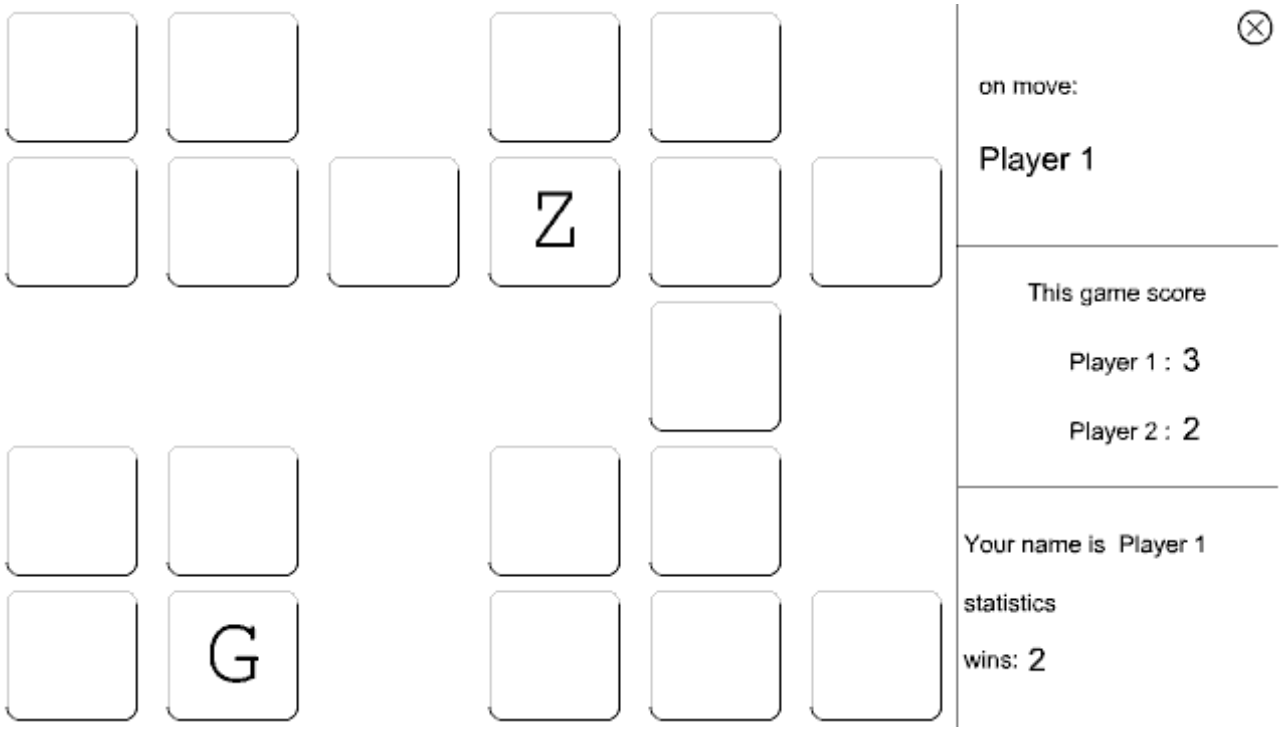


Rezultati govore, pre svega, u korist novog plejera, oko 9 (!) puta bolje vreme u odnosu na stari plejer (oznaka 1 je izvršni fajl sa ugrađenim plejerom Flasha 5, 4 je .swf Flasha 5 iz njegovog plejera, i oznaka 5 .swf iz Flasha MX za Player 5 iz Playera 5). Takođe, vrednosti merenja verzije iz Flasha 5 (1, 4, 6, 9, i 12) i Flasha MX za Player 5 (3, 5, 8, 11 i 14) se gotovo poklapaju, dok verzija iz Flasha MX za Player 6 (2, 7, 10 i 13) daje vrednosti slabije za oko 5%. Vrednosti postignute u projektorima (1-3) bliske su vrenostima postignutim u samostalnim plejerima (4-8), a merenja u Internet Exploreru (9-11) daju rezultate slabije za oko 2%, u Operi (12-14) za oko 25% slabije rezultate u odnosu na samostalni plejer.

