TALLINNA ÜLIKOOL

Matemaatika-Loodusteaduskond

Informaatika osakond

ActionScript'i harjutusülesanded

Bakalaureusetöö

Autor: Tanel Jõeäär

Juhendaja: Andrus Rinde

Autor: ..... "....." ..... 2006. a.

Juhendaja: ..... "....." ..... 2006. a.

Osakonna juhataja: ..... "....." ..... 2006. a.

Tallinn 2006

Sisukord

1. Sissejuhatus

Macromedia Flash on üks enimkasutatavaid vahendeid veebianimatsioonide loomiseks. Flash sobib ka interaktiivse materjali loomiseks. Enamus kasutajaid suudavad lihtsamad töövõtted ise omandada ja näiteks *bannereid* luua. Keerukamate materjalide loomiseks on vaja kasutada ka Flash'i programmeerimiskeele ActionScript, millest praeguseks on kasutusel juba versioon 2.0, vahendeid. ActionScript'i iseseisvalt omandamine pole paljudele jõukohane. Algajatele sobivaid materjale selle kohta eriti palju leida pole ning eestikeelsed puuduvad täiesti.

Töö eesmärgiks on koostada õppematerjal koos harjutusülesannetega ActionScript 2.0 olulisemate vahendite ja võimaluste kasutamiseks Flash rakenduses pädevale, kuid ActionScriptiga vähe kokkupuutunud algajale programmeerijale. Käesolev materjal ei püüa anda terviklikku ülevaadet antud programmeerimiskeele vahenditest, vaid pigem annab õppijale aimu selle võimalustest ning julgustab teiste materjalide abil lisa otsima.

Õppematerjal on ülesehitatud ülesannetele, mille eesmärk on tutvustada ActionScripti erinevaid kasutusvõimalusi.

Selleks:

- käsitleb autor Macromedia Flashi kasutamist sellisel määral, et oleks võimalik interaktiivse materjali loomine ja ActionScripti kasutamine
- uurib, milliseid õppematerjale on Flashi ja ActionScripti kohta võimalik leida ja selgitada olulisemad võimalused ja vahendid mida kindlasti käsitleda.
- koostab ülevaate ActionScript 2.0 keelest ja harjutusülesanded koos näidistega.

2. Olemasolevad õppematerjalid

Macromedia Flash 8 e-raamatud

Macromedia Flash'i veebilehelt[1] saab e-raamatutena tasuta alla laadida kollektsiooni kogu tarkvara dokumentatsioonist. Nende hulgas ka ActionScripti puudutavad materjalid "Learning ActionScipt 2.0 in Flash" ja "ActionScript 2.0 Language Reference". Raamatud on mahukad (vastavalt 830 ja 1378 lehekülge), ning sisaldavad põhjalikku informatsiooni kõige ActionScripti puudutava kohta, seda kahjuks tehniliselt. "ActionScript 2.0 Language Reference" on sobiv allikas siis, kui on soov lisainformatsiooni leida mingi kindla ActionScripti funktsiooni kohta. Raamatud on saadaval ka paberversioonis.

Actionscript.org Macromedia Flash Resources and Tutorials[2]

Esmapilgul mahuka allikana (üle 200 õppetüki) tundunud veebileht valmistas lähemal uurimisel pettumuse – mõningad õppetükid on poolikud või liiga vähe selgitusi sisaldavad. Siiski võib kindlale probleemile lahenduse otsija siit sobiva leida. Õppetükid on jaotatud kolme taseme vahel: algaja tase (*beginning level*), kesktase (*intermediate level*) ja edasijõudnute tase (*advanced level*). Algus on kena (õpetatakse muutujaid omistama, instantse looma jne), hiljem muutuvad õppetükid ebakorrapäraselt jaotunuiks – nii näiteks õpetatakse XML-baasil menüüd looma algaja tasemel, XML-i tutvustuse leiab aga kesktasemelt.

Küll on aga veebisaidil saadaval põhjalik skriptikogu eri valdkondade kohta on koodinäiteid saadaval pea 700.

Kirupa.com[3]

Leitud veebimaterjalidest kõige põhjalikum. Õppetükid on jaotatud teemade, mitte konkreetsete ülesannete kaupa. Teemad on enamasti lahti seletatud näiteülesannete varal. Kahjuks puudub samm-sammult lähenemine, õppijal tuleb probleemide korral lisa otsida mujalt. Lisaks leiab veebilehelt ka põhjaliku foorumi, mida õppijat iga õppetüki lõpus külastama julgustatakse.

[1] URL: http://www.macromedia.com/support/documentation/en/flash/
[2] URL: http://www.actionscript.org
[3] URL: http://www.kirupa.com

Senocular.com[4]

Saadaval valik ActionScript'i faile koos asjakohaste ning parajalt põhjalike kommentaaridega.

CartoonSmart.com[5]

Õppijal on võimalus alla laadida neli tasuta õppevideot algajatele, edasised on juba tasulised.

LearnFlash.com[6]

Samuti õppevideotel baseeruv tasuline õppekeskkond. Mõned videod on vaatamiseks saadaval veebilehel kirupa.com.

Kokkuvõtteks peab autor tõdema, et head, ülevaatlikku, algajale sobivat materjali ei leidu. Ülipopulaarse Macromedia Flash tarkvara kohta leidub veebis siiski üllatavalt vähe õppematerjale, veelgi vähem on neid Flashis kasutatava programmeerimiskeele ActionScript kohta. Eestikeelsed materjalid puuduvad täielikult.

[4] URL: http://www.senocular.com
[5] URL: http://www.cartoonsmart.com
[6] URL: http://www.learnflash.com

3. Harjutusülesannete koostamine

Harjutusülesannete valikul on lähtutud teistest vaatluse all olnud õppematerjalidest, saamaks aimu, mida algajatele reeglina õpetatakse. Kohaliku arvamuse saamiseks konsulteeris autor õppejõuga, kes samuti soovitas näiteharjutusi. Lisaks on lähtutud teiste kasutajate kogemusest ning isiklikust arvamusest.

Reeglina algavad programmeerimisõpikud ikkagi ühtemoodi – õpetatava keele tutvustusega, muutujate deklareerimise ning lihtsamate käskudega. Ka leitud ActionScripti õppematerjalid algasid samamoodi, kahjuks läks aga tee sealt edasi, ActionScripti keskpärase omandamiseni, erinevate autorite vaatevinklis üsnagi erisuguseks.

Kõigest eelnevast lähtuvalt koostas autor selgitustega harjutusülesannete kogumiku.

4. Flashi võimalused ja puudused

Flash on loomevahend [Vallaste] (*authoring tool*), mille abil on võimalik luua presentatsioone, multimeediumirakendusi mis sisaldavad heli, videot ning eriefekte ning interaktiivset sisu. Flashi projektid võivad sisaldada lihtsaid animatsioone, videopilti, komplektsemaid rakendusi ning kõike vahepealset. [GSF, lk. 5]

Enne kui alustada tööd Flashiga (ja sh ActionScriptiga), tuleks selgeks teha, mida on võimalik Flashi abil teostada ning mida mitte.

- **Luua uskumatult väikese failimahuga kõrgkvaliteedilisi animatsioone.** Flash on laialdaselt kasutusel videote, lühikeste animafilmide ning reklaamibännerite levitamiseks internetis.
- **Integreerida peaaegu kõiki multimeediumiformaate.** Flash dokumenti saab importida rastergraafika faile (GIF, JPEG, PNG, PCT, TIFF), vektorgraafikat (teiste hulgas FreeHand, EPS, Illustrator ja ka PDF-failid) ning heliformaate (WAV, AIF ja MP3). Selleks pole vaja mingeid "lisasid" ega pluginaid.
- **Luua kujundust, mis on kõikjal ühesugune.** HTML-kujundus on erinevate veebilehitsejatega erinev, Flash'iga võib aga kindel olla, et see näeb ühesugune välja igal seadmel, mis toestab Flash Player'it.
- **Kuvada ja manipuleerida andmetega välistest allikatest, nagu andmebaasid.** Andmebaas, mis on ligipääsetav mõne rakendusserveri (*application server*), nagu Macromedia ColdFusion'i või PHP poolt, võimaldab Flash animatsioonil töötada dünaamiliste andmetega.
- **Võimaldab mitme kasutaja poolset interaktiivsust mängudes, jututoasessioonides jne.** Mitu inimest saavad omavahel suhelda läbi Flashi animatsiooni. Alates Flash Player 5-st on võimalik kasutada XML-porte (*sockets*) viivituseta suhtluseks animatsiooni ja serveri vahel. Flash Player 6 ja selle edasiarendustega saab kasutada Macromedia Flash Communication Server'it, sünkroniseerimaks reaalajas edastatavaid andmeid mitme osaleja vahel.
- **Laadida teisi Flash animatsioone ühe suurde esitlusse.**
- **Jooksvalt laadida JPEG, MP3 või FLV** (Flashi videofail) **faile Flash-animatsiooni,** kasutades Flashi komponente ja ActionScripti.

- **Luua malle (*templates*).** Kasutades MovieClip-sümboleid, mis toetavad kohandatavaid parameetreid, on võimalik luua taaskasutatavaid liidese elemente. Lisaks on Flashil (alates MX 2004) eelnevalt valmistehtud mallid dünaamiliste Flashi animatsioonide loomiseks. Mallide kasutamine tõstab produktiivsust, võimaldades muuda seadistusi ning spetsifikatsioone vajaduseta kogu projekti otsast peale alustada.
- **Käivitada Flash animatsioone paljudel erinevatel platformidel ja seadmetel.** Flash Player'i versioonid on saadaval operatsioonisüsteemidele Windows, Macintosh, Linux, Solaris, OS/2, SGI IRIX ja Pocket PC-le. Samuti suudavad paljud mobiiltelefonid suudavad Flash-animatsioone maha mängida. Tähele tasuks siiski panna opsüsteemide/seadmete erinevaid ekraanisuurusi.
- **Käivitada animatsiooni eraldiseisvate presentatsioonidena – projektoritena (*projectors*).** Projektorid on Flash-animatsioonid sisseehitatud mängijaga – nende vaatamiseks pole vaja veebibrauserit ega Flash Playerit.
- **Saata sisu printerile.** PrintJob klassi kasutades on võimalik lasta printida terve kaadri sisu või dünaamilise sisu jooksvalt. Näiteks on võimalik luua animatsioon, mis tutvustatava toote kohta prindib välja sooduskupongi.
- **Saata heli- või videovoog ühelt kasutajalt teisele.** Macromedia Flash Communicaton Server'i abil on võimalik ühenduda kasutaja veebikaamera ning mikrofoniga, luues reaalajas videokonverentse.

Lisaks eelistele on Flashil ka mõningaid puudusi, mida tuleks rakenduse loomiseks vajalikku tehnoloogiat (olgu selleks siis Flash või midagi muud) valides silmas pidada:

- **Flashi animatsioonid vajavad enamuse brauserite puhul siiamaani plugina allalaadimist ja paigaldamist.**
- **Sõltuvalt brauseri tüübist võib Flash animatsiooni funktsionaalsus** (enamasti skripti ja andmete interaktiivsus) **olla piiratud.**
- **Veebibrauserid ei suuna automaatselt alternatiivsele allikale kui Flashi plugin on paigaldamata.**
- **Flash animatsiooni ei saa importida ega seal kuvada tõepäraseid 3D animatsioone.** 3D-efekte saab küll vektoranimatsioonis kaader-kaadrihaaval imiteerida või kasutada selleks mõningaid ActionScripti funktsioone.

