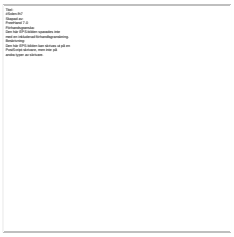


Datavetenskap

Jimmy Alfredsson och Niclas Börne

Shockwave och Flash,

en jämförelsestudie

Examensarbete, C-nivå

2003:22

Shockwave och Flash,

en jämförelsestudie

Jimmy Alfredsson och Niclas Börne

Denna rapport är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i denna rapport, vilket inte är vårt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Jimmy Alfredsson

Niclas Börne

Godkänd, 2003-06-04

Handledare: Torbjörn Andersson

Examinator: Stefan Lindskog

Sammanfattning

I vårt examensarbete har vi gjort en studie över vilka likheter och skillnader som finns mellan de båda verktygen Shockwave och Flash, som är vida använda för att visa multimedia på webben. Det finns många webbsidor där dessa verktyg används för att besökaren ska få en förhöjd upplevelse. Många av dem kan till synes likna varandra mycket, vilket kan medföra att det till det yttre inte verkar finnas någon skillnad mellan verktygen och hur man skapar multimediaproduktioner i dem. Det har gjort att det är ganska intressant att försöka ta reda på vad som egentligen skiljer dem åt.

Resultatet som vår uppdragsgivare, Hög-Data, förväntas få ut av vår studie är att få en väl underbyggd rekommendation av vilket format som passar deras syfte bäst och deras främsta anledning uttrycks bäst med deras egna ord: Hög-Data, i likhet med de flesta andra produktorienterade IT-företag som vill hålla sig i frontlinjen med ny teknologi, söker ständigt efter nya sätt att visa en optimal bild av sig själv till potentiella kunder.

I vår studie gör vi en allmän beskrivning av Shockwave och Flash för att ge läsaren en bild av vad för slags verktyg de är och hur utvecklingen går till i deras respektive utvecklingsmiljö. Vi har också gjort en enklare spelkonstruktion i syfte att undersöka utvecklingsmiljöerna och för att få den praktiska erfarenhet vi behövde. Resultatet som vi kom fram till är att Flash är det som är bäst för multimediaproduktioner för webben.

Shockwave and Flash, a comparison study

Abstract

In our Bachelor project we have made a study in which we have looked upon what similarities and differences there are between the tools Shockwave and Flash, which are widely used for presenting multimedia on the World Wide Web. There are many web pages where these tools are used to enhance the experience for the user. Many of those pages may be similar in their appearance and therefore one might think that there is no difference between them and how to produce multimedia with them. This has made it interesting to try and find out the differences between them.

The result our assigner, Hög-Data, expects from our study is a well-supported recommendation of what tool that fits their purpose best and as they say with their own words: Hög-Data, similar to most other product oriented IT-companies that wants to be in the frontline with new technology, is constantly seeking for new ways to show an optimal representation of themselves to potential customers.

In our study we make a general description of Shockwave and Flash, to give the reader an idea of what kind of tools they are and how to develop in their development-environment. We have also made a simple game in purpose of investigating each development-environment and to acquire experience. We came to the result that Flash is best for multimedia productions for the web.

Innehållsförteckning

Figurförteckning

Tabellförteckning

1 Inledning

Fler och fler använder idag mjukvarorna Shockwave och Flash för att göra animeringar och för att utöka interaktiviteten i presentationer på webbsidor. De främsta skälen till att ge besökarna en större möjlighet att själva aktivt påverka vad de kan få ut av innehållet, är dels att användaren enklare kan navigera genom innehållet och dels att ge besökarna ett såpass positivt intryck att de gärna återvänder till webbsidan.

Vår uppdragsgivare, Hög-Data, som är ett företag som bedriver försäljning av datorer, tillbehör och tjänster, vill ha just Shockwave eller Flash på deras hemsida, därför att de funnits i några år nu och inte längre kan ses bara som en modefluga, som snart kan förkastas av användarna.

Uppdragen var flera och bestod av att lära oss utvecklingsverktygen för respektive mjukvara och göra en experimentell spelkonstruktion med dem, samt att göra en allmän beskrivning av deras utvecklingsprogram. Vidare skulle vi göra en jämförelse av de båda och slutligen förfina någon av de två spelkonstruktionerna.

1.1 Bakgrund

I slutet av 60-talet startades det globala nätverket ARPA-NET på initiativ av USA's försvarsdepartement. Namnet kommer från dess finansiär Defence Advanced Research Projects Agency (DARPA) och var början på det som idag kallas Internet. Syftet var att skapa en bombsäker decentraliserad datakommunikation. Under 70- och stora delar av 80-talet användes nätverket enbart för att koppla samman olika universitetslaboratorier och militära forskningsanläggningar i USA. Men för att Internet skulle nå den popularitet som den nått idag så behövde de utspridda informationsresurserna, som kunde nås på många olika sätt och med många olika protokoll, förenas till en lättanvänd användaranslutning som kallas World Wide Web (WWW)[1][2].

WWW eller webben som den också kallas här i Sverige, utvecklades 1990 av Tim Berners Lee som ett förslag till ett Internetbaserat hypertextsystem, i vilken ett valfritt ord kan kopplas med en länk som hänvisar till annan information. Den metod som binder samman allt till ett nät av dokument sammanknutna av länkar bygger på Ted Nelsons hypertext-idé från 1965. Berners Lee utvecklade HyperText Markup Language (HTML) för att demonstrera hypertext. HTML kan t.ex. berätta vilken del av ett textdokument som är rubrik, framhävning, citat mm.

Ett HTML-dokument tolkas av användarens webbläsare och presenteras på dennes dataskärm[2].

I stort sett all utveckling av webben, fram till 1993, skedde vid kärnforskningslaboratoriet CERN, i Geneve, där Tim Berners Lee jobbade. Då startade Marc Andreessen, som studerade vid The National Center for Supercomputing Applications (NCSA), ett projekt med att ta fram en webbläsare. Det här var en av de första webbläsare som kunde hantera flertalet av Internet tjänster som t.ex. news, www, email osv. Presentationen var grafisk och byggde på peka och klicka principen med hjälp av musstyrning. Tidigare kunde webbläsarna visa monochrome text på en svart skärm och det var inte alls lika lätt att använda webben som idag. Användaren var tvungen att koppla upp sig mot servern i CERN via protokollet Telnet, för att då enbart mötas av en massa text. Genom webbläsaren Mosaic revolutionerades webben och många utanför den akademiska världen fick upp ögonen för den[3].

Privata intressen gjorde att många nya egenskaper utvecklades för nästa version, Mosaic Netscape. Bland annat utökades layoutkontrollen (CENTER-taggen) som idag känns självklar och möjligheten att öppna multipla Transmission Control Protocol- (TCP) uppkopplingar var en mycket viktig egenskap för att reducera tiderna för nedladdningen av webbsidor mot tidigare då webbläsaren bara använt sig av en TCP-uppkoppling. TCP är ett databärande protokoll som i sin tur används av ett protokoll som heter HyperText Transfer Protocol (HTTP). Det som förbättrades i webbläsaren var att den kunde öppna flera HTTP-sessioner för att hämta hem innehållet från en webbsida (HTML-sida). Företaget Mosaic Communication Corporation ville ge något tillbaka till de som hjälpt till med utvecklingen, så de bestämde sig för att dela ut webbläsaren gratis till alla lärare, studenter och forskare över hela världen. Det här visade sig vara en viktig faktor till den enorma succé som Netscape gjorde[3].

På sommaren 1995 kom företagen Netscape och Macromedia överens om att integrera Macromedias multimediamjukvara, Shockwave, med webbläsaren Netscape Navigator. Det här kom att ändra karaktären på webben från statiska sidor till interaktiv multimedia. Dessutom, genom integrationen, utan att behöva starta något annat program än webbläsaren[4]. Ett år senare kom ett annat företag, Microsoft, överens med Macromedia att integrera Shockwave med Internet Explorer 3.0[5].