5 TREĆI PRIMER: WEB MEMORIJA

Memory blocks je poznata igra uparivanja slika. Igra se sastoji od delova sa slikama. Svaka slika pojavljuje se tačno na dva dela (tako da je broj delova paran). Na početku igre su svi delovi okrenuti na svoja naličja, tako da se ne zna gde se koja slika nalazi. Potez igrača sastoji se u sukcesivnom okretanju dva dela na lice, po igračevom izboru. U slučaju da se na delovima okrenutim na lice nalazi ista slika, nađen je par, delovi se uklanjaju i to se računa kao pogodak, a u slučaju različitih slika, delovi se ponovo okreću na naličje, što se beleži kao promašaj.

Web memorija je varijanta igre predviđena za dva igrača preko Interneta. Umesto slika koriste se slova engleskog alfabeta. U svakom trenutku na potezu je jedan igrač dok drugi čeka na svoj red. Ako igrač na potezu pogodi par, broji mu se pogodak i ima pravo na sledeći potez, a ako promaši – šansu dobija igrač koji je čekao. Kraj igre je kada se upare svi delovi i tada je pobednik igrač sa više uparivanja, ili ako jedan igrač napusti igru (što se računa kao predaja) kada je pobednik igrač koji je ostao u igri.



Slika prikazuje scenu iz igre. Glavni deo površine zauzimaju delovi koje igrači okreću, a sa desne strane prikazane su korisne informacije za igru: ko je na potezu, broj pogodaka oba igrača i statistika (broj pobeda) igrača. Igrač koji igra upisao se pod imenom Player 1 i ima dve pobede i tri okrenuta para, njegov protivnik se predstavio kao Player 2 i pogodio je dva para, na potezu je Player 1 koji je okrenuo delove sa

slovima Z i G, što znači da je promašio i da sledeći potez igra Player 2. U gornjem desnom uglu nalazi se “dugme za predaju”.

5.1 KOMUNIKACIJA S OKRUŽENJEM

Flash ima dobre mogućnosti za saradnju sa “spoljnim svetom”. Jedna mogućnost jeste instrukcija `loadVarijables` koja treba da učita vrednosti promenljivih sa zadate lokacije. Ako se kao lokacija zada neka aktivna strana (PHP, ASP i slično) ona može da prosledi generisan sadržaj koji može da zavisi od primljenih parametara. Elegantnija rešenja postižu se korišćenjem objekata `XML` i `XMLSocket`.

5.1.1 Objekat XML

Objekat `XML` omogućava komunikaciju kroz slanje i prijem poruka definisanim u jeziku XML, opšteprihvaćenom standardu za razmenu podataka. Ulazno-izlazne metode objekta su `send` (slanje XML poruke na određenu adresu), `load` (učitavanje sa specifirane adrese) i `sendAndLoad` (slanje poruke na adresu i učitavanje sa adrese u neki `XML` objekat). Za indikaciju prijema poruke koristi se logička osobina objekta `loaded` (samo za čitanje – postavlja se na true po kompletiranju prijema). Takođe se može koristiti hendler `onLoad (success)`, koji program poziva po prijemu podataka, za definisanje akcija u zavisnosti od uspeha prijema poruke (hendleru se dodeljuje potrebna funkcija).

5.1.2 Objekat XMLSocket

Kada aplikacija u Flashu zahteva brzi odziv koristi se objekat `XMLSocket` koji ostvaruje direktnu vezu sa serverom, što omogućava serveru slanje poruka i bez prijema zahteva za slanje od strane klijenta. Server i klijent tada razmenjuju XML poruke korišćenjem full-duplex TCP/IP protokola. Klijent je na server vezan preko određenog porta. Server mora da ima aktivan program zadužen za ovaj port.