- **Veebi otsingumootorid ei suuda Flash animatsiooni sisu indekseerida.** Luues 100% Flashil baseeruvaid veebilahendusi, tuleb siiski lisada ka teksti või HTML-koodi, kui soovime, et veebilehed saaks indekseeritud.
- **Flash ei tohiks asendada tekstil baseeruvaid veebisaite.** Kui veebirakenduse peamiseks sisuks on tekst lihtsa graafikaga, ei tohiks selleks kasutada Flashi. Paljud kasutajad ei pea teksti selekteerimist ja printimist Flashi animatsioonidest mugavaks.
- **Kui arendaja(d) ei ole kogenud Flashi kasutaja(d) ning arendusperiood on lühike, ei ole Flash parim lahendus.** [Bible, lk. 22-26]

5. Flashi terminid

Kaadrid (*Frames*)

Iga kiht ajateljel koosneb kaadrite jadast. Eristatakse staatilisi (ehk tavalisi) kaadreid (*frame*) ning võtmekaadreid (*keyframe*). Võtmekaadrid tähistavad sisu muutumist, olgu selleks kaader-kaadrihaaval animeerimine (muudetakse sisu igal võtmekaadril) või graafilist interpoleerimist (*tweening,* kahe võtmekaadri vahelist sisu animeeritakse vastavalt nende erinevusele), staatilised kaadrid kordavad eelmise kaadri sisu. [Bible, lk.12]

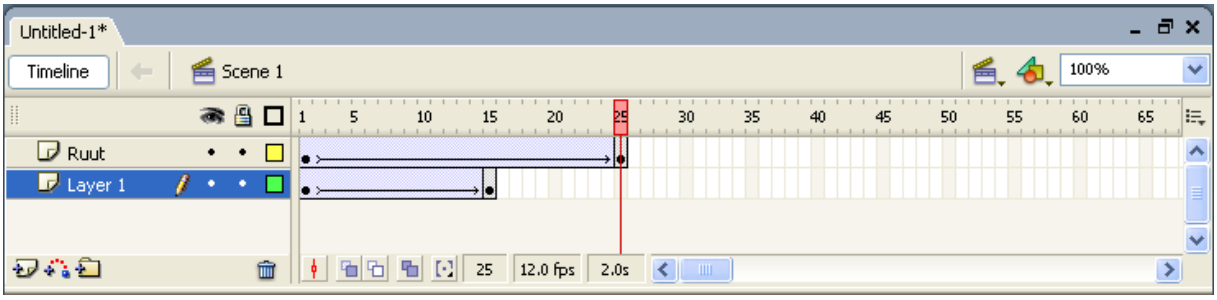
Stseenid (*Scenes*)

Stseenid on Flashi ajatelje osadeks või segmentideks. Iga stseen võib sisaldada erineva arvu kaadreid (sh võtmekaadreid), Flash Player laeb stseenid järjestikuliselt. [Bible, lk.12]

Kihid (*Layers*)

Kihid hoiavad Flashi dokumendi sisu ja organiseerivad elementi stseenis, kus kõige pealmine kiht kuvatakse laval kõige pealmisena ning madalaim kõige taga. Kihi ajatelg algab alati võtmekaadriga. Kihtide paremaks organiseerimiseks võib kasutada kaustu. [Bible, lk.12]

Kaadrite, stseenide ning kihtide paiknemist selgitab joonis 1.



Joonis 1 - Kaadrid, stseenid ja kihid Flashi ajateljel. Kihid võivad kanda vaikimisi nimesid (*Layer 1*) või olla kasutaja poolt määratud (*Ruut*).

Lava *(Stage)*

Lava on suur ristküliku-kujuline ala, millele lisatakse kogu sisu, mida soovitakse animatsioonis näha. Kõik, mis asub väljaspool lava, ei ole animatsiooni ajal nähtav. [VTHC]

Sümbol (*Symbol*)

Sümboliks võib olla iga objekt või hulk objekte, animatsioon või nupp. Sümboleid saab luua

ka üksteise sisse – niimoodi üks animeeritud MovieClip-sümbol koosneda mitmest graafilisest sümbolist, tekstist jne.

Sümbolid jagunevad kolme liiki:

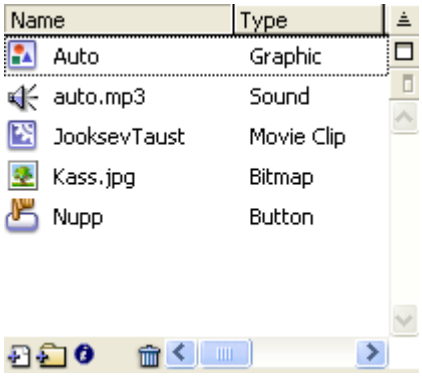
- Graafilised sümbolid on kõige lihtsamat ja loogilisemat tüüpi sümbolid, neid saab kasutada staatiliste objektidena laval või animatsioonis.
- Movie clip on animatsioon animatsioonis, mida saab interaktiivseid ohjureid kasutades kontrollida (ActionScript).
- Nuppe (*button*) kasutatakse Flashis interaktiivsuse loomiseks – kasutaja saab liikuda kaadrite vahel; valida, kas soovib animatsiooni näha; juhtida mängu käiku jm. Kui teistel sümbolite ajatelg on sarnane üldise ajateljega, siis nupul on see teistsugune. Tavapärased kaadrid on asendatud kaadritega *Up*, *Over*, *Down* ja *Hit* – niimoodi on võimalik muuta nupu väljanägemist vastavate olekute korral ning vajadusel ka lisada skript. [Dummies]

Instants (*Instance*)
Kui lohistame sümboli lavale, oleme loonud instantsi antud sümbolist. Niimoodi võib laval olla ka mitu instantsi, tihtilugu ongi see vajalik. Instantsidele saab anda erinevad nimed ning seega omistada ActionScripti abil neile erinevaid käsklusi.

Instantside kasutamine vähendab tunduvalt failimahtu – Flash hoiab failis vaid sümbolit ennast, ning kasutavatel instantsidel vaid viidatakse sümbolile. [Dummies]

Objektikogu (*Library*)
Objektikogu koondab endas kõiki animatsioonis kasutusel olevaid, aga ka loodud, kuid kasutamata objekte.

Lisaks sümbolitele saab objektikokku lisada multimeediumi; nagu pilte, heli ja videot; ka väljastpoolt (vt joonis 2). Selleks tuleb menüüst valida File ⇨ Import ⇨ Import to Library... ning valitud failid ilmuvad objektikokku. Kui objektikogu ekraanil näha pole, tuleks kontrollida kas Window ⇨ Library ees on linnuke või vajutada klahve Ctrl + L.



Joonis 2 – Erinevad sümbolid objektikogus

Autor (*Author*)

Autorina käsitletakse Flash rakenduse loojat.

6. Failitüübid Flashis

Käesolevas õppematerjalis mainitakse ülesannete sisus mitmel korral erinevaid Flashi failitüüpe. Olgu nad siinkohal selgitusena väljatoodud:

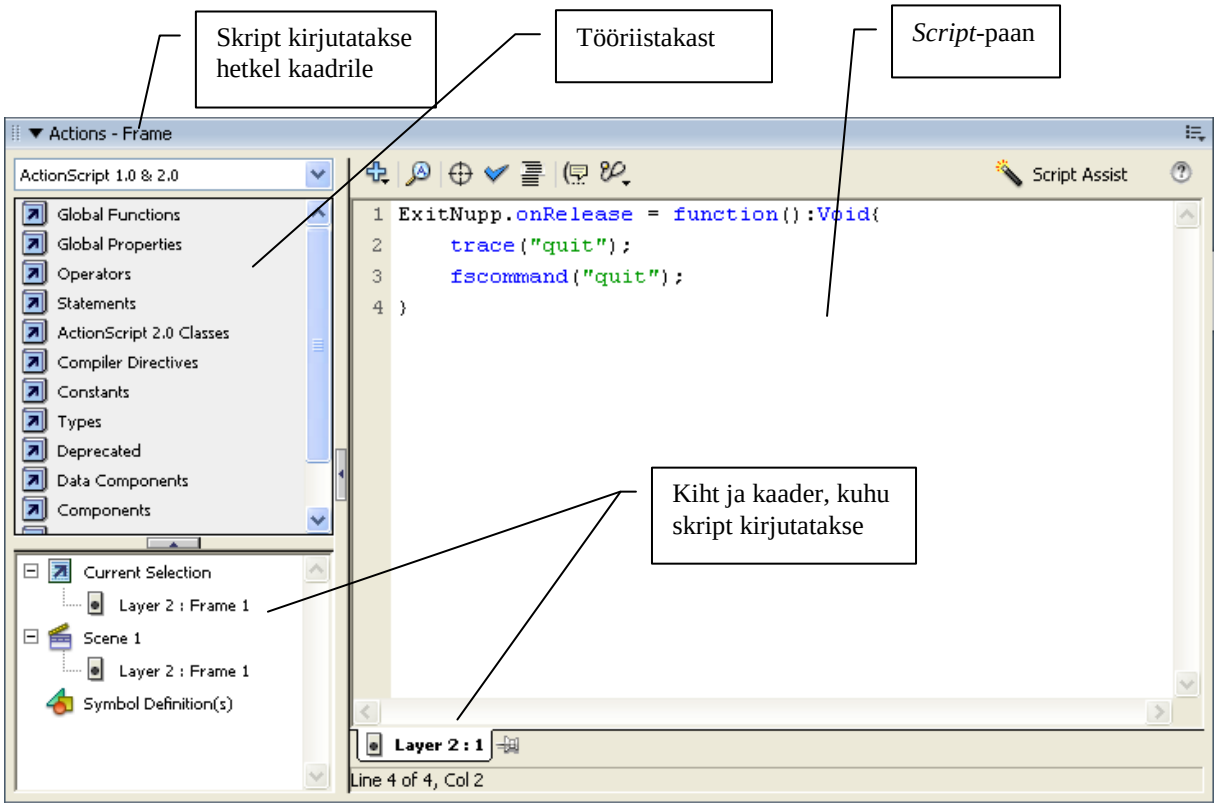
- **.FLA** – failid, millega töötakse, luues või muutes multimeediumisisu Flashis. Peamine failitüüp Flashis.
- **.SWF** – Flash-film (*Flash movie*), FLA-failide kompresseeritud versioon, need käivitatakse Flashi plugina või iseseisva pleieri (*stand-alone player*) abil. Flash-filmi salvestatakse ainult need objektid, mida laval kasutatakse, optimiseerimaks faili suurust. Kõik objektikogus olevad, kuid kasutamata elemendid filmi üle ei tooda ning taaskasutusel olevad salvestatakse ühekordselt, hiljem neile refereerides. Kompressioon ei ole sarnane ZIP-pakkimisele, kuna pildid ning heli kompresseeritakse eraldi. Kompressiooniseadeid saab muuta, valides File ⇨ Publish Settings... ⇨ Flash. [Bible, lk. 13-14]
- **.AS** – ActionScripti eraldiseisvad failid. Seda failitüüpi saab kasutada, kui soov hoida osa või kogu ActionScripti kood FLA-failist eraldi. See on kasulik organisatoorsetel eesmärkidel ning samuti siis, kui ühe projekti kallal töötab mitu inimest.
- **.SWC** – Failitüüp taaskasutatavate Flashi komponentide hoidmiseks. SWC-fail sisaldab kompileeritud filmiklippi, ActionScripti koodi ning muid vajalikke lisasid.
- **.FLV** – Flashi videofail.
- **.ASC** – ActionScripti käivitamiseks arvutil, mis jooksutab Flash Communication Serverit. Need failid võimaldavad kasutada serveripoolset loogikat, mida saab rakendada koos ActionScriptiga SWF-failis.
- **.JSFL** – JavaScripti failid.
- **.FLP** – Flashi projektifailid (ainult Flash Professionalis). Võimaldavad hallata erinevaid Flashi dokumente ühes projektis. Võimaldab kaasata mitmeid, seostatud faile ühe komplektse rakendusena. [Bible, lk. 13-14] [GSF, lk. 50-51]

7. ActionScript

ActionScript, millele käesolev õppematerjal keskendub, on programmeerimiskeel, mida saab lisada Flashi dokumentidele. ActionScripti abil saab meediale lisada interaktiivsuse ning Flash dokumendi käitumist paremini kontrollida. Niimoodi on näiteks võimalik nupul klõpsamise peale näidata uut pilti. Loogika võimaldab rakendusel käituda kasutaja tegevusele ning teistele konditsioonidele vastavalt. Ilma ActionScriptita on Flashis võimalik paljutki teostada, kuid ActionScript lisab palju teisi võimalusi. Flash sisaldab kahte versiooni ActionScriptist, mõlemad kohaldatud autori vajadustele. [LAS2, lk. 35]

Skripti lisamiseks kirjutatakse ActionScript otse Actions-paneeli. Väliste skriptide lisamiseks, mida hiljem rakendusse lisada või importida, võib kasutada spetsiaalset skriptiakent (File ⇨ New ⇨ ActionScript File) või vabalt valitud tekstiredaktorit. [LAS2, lk. 35]

Nii paneelil *Actions* kui ka aknal *Script* on *Script*-paan (see, kuhu kirjutatakse koodi; vt joonis 3) ning Actions tööriistakast. *Actions* paneelil on veidi rohkem assisteerimisvõimalusi kui *Script* aknal. Avamaks *Actions* paneeli, tuleb valida menüüst Window ⇨ Actions või vajutada F9.



Joonis 3 – *Actions*-paneel

ActionScript baseerub ECMAScript'i programmeerimiskeelel, sama süntaksi kasutab ka veebis laialt levinud JavaScript. Seetõttu leidub neis ka sarnasusi. [Wikipedia[6]]

7.1. ActionScript 1.0 või 2.0?

Enne Flashis tööle asumist tuleb otsustada, kuidas valmiv dokument või rakendus ning sellega seotud failid organiseerida. Näiteks võib keerulisemaid Flash-rakendusi luues vaja minna klasside kasutamist (selgitatud hilisemates harjutusülesannetes). Väikesemahuliste rakenduste loomisel ei ole klasside kasutamine funktsionaalsuse tagamisel lihtne ega hea lahendus. Kasulikum on paigutada ActionScript loodava dokumendi sisse. Sellisel juhul peaks kogu kood asetsema ajateljel, võimalikult vähestel kaadritel, vältida tuleks koodi paigutamist instantsidele (nagu nupud või MovieClip-sümbolid).