Shockwave är inget program i sig självt, utan snarare en teknik för att utveckla och presentera interaktiv multimedia, främst för webben. Utvecklingen sker genom att integrera och styra många olika multimediafiltyper med programmet Director Shockwave Studio, vanligen kallat enbart Director. Programmet innehåller även verktyg för att rita enklare figurer

och för att göra animationer. Vidare innehåller det ett inbyggt programmeringsverktyg som heter Lingo. För att kunna se presentationer skapade med Director i webbläsaren så måste webbläsaren integreras med ett plug-in som heter ”Shockwave player”. Ett plug-in är ett litet program som är gjort för att kopplas samman med, och styras av, ett annat program. Eftersom Shockwaveformatet begränsar filstorleken på filerna som skapas, mer än många andra filformat för multimedia på Internet, så är det populärt att använda för att t.ex. skapa interaktiva spel för webbsidor[6].

Flash är ett multimediaprogram som utvecklats speciellt för webben (men kan även användas på andra områden). Det är idag ett av de mest populära verktygen för att skapa webbsidor rikt med grafik- och multimediaupplevelser. För själva skapandet används programmet Flash. För att kunna visa upp det som skapats i en webbläsare måste webbläsaren, precis som med Shockwave, integreras med ett plug-in. Det plug-in som behövs kallas för ”Flash player”. Flash arbetar med grafik i ett filformat som begränsar filstorleken mer än många andra filformat för grafik på Internet. Flash kan även hantera de här filformaten och utöver statisk grafik kan det hantera och skapa animationer och ljud och med det inbyggda programmeringsverktyget programmeras den interaktivitet som önskas[7].

1.2 Syfte och Mål

Hög-Data, i likhet med de flesta andra produktorienterade IT-företag, vill hålla sig i frontlinjen med ny teknologi. Företaget söker ständigt efter nya sätt att visa en optimal bild av sig själv till potentiella kunder och eftersom ansiktet utåt för ett IT-företag i mångt och mycket är dess hemsida så vill de finna ett lämpligt verktyg för att vidareutveckla den. Av den anledningen vill de ha en jämförelse mellan de båda multimediabärarna Shockwave och Flash, som utreder skillnaderna. Jämförelsen ska vara ett beslutsunderlag. De nämnde inget om vad de specifikt skulle använda programmen till och det gjorde att vi siktade in oss på att göra jämförelsen mer generell.

De hade även under en längre tid funderat på att göra ett spel till deras hemsida. Därför föll det sig naturligt att i utredningsarbetet konstruera ett spel för att kunna göra iakttagelser i samband med användning av verktygen. Att konstruera ett spel som grund för projektet är bra eftersom det kan innehålla många sorters svårigheter som kan uppkomma vid konstruktion av en fräck, intresseväckande och användarvänlig hemsida och på samma gång kanske få fram ett spel som Hög-Data kan använda sig av. Ett spel kan innehålla allt från vanlig text till möjligheten för användaren att aktivt vara med och påverka. Projektmålen visas i Tabell 1:1.

	Beskrivning
MÅL-1	Lära oss de båda verktygen för att kunna skapa enklare applikationer.
MÅL-2	Konstruera ett enklare spel med båda verktygen.
MÅL-3	Få en uppfattning om vad verktygen innehåller och kan åstadkomma
MÅL-4	Komma fram till en rekommendation åt Hög-Data.
MÅL-5	Färdigställa spelet vi startat utveckla under konstruktionsarbetet för att nå mål två.

Tabell 1:1: Projektmål

1.3 Avgränsningar

Vi har bestämt oss för att inte använda några andra utvecklingsprogram än Flash och Director i den experimentella konstruktionen. Anledningen till det här är att vi vill lära oss använda de verktyg programmen innehåller för att sedan jämföra likheter/skillnader mellan dem. Vi ska försöka så långt det går att inte importera färdiga filer gjorda i andra program, så att verktygen får visa vad de går för.

1.4 Genomförande/Arbetspaket

Arbetspaketen i Tabell 1:2 beskriver hur vi tänkt uppnå målen vi satt upp i Tabell 1:1.

	Beskrivning
AP-1	I inledningen läsa in oss på ämnet och lära oss grunderna för hur verktygen används.
AP-2	Först tänka ut en spelidé och sedan konstruera den med de båda verktygen på ett sätt så att de liknar varandra så mycket som möjligt.
AP-3	Samla in data om verktygen.
AP-4	Göra en jämförelse av de båda verktygen grundade på arbetet i AP 1-3.
AP-5	Förändra, förbättra och utöka funktionaliteten i spelet med det rekommenderade verktyget.

Tabell 1:2: Genomförande för att uppnå målen

1.5 Disposition

I kapitel 1 (Inledning) beskrivs kort Shockwave och Flashs bakgrund och historia. Samma kapitel beskriver även syftet med examensarbetet, vilka mål och begränsningar examensarbetet har, och hur dessa mål är tänkta att uppnås.

I kapitel 2 (Allmänt om Shockwave och Flash) förklaras allt som läsaren behöver för att kunna förstå beskrivningen av den experimentella spelkonstruktionen. Här beskrivs de gemensamma begrepp som programmen har och de begrepp som har liknande betydelse i programmen, men har olika namn. De senare begreppen ”mappas” även här till ett gemensamt namn för att förenkla för läsaren. Efter det följer två avsnitt om Shockwave och Flash där respektive utvecklingsmiljö och hur man utvecklar beskrivs.

I kapitel 3 (Experimentell spelkonstruktion) förklaras först varför vi gjorde ett spel. Därefter beskrivs den spelidé vi hade och till sist detaljerat hur vi konstruerade spelet.

I kapitel 4 (Jämförelse och rekommendation) beskrivs de skillnader och likheter som finns mellan Shockwave och Flash, mestadels baserat våra egna erfarenheter. Först skriver vi vad vi baserade vår jämförelse på. Efter det beskrivs de likheter och skillnader vi hittade i programmens verktyg. Vidare beskrivs de jämförelser vi gjorde riktade till en tjänsteförmedlare och utvecklares kostnader. Sist sammanfattar vi alla jämförelser och ger våra slutgiltiga åsikter om vilket program vi tycker lämpar sig bäst.

I kapitel 5 (Sammanfattning) sammanfattar vi hela vårt examensarbete.

2 Allmänt om Shockwave och Flash

Kapitlet är en redovisning av en del av arbetet för att uppnå MÅL-1 och MÅL-3 och AP-1 och AP-3, beskrivet i Tabell 1:1 och Tabell 1:2.

Multimedia är idag ett vida spritt begrepp. Ordet är en sammanslagning av orden "multi", som betyder *många* och "media", som betyder *kanaler för informationsspridning*. De viktigaste beståndsdelarna som brukar räknas upp är: texter, rubriker, bilder, animationer, fotografier, filmsnuttar, interaktivitet och ljud. Dessa begrepp har de flesta stött på någon gång och vet dess betydelse. Det som är intressant är hur dessa beståndsdelar slås ihop till en integrerande enhetlig produktion och hur de skapas var för sig[14]. I det här kapitlet berättar vi hur det går till i Flash och Director, genom att beskriva deras utvecklingsmiljöer. Men innan vi börjar med den beskrivningen krävs det att några väsentliga och gemensamma begrepp förklaras.

2.1 Gemensamma begrepp för Flash och Shockwave

Webbutvecklare världen över strävar oftast efter att hålla webbsidor så kompakta som möjligt eftersom den genomsnittliga webbsurfaren inte vill vänta mer än högst 10 sekunder på att en webbsida ska ha hunnit laddats klart och visats[12]. Med kompakt menas att de objekt som finns på en webbsida, exempelvis bilder, ljud, animationer osv. ska ha en så liten filstorlek som möjligt, men samtidigt vara av hög kvalité.