Iz sigurnosnih razloga uvedena su određena ograničenja nad ovim objektom. Objekat se može povezati samo na portove sa brojem većim ili jednakim 1024. Drugo ograničenje je da se objekat može konektovati samo na kompjutere istog poddomena gde se nalazi i Flash film fajl koji traži konekciju.

Objekat se kreira svojim konstruktorom, povezuje sa serverom metodom `connect`, šalje poruke metodom `send`, diskonektuje sa `close`. Obrada događaja obezbeđuje se dodelom potrebnih funkcija hendlerima `onConnect` (šta raditi nakon uspešne ili neuspešne konekcije), `onXML` (po prijemu poruke) i `onClose` (pri diskonekciji).

5.1.3 Server

Klijent je na server vezan preko određenog porta. Server mora da ima aktivan program zadužen za ovaj port. Jedno od pogodnih rešenja za ovu serversku aplikaciju jeste korišćenje programskog jezika Java koji poseduje klasu ServerSocket čiji objekti

preuzimaju kontrolu nad određenim portom na računaru, kao i mogućnost rada sa više niti (multithread) što omogućava opsluživanje više klijenata u isto vreme.

Kao serverska aplikacija korištene su (autor Derek Clayton) Java klase CSClient (realizuje opsluživanje klijenta) i CommServer (drži port i stara se o klijentima). Server se podiže pozivanjem klase CommServer i parametrom za broj porta. Server primljenu poruku od nekog klijenta prosleđuje svim svojim klijentima. Pri konekciji ili diskonekciji nekog klijenta, server šalje svim klijentima poruku o trenutnom broju klijenata na serveru.

5.2 TOK PROGRAMA

Za ostvarivanje veze sa serverom korišćenjem objekta `XMLSocket` neophodno je znati dva podatka, adresu računara servera (može se koristiti DNS ime ili IP broj) i broj porta koji je rezervisan za serversku aplikaciju. Stoga je zgodno ove podatke setovati kao promenljive (konstantne) na samom početku koda klijentske aplikacije u ActionScriptu. Prilikom razvoja može se koristiti računar na kojem se razvija aplikacija, tada se postavlja `host = "localhost"`, a nakon uspešne realizacije klijenta, pre postavljanja cele aplikacije na server promeniti vrednost promenljive `host` na stvarnu adresu servera. Uslov za podizanje serverske aplikacije jeste instalirana Java, kako na razvojnom računaru, tako i na računaru servera.

5.2.1 Povezivanje na server

Da bi igrač mogao da igra igru, potrebno je konektovati se na server. Prva stvar koja se očekuje od korisnika jeste proces logovanja. Korisnik u tekstualno polje unosi ime pod kojim se prijavljuje na server i pritiska dugme za konekciju.

Ako je konekcija uspešna, korisnikov program je ostvario vezu sa serverom. Server šalje poruke o trenutnom broju klijenata svaki put kada dođe do promene ovog broja (kada se prijavi novi korisnik ili neki napusti server). Svi klijenti primaju ovu poruku od strane servera i u zavisnosti od stanja u kom se trenutno nalaze, izvode određene akcije. Ako na serveru nije bilo klijenata, korisnik ostvarenom konekcijom postaje prvi klijent i, pošto je Web memorija igra za dva igrača, prelazi u stanje čekanja na drugog igrača. Ako se korisnik loguje kao drugi klijent, to znači da igra može da počne: drugi klijent šalje poruku kojom se predstavlja, na šta prvi klijent šalje poruku kojom se takođe predstavlja i izmešani niz karaktera koji odgovara karakterima na delovima igre, na osnovu koje se generiše tabela za igranje i prvi igrač dobija pravo na potez. Logovanje trećeg i ostalih klijenata ih tera na napuštanje igre, prelazak u stanje za rekonekciju, dok igrači ove poruke ignorišu. Ovo zato što nije vršena modifikacija klasa u serverskoj Java aplikaciji, koja je pisana kao opštija i sa mogućnošću opsluživanja većeg broja klijenata.