ActionScript 2.0 poolt pakutav süntaks on sarnane teistele programmeerimiskeeltele. See teeb keele õppimise lihtsamaks ning väärtuslikumaks. ActionScript võimaldab arendamisel kasutada objekt-orienteeritud lähenemist, kasutades täiendavaid keeleelemente.

Mõningatel juhtudel ei saa valida, millist ActionScripti versiooni kasutada. Luues Flash-faili, mis on suunatud vanemale Flash Player'ile või näiteks mobiilsele seadmele, tuleb kasutada ActionScripti versiooni 1.0, mis on ühilduv paljude seadmete Flash-esitajatega.

Hoolimata kasutatava ActionScripti versioonist tuleb järgida häid programmeerimistavasid. Paljud sellised tavad, nagu tõstutundlikusele (*case sensitivity*) kindlaks jäämine, loetavuse parandamine, võtmesõnade vältimine instantsidele nime andmisel ning läbivate nimereeglite (*naming conventions*) kasutamine, kehtivad mõlemale versioonile.

Kui on plaan uuendada oma rakendus tulevastesse Flash'i versioonidesse või arendada seda suuremaks ning kompleksemaks, tuleks kasutada ActionScripti versiooni 2.0 koos klassidega. Sel moel on rakenduse muutmine ja uuendamine laiemas perspektiivis lihtsam.

Kui kompileerida ActionScript 2.0 sisaldav SWF-fail publitseerimisseadetega Flash Player 6 ja ActionScript 1.0, funktsioneerib kood senikaua, kuni seal pole kasutusel ActionScript 2.0 klassid. Range andmetüüpimine (*strict data types*) on nõutav publitseerides rakenduse Flash Player versioonile 7 või 8, koos ActionScripti versiooniga 2.0. [LAS2, lk. 69-70]

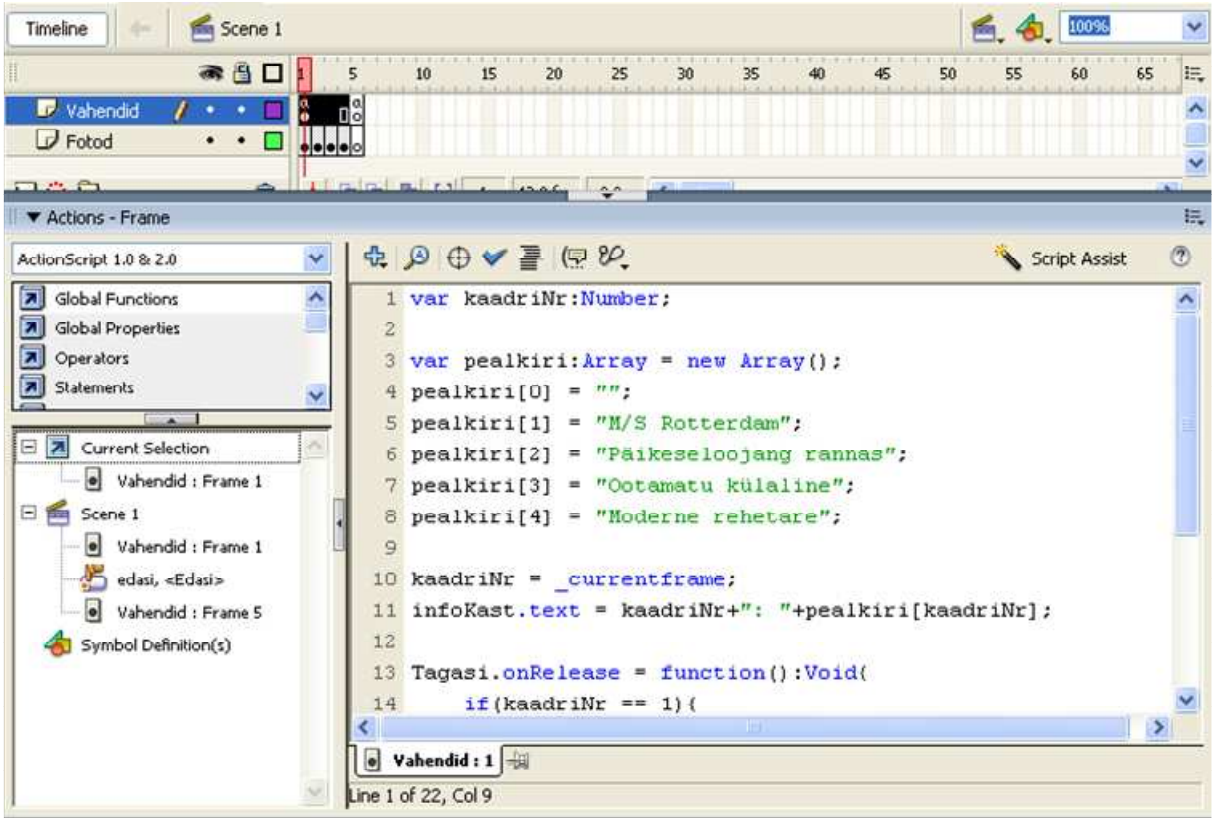
ActionScript 2.0 range andmetüüpimine võimaldab muutuja tüübi selle loomisel selgesõnaliselt dekraleerida. Kuna andmetüübi kokkusobimatused tekitavad kompilaatorivigu,

aitab range andmetüüpimine leida koodis vigu juba kompileerimise ajal, takistades olemasolevale muutujale valet tüüpi andmete omistamist. [LiveDocs]

8. ActionScript'i õppematerjal

8.1. Sissejuhatus ActionScript'i

ActionScript'i kirjutades tuleb jälgida, kuhu kood paigutatakse. Kood võib paikneda ühel kaadril, ühistel kaadritel või olla omistatud objekti(de)le. Joonisel 4 redigeeritakse parasjagu *Vahendite* kihi 1. kaadri ActionScript'i, mis samaaegselt mõjub kaadritele 1-4, ajateljel on näha, et ka 5. kaadril on ActionScript. Samuti on näha, et ka nupule *edasi* on võimalik omistada ActionScript'i. ActionScript'is kasutatakse pöördumiseks instantsinime (*Instance name*) `Edasi`, instantsinime saab määrata objekti seadistuste (*Properties*) paneelil. Erinevad objekti- ja instantsinimed on selles õppematerjalis kasutusele võetud selleks, et rõhutada nende lahusolekut.



Joonis 4 - ActionScripti võib kirjutada kaadrile, nende kogumile või objektile.

Soovituslik on ActionScripti kood kirjutada eraldi kihile, ning eraldi (tühjadele) kaadritele. Joonisel 4 paikneb ActionScript kihil *Vahendid*. Ebasoovitatav on kirjutada ActionScript

instantsidele (näiteks joonisel 4 nähaolev nupp *edasi*), kuna see suurendab veaohtu ning raskendab ülevaadet.

8.2. Sündmused (events)

Sündmused saavad olla kas kasutaja- või süsteemipoolsed. Kasutajad klõpsavad hiirenuppe ja vajutavad klahve, süsteem päästab sündmusi valla juhul, kui jõutakse mingisse olukorda või mingi protsess on läbitud (SWF-fail laeb, ajateljel jõutakse teatud kaadrini jne). ActionScript'is kirjutatud sündmuseohjur (*event handler*) püüab sündmuse kinni ja tegutseb vastavalt. Sündmuseohjur ja sündmus saavad eksisteerida iseseisvalt, nagu uksekell ja kellanupp. Vajutades nuppu (sündmus) ei juhtu midagi, kuni pole kella (sündmuseohjur), mis saaks nupu vajutamisel heliseda (tegevus). [Bible, lk. 68] [LAS2, lk. 32, 35]

Flashis saab sündmused jagada kolme kategooriasse: hiire ja klaviatuuri sündmused (kui kasutaja suhtleb Flash-rakendusega), MovieClip'i sündmused, ning kaadri sündmused.

Sündmused jagunevad:

- *Key* (klaviatuuriklahv)
- *Mouse* (hiir)
- *Selection* (selekteerimine)
- *Stage* (lava*)*
- *TextField* (tekstiväli)

Hiirega käivitavad sündmused

Tabel 1 - Hiirega käivitatavad sündmused [Bible, lk. 225]

`onPress()`
`onRelease()`
`onReleaseOutside()`
	Enimkasutatavad sündmusehaldurid. `onPress()` ja `onRelease()` meetodeid kasutatakse juhul, kui soovitakse sündmust käivitada, kus kasutaja klõpsab objektil hiirega või vabastab hiireklahvi. Vahel võib mõlemaid meetodeid kasutada koos, näiteks luues lohistatavat MovieClip'pi, kus on vajalik hiirt klõpsates alustada objekti lohistamist ning hiirenupu vabastades see lõpetada. `onReleaseOutside()` käivitub, kui kasutaja on klõpsanud hiirega instantsi sees, kuid vabastab hiire sellest väljaspool.

`onRollOver()` `onRollOut()`	Käivitakse, kui kasutaja sõidab instantsist hiirega üle või sellest välja.

`onSetFocus()` `onKillFocus()`	Kui nupp või MovieClip fokuseeritakse, käivitakse `onSetFocus()`. Kui instants fookuse kaotab, käivitatakse `onKillFocus()`. Fokusseeritud objekt saab vastu võtta klaviatuurilt käivituvaid sündmusi (klahvivajutus).
`onDragOver()` `onDragOut()`	`onDragOut()` käivitakse, kui kasutaja klõpsab hiirega instantsil, lohistab seejärel hiire instantsist välja ning vabastab hiire, `onDragOver()` peab kasutaja hiire vabastama tagasi instantsis olles. Nende meetodite puhul tuleb tähele panna, et ei tööta, kui fookus on mõnel teisel instantsil.

MovieClipi sündmused

Tabel 2 - MovieClip sündmused [AS2LR, lk. 901-911]

`onEnterFrame()`	Kutsutakse välja korduvalt, vastavalt SWF-faili kaadrisagedusele (*frame rate*, *fps*). Näide: kui SWF-faili kaadrisagedus on 12 kaadrit sekundis, kutsutakse `onEnterFrame()` välja sekundi jooksul 12 korda.
`onLoad()` `onUnload()`	Kutsutakse välja vastavalt kui MovieClip käivitatakse laval või kui see sealt eemaldatakse.
`onData()`	Käivitub, kui MovieClip saab andmeid `MovieClip.loadVariables()` või `MovieClip.loadMovie()` kaudu.

`onMouseDown()`
`onMouseUp()`
`onMouseMove()`
	Kutsutakse välja vastavalt hiirenupu vajutuse, vabastamise või hiire liigutamise järel. Liigitatakse *Mouse*-klassi alla.

`onKeyUp()` `onKeyDown()`	Kutsutakse välja klahvivajutuse või –vabastuse järel. Liigitatakse *Key*-klassi alla.

Sündmusekuularid

Sündmuseid ning sündmuseohjureid käsitlesime juba varasemas harjutusülesandes. Alates Macromedia Flash versioonist MX on võimalus lisaks Flashis eeldefineeritud nuppudele ja MovieClip-sümbolitele omistada objektidele ka endaloodud sündmusi. Seda `AsBroadcaster` nimelise klassi abiga. [Kirupa[2]]

Niipea kui mingi objekt saab kuulari omadused, suudab see tuvastada levitava (*broadcasting*) objekti poolt väljasaadetava sündmuse. [Kirupa[2]]

Ülalmainitud objektidele saab lisada kuulareid, kasutades `addListener()` meetodit. [Kirupa[2]]

Sündmusekuulari töö paremaks selgituseks on järgnev näide:

```
// Loome uue objekti
yksSuurK6rv = new Object();

// Defineerime funktsiooni
K6rvakiil = function(){
      trace("Ai, ma sain just kõrvakiilu!");
}

// Teeme objekti yksSuurK6rv hiire kuulariks...
Mouse.addListener(yksSuurK6rv);

// ...ning määrame sellele sündmuse onMouseDown.
// Kuna yksSuurK6rv on justnimelt hiire objekti kuular,
// tunneb see hiire sündmuse ära sarnaselt MovieClip-sümbolile
yksSuurK6rv.onMouseDown = K6rvakiil;
```

Antud näite toimimist saab ise katsetada. Kõikjal laval klõpsates avaneb väljundiaknas teade "Ai, ma sain just kõrvakiilu!". [Kirupa[2]]

8.3. Muutujad ja nende kasutamine ActionScriptis

Muutuja kasutuselevõtmisel tuleb see esmalt deklareerida. Muutuja deklareeritakse skripti koodis kujul

```
var muutujanimi:andmetüüp;
```

Muutuja nime võib ise valida kuid soovitav oleks järgida mõningaid reegleid:

- Muutuja nimi ei tohi alata numbriga; muutujad peavad algama tähe, allkriipsu (_) või dollari märgiga ($).
- Muutuja nimes ei tohi esineda tühikuid.
- Muutuja nimeks ei tohi olla Flashi poolt kasutatav märksõna või mõni muu eriväärtus. (Nt muutujanimena ei tohi kasutada `MovieClip`, `true`, `String`, `undefined` jne, kuna neil on ActionScriptis juba antud tähendused.)