2.1.1 Bitmappgrafik

En webbsidas vanligaste komponenter/objekt är grafik och text. Den grafik som förekommer oftast på webbsidor är bilder i filformaten ”jpeg” och ”gif”[13]. Dessa två filformat för grafik kallas för bitmappgrafik. Det kallas för bitmappgrafik eftersom grafiken består av ett stort antal punkter sk "bitmappar". Bitmappar är ett generaliserat namn av dessa punkter. En punkt i bilden är egentligen en ”pixel”, vilket är en bitmapp med ett färgvärde. Varje pixel har tilldelats en plats och ett färgvärde i bilden. En bitmappbilds filstorlek bestäms av antalet pixlar bilden innehåller, hur den komprimeras vilket är olika beroende på filformat och hur många olika sorters färger som finns i den. En fördel med bitmappbilder är att de kan vara väldigt detalj- och färgrika. En nackdel är att ju mer detalj- och färgrika de är, desto större

filstorlek har de. Denna nackdel är anledning till att grafiken som skapas i Flash huvudsakligen är baserad på vektorgrafik.

2.1.2 Vektorgrafik

Vektorgrafik är ett matematiskt sätt att visa grafik på skärmen. Det är uppbyggt så att det är en formel som räknar ut var en linje börjar och var den slutar, samt om det eventuellt är kurvor på linjen. Det här innebär att det som ritats kan förstoras och förminskas hur som helst utan att bilden tappar kvalitet. Vektorgrafik byggs alltså upp genom att ge datorn instruktioner om hur bilden ska se ut. Dessa instruktioner tar mindre fysiskt utrymme på en dator än de ”bitmappar” som en bitmappbild byggs upp av. Det här är anledningen till att vektorgrafik är mer kompakt än bitmappgrafik[17].

2.1.3 Animation

En bild är statisk och har ingen rörelse i sig. En animation är en serie av bilder som ses snabbt efter varandra för att ge illusionen av rörelse. Bilder och animationer är passiva objekt i begreppet multimedia. Med det menas att användaren tittar på dem som en film och kan inte styra vad som visas.

Det finns olika metoder att skapa en animation i Director och Flash, men det grundläggande tillvägagångssättet som utvecklaren kan skapa en animation med i både Director och Flash är att den görs ”frame-by-frame”, förklarat i avsnitt 2.2. Annan animation går att göra bild för bild m.h.a inbyggda verktyg som finns i bådas utvecklingsmiljöer. Mer om det här i avsnitt 2.3 (Shockwave) och 2.4 (Flash).

2.1.4 Interaktivitet

När användaren får vara med och bestämma genom att exempelvis trycka på en knapp och det genererar en händelse, exempelvis ett ljud som spelas eller en bild som visas, så kallas det för interaktivitet. Att införa interaktivitet i det som skapas med Director och Flash kan göras med färdiga verktyg, men för att skapa mer specifik interaktivitet så kan utvecklaren programmera i ett scriptspråk. Skillnaden mellan ett scriptspråk som Lingo för Shockwave och Actionscript för Flash och programspråk som t.ex. C++ eller Java, är att scriptspråken inte är kompilerade i förväg utan tolkas i realtid under körningen. Scriptspråket Lingo för Director och Actionscript för Flash, skiljer sig på många punkter och scriptspråken förklaras var för sig när vi senare i kapitlet beskriver programmen.

2.2 Ordlista

För att det inte ska bli för rörigt med alla nya namn och begrepp så har de begrepp som har liknande betydelse i Shockwave och Flash fått ett gemensamt namn. Det här stycket kan ses som en ordlista som du som läsare kan hoppa tillbaka till under läsning.

- Film (movie)

Det som skapas i Flash och Director.

- Ramar (frames)

En film är en samling ramar (bildrutor). En inställning som kan göras för en film är visningshastigheten, d.v.s. hur många ramar per sekund eller frames per seconds (fps), som den ska visa. I verkligheten så kan fps och verklig visningshastighet vara olika. Om en dator med för låg hårdvarumässig prestanda används så kan det hända att den inte har kapacitet nog för att visa angivet antal fps.

- Nyckelram (keyframe)

En ram som indikerar en händelse. Kanske att ett objekt ska börja visas, ett script startar osv.

- Frame-by-frame animation

Innebär att nyckelram på nyckelram läggs efter varandra. Nu kan utvecklaren ändra position, form, färg, storlek mm för objektet i varje nyckelram för att ge en illusion av rörelse.

- Element

Kallar vi de beståndsdelar som finns i begreppet multimedia (bilder, ljud, animationer osv).

- Objekt

När elementen används i filmen kallar vi dem för objekt.

- Scenen (stage)

Själva arbetsytan, här placeras de objekt som används i filmen. Scenen representerar vad som kommer att vara synligt i filmen.

- Tidslinjen (timeline)

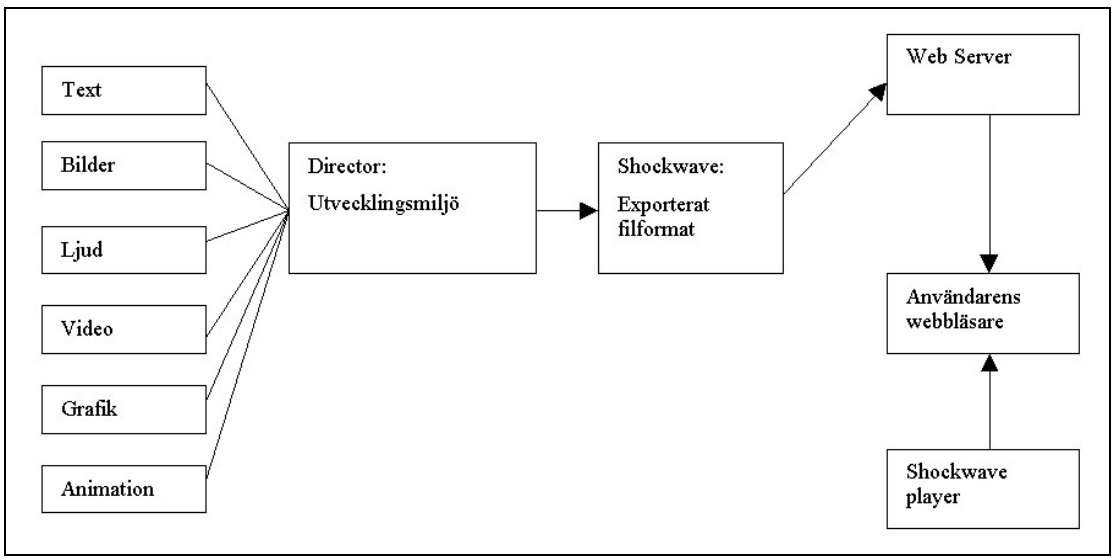
Som namnet antyder representerar tidslinjen översiktligt vad som händer i din film under tidens gång. Exempelvis vilka objekt som ska visas på en ram.

2.3 Shockwave

Som det står i inledningen är Shockwave inget program, utan mer en teknik för att visa upp multimediaproduktioner skapade i Macromedia Director för webben. Programmet Director är en vida använd utvecklingsmiljö för att skapa interaktiv multimedia, utbildningsapplikationer,

e-handelspresentationer, underhållande spel och filmer etc.[11]. Styrkan ligger i att utvecklaren på ett enkelt sätt kan importera och kombinera i stort sett alla tänkbara medietyper som text, bitmapp-bilder, vektorgrafik, ljud, video, animationer mm, se Figur 2:1. Director kan dessutom styra de här elementen under tiden som de spelas/visas upp, genom att utvecklaren eller användaren kontrollerar egenskaper som storlek, färg, placering på vyn etc. Director tillåter också att utvecklaren programmerar elementen så att de interagerar med varandra och användaren för att filmen ska uppföra sig på ett speciellt sätt när en viss händelse inträffar. Den metafor som används är just den att utvecklaren genom Director fungerar som en regissör som styr ett uppträdande. Förutom att importera färdiga element av olika mediatyper så kan utvecklaren skapa egna element i några av dessa medierna med hjälp av inbyggda verktyg[6].