5.2.2 Igra dva igrača

Igrač na potezu pritiskom na izabrani deo "okreće" deo (postaje vidljiv pridružen mu karakter). Klijentska aplikacija šalje poruku o okrenutom delu, pa deo postaje vidljiv kod oba igrača. Igrač je i dalje na potezu jer treba da odredi deo za koji pretpostavlja da je par okrenutom delu. Po pritisku na drugi odabrani deo se takođe šalje poruka, drugi deo postaje vidljiv kod oba igrača pa se, nakon kraće pauze koja daje igračima vreme da uoče karaktere na okrenutim delovima, u zavisnosti od uspeha uparivanja okrenutih delova – ako su delovi upareni na potezu ostaje isti igrač, delovi se "gase" i broji se pogodak, a ako delovi nisu upareni pravo poteza dobija protivnik, a delovi se "okreću na naličje".

5.2.3 Pobeda ili poraz?

Nakon uparivanja poslednjeg preostalog para, igrač sa manjim brojem pogodaka se diskonektuje sa servera, briše mu statistika o broju pobeda i nudi mu se mogućnost rekonekcije, dok pobednik, igrač sa većim brojem uparenih delova, povećava za jedan broj svojih pobeda i prelazi u stanje čekanja na protivnika pošto je sam na serveru.

5.3 PORUKE

Tokom programa razmenjuju se XML poruke između servera i klijenata. To su u stvari XML dokumenti. Poruke koje se koriste u programu:

- `<NUMCLIENTS>numOfClients</NUMCLIENTS>` – poruka koju server šalje pri konekciji i diskonekciji klijenata
- `<SECOND>playerName</SECOND>` – poruka drugog igrača, prosleđuje ime
- `<DEALER>playerName</DEALER><ARRAY>startArray</ARRAY>` – odgovor prvog igrača drugom, predstavlja se sa `playerName`, a `startArray` predstavlja redosled karaktera koji se dodeljuju delovima
- `<FROM>ordNum</FROM><MOVE>pieceID</MOVE>` – regularan potez. Šalje se pri okretanju nekog dela. Vrednost `ordNum` je redni broj igrača koji je okrenuo deo, a `pieceID` identifikator okrenutog dela.

XML dokument je skup tagova (oznaka) koji mogu imati svoju vrednost, atribute (osobina taga koja može imati vrednost) i druge tagove. Objekat `XML` ima za cilj da verno predstavi strukturu XML dokumenta, tako da je sam objekat u stvari kolekcija, odnosno niz `childNodes`, `XML` objekata, svi tagovi dokumenta reprezentuju se kao `XML` objekti u okviru ovog niza.

5.4 PROGRAM

Za realizaciju je potreban jedan objekat `XMLSocket` kojim se ostvaruje veza sa serverom. U promenljivim `host` i `port` čuvaju se adresa računara servera i port preko kog se ostvaruje konekcija. Zahtev za logovanje, pritiskom na dugme za konekciju, poziva funkciju `connect`.

5.4.1 Konekcija

```
function connect () {
      mySocket = new XMLSocket();
      mySocket.onConnect = handleConnect;
      mySocket.onClose = handleClose;
      mySocket.onXML = handleIncoming;
      if (!mySocket.connect (host, port)) {
            gotoAndStop ("notConnected");
      {
}
```

Kreira se objekat, određuju njegovi hendleri i uspostavlja se konekcija metodom `connect`. U slučaju neuspešne konekcije ostaje da se pokuša ponovo ili odustane. Ako je konekcija uspela, može se početi sa prijemom i slanjem XML poruka.

5.4.2 Slanje poruke

```
function sendMessage (message) {
      var messageObj = new XML();
      messageObj.parseXML (message);
      if (mySocket && mySocket.connected) {
            mySocket.send (messageObj);
      } else {
            quit ();
      }
}
```

Argument funkcije `sendMessage` je string, tekst XML poruke. Kreira se `XML` objekat, parsira poruka i šalje putem `mySocket` na server.