- Ja muidugi ei tohi kaks erinevat muutujat olla sama nimega, kuna hilisem kirjutab varasema sel juhul lihtsalt üle. [Bible, lk. 94-99]

Tähele tasub panna, et muutujanimed (nagu ka funktsioonide jm nimed) on ActionScriptis tõstutundlikud (case sensitive). Niimoodi ei ole `eesnimi` ja `EesNimi` üks ja seesama.

Muutujatele nime andmisel võib lähtuda mõnest üldtuntud kontseptsioonist, üheks selliseks on Ungari notatsioon.

Ungari notatsiooniks kutsutakse sellist muutujatele nimede andmise kontseptsiooni, kus muutuja nime algus näitab ära tema tüübi. Suuremate ja keerukamate projektide juures, eriti juhul, kui üht programmi arendab mitu inimest ja suhteliselt sarnaste nimedega muutujaid on lähtekoodis palju, on see sageli suureks abiks. [Kusmin]

Tabel 3 - Ungari notatsioon ActionScriptis [Bible, lk. 98]

Prefiks	**Andmetüüp**	**Selgitus**
`a`	`Array`	massiiv
`b`	`Boolean`	loogikamuutuja
`bt`	`Button`	nupp
`c`	`Color`	värv
`cam`	`Camera`	-
`cm`	`ContextMenu`	kontekstimenüü
`cmi`	`ContextMenuItem`	kontekstimenüü punkt
`d`	`Date`	kuupäev
`lc`	`LocalConnection`	lokaalühendus
`lv`	`LoadVars`	*LoadVars* klass
`mc`	`MovieClip`	-
`mcl`	`MovieClipLoader`	-
`mic`	`Microphone`	-
`n`	`Number`	-
`nc`	`NetConnection`	võrguühendus
`ns`	`NetStream`	võrguvoog
`o`	`Object`	-
`pj`	`PrintJob`	-

`rs`	`RecordSet`	-
`s`	`String`	-
`snd`	`Sound`	-
`so`	`SharedObject`	jagatud objekt
`t`	`TextField`	-
`tf`	`TextFormat`	-
`vid`	`Video`	-
`xml`	`XML`	-
`xmls`	`XMLSocket`	-

Niimoodi võib millegi kogust näitava muutuja nime defineerida `nKogus` (või `nQuantity`), kasutaja eesnime hoidvat muutujat aga `sEesNimi` (`sFirstName`), vastavalt sellele, et kogus on arvuline väärtus, eesnimi aga sõne (*string).* Siinkohal tasub tähele panna, et Ungari notatsiooni kasutamine annab muutujate tüübi kohta selgust inimesele, küll aga mitte ActionScriptile.

Andmetüübid

Tabel 4 - ActionScript'is kasutusel olevad andmetüübid [LAS2, lk. 72-81]

Andmetüüp:	**Selgitus:**
Boolean	Loogikamuutuja, mille väärtuseks saab olla `true` (tõene) või `false` (väär), vastavalt 1 või 0. Deklareeritud, kuid algväärtustamata muutuja väärtuseks on `false`.
MovieClip	Võimaldab kontrollida filmiklipi sümboleid, kasutades selleks MovieClip-klassi meetodeid. Järgnev näide kutsub välja `startDrag()` meetodi laval asuvale filmiklipile *minu_film*: `minu_film.startDrag(true);` MovieClip meetodid kutsutakse välja, kasutades operaatorina punkti (.).
null	Tõlkes „tühimärk”. Saab kasutada väärtusena märkimaks, et muutuja eksisteerib, kuid ei oma (veel/enam) väärtust. Funktsiooni väärtusena näitab see, et funktsiooni poolt pole võimalik ühtki väärtust tagastada. Funktsiooni parameetrina, et parameetri andmine on vahele jäetud. **Mitte segi ajada arvuga 0, mis iseenesest on väärtus!**
Number	Täis- ja ujukomaarvude tähistamiseks.

Object	Object-klassi poolt defineeritud andmetüüp.
String	*String* e. sõne on tähemärkide jada. Sõned tuleb panna jutumärkide (") või ülakomade (') vahele.
Void	Void andmetüüp kasutab ainult üht väärtust: `void`. Seda andmetüüpi võib kasutada nende funktsioonide tähistamiseks, mis ei väljasta mingit väärtust.

Andmetüüpide määramine

Kui jätta muutuja andmetüüp täpsustamata, püüab Flash muutujale antud väärtuse järgi andmetüübi ise paika panna. Järgnevas koodireas antakse muutuja `x` väärtuseks 3, ilma etteantud andmetüübita kohtleb Flash seda kui numbrit:

```
var x = 3;
```

Kuna muutuja `x` andmetüüp polnud deklaleeritud, võib Flash sellele ükskõik millise andmetüübi omistada. Hilisem muutuja väärtuse muutus võib seega muuta ka selle tüüpi, näiteks kui muutuja `x` saab väärtuseks `x = "tere"`, muutub muutuja andmetüüp *stringiks*. ActionScript teisendab primitiivseid andmetüüpe (*Boolean*, *Number*, *String*, *null* või *undefined*) automaatselt, kui kood seda nõuab ning andmetüüp pole täpselt määratud.

Tähele tasuks panna ka järgmist olukorda:

```
var tekst:String = "abc";
tekst = 12;
```

ActionScript 2.0 annab kompileerimisel veateate, kuna sõne-tüüpi muutujale soovitakse omistada numbrilist väärtust. Lahenduseks tuleks paigutada 12 jutumärkide vahele. Tõsi küll, numbrilisi arvutusi sõnega siiski teha ei saa.

8.4. Funktsioonid

Funktsioone võib võtta kui kokkugrupeeritud koodiridu, mis käivitatakse alles funktsiooni väljakutsumisel ning mis seejärel teostavad mingi kindlaksmääratud protseduuri.

Funktsioonidel on palju eeliseid struktureerimata programmeerimise ees, näiteks:

- Kood muutub loetavamaks.

- Programm muutub tõhusamaks, kuna funktsiooni on lihtsam taas välja kutsuda kui kirjutada iga kord sama koodi.
- Et hiljem mõnda protseduuri muuta, piisab funktsiooni sisu muutmisest ning protseduur ongi muudetud kõikjal, kus see välja kutsutakse.
- Hästi kirjutatud protseduure saab kasutada mitmetes programmides. Niimoodi saab luua protseduuride kogu (näiteks eraldi ActionScripti faili), mida saab kasutada paljude programmide loomiseks, ilma, et peaks iga kord alustama samade protseduuride kirjeldamisest. [Bible, lk. 125-141]

Funktsioonide loomine

Funktsiooni loomist nimetatakse ka funktsiooni defineerimiseks.

```
function funktsiooninimi():andmetüüp {
	koodiread
}
```

Funktsioonile nime andmisel tuleks järgida samu reegleid kui muutujate nimetamisel. Sarnaselt muutujatele on kasulik oma funktsioonile nimi anda selle järgi, mida see teeb. Anda funktsioonile nimeks `mingiFunktsioon` pole nii hea mõte, kui näiteks `joonistaUusRing`.

Funktsiooni nime järel peavad asetsema ümarsulud, niimoodi on funktsioonile võimalik lisada parameetreid. Hoolimata sellest, kas funktsioonile antakse parameetreid või mitte, on ümarsulud kohustuslikud. Parameetrite andmist funktsioonile käsitleme hiljem.

Viimasena tuleb defineerida funktsiooni andmetüüp, mis määrab ära, millist tüüpi andmeid funktsioon väljastab. Andmete väljastamist käsitleme hiljem. Kui funktsioon andmeid väljastama, kasutame andmetüüpi `Void`. [Bible, lk. 125-141]

Näitena vaatame väga lihtsat funktsiooni:

```
function ytleTere():Void {
	trace("Tere!");
}
```

Funktsioon `trace()` väljastab teate Flashi väljundipaneelile. See on kasulik näiteks veaotsimisel, kui on tarvis teada, millal (või kas üldse) mõnda koodiosa läbitakse. [Bible, lk. 66-67]

Funktsiooni väljakutsumine

Kui funktsiooni loomise osas kirjeldatud koodinäide tööle panna, ei toimuks justkui midagi, kuigi `trace` on koodis sees. Seda seetõttu, et funktsioon `ytleTere()` pole välja kutsutud.

Funktsiooni väljakutsumiseks tuleb kirjutada defineeritud funktsiooni nimi koos sellele järgnevate sulgudega. Järgnev koodirida defineerib funktsiooni ning kutsub selle seejärel välja [Bible, lk. 125-141]:

```
function ytleTere():Void {
      trace("Tere!");
}
ytleTere();
```

Funktsioonile parameetrite andmine

Paljud funktsioonid vajavad ka parameetreid. Näiteks võiksime teha `ytleTere()` funktsiooni märksa huvitavamaks, kui muudaksime tervituse isikupärasemaks.

```
function ytleTere(sEesnimi:String):Void {
      trace("Tere, " + sEesnimi + "!");
}
```

Kui selline funktsioon on defineeritud, saab seda välja kutsuda ning anda parameetrina erinevaid väärtusi, näiteks:

```
ytleTere("Mari");
```

väljastab:

```
Tere, Mari!
```

Funktsioonile saab anda korraga ka mitu parameetrit, sellisel juhul tuleb need eraldada komaga. [Bible, lk. 125-141]

8.5. Viide "this"

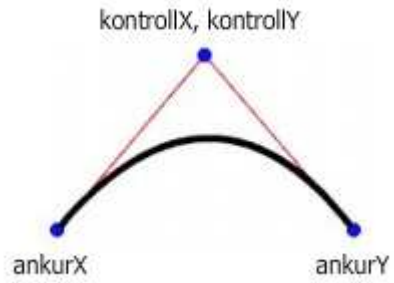
Paljud objektid kasutavad oma töös samade nimedega meetodeid, prototüüpe või funktsioone, võivad nad kõik muuta samu muutujaid. Et seda vältida, tuleb Flashile selgitada, millised muutujad sõltuvad antud objektist, deklareerides neid viitega *this* [Kirupa[1]]:

```
onClipEvent (enterFrame){
   this.muutuja = 5;
}
```

8.6. Joonistamise rakendusliides (Drawing API)

Graafiliste kujundite loomine ActionScripti abil on keerulisem kui see olema peaks. Siiski on võimalik see kord selgeks saades luua kasvõi üdini koodil baseeruvaid animatsioone.
Põhilised joonistamise rakendusliidese funktsioonid on:

- `lineStyle(`*`joonePaksus, värv, alpha`*`)` – Kuigi lineStyle parameetreid on koguni 8, käsitleme neist vaid kolme esimest. Joone paksus tuleb anda vahemikus 0-255. Värvi täpsustamine on omajagu raske – need tuleb määrata 16-süsteemis RGB-koodina. Niimoodi on punase värvi koodiks 0xFF0000 ning sinise 0x0000FF. *Alpha* (0-100) ehk läbipaistvus ei ole kohustuslik parameeter. [AS2LR, lk. 885-888]
- `moveTo(`*`x, y`*`)` – Liigutab kujutletava pliiatsiotsa soovitud koordinaatidele. Kui seda ei määrata, alustatakse joonistamist koordinaatidelt (0, 0), ehk siis lava vasakust ülanurgast.
- `lineTo(`*`x, y`*`)` – Tõmbab joone pliiatsiotsa asukohast etteantud lõpukoordinaatideni. Tähele tasub panna, et ekraanile ei ilmu midagi, kui funktsiooniga `lineStyle()` pole määratud joone stiil. Pärast joone tõmbamist jääb pliiatsiots joone lõpukoordinaatidele, funktsiooni `moveTo()` pole vaja uuesti kasutada.
- `curveTo(`*`kontrollX, kontrollY, ankurX, ankurY`*`)` – kõvera tõmbamiseks on vaja ette anda nii ankurpunkti (*anchor point*) koordinaadid, milleks on kõvera lõpppunkt, kui ka kontrollpunkt (*control point*), mis moodustub kõvera puutujate lõikepunktis (vt joonis 8).