Vanligtvis placeras filmerna, som kan skapas i olika filformat med Director, på en CD-ROM för att visa upp innehållet på en lokal dator. Men genom att skapa filmerna i filformatet Shockwave kan samma innehåll göras tillgängligt, distribueras, via webben. Det här illustreras i Figur 2:1 av pilarna från "Utvecklingsmiljö" till "Exporterat filformat" och vidare till "Webbservern". Filen/filerna hämtas från webbservern med hjälp av ett plug-in till Webbläsaren som heter Shockwave player”[10] och illustreras av pilarna från "Webbservern" och "Shockwave player" till "Webbläsaren" i Figur 2:1. Det som gör att Shockwave-formatet lämpar sig bra för webben är att filerna komprimeras kraftigt och att viss funktionalitet inte görs tillgänglig, av säkerhetsskäl. Exempelvis så är inte många systemrelaterade kommandon tillgängliga av den enkla anledning att ingen ska kunna skapa filmer som gör att obehöriga kan ta sig in i en användares system, vilket kan ge oönskade effekter[8].



Figur 2:1: Utveckling och distributionsdiagram[6]

2.3.1 Utvecklingsmiljö

I Director går de flesta av fönstren att koppla samman i grupper för att spara plats. Det finns två typer av fönster, dokumentfönster och verktygspaneler. Dokumentfönster används för att skapa eller redigera element och verktygspaneler används för att se på och modifiera vissa egenskaper för det elementet. Följande är en lista med några av de dokumentfönster som finns i Director.

- Stage (scenen)

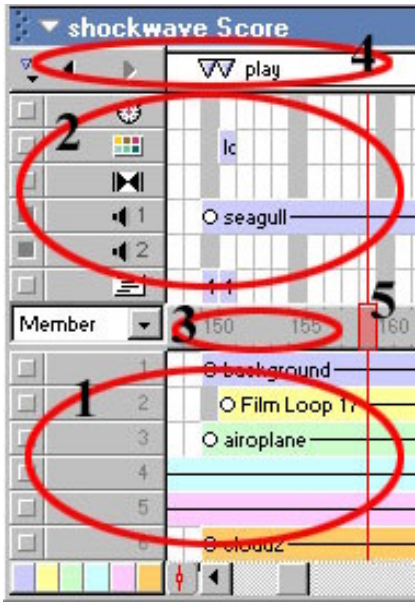
Det är på scenen som händelserna som ska vara synliga i filmen utspelar sig och det är här som utvecklaren skapar eller placerar ut de element som han/hon vill ska vara med i den färdiga filmen. Genom att zooma ut eller in kan utvecklaren välja om han/hon vill se mer eller mindre än den exakta filmytan. Som hjälp, när olika objekt placeras ut, kan scenen visas med ett rutnät med möjlighet för objekt att automatiskt ”snäppas” fast mot rutnätet. Ett objekt kallas i Director för en ”sprite” men vi har valt att använda det mer generella ordet objekt i det här dokumentet. Att ”snäppa” fast ett objekt innebär att objektet dras över scenen mot ett streck i rutnätet så att objektet "hoppar" till strecket när det kommer nära. På det här viset blir alla objekt som fästs mot samma streck placerade på exakt samma x eller y koordinat på scenen. I standardvisning har scenen en kontrollpanel för uppspelning av filmen under utvecklingsarbetet. Kontrollpanelen kan även fås som ett flytande fönster och då med den extra informationen om vilken visningshastighet (fps) som det är meningen att filmen ska visas i och information om filmens verkliga visningshastighet under uppspelning. Se Figur 2:4.

- Cast

Här kan utvecklaren se alla de element som används i filmen antingen i form av en lista eller i form av "thumbnails" (en liten grafisk bild av elementet). Ett element kallas i Director för en ”cast-member” men vi har valt att kalla dem för element i det här dokumentet. Cast kan användas som ett bibliotek eftersom elementen kan organiseras för att vara lättare att hålla reda på. Förutom de element som syns på scenen som objekt så visas också de element som bara används i Score som exempelvis script, typsnitt och övergångar. Elementen kan vara egentillverkade i Director eller importerade. Alla element i det här fönstret behöver inte användas i filmen men för att ett objekt ska kunna finnas med i filmen måste det återfinnas som ett element här. Flera objekt kan skapas från samma element. Se Figur 2:4.

- Score (Figur 2:2)

Objekten placeras i kanaler (1), som finns i ett begränsat antal (tusen stycken), under tidslinjen (3) som är indelad i ramar. På tidslinjen finns ett spelhuvud (5) som visar på vilken ram filmen befinner sig. Förutom kanalerna som gör att utvecklaren kan kontrollera synliga objekt på scenen finns ovanför tidslinjen sex specialeffektskanaler (2) för att kontrollera filmens visningshastighet, färger, ljud (vänster och höger kanal), övergångar och script. Dessutom kan utvecklaren sätta markeringar (4) på ramarna för att enklare kunna referera till dem i ett script. Score är det viktigaste fönstret vid utvecklingsarbetet eftersom det är där utvecklaren kan bestämma *när* olika saker ska hända med objekten på scenen.



Figur 2:2: Score

- Script (Figur 2:3)

Lingo heter det objektorienterade scriptspråket som är inbyggt i Director. I scriptfönstret kan script skapas på fyra olika sätt, vilket beskrivs i avsnitt 2.3.2. Kod kan skrivas direkt för hand eller genom att ta hjälp av den inbyggda ordlistan. Drop-down knapparna för ordlistan syns inringade i Figur 2:3. Där listas nyckelord, funktioner och termer som det finns exempelkod för. Om utvecklaren t.ex. vill lägga till ljud är det bara att välja "sound" i ordlistan. Då skrivs en exempelkod ut i scriptfönstret som även förklarar hur funktionerna ska skrivas färdiga. Ordlistan listar innehållet både i alfabetisk ordning och efter olika kategorier. Därtill finns listning över de tredje-parts “Xtras” som har installerats. Xtras kan beskrivas som hjälpfiler eller plug-ins som Director kan använda sig av för att spela upp Shockwave-filer korrekt. I scriptfönstret finns också hjälp att tillgå för att felsöka scriptkoden.

Figur 2:3: Scriptfönstret

- Paint

Används för att skapa eller förändra bitmappbilder. Flera verktyg ingår som gör att grundläggande geometriska- eller oregelbundna former kan ritas. Verktygen är liknande dem som finns i vanliga bildbehandlingsprogram som exempelvis Microsoft Paint.

- Vector Shape

Används för att på frihand rita grundläggande geometriska och oregelbundna former av typen vektorgrafik. För frihandsritning finns verktyget ritstift.

- Text

Som namnet antyder skriver utvecklaren in text som ska ingå i filmen i det här fönstret. De vanligaste textformateringsalternativen som finns i en vanlig ordbehandlare kan utföras (kursiv, fet, typsnitt etc.). Text som skrivs in i det här fönstret kallas för ”Rich Text”. Förutom att skriva in ny text kan utvecklaren även förändra importerad text som är skapad i en ordbehandlare och sparad i ”Rich Text” formatet. Med den här typen av text behöver utvecklaren inte bekymra sig över om användaren har typsnittet installerat i sin dator, eftersom texten så att säga ”bränns” in i filmen.