5.4.3 Prijem poruke

Prijem poruke poziva hendler `onXML`. Tom prilikom pristiže i XML objekat sa servera, koji hendler treba da obradi.

```
function handleIncoming (messageObj) {
      var leadNode = messageObj.firstChild.nodeName;
      if (leadNode == "NUMCLIENTS") {
//...
```

Znači, `leadNode` je naziv vodećeg taga XML dokumenta koji je primljen i na osnovu njega može se tačno odrediti koji tip poruke je primljen. Dalje se prati opisano ponašanje i izvlače potrebni podaci.

```
messageObj.childNodes [i].firstChild.nodeValue
```

Ovo je pristup vrednosti i-tog taga primljenog dokumenta.

5.4.4 Zaključavanje i uslovne promenljive

Pošto se igra odvija tako što je u svakom trenutku na potezu tačno jedan igrač, pri čemu drugi čeka na svoj potez, postoji potreba za inplementacijom mehanizma koji obezbeđuje ovakvo ponašanje. U tom smislu, uvedeno je nekoliko logičkih uslovnih promenljivih čije stanje definiše ponašanje u posmatranom trenutku igre.

```
var myMove, even, locked;     //    boolean
```

Promenljiva myMove signalizira da li je igrač na potezu ili ne. U svakom trenutku, jedan od igrača ima vrednost ove promenljive postavljenu na true, dok je kod drugog setovana na false. Inicijalno stanje uspostavlja se pri početku partije kada prvoprijavljeni igrač dobija pravo na potez.

Kako se svaki potez igrača u stvari sastoji od okretanja dva dela, promenljiva even vodi računa o parnosti broja okrenutih delova. Pri okretanju svakog dela ova promenljiva negira svoju vrednost. Sledi kod koji se dodeljuje dugmadima u sastavu delova. Vidi se da se kod nalazi u okviru hendlera on i da se kod izvršava po otpuštanju levog dugmeta miša sa dugmeta dela web memorije. Dugme reaguje samo ako se igrač nalazi na potezu (_root.myMove):

```
on (release) {
      if (_root.myMove) {
            //    statements
      }
}
```

Ako je igrač na potezu, ispituje se da li je odigrani broj poteza bio paran ili ne i u zavisnosti od toga izvode određene akcije. (sledeći kod nalazi se na mestu gorenavedenog komentara) Ako je even:

```
if (_root.even) {
      _root.newMove (this.row, this.column);
      value._visible = true;
      _parent.el = this;
      _root.even = false;
}
```

Funkcija newMove generiše sadržaj poruke FROM:MOVE i prosleđuje ga na slanje u sendMessage. Naredna linija koda uključuje karakter posmatranog dela tako da igrač vidi koji znak je dodeljen delu. Osobina el objekta memory, koji sadrži sve delove pa se odavde i vidi kao _parent objekat, pokazuje se na posmatrani deo, što će biti od koristi pri okretanju drugog dela i upoređivanja vrednosti dodeljenih im karaktera. Na kraju se even postavlja na false, jer odigran je neparni potez. Ako pak nije even (else grana prethodnog if):

```
if (!_root.locked && (this != _parent.el)) {
      _root.newMove (this.row, this.column);
      value._visible = true;
      _root.locked = true;
      attachMovie ("waiter", "waiter", 432);
}
```

Ako nije zaključano (treća uslovna promenljiva, locked, objašnjena je u nastavku) i ako je pritisnuti deo različit od prethodno pritisnutog dela (koji se pamti u promenljivoj el), takođe se šalje poruka i uključuje karakter pritisnutog dela. Potom se vrši zaključavanje setovanjem locked na true i kontrola se predaje movieClipu waiter. Ovaj element pravi vremensku pauzu jer se sastoji od nekoliko praznih frejmova pre frejma u kome se nalazi kod koji izvodi dalja ispitivanja. U ovom trenutku, pri kačenju klipa waiter, onemogućena je svaka interakcija (igrač koji nije na potezu ima myMove na false, igrač na potezu ima even na false i locked na true). Pre nego što vidimo o čemu se stara waiter, da vidimo kako se obrađuju pristigle poruke o odigranom potezu. Kod je iz funkcije handleIncoming koja je

pridružena hendleru onXML objekta mySocket. Kada poruka nije ni `NUMCLIENTS`, ni `SECOND`, ni `DEALER`:

```
if (userOrdNum != messageObj.firstChild.firstChild.nodeValue) {
      //    statements
}
```