Joonis 5 - Funktsiooni `curveTo()` parameetrid

- `beginFill(rgb, alpha)` / `endFill()` – Loodava kujundi värviga täitmiseks pannakse funktsioon `beginFill()` enne `moveTo()` ja `lineTo()` käske. Parameetrite definitsioonid samad kui funktsioonil `lineStyle()`.
- `clear()` – Puhastab viidatava MovieClip-sümboli. [Kirupa[3]]

8.7. Lause with()

Tihti tuleb ühele MovieClip-sümbolile anda palju seadistusi. Niimoodi näeks MovieClip-sümbolile Sisu seadistuste andmine välja niimoodi:

```
Sisu.tekst = "Põdral maja metsa sees...";
Sisu._x = 456;
Sisu._y = 345;
```

Parem ja selgem meetod ühele kindlale objektile viitamisel oleks `with()`-lause. See kutsub objekti välja ühel korral ning võimaldab ligipääsu kõigile objekti seadistustele korraga. Näitena näeksid samad seadistused välja nii:

```
with(Sisu) {
      tekst = "Põdral maja metsa sees...";
      _x =456;
      _y =345;
}
```

Lisaks esteetilisele välimusele saab `with()` lauset kasutades hõlpsasti kokku grupeerida objekti seadistused. [Kirupa[4]]

9. Harjutusülesanded

9.1. Pildigalerii navigeerimisnuppudega

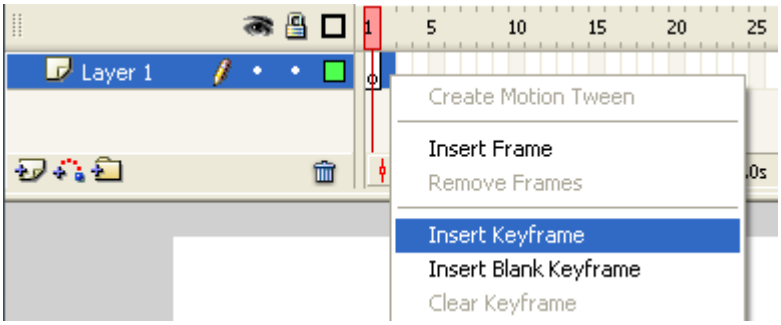
Loome väikese fotogalerii, kus iga pilt oleks eraldi võtmekaadril. Piltide sirvimiseks lisame navigeerimisnupud ning kirjutame nende ActionScripti koodi. Iga pildi alla kuvame pealkirja andmete massiivist.

Õpime:

- Kasutama muutujaid ning massiive
- Kuvama infot dünaamilisse tekstikasti
- Kasutama animatsioonis liikumiseks nuppe
- Kirjutama ActionScripti mitmele kaadrile

Näidetest fail nav.fla

1. Esmalt impordime pildid Flashi objektikogusse (*Library*).
2. Lohistame iga pildi eraldi võtmekaardile esimesel kihil. Peale viimast pildiga võtmekaadrit teeme veel ühe võtmekaadri, mis edaspidi hakkab suunama esimesele võtmekaadrile.
3. Lisame uue kihi (*Insert Layer*). "Layer 1" võib selguse mõttes ümber nimetada kihiks "Fotod", tekitatud "Layer 2" kihiks "Vahendid".



Joonis 6 - Võtmekaadri lisamine

4. Loome uued objektid: nupp *edasi*, nupp *tagasi* ning graafilise sümboli *tekstiKast*. Sümbol *tekstiKast* koosneb ristkülikust (*rectangle*), staatilisest tekstikastist fotode autori nimega ning dünaamilisest tekstikastist *infoKast*.
5. Lohistame vastloodud objektid selleks loodud uuele kihile soovitud kohtadele laval. Uus kiht peab olema pikendatud eelviimase kaadrini, viimaseks kaadriks samamoodi tühi võtmekaader.
6. Anname nuppudele *edasi* ja *tagasi* instantsinimed `Edasi` ja `Tagasi`. Selleks tuleb vastav objekt selekteerida ning seadistuste (*Properties*) paneelil instantsinime (*Instance* name) lahtrisse (valiku *Button* all) kirjutada vastavalt `Edasi` ja `Tagasi`.
7. Avame Actions-paneeli, valides menüüst Windows ⇨ Actions või vajutades klaviatuuril F9.
8. Jälgida tuleb, et koodi hakataks kirjutama Vahendite kihi esimesele (võtme)kaadrile. Esmalt paneme tööle nupud. Eelmisele/järgmisele kaadrile liikumiseks on kasutada funktsioonid `nextFrame()` ja `prevFrame()`, teeme nii, et vastav funktsioon käivituks siis, kui vabastame hiireklahvi nupul `Edasi` või `Tagasi`.

   Kasutame näites sündmust `onRelease()`, kirjutame:

```
Edasi.onRelease = function(){
            nextFrame();
};

Tagasi.onRelease = function(){
      prevFrame();
};
```

   Kontrollida tulemust.

   Eeldame, et tahame piltide all näha ka pealkirju või kommentaare. Taolisi fotogaleriisid luues võivad fotograafid soovida, et näha oleks ka foto pealkiri, tehnilised näitajad (nagu säriaeg, diafragma arv) jm.

   Probleemi võib lahendada kahte moodi. Võib kopeerida igale kaadrile kindlasse kohta staatilise tekstivälja ning sisestada sinna soovitud info. Meie näites on kasutusel dünaamiline tekstiväli, mis väljastab info andmemassiivist vastavalt kaadrile, millel parasjagu asutakse.

9. Dünaamiline andmeväli sai graafilisse objekti *tekstiKast* lisatud punktis nr. 4, nüüd pole muud, kui see soovitud informatsiooni kuvama saada.Esmalt delegeerime muutuja `kaadriNr` (numbriline andmetüüp) ning omistame talle väärtuseks `_currentframe`, mis on Flashi enda konstant. Senise koodi algusesse kirjutame:

```
var kaadriNr:Number = _currentframe;
```

10. Nupul klõpsates soovime, et kaadri number suureneks või väheneks ühe võrra. Selleks kirjutame vastavate funktsioonide juurde lisaks

    `kaadriNr++;` (edasi) ja

    `kaadriNr--;` (tagasi).

    `kaadriNr++` on sama, mis kirjutada: `kaadriNr = kaadriNr + 1`.

11. Kuvame kaadri numbri ka meie infokasti. Et kaadri numbrit uuendatakse, on järgnev koodirida vaja kopeerida nii nuppude funktsioonide sisse kui ka põhikoodi (et Flashi esitlusse sisenedes kuvataks tekst ka siis, kui hiirega polegi veel kuhugi klõpsatud).

```
infoKast.text = kaadriNr;
```

12. Teeme massiivi piltide pealkirjadest. Massiivi saab hoida hulka ühetüübilisi andmeid. Massiivi esimeseks elemendiks on null-element. Jätame selle lihtsuse mõttes tühjaks, et kaadrile 1 vastaks pealkiri[1], mitte pealkiri[0] jne.

```
var pealkiri:Array = new Array();
pealkiri[0] = "";
pealkiri[1] = "M/S Rotterdam";
pealkiri[2] = "Päikeseloojang rannas";
pealkiri[3] = "Ootamatu külaline";
pealkiri[4] = "Moderne rehetare";
```

13. Viimaks lisame pealkirja kuvamise praegusele kaadrinumbri näitamisega:

```
infoKast.text = kaadriNr + ": " + pealkiri[kaadriNr];
```

14. Viimase sammuna lisame "Vahendite" kihi viimasele kaadrile koodi:

```
gotoAndStop(1);
```

    Niimoodi hüpatakse peale viimasel pildil edasi klõpsamist tagasi esimesele kaadrile.

Kogu valminud rakenduse kood:

- Kiht "Vahendid", kaader 1:

```
var kaadriNr:Number = _currentframe;

var pealkiri:Array = new Array();
pealkiri[0] = "";
pealkiri[1] = "M/S Rotterdam";
pealkiri[2] = "Päikeseloojang rannas";
pealkiri[3] = "Ootamatu külaline";
pealkiri[4] = "Moderne rehetare";

infoKast.text = kaadriNr+": "+pealkiri[kaadriNr];

Edasi.onRelease = function():Void {
      kaadriNr++;
      infoKast.text = kaadriNr+": "+pealkiri[kaadriNr];
      nextFrame();
}

Tagasi.onRelease = function():Void{
      if(kaadriNr == 1){
      }else{
      kaadriNr--;
      infoKast.text = kaadriNr+": "+pealkiri[kaadriNr];
      prevFrame();
      }
};

stop();
```

- Kiht "Vahendid", viimane kaader:

```
gotoAndStop(1);
```

9.2. Pildigalerii ajastusega

Eelmine ülesanne käsitles pildigaleriid navigeerimisnuppudega, nüüd aga loome pildigalerii, mille pildid vahetuksid ajastusega. Samal moel töötavad ka paljud Flashis loodud ribareklaamid – *bannerid*, kus pildid ja tekstid teatud aja järel vahetuvad.

Õpime:

- Mis on funktsioonid ja kuidas neid välja kutsuda.
- Kuidas sama nimega funktsioon võib eri kaadritel toimida erinevalt.

Näidetest fail banner.fla

1. Taaskord vajame esmalt objektikogusesse imporditud pilte ning nende kihtidele lohistamist. Soovi korral võib teistele kihtidele luua tekstikastid või kasutada eelmises ülesandes selgitatud dünaamise tekstikasti ja massiivi näidet.
2. Loome uue kihi ActionScripti tarvis.
3. Selleks loodud tühjale kihile kirjutame kaks ActionScripti, esimese esimesele kaadrile, mõjualaga eelviimase kaadrini ning teise viimasele kaadrile.
   - Esimene kaader:

```
function edasi():Void {
      nextFrame();
}
var intervalID:Number = setInterval(edasi, 5000);
stop();
```

   Funktsiooni `edasi()`ülesandeks on väljakutsumisel liikuda järgmisele kaadrile.

   `setInterval()` funktsiooni kasutatakse funktsiooni või meetodi välja kutsumiseks teatud perioodi järel, mil SWF-fail mängib, parameetriteks on funktsiooni/objekti nimi ning numbriline intervall. Arv 5000 märgib millisekundeid, seega on intervalli pausiks 5 sekundit.

   Muutujat `intervalID` kasutatakse `setInterval()` poolt tagastatava väärtuse (intervalli numbri) hoidmiseks.

- Viimane kaader:

```
function edasi():Void {
      clearInterval(intervalID);
      gotoAndPlay(1);
}

clearInterval(intervalID);
var intervalID:Number = setInterval(edasi, 5000);
stop();
```

Nagu näeme, omab funktsioon `edasi()`järgmises kehtivuspiirkonnas, milleks on viimane kaader, hoopis teistsugust ülesannet, nimelt lõpetab see intervalli ning suunab tagasi esimesele kaadrile. Kehtivuspiirkondi nimetatakse ka skoopideks.

`clearInterval(intervalID)`, nagu nimigi ütleb, lõpetab muutujaga `intervalID` ette antud intervalli tegevuse.

Et enne uue intervalli loomist eelmist lõpetada, tuleks `clearInterval()` kindlasti välja kutsuda enne `setInterval()` funktsiooni. See kindlustab, et `intervalID` muutuja, mis on ainus viide hetkel käimasolevale intervallile, ei kirjutata üle ega hävitata muul moel. [AS2LR, lk. 65]

Topelt kood (kustutame intervalli, seejärel loome uue, et see siis uuesti kustutada funktsioonis `edasi()`) , on vajalik selleks, et ka viimasel kaadril peatutaks, vastasel korral liigutakse sellest lihtsalt üle, kui vaid `edasi()` funktsioon välja kutsuda.

4. Ongi valminud vahetuvate piltidega pildigalerii.

9.3. Sümbolitega manipuleerimine ActionScripti abil

Luues interaktiivset rakendust on vajalik, et Flashi objekte saaks animeerida ka vastavalt kasutaja soovidele. Selles harjutusülesandes õpimegi, kuidas sümbolit laval animeerida ActionScripti vahenditega. Lisaks uurime, mil viisil on võimalik suhelda Flash Playeri või muu projektoriga.

Õpime:

- MovieClip-sümboli animeerimine laval ActionScripti vahenditega
- Muutujate väärtuste hõlpsat suurendamist

- Väliskeskkonnaga suhtlemine funktsiooni fscommand() abil

Näidetest fail anim.fla

1. Joonistame animeerimiseks ringi ning muudame selle MovieClip-sümboliks nimega *ring*.
2. Lohistame vastloodud sümboli lavale (kui see seal juba pole) ning anname sellele instantsinime `Ring`. Jälgida tuleb, et loodud sümboli keskkoht MovieClip-sümboli sees oleks ikka tõepoolest keskel, selle järgi liigutatakse sümbolit laval (vt joonis 6).

Joonis 7 – Sümboli keskkoht on paigutatud MovieClip'i keskele

3. Loome uue kihi, millele hakkame ActionScripti kirjutama.
4. Skript:

```
var Serv:Number = Stage.height - (ring._height / 2);
```

Muutuja `Serv` asemel võiksime *if*-tingimuses kasutada ka lihtsalt parameetrit `Stage.height`, kuid sellisel juhul lõpetatakse sümboli liigutamine alles siis, kui selle keskkoht puutub kokku lava alumise servaga. Kui soovime, et liikumine lõppeks siis, kui MovieClip'i serv on jõudnud lava servani, peame lava pikkusest lahutama pool MovieClip'i pikkust (s.o. keskkohast servani, vt joonis 7).