- Field

Här skriver utvecklaren in och formaterar text på liknande sätt som i Textfönstret. Text som skrivs in i det här fönstret kallas för ”Field Text”. Skillnaden på ”Field Text” och ”Rich Text” är att i ”Field Text” kan texten när som helst manipuleras, även i den färdiga filmen bl.a. med hjälp av Lingo script. Här får dock utvecklaren passa sig för att använda ovanliga typsnitt eftersom de måste finnas i användarens dator för att kunna återges.

Nedan följer en lista med några av de verktygspaneler som finns i Direktor.

- Property inspector (egenskapsfönstret)

Här presenteras egenskaperna för ett objekt, ett element eller själva filmen antingen grafiskt eller som en lista. Utvecklaren kan på ett smidigt sätt se eller ändra på egenskaperna genom att ange nya värden där så tillåts.

- Tool palette (verktygspaletten)

Här finns verktyg för att rita grundläggande geometriska former som rektanglar, cirklar osv.. Det har även verktyg för att skapa radioknappar, kryssrutor, tryckknappar och textfält. Allt det här görs direkt på scenen i vektorgrafik. Elementet som skapas av verktygspaletten kallas för ”Shape” i Cast och skiljer sig bara något från elementen som skapas i fönstret Vector Shape genom att kvalitén är något lägre.

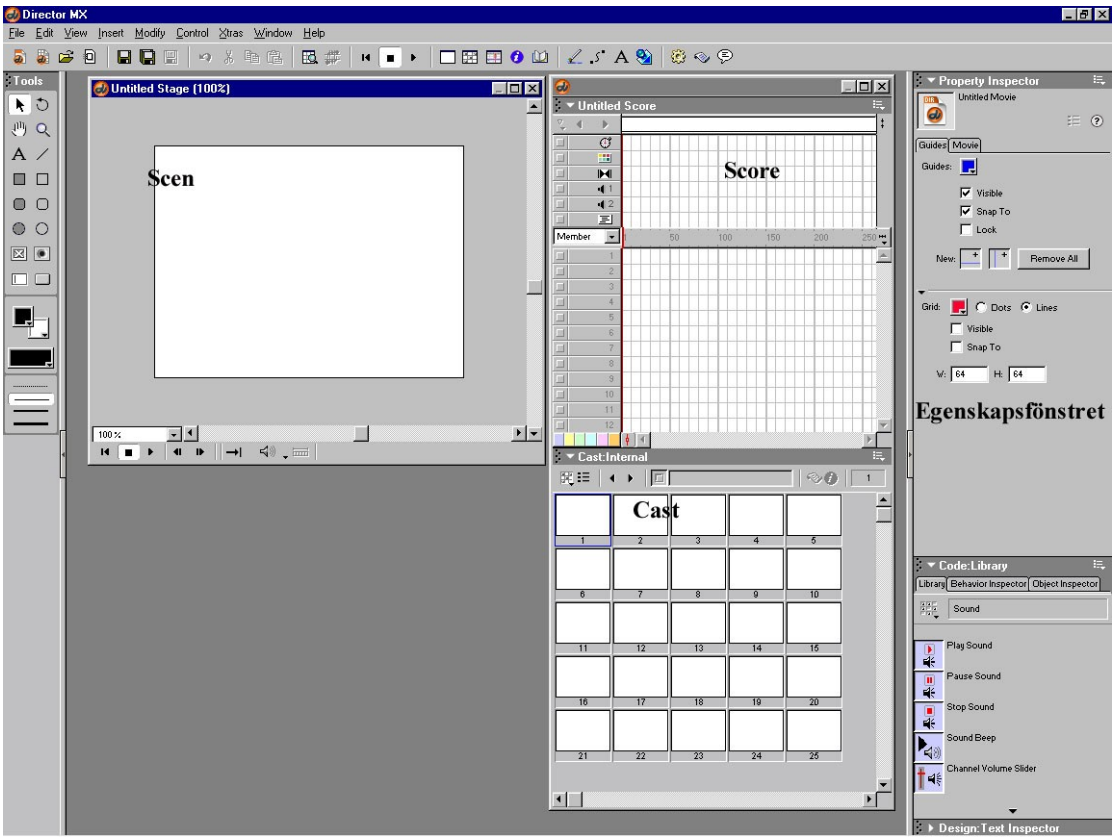
- Onion Skin

Används i kombination med ritfönstret Paint för att skapa bilder som följer en annan bilds form. Jämför med traditionell animation där animatören ritade på papper och sedan lade på ett nytt papper som det gick att se den första bilden igenom. Den första bilden kunde sedan ritas av och någon liten detalj förändras.

- Align

Fönster för att ordna olika objekts inbördes positionering på scenen.

De fyra fönstren i Directors utvecklingsmiljö som utvecklaren använder hela tiden, visas i Figur 2:4, är Scenen, som är huvudfönstret, Cast, Score och Egenskapsfönstret. De fönstren som används för att skapa grafik är också viktiga men behövs inte om elementen som ska användas skapas i ett externt utvecklingsprogram och importeras till Cast[6][9][15].



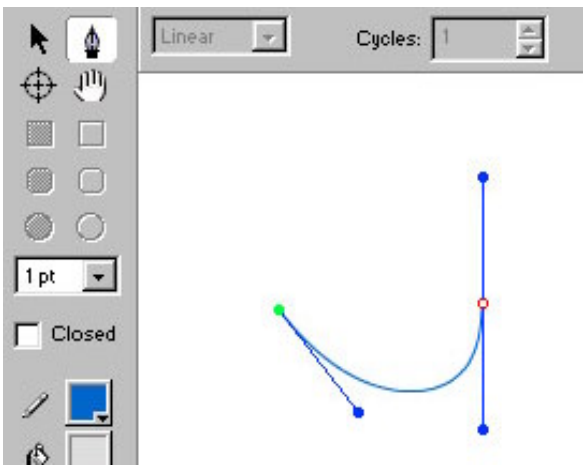
Figur 2:4: Directors utvecklingsmiljö

2.3.2 Utveckling

Utplacering av element görs från Cast genom att klicka och dra elementen till scenen eller till en kanal. När det görs skapas ett objekt av det aktuella elementet. Om objektet skapas genom att elementet dras till scenen så visas objektet också i den översta lediga kanalen i Score. Om objektet istället skapas genom att elementet dras till en kanal i Score visas det också i centrum av scenen. Det här gäller dock inte om det är ett osynligt element som t.ex. ljud, då görs samma sak fast till en specialeffektskanal i Score. Generellt för alla objekt är att dess existens i filmen bestäms genom att objektens respektive ände dras ut längs tidslinjen i Score. Det objekt som är placerat i den översta kanalen i Score är det objekt som är placerat längsta bak på scenen, dvs. ju längre ner bland kanalerna i Score desto längre fram på scenen. Objektens inbördes ordning bland kanalerna ändras dock enkelt genom att klicka och dra dem till önskad ledig kanal. Genom Lingoscript kan objekt också placeras ut och placeras om under filmens gång. Under utvecklingsarbetet kan varje kanal stängas av om så önskas, men de objekt som är placerade i kanalen kommer alltid med på den slutliga filmen. Avstängningen har alltså bara verkan under uppspelning i utvecklingsmiljön och kan vara bra om ett beteende för ett särskilt objekt på scenen behöver kontrolleras och utvecklaren inte vill se alla andra objekt.