Ako je poruka od protivnika, radi se sledeće (umesto `statements`):

```
var el = memory ["e"+messageObj.childNodes[1].firstChild.nodeValue];
el.value._visible = true;
if (even) {
      memory.el = el;
      even = false;
} else {
      locked = true;
      el.attachMovie ("waiter", "waiter", 432);
}
```

U lokalnu promenljivu `el` smešta se deo koji odgovara delu koji je protivnik okrenuo (elementi objekta `memory` nazvani su `"e" + vrstaDela + kolonaDela`, što određuje poziciju dela na polju igre, imena počinju sa "e" da bi mogli biti identifikatori), njegov karakter postaje vidljiv igraču i, slično kao i kod protivnika, ispituje se da li je broj poteza bio paran ili ne. Takođe i preduzima potpuno iste akcije: ako jeste paran, postavlja `memory.el` i negira `even`, a ako nije – zaključava i kači `waiter`.

Objekat `waiter` kači se na drugi okrenuti deo jer sadrži u `_parent` referencu na njega:

```
var e0 = _parent._parent.el;
var e1 = _parent;
```

Ispituje se da li je postignut pogodak:

```
if (e1.value.value == e0.value.value) {
      e1._visible = false;
      e0._visible = false;
      if (_root.myMove) {
            _root.myHit++;
      } else {
            _root.opponentHit++;
      }
      _root.left--;
      if (_root.left == 0) {
            if (_root.myHit < _root.opponentHit) {
                  _root.quit ();
            } else {
                  _root.wins++;
            }
      }
}
```

Ako su vrednosti karaktera dva okrenuta dela jednaki, postignut je pogodak. Delovi se isključuju i treba brojati pogodak. Ako je igrač bio na potezu, to je njegov pogodak (`myHit++`), inače je pogodak protivnika (`opponentHit++`). Promenljiva left drži informaciju o preostalom broju parova koje treba pogoditi i, kako je postignut pogodak, umanjuje se za jedan. Ako je pogođeni par ujedno i poslednji, donosi se odluka o pobedniku (u igri je 30 delova, odnosno 15 parova, što garantuje mogućnost odluke o pobedi igrača sa većim brojem pogodaka). Ako je broj pogodaka igrača manji

od pogodaka protivnika, on je izgubio pa se diskonektuje, inače je pobedio i broji pobedu više.

Inače, ako se vrednosti karaktera dva okrenuta dela ne podudaraju:

```
e1.value._visible = false;
e0.value._visible = false;
_root.myMove = !_root.myMove;
_root.updatePlayer ();
```

Gase se karakteri okrenutih delova, menja se igrač koji je na potezu, `updatePlayer` postavlja ispis imena igrača na potezu.

Posle svega, još malo administracije:

```
_root.even = true;
_root.locked = false;
this.removeMovieClip ();
```

Odigran je paran broj poteza, uklanja se zaključavanje igre, a `waiter` je odradio posao i može se ukloniti.

ZAKLJUČAK

Činjenica da predstavljeni primeri imaju veličinu korisničkog fajla (.swf filma) od tridesetak KB, a da u potpunosti definišu ponašanje aplikacije i realizuju kompletan korisnički interfejs, nedvosmisleno govori u prilog tvrdnji da Flash filmovi jesu pravo rešenje u svetu Interneta.

Vektorska grafika i animacija Flasha i jezik ActionScript svojom raznovrsnošću omogućavaju rešavanje širokog spektra web problematike. Od kreiranja navigacionih rešenja u okviru HTML stranica, animiranih reklamnih sličica, do izrade kompletnih web-prezentacija u Flashu i direktnu razmenu podataka sa web serverom i njihovu dalju obradu.