Joonis 8 – Stage.height ja muutuja Serv kasutamise erinevus meie näites

Meie harjutusülesandes on kasutusel MovieClip'i sündmuseohjur `onEnterFrame()`, kuid sarnaselt saab reageerida ka näiteks kasutaja nupuvajutusele. Sündmus `onEnterFrame()` kutsutakse välja kaadrite arv kordi sekundis (meie näites 12). Niimoodi on lihtne arvutada, et esimeses *if*-tingimuse pooles liigutatakse ringi sekundis 24 (12 * 2) piksli võrra allapoole. Kui ringi serv on jõudnud lava alumise

ääreni, vähendatakse ringi kõrgust, samas seda suuremaks skaleerides ning lavalt minema nihutades. Niimoodi saavutatakse nn. "laialivalgumise" efekt.

```
onEnterFrame = function(){
	if(ring._y < Serv){
		ring._y += 2;
	}else{
		if(ring._height > 0){
			ring._height -= 2;
			ring._xscale += 2;
			ring._y += 1;
		}
	}
}
```

Tabel 5 - Parameetrid sümboliga manipuleerimiseks [Bible, lk. 228-232]

`_x`	x-koordinaadi muutus pikslites
`_y`	y-koordinaadi muutus pikslites
`_width`	Objekti laius pikslites
`_height`	Objekti pikkus pikslites
`_xscale`	Skaleerimine x-telge pidi protsentides
`_yscale`	Skaleerimine y-telge pidi protsentides
`_alpha`	Objekti läbipaistvus protsentides
`_visible`	Objekt nähtav või nähtamatu (`true`/`false`)
`_rotation`	Objekti pööramine kraadides
`_xmouse` `_ymouse`	Hiirekursori asukoht konkreetse objekti koordinaatväljal (*coordinate space*), kasutatakse peamiselt nn *rollover* jm efektide loomisel. Need parameetrid on kirjutuskaitstud (*read-only*)!

Koodis esinevad

```
ring._height -= 2;
ring._xscale += 2;
```

ei ole tegelikult muud kui antud väärtusele juurde või maha arvutamine, pikemalt välja kirjutades:

```
ring._height = ring._height + 2;
ring._xscale = ring._xscale - 2;
```

Kokkuvõtlikult:

`+=` suurendamine väärtuse võrra

`-=` vähendamine väärtuse võrra

`*=` suurendamine väärtus korda

`/=` jagamine väärtusega

5. Lisame meie animatsioonile omaduse, mis võimaldab selle laval asuvale nupule vajutades sulgeda. Loome lihtsa nupu *Nupp* ning lohistame selle lavale, andes instantsile nimeks `exitNupp`.

Funktsioon fscommand() [AS2LR, lk.54-58]

`fscommand()` võimaldab Flash-failil suhelda Flash Playeri või selle peremeesprogrammiga (*host*), nagu näiteks veebibrauser.

Funktsioon kirjeldatakse kujul `fscommand("käsk", "parameeter")`, viimased on kirjeldatud tabelis 6.

Tabel 6 - fscommand() atribuudid

Atribuut	**Parameeter**	**Selgitus**
`quit`	(puudub)	Sulgeb projektori (Flash Player).
`fullscreen`	`true` või `false`	Määrates `true`, läheb Flash Player täisekraanvaatesse, `false` naaseb normaalvaatesse.
`allowscale`	`true` või `false`	Määrates `false` seadistab Flash Playeri SWF-faili alati originaalsuuruses näitama ning keelab selle skaleerimise. `true` skaleerib animatsiooni 100%-le Playeri suurusest.
`showmenu`	`true` või `false`	Määrates `true` võimaldakse Playeri täiskontektsimenüü. `false` peidab kõik menüüelemendi, välja arvatud seadistused (*Settings*) ning info (*About Flash Player*).
`exec`	failiteekond	Käivitab failiteekonnaga etteantud programmi. Failiteekond võib sisaldada vaid numbreid 0-9, tähti A-Z, a-z (mitte täpitähti), punkti (.) ning allkriipsu (_).
`trapallkeys`	`true` või `false`	Määrates `true` saadab Flash kõik klahvisündmused `onClipEvent(keyDown/keyUp)` ohjurile Flash Playeris.

Tähelepanuks:

- Kõik käsud töötavad vaid iseseisvas rakenduses, projektoris.
- Vaid `allowscale` ja `exec` töötavad testkeskkonnas.

6. Skript:

```
ExitNupp.onRelease = function():Void{
      fscommand("quit");
}
```

   Rakenduse testimiseks tuleb fail publitseerida.

9.4. Interpoleerimine (tweening) Tween-klassi abil

Alates Flash versioonist MX 2004 saab Flashi interpoleerimise (*tweening*) võimalusi kasutada ka ActionScripti vahenditega. See harjutusülesanne keskendubki *Tween* klassi abil erinevate liikumisefektide loomisele.

Õpime:

- Klasside importimist
- ActionScripti abil graafilist interpoleerimist (*tweening*)
- Viide "*this*"

Näidetest fail tween.fla

1. Esmalt tuleks luua lihtne *MovieClip*-tüüpi sümbol, millele hiljem ActionScripti abiga efekte rakendame. See ei pea olema midagi erilist, piisab näiteks ka lihtsalt ühest teksti tööriistaga loodud sõnast.
2. Lohistame sümboli lavale ning anname talle instantsinimeks `mcTekst`.
3. Loome uue kihi, mille esimesele kaadrile kirjutame ActionScript koodi:

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
```

Need kaks koodirida impordivad Flashi dokumenti klassi *Tween* ning kõik klassid kaustast *easing*. Antud klassid asuvad ActionScript failidena Flashi kaustas \en\First Run\Classes\mx\ transitions\.

Meetod on sisuliselt funktsioon, mis asub klassi sees. Lihtsustatult öeldes on klass meetodite (funktsioonide) kogum, mis objekti luues sellele omistatakse. [PHP]

4. Järgmised kaks rida loovad kaks interpoleerimise instantsi ning hoiavad neid muutujatena `Keera` ja `Asukoht`.

```
var Keera:Tween = new Tween(mcTekst, "_rotation", Elastic.easeOut, 0, 360, 3, true);
var Asukoht:Tween = new Tween(mcTekst, "_x", Bounce.easeOut, 0, Stage.width, 3, true);
```

Esmajoones segasena tunduvad koodiread ei kujuta endast tegelikult midagi keerulist. Tween-klassi välja kutsudes tuleb määrata 8 parameetri väärtused.

a. **objekt** – MovieClip-sümboli instantsi nimi, millele soovitakse interpoleerimist luua. Meie näites on instantsiks `mcTekst`.

b. **omadus** – Omadus, mille järgi interpoleerida. Selleks võib olla `_alpha` (läbipaistvus), `_xscale` (x-skaala), `_yscale` (y-skaala), `_x`, `_y`, `_rotation` (pöörlemine), `_height` (pikkus), `_width` (laius). Meie näites vastavalt `_rotation` ja `_x`.

c. ***ease*-efekti klass ja meetod** – Interpoleerimise tüüp. Flashil on kuus sisseehitatud *ease*-klassiga.

   1) ***Back*** – Laiendab animatsiooni üle ühe või mõlema interpoleerimise ääre.

   2) ***Bounce*** – Loob põrkuva efekti.

   3) ***Elastic*** – Loob põrkuva ja laiendava efekti segu. Animatsiooni laiendatakse ja põrgatatakse tagasi.

   4) ***Regular*** **-** Aeglasem liikumine.

   5) ***Strong*** – Sarnane eelmisele, kuid kombineeritav erinevate *ease*-meetoditega.

   6) ***None*** – Lihtne lineaarne üleminek a-lt b-le.

Eelpoolnimetatud klassidel on omakorda 3 meetodit:

- ***easeIn* –** animeeritakse ainult interpoleerimise algust.
- ***easeOut* –** animeeritakse ainult interpoleerimise lõppu.
- ***easeInOut* –** animeeritakse mõlemaid pooli.

Kombineerides kuus *ease*-klassi ja kolm meetodit, saame 18 erinevat efekti. Meie näites on kasutusel `Elastic.easeOut` ja `Bounce.easeOut`.

d. ***begin*** – interpoleerimise algus.

e. ***end* –** interpoleerimise lõpp. Nagu näitest näha, ei peaks väärtused tingimata olema kindla arvuna ette antud (`Stage.width`).

f. ***kestvus*** – interpoleerimise kestvus. Selle väärtuse võib anda nii kaadrite arvuna kui ka sekundite arvuna, olenevalt järgmisest parameetrist.

g. ***kasuta sekundeid* –** Määrab ära, kas kestvus on antud kaadrite või sekunditena. Väärtustena tuleb kasutada `true` (kasutusel on sekundid) või `false` (kasutusel on kaadrid). Meie näites on kestvuseks 3 sekundit. [CLR, lk. 1313-1315]

5. Edasi kasutame meetodit `yoyo`, mis loob interpoleerimise lõpp-punktist tagasi alguspunkti. [CLR, lk. 1338] Kasutades *Tween*-klassi sündmuseohjurit (*event handler*) `onMotionFinished`, mis kutsub yoyo-meetodi välja iga kord, kui interpoleerimine `Keera` on lõpetatud. Seega saame lõputu interpoleerimise edasi ja tagasi:

```
Keera.onMotionFinished = function() {
      this.yoyo();
};
```

6. Sarnaselt talitame ka interpolarisatsiooniga `Asukoht`, kuigi nüüd kasutame meetodit `continueTo`, mis suunab interpolarisatsiooni asukoha määratud aja jooksul teise asukohta. `continueTo` vajab selleks kahte parameetrit: finiš – ehk kuhu peab jõudma ning kestvus, kaua selleks aega kulub. [CLR, lk. 1319] Kestvus võib olla nii kaadrites kui sekundites, olenevalt eelpool `Asukoht:Tween` muutujas määratust.

```
Asukoht.onMotionFinished = function() {
      this.continueTo(Stage.width / 2, 3);
};
```

7. Nagu näha, on *Tween*-klassi on soovitud efekti saavutamiseks üsnagi lihtne kasutada.

9.5. MovieClip-sümboli loomine ActionScripti vahenditega

Selles harjutusülesandes käsitleme dünaamilise MovieClip-sümboli loomist ning hiirekuularit.

Õpime:

- Looma sündmusekuulareid meetodi `addListener()`abil
- Kasutama ActionScripti joonistamise rakendust
- Kasutama objektile seadistuste andmiseks lauset `with()`
- Laadima üht Flash-animatsiooni pilti või teist animatsiooni

 **Näidetest fail movieclip.fla**

1. Selles ülesandes vajame vaid ühte kihti ning ühte kaadrit, millele kirjutame ActionScripti. Kood genereerib lavale dünaamiliselt ühe MovieClip-tüüpi sümboli.
2. Esmalt impordime juba tuttavaks saanud *Tween*- ja *easing*-klassid.
   ```
   import mx.transitions.Tween;
   import mx.transitions.easing.*;
   ```
3. Loome objekti, mille ülesandeks on "kuulata", millal kutsutakse välja sündmus `loadInit`. "`mc`" tähistab muutujanimedes *MovieClip*'i ning "`mcl`" *MovieClip Loader*'it ehk *MovieClip*'i laadijat.

   ```
   var mclKuular:Object = new Object();
   ```

   Järgnevalt paneme paika, mis toimub siis, kui `mclKuular` püüab sündmuse `onLoadInit` (laadimise algus):

   ```
   mclKuular.onLoadInit = function(siht_mc:MovieClip) {
         siht_mc._visible = false;

         //Joondame MovieClip-sümboli lava keskele
         siht_mc._x = (Stage.width - siht_mc._width) / 2;
         siht_mc._y = (Stage.height - siht_mc._height) / 2;
   ```

   Loome MovieClip-sümboli `maskClip` ning määrame selle ala raamiga.

```
                var maskClip:MovieClip =
                siht_mc.createEmptyMovieClip("mask_mc", 20);
                with (maskClip) {
                      beginFill(0xFF00FF, 100);
                      moveTo(0, 0);
                      lineTo(siht_mc._width, 0);
                      lineTo(siht_mc._width, siht_mc._height);
                      lineTo(0, siht_mc._height);
                      lineTo(0, 0);
                      endFill();
                }
                siht_mc.setMask(maskClip);
                siht_mc._visible = true;

                var mask_tween:Object = new Tween(maskClip, "_yscale",
Strong.easeOut, 0, 100, 2, true);
            };
```

Ning viimaks loome tühja MovieClip-sümboli `pilt-mc`, lisame kuulari, mis aktiveerub `pilt_mcl.loadClip()` käivitumisel ning edasi asub tööle ülalkirjeldatud funktsioon `mclKuular.onLoadInit`.