- Rita

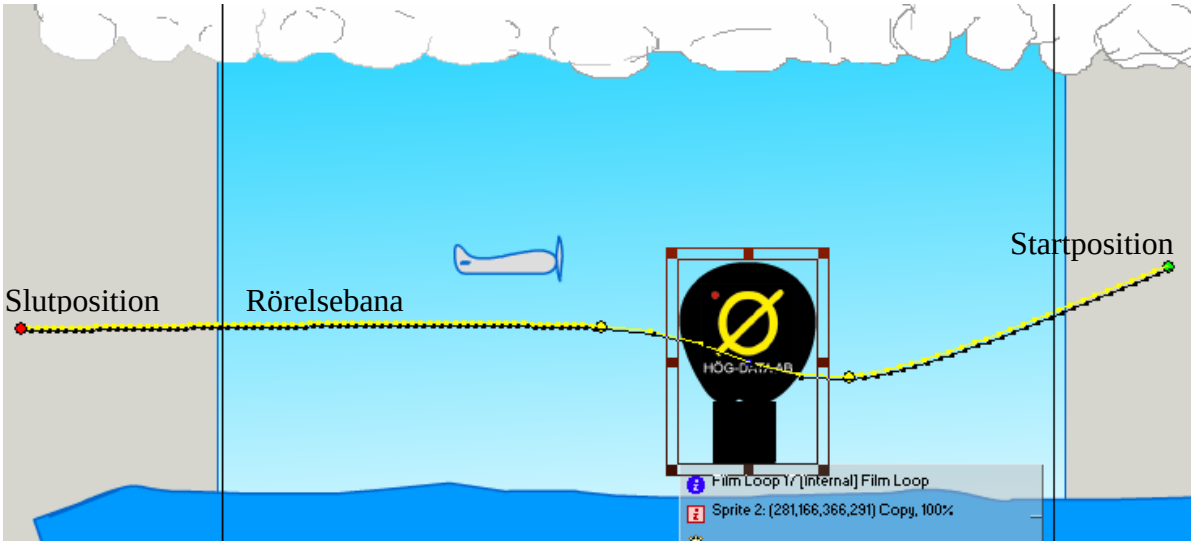
Ritar gör utvecklaren antingen direkt på scenen med verktygspaletten eller i fönstren Paint och Vector Shape, se beskrivning i avsnitt 2.3.1. Ritar man med verktygspaletten direkt på scenen kan det vara bra att först markera den ram där det är tänkt att objektet ska börja visas i filmen, så behöver den inte flyttas senare. Här finns det inte något ritstift för att rita oregelbundna figurer på frihand och bilderna blir inte lika jämna och fina som de som ritas i Vector Shape, fast båda ger vektoriserad grafik. Av de två ritfönstren Paint och Vector Shape är Vector Shape också att föredra, tack vare att bilderna skapas i vektorformatet mot bitmappformatet i Paint. Därför är det synd att det ritstift som finns där för att rita oregelbundna figurer på frihand inte är mer lätthanterligt. Det går säkert att få det som ska ritas att bli snyggt om den som utvecklar har gott om tid och tålamod. När ritstiftet används placeras noder ut så att när det finns minst två noder bildas en linje mellan noderna. Genom att dra ut ”handtag” för noden bestäms det i vilken riktning som linjen ska ta igenom noden. Jämför med en tangent i matematikens geometri, där tangenten i en graf för en ekvation är en rät linje som beskriver lutningen i en given punkt. Genom det här sättet att dra ut ”handtag” får utvecklaren först göra en grovskiss över det som ska ritas för att sedan gå in och justera nodernas positioner och "handtagens" riktningar, det här tillvägagångssättet illustreras i Figur 2:5. Om utvecklaren vill lägga till fler noder efteråt för att bättre kunna justera figuren så är det bara att klicka dit dem med ritstiftet. Att rita på det här sättet tar ganska lång tid men resultatet kan bli riktigt snyggt. Oavsett var figurerna ritas så placeras det som ett nytt element i Cast.



Figur 2:5: Ritstiftet i fönstret Vector Shape

- Animation

Animation i form av "frame-by-frame"-animering kan skapas manuellt med något av ritverktygen eller med Onion skin-verktyget. Det kan också göras mycket enklare genom att använda programmets inbyggda animeringsmekanism, tweening. Det går till så att utvecklaren drar ut startposition och slutposition för animeringen, till en rörelsebana, med hjälp av objektets animeringshandtag, illustreras av Figur 2:6.



Figur 2:6: Exempel på animeringsmekanismen, tweening, i Director

På det här viset kan alla objekt animeras, men det kan se konstigt ut om t.ex. en fågel flyger över scenen utan att flaxa med vingarna. Det görs genom att kombinera den här mekanismen med ren "frame-by-frame"-animering där först ett flertal fåglar skapas, som på samma position på scenen endast har olika lägen på vingarna. Utifrån dem skapas ett nytt element, en filmloop, som utför animationen. En filmloop är ett verktyg som gör det möjligt att förena flera fristående element/objekt till ett enda element. Filmloopen kan i sin tur placeras ut som ett objekt och animeras med den inbyggda mekanismen och då har en fågel som flaxar längs en förbestämd bana skapats (se även frame-by-frame i ordlista, avsnitt 2.2).

- Interaktivitet

Interaktivitet och beteende av enklare strukturer kan skapas i beteendefönstret. Lite mer avancerade beteenden kan skapas med Lingoscript. De fyra olika sätten att skapa script är att skriva dem för en film, för ett element, för ett objekt eller för en ram. Alla utom det sista sättet skapar ett nytt eget element i Cast.

Filmscript är kopplade till hela filmen och är tillgängliga i alla objekt och ramar under uppspelning av filmen. Den här typen av script kan också anropas av andra typer av script.

För att skriva in koden öppnar man scriptfönstret i fönstermenyn och väljer att skapa ett nytt script. Sedan väljs typen Movie i egenskapsfönstret.

Elementscript är kopplade direkt till ett element så att oberoende av i vilken ram filmen befinner sig är scriptet tillgängligt för ett objekt skapat från det elementet. För att skriva ett script för ett element markeras elementet i Cast och sedan klickar man på script-iconen.

Objektscript är kopplade till ett objekt under den tid som objektet finns på scenen. Genom att först markera objektet och sedan välja att öppna scriptfönstret med modifieringsalternativet i menyn så kan kod skrivas in.

Ramscript är kopplade till en ram i Score och skapas genom att dubbelklicka på den ram i scriptkanalen (specialeffektkanal i Score) där scriptet ska gälla. Scriptfönstret öppnas och utvecklaren kan skriva in sin kod.

När filmscript skrivs så skapas ett element av typen *moviescript.* När ett elementscript skrivs så skapas inget nytt element i Cast, men det kan ändå ses där i form av en ikon bredvid elementet, om grafisk visning av elementen är vald. När objekt- eller ramscript skrivs så skapas ett element av typen *behaviorscript.* Det finns ytterliggare en typ av script som skapas som ett element i Cast och det är *parent-script*. Med ett sådant script kan objektorienteringens fördelar användas för att skapa nya scriptobjekt som har liknande beteenden men som kan fungera oberoende av varandra.

Generellt när utvecklaren skriver kod så gäller att han/hon kan få hjälp av den inbyggda ordlistan, som är åtkomlig via de fönster som används för att skriva Lingokod. För beskrivning av scriptfönstret se avsnitt 2.3.1.