Svojom rasprostranjenošću Flash već danas jeste standard, a sutra je već počelo, jer tu je Flash MX, koji definitivno zaokružava priču koju je započeo Flash 5 uvodeći ActionScript.

LITERATURA

- Standard ECMA-262, 3rd Edition – December 1999
- Philip Kerman : ActionScripting in FLASH, SAMS Publishing, 2001
- Macromedia : Using Flash MX, 2002

BIOGRAFIJA AUTORA

Dejan Katašić je rođen u Novom Sadu 3. juna 1970. godine, od majke Dragice i oca Petra, i ima i sestru Jovanku-Vanju. Od 27. januara 1999. godine je u braku sa Milkom (devojačko Stanarević), a 21. marta iste godine postaje srećni otac Helene.

Osnovno obrazovanje stekao je u školama "Petefi Šandor"i "Prva vojvođanska brigada" iz Novog Sada. Zajedničko srednje obrazovanje stiče u SMŠ "7. april" iz Novog Sada. Na proleće 1987. osvaja 26 poena na pokrajinskom takmičenju iz matematike. Usmereno srednje obrazovanje završava u "Tehničkoj školi za vojna usmerenja" (TŠVU) u Zagrebu, 13. jula 1989, kada stiče zvanje "Radnik IV stepena stručnosti za električne uređaje na motornim i borbenim vozilima". Studije na smeru "diplomirani informatičar" na Institutu za matematiku i informatiku, PMF Novi Sad, započeo je 1992. godine kao redovan student, gde je položio sve ispite predviđene nastavnim planom i programom.

Od 31. jula 1989. do 31. jula 1993. zaposlen je u Novom Sadu u svojstvu građanskog lica na službi u JNA (kasnije civilno lice na službi u VJ) na radnom mestu "Elektromehaničar za vozila guseničare, ujedno i točkaše".

Od aprila 1997, pa do kraja te godine, zaposlen je u firmi DD "infoMedia" iz Novog Sada, gde radi na izradi prezentacija za Internet i održavanju web-sajta **yumedia.com**. Projekati rađeni u tom periodu su: jedna od prvih web-galerija slika novosadskih umetnika ("Alfa art media gallery"), prvi filmski festival iz Jugoslavije na Internetu (6. jugoslovenski festival igranog filma "Novosadska arena"), jedne od prvih domaćih koncertnih manifestacija na Internetu ("NS plus Štrand festval '97" i "Koncert godine '97"), prezentacija slika novosadskog slikara Marka Milovića i probno Internet izdanje Vojvođanskog građanskog lista "Nezavisni".

Godine 1998. nastavlja samostalan rad na web prezentacijama. Od maja 1998. do jula 1999. održava nedeljno web-izdanje lista "Nezavisni" (**www.nezavisni.co.yu**). Od septembra 1998. do aprila 2000. vodi sajt novisad.com. Od septembra 1998. održava sajt DD "Zvezda Film" iz Novog Sada (**zvezdafilm.novisad.com**), gde je do sada ispraćeno četiri festivala domaćeg igranog filma "Novosadska arena", više manifestacija "Fokus" i "Cinemania", i predstavljeno stotinak filmova. Početkom 2001. predstavlja pekaru "Milan" iz Novog Sada na Internetu (**www.pekara-milan.co.yu**). Tokom 2000. godine učestvuje u projektu organizacije "aDigital" iz Novog Sada na računarskom opismenjavanju kadrova iz nevladinih organizacija, gde vodi kurseve Windowsa i izrade Internet prezentacija.

Poznaje jezike HTML i JavaScript, upoznat je sa problematikom prezentacije različitih sadržaja na Internetu, klijentskog i serverskog skriptovanja, korišćenjem baza podataka na Internetu i drugih tema vezanih za Internet. Takođe, zna Flash i ActionScript.