```
      this.createEmptyMovieClip("pilt_mc", 10);
      var pilt_mcl:MovieClipLoader = new MovieClipLoader();
      pilt_mcl.addListener(mclKuular);
      pilt_mcl.loadClip("http://www.liiklus.net/liiklus.png",
pilt_mc);
```

[AS2RL, lk. 956-957]

Sarnaselt võib `pilt_mc` asemel laaditavaks objektiks olla ka teine Flash-animatsioon.

9.6. Vormid ja integratsioon andmebaasiga

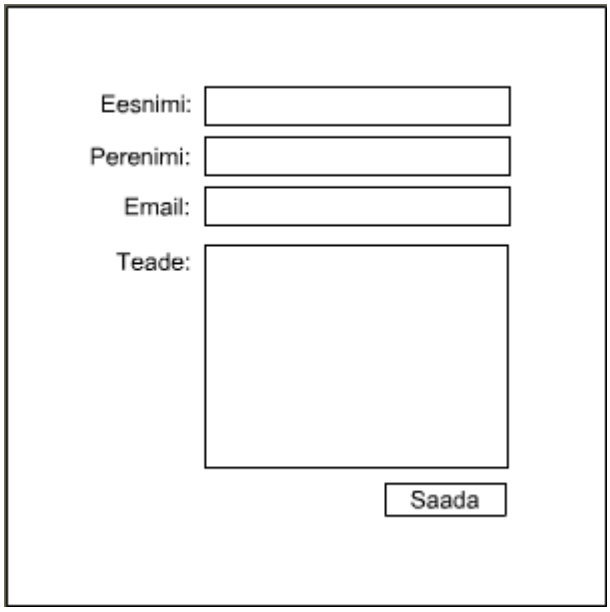
Juba varastest versioonidest (Flash 5) toetab Flash võimalust intergreerida rakendus andmebaasi või mõne muu välise andmete töötlemisvahendiga. Vaatleme, mil viisil on võimalik andmeid Flash rakendusest välja saata ning koostame lihtsa tagasiside vormi.

Õpime:

- Kuidas koostada vorme
- Kuidas saata andmeid välistele rakendustele

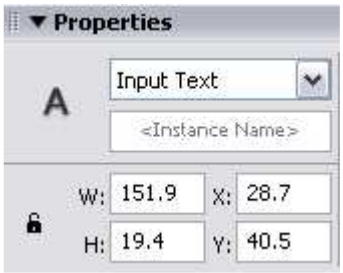
Näidetest failid vorm.fla, vorm.php, vorm_mysql.php

1. Kujundame vormi, nagu näha joonisel 9.



Joonis 9 - Lihtne vorm Flashis

Selleks loome kokku kaheksa tekstikasti, esimesed neli lahtrite selgituseks ning teised neli etendavad sisendikastide rolli. Et tavalisest tekstikastist saaks kasutaja teksti sisendikast, tuleb see seadistuses vastavalt määrata (vt. joonis 10), valides teksti tüübiks sisendteksti (*Input text*). [Kirupa[5]] Varasemates harjutusülesannetes oleme kasutanud staatilisi (muutumatu sisuga) ja dünaamilisi (ActionScripti abil muutuva sisuga) tekstikaste.



Joonis 10 – Sisendkasti loomine

Lisaks vajame veel nuppu, millele klõpsamisel andmed edasi saadetakse.

2. Sisendikastidele tuleb anda muutujanimed, kirjutades need vastava tekstikasti seadetes lahtrisse *Var*. Nendeks on vastavalt: *eesnimi*, *perenimi*, *email*, *teade*. Samanimelised muutujad väljastatakse hiljem teisele rakendusele.

3. Kirjutame nupule uuel kihil järgneva skripti:

```
saadaNupp.onRelease = function():Void{
      getURL("http://koduleht.com/vorm.php",0,"post");
}
```

   Selleks on eelnevalt lavale lohistatud nupule vaja anda instantsinimi `saadaNupp`.

   Funktsiooni `getURL()` saab kasutada dokumendi laadimiseks kindlalt URL-ilt või muutujate saatmiseks teisele rakendusele selleks defineeritud URL-il. [AS2LR, lk. 59-61] Parameetriteks on *url*, *aken* - määrab ära akna või HTML-freimi, kuhu dokument peaks laaditama (meie näites lihtsalt "0") ning *meetodi*, kuidas muutujad saata (GET või POST, erinevad sellepoolest, kas meetodid saadetakse kaasa URL'iga või mitte).

   Nagu näha, piisab andmete väljaspoole saatmiseks sisuliselt vaid ühest koodireast. Sarnaselt võimaldab funktsioon `loadVariablesNum()` ka andmeid välisallikast (ColdFusion'i, CGI, ASP või PHP programmeerimiskeelte abil genereeritud tekstist või failist) sisse lugeda. [AS2LR, lk. 74-76]

4. Flashi-poolne osa on sellega tehtud – andmed saadetakse välja. Näitena kirjutame PHP programmi. PHP koodi jooksutamiseks on tarvis Apache't või mõnda muud veebiserverit ning PHP-d. MySQL andmebaasi võimaluste kasutamiseks peab rakendatud olema MySQL-server.

Flashiga saadetud andmete kuvamine PHP abil

Antud kood kuvab veebilehel Flashist saadud andmete põhjal ees- ja perekonnanime. See ei õigusta küll tagasiside vormi kasutamist, kuid näitab, mil viisil Flashi poolt väljastatud andmeid kuvada.

```
<?php
echo '<html><head>';
echo '<meta http-equiv="content-type" content="text/html;
charset=utf-8" />';
echo '</head><body>';
```

```
echo 'Tere,'.$_REQUEST[eesnimi].' '.$_REQUEST[perenimi].'!<br />';
echo 'Töötab!';

echo '</body></html>';
?>
```

Vaikimisi eeldavad Flash Player 7 ning selle hilisemad versioonid, et kogu tekst, millega need kokku puutuvad, on UTF-8 kodeeringus. Seetõttu peaksid välised teksti- ja XML-failid, mida rakendus kasutab, olema sarnaselt salvestatud UTF-8 kodeeringus. [UF, lk. 365] Vastasel korral on probleeme näiteks täpitähtede kuvamisel.

Flashist saadud andmete saatmine MySQL andmebaasi

Järgnev kood on näide PHP abil MySQL andmebaasiga ühendamisest. Esmalt kontrollitakse *if*-tingimuse abil, kas lehele saabudes on kaasa antud muutuja `teade`, kui see on tõene, luuakse ühendus andmebaasiga ning salvestatakse kirje. Antud juhul ei ole tegemist ainuõige lahendusega, vaid katsetamiseks mõeldud koodiga. Väljad *server*, *kasutajanimi*, *parool*, *andmebaas* ning *tabel* tuleks koodis asendada kehtivatega.

```
<?php
$REQUEST[teade] = $teade;

if(isset($teade)){
      mysql_connect("server", "kasutajanimi", "parool");
      mysql_select_db("andmebaas");

      $lause="INSERT INTO tabel (eesnimi, perenimi, email, teade)
VALUES ('$_REQUEST[eesnimi]', '$_REQUEST[perenimi]', '$_REQUEST[email]',
'$_REQUEST[teade]')";
      mysql_query($lause);
      mysql_close();

      echo "Korras!";
      }
?>
```

9.7. Flash ja XML

XML (eXtensible Markup Language) on erinevate andmete struktureerimiseks mõeldud märgistuskeel. [Vallaste] XMLi kunagisest põhieesmärgist – andmete kirjelduskeelest WWW rakenduste jaoks on välja kasvanud universaalse andmekirjeldus ning -edastus keele idee. Üha enam kasutatakse XMLi ärirakendustes süsteemi erinevate osade vahel informatsiooni vahetamiseks, samuti suhtlemiseks teiste süsteemidega. Kuna XML andmeid hoitakse tekstikujul, on neid lihtne vahetada paljude erinevate platvormide vahel taskuarvutist suurte serveriteni – tavaline tekstitöötlus on neile kõigile jõukohane. [Haamer]

XMLi saab Flashis ära kasutada mitmel moel – alates uudisesöödetest [Sõnastik], kasutajahalduste, sisukordade ning isegi jututubadeni. [Bible, lk. 631] Eeldan, et lugejal on piisavad teadmised XML-ist või suudab ta need teiste materjalide abil omandada, XMLi juhendit käesolev materjal ei korva.

Oletame, et veebilehel on Flashiga loodud menüü. Kui tahaksime seal midagi muuta, tuleks avada Macromedia Flash programm, vastav link muuta või lisada, seejärel kompileerida uus *swf*-fail ning see uuesti veebisaidile laadida.

Õnneks on olemas XML. Kasutame XMLi, et menüüstruktuuri defineerimiseks. Näitena on toodud menüü, mis võiks olla ühe informaatika eriala tudengi kodulehel.

Õpime:

- Kuidas XML'i Flashis ära kasutada
- Parameetritega funktsioonid praktikas
- ActionScript MovieClip-sümboli sees

Näidetest fail menu.fla, menu.xml

1. Loome esmalt menüüfaili menu.xml:

```
<?xml version="1.0" encoding="iso-8859-1"?>
      <lingikogu>
            <link nimi="Esilehele" url="index.html"/>
            <link nimi="Foto" url="#">
                  <alamLink nimi="Pildigalerii" url="pildid.html"/>
                  <alamLink nimi="Minu kaamera" url="a95.jpg"/>
            </link>
```

```
        <link nimi="Programmeerimine" url="#">
              <alamLink nimi="XML" url="xml/index.html"/>
              <alamLink nimi="Flash" url="flash/index.html"/>
        </link>
        <link nimi="3D disain" url="#">
              <alamLink nimi="3DCanvas" url="3d/3dcanvas.html"/>
              <alamLink nimi="POV-Ray" url="3d/povray.html"/>
        </link>
        <link nimi="Ülikool" url="http://www.tlu.ee"/>
</lingikogu>
```

Selle XML-dokumendi juurelemendiks on element "lingikogu". Kõik selle elemendid sees on selle alamelementideks. Tähele tasub panna, et element "link" võib olla ise menüüpunktiks või omada alamlinke (sellisel juhul on lingi URLina märgitud "#", mis ei vii hiirega klõpsates kuhugi). Linkimisel piiranguid pole, võib linkida nii samas kaustas või alamkaustas olevatele failidele või sootuks välislingile. Menüüpunkti alamlingid on tähistatud elemendina "alamLink". XML ei sea piiranguid elementide või atribuutide nimede kasutamises, oluline on hiljem õigesti viidata.

XML-faili lugemine ActionScripti abil

Peamine objekt XML-failiga manipuleerimiseks on Flashis spetsiaalne XML-objekt:

```
var lingidXML = new XML();
```

See lause loob uue XML-objekti ning omab paljusid meetodeid, seadistusi ja sündmusi, peatume neist vajalikel tabelis 7.

Tabel 7 – XMLi meetodid [xmlMenu]

`lingidXML.ignoreWhite = true;`	Muudab elemendid, mille ainsaks väärtuseks on tühik: `<element> </element>` kujule: `<element/>`
`lingidXML.load("links.xml");`	Laeb failist XML-struktuuri.
`lingidXML.onLoad = checkLoading;`	See sündmus kutsutakse välja XML faili laadimise ajal. Alles pärast faili laadimise lõppu saab objekti siseneda. Seda selleks, edasi tegutsetaks alles siis, kui

	kogu XML-fail on laetud, muidu oleks näha ainult tühi XML-fail.

`function checkLoading(success){`
`  if (success == true){`
`  /*XML-töötlus siia*/`
`  }`
`}`
	See funktsioon sisaldab tingimust: kui `checkLoading` on tõene, siis alusta XML-failist saadud info töötlemist. Kasulik on laadida kõik vajalikud XML-failid esimesel kaadril.

`if(success == true){`
`  gotoAndPlay(2);`
`}`
	Kui andmed on käes, hüppame teisele kaadrile.

Kuidas nüüd aga XML-objektis olevatele andmetele ligi pääseda? Tähele tasub panna, et enam ei tegele me XML-failidega vaid Flashi loodud XML-objektiga.
Pääsemaks ligi atribuudi väärtusele elemendis:

```
<link nimi="Ülikool" url="http://www.tlu.ee"/>
```

```
trace (aNode.attributes.nimi);
```

Väljastakse: "Ülikool"

Meie näiteks on kogu info toodud elementide atribuutidena, kui aga peaks väljastama atribuudi enda väärtust:

```
<koduloom liik="kass">Miisu</koduloom>
```

```
trace (aNode.nodeValue);
```

Annab väärtuseks "Miisu".

2. Järgnevalt vajame Flashi faili. Loome pikliku paneeli, muutes selleks vastavalt lava suurust.
3. Loome MovieClip-tüüpi sümboli, millele paneme nimeks `lingiKast`. Sümbol peaks sisaldama ühe dünaamilise tekstikasti, mille nimeks `lingiTekst`.
4. Lohistame vastloodud sümboli `lingiKast` stseenile, kuid mitte lavale, vaid sellest eemale ning paneme instantsile nimeks `baasLink`. See on vajalik sellepärast, et hiljem hakkame ActionScripti abil sellest uusi sümboleid dubleerima, kuid me ei soovi, et baassümbol ise nähtavale kohale segama jääks.

5. Kuigi ka vastloodud MovieClip-sümbol `lingiKast` hakkab sisaldama ActionScripti koodi, kirjutame esmalt koodi stseenile. Nagu tavaks saanud, uuele kihile.

6. Esmalt defineerime paar muutujat, algväärtustades dubleeritavate MovieClip-sümbolite positsiooni ning duplikaatideloenduri:

```
var yPos:Number = 20;
var dubLoendur:Number = 1;
var aLink:String;
```

7. Kirjutame ära ka juba eelnevalt selgitatud XML-faili laadimise:

```
var linksXML = new XML();
linksXML.ignoreWhite = true;
linksXML.load("menu.xml");
linksXML.onLoad = checkLoading;
```

8. Järgnevalt asume juba tõtakalt selle kallale, mis peaks juhtuma siis, kui XML-fail on edukalt laetud ning paneme kirja, mida saadud infoga peale peaks hakkama.