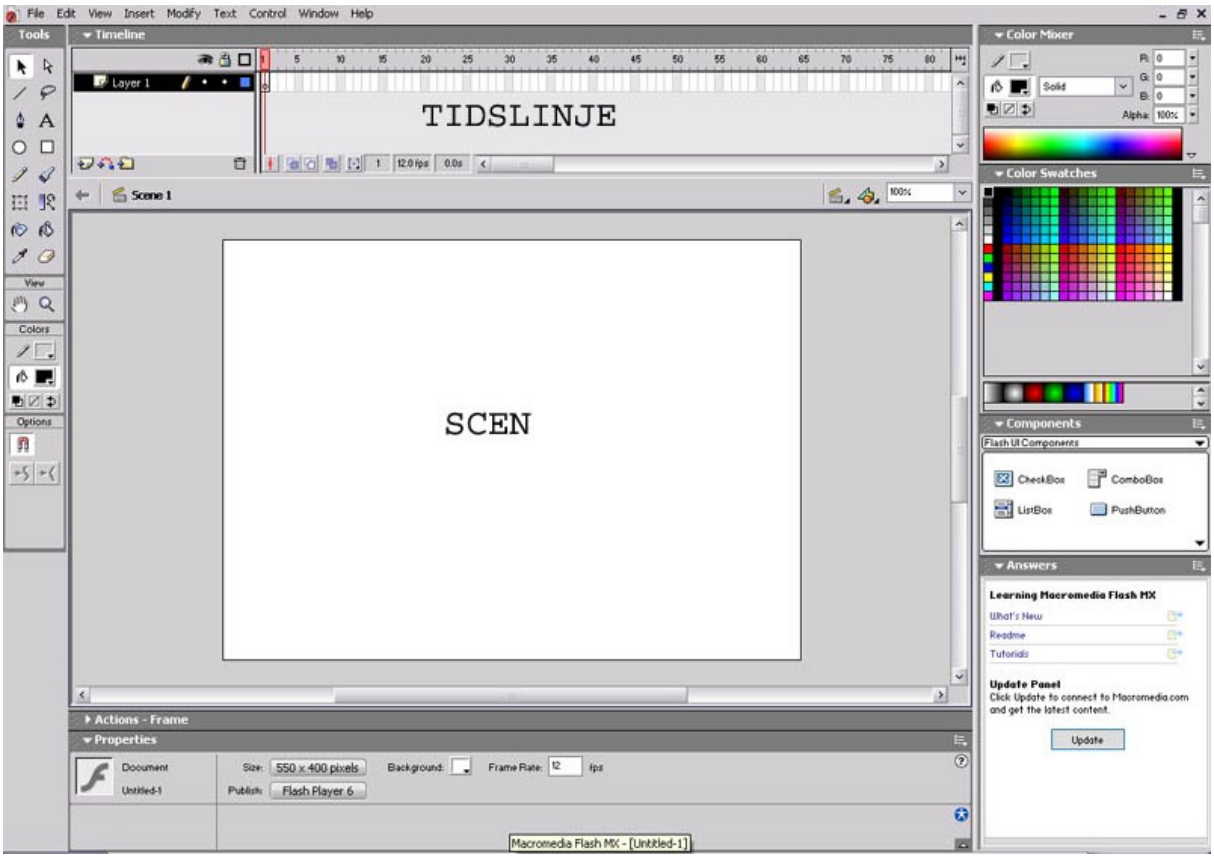
När en film skapas så används egenskapsfönstret (Property Inspector) för att se eller förändra egenskaperna för markerade objekt, element, hela filmer eller fönster. I egenskapsfönstret kan visningen av egenskaperna fås grafisk eller som en lista. Till skillnad från den grafiska visningen så benämns egenskaperna i den listade visningen i termer som används i scriptspråket. Exempelvis är ett objekts höjdposition i den grafiska visningen benämnt med "Y", medan den benämns "locV" i den listade. Den listade visningen ger alltså mer information om utvecklaren använder mycket script, som vid mer avancerade produktioner[6][8].

2.4 Flash

Innan Macromedia tog över utvecklingen av Flash 1997, kallades det för FutureSplash. På den tiden gjordes det reklam för Flash som ett program för att skapa grafiskt material för webben. Idag när datoranvändare pratar om Flash så är det inte nödvändigtvis programmet Flash det pratas om utan kanske själva tekniken. Undersökningar visar att runt 80-90 procent av webbsurfarna världen över kan se Flash-filmer[16]. Dessa filmer är oftast inbakade i en webbsida och visas m.h.a ett plug-in för webbläsaren, som heter ”Flash player”. I programmet Flash finns det verktyg för att skapa vektorgrafik, men det går även att importera annan grafik av olika filformat. Förutom att skapa grafik kan animationer, användargränssnitt, reklam (banners), logotyper, tecknade filmer mm skapas[13].

2.4.1 Utvecklingsmiljö

I avsnitt 2.2 beskriver vi scen och tidslinje och i Flash ser de ut som i Figur 2.7.

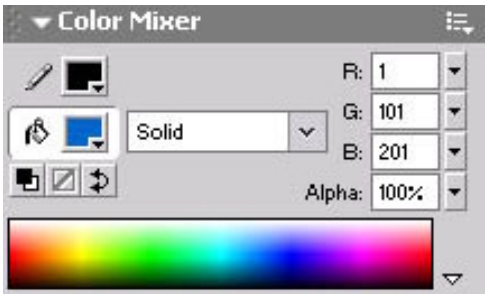


Figur 2.7: Flashs utvecklingsmiljö

I det avsnittet skrev vi att på scenen placeras de objekt som används i filmen. Ett ytterliggare tillägg till detta är att i Flash sker all utveckling av visuella objekt, som grafik och

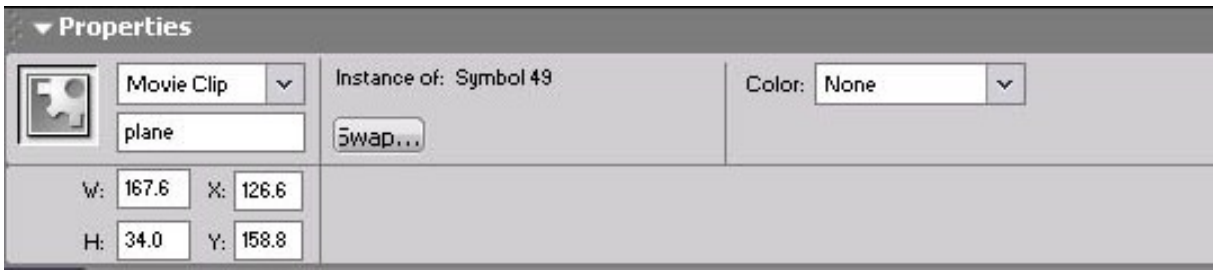
animation, direkt på scenen. Det finns alltså inga extra dokumentfönster för det här. Däremot finns det andra fönster, ritverktyg och paneler, som hjälp vid utveckling av visuella objekt.

Paneler är små fönster som kan läggas till och tas bort från Flashs arbetsyta. Dessa paneler kan utföra speciella uppgifter t.ex. välja objekt, ändra färg på objekt osv. Figur 2.8 visar en bild på panelen Color Mixer. Allt som har med objektet/objektens färger att göra ändras/kontrolleras här.



Figur 2.8: Color Mixer

En annan panel och kanske den viktigaste i Flash är Property inspector, Figur 2.9. Med den ändras egenskaperna för ett objekt. Property inspector ändrar utseende beroende på vilket objekt som har markerats. Om exempelvis en text markerats så kan typsnitt ändras.

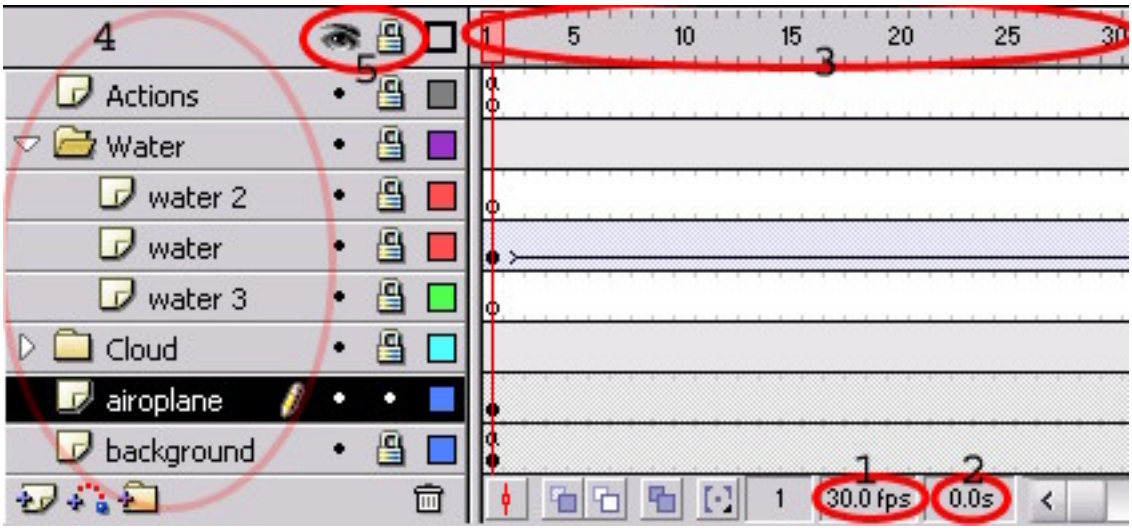


Figur 2.9: Property inspector

Ritverktygen i Figur 2.10 används för att skapa vektorgrafik. Här finns de vanligaste verktygen som används i många bildbehandlingsprogram. En intressant detalj med textverktyget är att den ”bränner” in texten i filmen. Texten bakas alltså in i filmen och det innebär att det inte spelar någon roll om typsnittet finns i den dator där filmen spelas.