UNIVERZITET U NOVOM SADU
PRIRODNO MATEMATIČKI FAKULTET
KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Redni broj	RBR	
Identifikacioni broj	IBR	
Tip dokumentacije	TD	Monografska dokumentacija
Tip zapisa	TZ	Tekstualni štampani materijal
Vrsta rada	VR	Diplomski rad
Autor	AU	Dejan Katašić
Mentor	MN	dr Zoran Budimac
Naslov rada	NR	Flash i ActionScript
Jezik publikacije	JP	srpski (latinica)
Jezik izvoda	JI	s/en
Zemlja publikovanja	ZP	SR Jugoslavija
Uže geografsko područje	UGP	Vojvodina
Godina	GO	2002
Izdavač	IZ	autorski reprint
Mesto i adresa	MA	Novi Sad, Trg D. Obradovića 4
Fizički opis rada	FO	(5/ 54/ 0/ 2/ 15/ 2/ 0)
Naučna oblast	NO	Računarske nauke
Naučna disciplina	ND	Internet tehnologije
Predmetne odrednice, ključne reči	PO UDK	Flash, ActionScript, Word Wide Web, Internet, vektorska grafika, animacija, interakcija, skriptovanje
Čuva se	ČU	
Važna napomena	VN	
Izvod	IZ	
Vektorska grafika i animacija Flasha i jezik ActionScript svojom raznovrsnošću omogućavaju rešavanje širokog spektra web problematike. Od kreiranja navigacionih rešenja u okviru HTML stranica, animiranih reklamnih sličica, do izrade kompletnih web-prezentacija u Flashu i direktnu razmenu podataka sa web serverom i njihovu dalju obradu.		
Datum prihvatanja teme od strane NN veća	DP	maj 2002.
Datum odbrane	DO	jun 2002.
Članovi komisije	KO	
Predsednik		dr Mirjana Ivanović, redovni profesor, Prirodno-matematički fakultet u Novom Sadu
Član		dr Miloš Racković, vanredni profesor, Prirodno-matematički fakultet u Novom Sadu
Član		dr Zoran Budimac, vanredni profesor, Prirodno-matematički fakultet u Novom Sadu

UNIVERSITY OF NOVI SAD
FACULTY OF NATURAL SCIENCES & MATHEMATICS
KEY WORDS DOCUMENTATION

Accession number	ANO	
Identification number	INO	
Document type	DT	Monograph documentation
Type of record	TR	Textual printed material
Contents code	CC	Graduation thesis
Author	AU	Dejan Katašić
Mentor	MN	dr Zoran Budimac
Title	TI	Flash i ActionScript
Language of text	LT	Serbian (Latin)
Language of abstract	LT	en/s
Country of publication	CP	FR Yugoslavia
Locality of publication	LP	Vojvodina
Publication year	PY	2002
Publisher	PU	Author's reprint
Publ. place	PP	Novi Sad, Trg D. Obradovića 4
Physical description	PD	(5/ 54/ 0/ 2/ 15/ 2/ 0)
Scientific field	SF	Computer Science
Scientific discipline	SD	Internet technologies
Subject Key words	SKW UC	Flash, ActionScript, Word Wide Web, Internet, vector graphics, animation, interactivity, scripting
Holding data	HD	
Note	N	
Abstract	AB	
Vector graphics and animation of Flash and ActionScript language with it's versitality enables solving of wide spectre web based problems. From creation of navigation inside HTML pages, animated banners, to comlete web presentations in Flash and direct data exchange with web server.		
Accepted on Scientific board on	AS	May 2002
Defended	DE	June 2002
Thesis Defend board	DB	
President		dr Mirjana Ivanović, full professor, Faculty of Natural Sciences and Mathematics, Novi Sad
Member		dr Miloš Racković, associate professor, Faculty of Natural Sciences and Mathematics, Novi Sad
Member		dr Zoran Budimac, associate professor, Faculty of Natural Sciences and Mathematics, Novi Sad