```
function checkLoading(success) {
      if (success == true) {
            var juurElement = linksXML.firstChild;
            var kokku = juurElement.childNodes.length;
```

Lugemist alustatakse esimesest elemendist (`juurElement`) ning loetakse kokku alamelemendite arv. Seejärel hakatakse tegutsema juba iga elemendiga üksikult. *for*-tsüklid saadavad parameetritena tulevase objekti nime ning sisu (muutuja `tnLink`, mis ei kujuta endast muud kui vastavat rida XML-koodist) allpool kirjeldatud funktsioonile `looLink`.

```
            var tLink = juurElement.firstChild;
            for (i=0;i<kokku;i++) {
                  looLink("tLink" + i, tLink);
                  var lasteArv = tLink.childNodes.length;
                  var tnLink = tLink.firstChild;
                  for (j=0;j<lapsteArv; j++){
                        looLink("tnLink" + j + ""+ i, tnLink);
                        tnLink = tnLink.nextSibling;
                  }
```

```
                tLink = tLink.nextSibling;
            }
            gotoAndStop(2);
        }
}
```

Funktsioon `createLink` dubleerib igast `link`-elemendist baasLink-sarnase MovieClip-sümboli, mis saab nimeks `uusObj` väärtuse (näiteks `tnlink13`, sellise nime saab link Pov-Ray'le). Funktsiooni `eval()` ülesandeks on Flashis viidata dünaamiliselt loodud objektile või MovieClip-sümbolile. [AS2LR, lk. 55-56]
Iga järgnev MovieClip-sümbol liigutatakse 30 pikslit allapoole.

```
function looLink(uusObj, aNode){
    duplicateMovieClip(_root.baasLink, uusObj, dubLoendur++);
    var tcl = eval(uusObj);
    tcl.aLink = aNode.attributes.url;
    yPos +=30
    if (aNode.nodeName == "link"){
        setName(tcl, aNode.attributes.nimi, 1);
        tcl._x = 50;
    }else{
        setName(tcl, aNode.attributes.nimi, 0.95);
        tcl._x = 60;
    }
    tcl._y = yPos;
}

gotoAndStop(1);
```

Funktsioon `setName()` seab objektidele parameetriteks kõrguse, laiuse (need arvutatakse koefitsendi järgi, mis tuleb eelnevast *if*-tingimusest) ning lingi nime.

```
function setName(obj, nimi, suurus){
    obj._height = obj._height*suurus;
    obj._width = obj._width*suurus;
    obj.TextLink.lingiNimi.text = nimi;
    obj.embedFonts = true;
}
```

9. Põhimõtteliselt meie rakendus juba töötabki, veel on vaid tarvis panna hiireklõps lingil ka soovitud aadressile minema. Silmailu mõttes lisame MovieClip-sümbolile veel ka lahenduse, mis näitaks hiirekursoriga osutatava lingi kõrval väikest noolekest.

   Kui varem oli sümbolil `lingiKast` vaid üks kiht, siis nüüd teeme neid kolm tükki juurde lisaks.

10. Ühe vastloodud kihi teisele kaadrile paigutame graafilise sümboli *nooleke*, mille võib oma maitse järgi kujundada. Lohistame selle vasakule, sümboli *lingiKast* ette.

11. Järgmised kaks kihti on ActionScipti tarvis. Esimese kihi esimesele kaadrile kirjutame skripti, mille mõjuala ulatuks ka teisele kaadrile:

```
onRelease = function () {
      getURL(aLink, "_top");
}
```

   Nagu koodist lugeda võib, liigutakse hiirenupu vabastamisel etteantud URL-ile. Funktsiooniga `getURL()` puutusime esmakordselt kokku juba eelmises harjutusülesandes, kus toimus andmete välja saatmine Flash rakendusest.

12. Teise ActionScipti kihi eesmärgiks on näidata noolekest lingi ees, millel hiirega peatume. Ilma selle funktsioonita piisaks ka teistel selle MovieClip-sümboli kihtidel vaid ühest kaadrist.

   1. kaadri kood:

```
this.onRollOver = function () {
      gotoAndPlay(2);
}
stop();
```

   2. kaadri kood:

```
this.onRollOut = function () {
      gotoAndPlay(1);
}
this.onDragOut = function() {
      gotoAndPlay(1);
}
stop();
```

Ilma sündmuseohjurita `onDragOut` jääksid hooletul hiire kasutamisel, kus kasutaja klõpsab lingil ning sellest allhoitud hiireklahviga üle sõidab, nooled ette ja need ei kaoks sealt enam ära.

Olemegi loonud toimiva, XML-i kasutava lahenduse. Sarnaselt, olles eelnevalt uurinud kasutatava XML-faili struktuuri, saab luua mitmeid erinevaid rakendusi, sh uudisteloendit või muid andmeid kuvava rakenduse.

Sõnaseletused

ASP – (*Active Server Pages*) Microsofti tehnoloogia loomaks dünaamilisi veebilehti. [Wikipedia[1]]

banner – ribareklaam; ristkülikukujuline, enamasti staatilise pildina realiseeritud reklaam veebilehel; toimib lingina reklaamiandja juurde. [Sõnastik]

CGI – (*Common Gateway Interface*) standardprotokoll välise tarkvara infoserveriga, milleks on enamasti veebiserver, ühendamiseks. [Wikipedia[2]]

ColdFusion – MacroMedia poolt arendatav tarkvarapakett veebisaitide tegemiseks ning nende serveerimiseks kasutajale. [Vikipeedia]

Flash Player – Macromedia poolt arendatav ning levitatav Flashis loodud multimeediumi ja rakenduste esitlusprogramm. [Wikipedia[3]]

HTML – (*HyperText Markup Language*) hüperteksti märgistuskeel, veebis kasutatav tekstivorming. [Sõnastik]

interpoleerimine – (*tweening*) tehnika, mis etteantud võtmekaadrite vahele genereerib kasutaja poolt etteantud arvu kaadreid, millede jooksul toimub järkjärguline üleminek esimeselt võtmekaadrilt teisele. Võttes arvesse kahe võtmekaadri erinevusi, kalkuleerib arvuti iga kaadri jaoks proportsionaalsed muudatused objektide asukoha, suuruse, värvuse, kuju jms jaoks. Mida rohkem kaadreid lastakse genereerida, seda väiksemad tulevad järjestikuste kaadrite vahelised erinevused ning seda sujuvam tuleb liikumine. [Rinde]

PHP – veebiprogrammeerimiskeel dünaamiliste veebilehtede loomiseks.

plugin – lisandprogramm [Sõnastik] veebilehitseja või muu programmi juurde, andmaks sellele lisavõimalusi (nt jooksutada Flash-filme).

URL – (*Uniform Resource Locator*) internetiaadress. [Vallaste]

Kokkuvõte

Macromedia Flash tarkvarapaketi näol on tegemist ühe levinuma interaktiivse multimeediumi loomevahendiga, mida kasutatakse laialdaselt mitmes tegevusvaldkonnas.

Käesoleva bakalaurusetöö eesmärgiks oli luua Flash programmeerimiskeele ActionScript harjutusülesannete kogumik koos seletavate materjalidega. Sellele eelneb autori analüüs saadaolevate materjalide kohta ning harjutusülesannete valiku põhjendus.

Töö koosneb kolmest osast: olemasolevate materjalide analüüs koos harjutusülesannete püstitusega, ActionScripti tutvustus ning harjutusülesanded.

Antud valdkonnas valminud tööd ei saa kindlasti pidada lõplikuks, seda edasiarendades on võimalik minna spetsiifilisemaks ning demonstreerida ActionScript-keele neid omadusi, mis sellest tööst välja jäid.

Materjal tehakse vabalt veebis kättesaadavaks.

Summary

Current Bachelor thesis gives an overview of the ActionScript programming language. In order to accomplish this, author analyzes studying material avalaible both on the Web and as in books. Based on the most commonly covered topics, author assembles an ActionScript tutorial.

The thesis is divided into three parts. The first part gives an overview to the materials currently available and raises the theory of tutorials that should be covered. The second part (Chapters 2-8) concentrates on the ActionScrips and gives a reader the possibility to get acquianted with the advantages and disadvantages of Macromedia Flash. The third part (Chapter 9) is wholly devoted to the tutorials.

The material will be available on the Web.

Kasutatud allikad

[AS2LR] Macromedia, Inc.; (2005); ActionScript 2.0 Language Reference; San Fransisco, USA; Macromedia, Inc.

[Bible] Reinhardt, R.; Lott, J.; (2004); Flash MX 2004 ActionScipt Bible; Indianapolis, USA; Wiley Publishing; ISBN 0764543547

[CLR] Macromedia, Inc.; (2005); Flash 8 Components Language Reference; San Fransisco, USA; Macromedia, Inc.

[Dummies] Leete, G.; Finkelstein, E.; (2002); Macromedia Flash MX for Dummies; Indianapolis, USA; Wiley Publishing; ISBN 0764508954

[GSF] Macromedia, Inc.; (2005); Getting Started with Flash 8; San Francisco, USA; Macromedia Inc.

[Haamer] Liivi Haamer; (2001); Laiendatav märgendikeel XML – proseminaritöö; Tallinn; Tallinna Pedagoogikaülikool

[Kirupa[1]] kirupa.com - _root, _parent and this; URL: http://www.kirupa.com/developer/actionscript/tricks/root_parent_this.htm (29.04.2006)

[Kirupa[2]] kirupa.com - Listeners and ASBroadcaster; URL: http://www.kirupa.com/developer/actionscript/asbroadcaster.htm (29.04.2006)

[Kirupa[3]] kirupa.com - The drawing API; http://www.kirupa.com/developer/actionscript/tricks/drawing_api.htm (29.04.2006)

[Kirupa[4]] kirupa.com - with() statement; URL: http://www.kirupa.com/developer/actionscript/with.htm (29.04.2006)

[Kirupa[5]] kirupa.com – Flash Forms and Database Integration; URL: http://www.kirupa.com/developer/actionscript/forms_database.htm (29.04.2006)

[Kusmin] Kusmin, M; Programmeerimine II; Teema 2; URL: http://www.kusmin.com/c/2/2.html (29.04.2006)

[LAS2] Macromedia, Inc.; (2005); Learning ActionScript 2.0 in Flash; San Fransisco, USA; Macromedia Inc.

[LiveDocs] Macromedia Flash LiveDocs; URL: http://livedocs.macromedia.com/flash/mx2004/main_7_2/wwhelp/wwhimpl/

js/html/wwhelp.htm?href=00000780.html (29.04.2006)

[PHP] PHP.ee – Objektorienteeritud programmeerimine PHP-s; URL: http://www.php.ee/4732 (29.04.2006)

[Rinde] Rinde, A.; Multimeedium; Tallinn; Tallinna Ülikooli informaatika osakond

[Sõnastik] Hanson, V.; Tavast, H.; (2001); Arvutikasutaja sõnastik; Tallinn; Kirjastus Ilo

[UF] Macromedia, Inc.; (2005); Using Flash; San Fransisco, USA; Macromedia, Inc.

[Vallaste] Vallaste, H.; e-teatmik: IT ja sidetehnika seletav sõnaraamat; URL: http://www.vallaste.ee (29.04.2006)

[Vikipeedia] ColdFusion - Vikipeedia; URL: http://et.wikipedia.org/wiki/ColdFusion (29.04.2006)

[Wikipedia[1]] Active Server Pages – Wikipedia, the free encyclopedia; URL: http://en.wikipedia.org/wiki/Active_Server_Pages (29.04.2006)

[Wikipedia[2]] Common Gateway Interface – Wikipedia, the free encyclopedia; URL: http://en.wikipedia.org/wiki/Common_Gateway_Interface (29.04.2006)

[Wikipedia[3]] Macromedia Flash Player - Wikipedia, the free encyclopedia; URL: http://en.wikipedia.org/wiki/Flash_player (29.04.2006)

[Wikipedia[4]] ActionScript - Wikipedia, the free encyclopedia; URL: http://en.wikipedia.org/wiki/Actionscript (29.04.2006)

[VTHC] Virtual Training Help Center; URL: http://iit.bloomu.edu/vthc/Flash/Stage.html (29.04.2006)

[xmlMenu] ActionScript.org Macromedia Flash Resources and Tutorials –XML menu; URL: http://www.actionscript.org/tutorials/beginner/xml_menu/index.shtml (29.04.2006)