Figur 2.10: Ritverktyg

Tidslinjen, Figur 2.11, innehåller några viktiga delar.



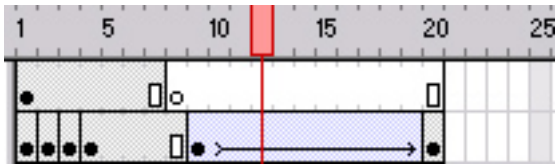
Figur 2.11: Tidslinjen

1. Här visas information om hur många ramar per sekund som filmen visar.
2. Här visas längden i sekunder mellan den första ramen till den *markerade* ramen.
3. På den här listan väljs vilken ram som ska visas på scenen. Indikatorn ”playhead” bestämmer vilken ram som är aktiv på scenen. Den står på ram 1 i Figur 2.11.
4. ”Layers” (lager) används för att organisera elementen i filmen. Det lager med objekt som ligger överst i listan av lager, är det lager med objekt som kommer att visas i fronten av de andra lagren. Säg att ett lager har ett grafikobjekt i form av en bild på fåglar. Ett annat lager har en bild på en bakgrund, kanske en himmel med moln, som täcker hela scenen. Om bakgrundslagret läggs överst i listan av lager kommer fåglarna aldrig synas, eftersom bakgrunder täcker hela scenen. Läggs däremot fågellagret överst i lagerlistan så syns de alltid.

5. Ögat markerar om ett lager med innehållande objekt ska synas eller inte. Markerar utvecklaren under ögat för ett lager syns dess objekt inte på scenen. Låset markerar om det ska vara möjligt att ändra, t ex: flytta, ändra storlek etc, på de objekt som lagret innehåller.

2.4.2 Utveckling

Det här avsnittet beskriver hur en film skapas i Flash genom att förklara hur utvecklaren ritar objekt, gör enklare animation och skapar interaktivitet. Men först passar det bra att förklara hur ramarna, Figur 2:12, på tidslinjen skapas/hanteras.



Figur 2:12: Ramar

En ram med en ring i markerar att det är en nyckelram. Om ringen är svart så betyder det att nyckelramen innehåller ett objekt, om den är vit finns det inget objekt i den. Om nyckelramen innehåller ett objekt, säg en bild, så är bilden synlig från den nyckelramens början till den svarta linjen, som markerar dess slut. Ramarna nummer 8-19 i det undre lagret på bilden markerar en motion tween, som är en slags animation (mer om det snart).

- Rita

Gör utvecklaren genom att välja den ram och lager där han/hon vill att objektet ska visas, vilket naturligtvis kan ändras senare. Sedan ritas objektet direkt på scenen. Om utvecklaren exempelvis vill rita fritt på scenen kan pennan användas för att rita vad som helst, eller kanske penseln för att rita lite tjockare. Ovalen eller rektangeln kan också användas för att rita ringar eller fyrkanter. Färgerna för objektet som ritats ändras/kontrolleras med panelen ”Color Mixer”.

- Animation

I Flash kan utvecklaren animera ”frame-by-frame”. En nackdel med det här är att när en ny nyckelram läggs till så skapas ett separat objekt för varje nyckelram och det innebär att filmens filstorlek ökar, vilket aldrig är bra. Men det finns ett alternativ till att ha ett nytt objekt för varje nyckelram, och det är symboler.

En symbol är ett grafiskt återvinningsbart objekt. Med det menas att om en nyckelram med ett objekt är konverterad till en symbol, så blir varje nyckelram som skapas efter denna i lagret en instans av den första symbolen. Instansen är inte en kopia av orginalsymbolen utan

refererar till den. Det innebär att om egenskaperna ändras, t ex: storlek, färg etc, på orginalsymbolen får alla instanserna samma ändring, men om förändringen sker på instanserna ändras inte orginalsymbolen.

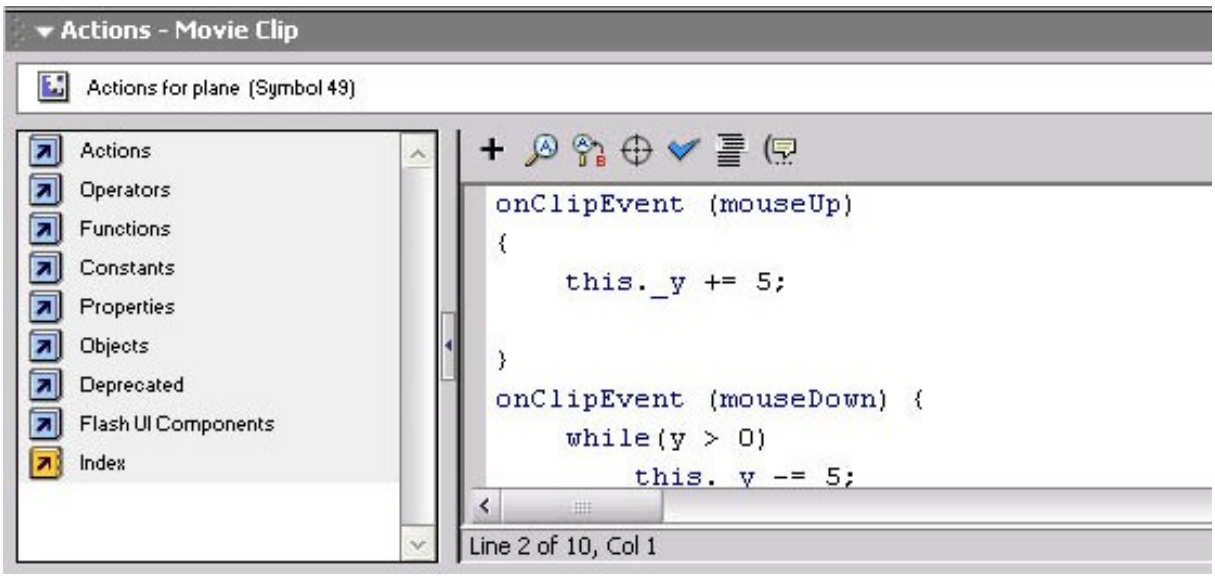
Alla objekt på scenen kan konverteras till en symbol. När ett objekt konverteras till en symbol så får utvecklaren välja vilket beteende den ska ha. Det finns tre olika: grafiskt, knapp eller filmklipp. Ett grafiskt beteende är det bästa valet för ett statiskt, ej rörligt objekt. Med knappbeteendet kan symbolen reagera visuellt vid intryckning av musknappen. Ett filmklippsbeteende liknar mycket det grafiska, men för ett rörligt objekt väljs filmklippsbeteendet.

”Frame-by-frame” animation är en manuell process. Utvecklaren lägger till nyckelram efter nyckelram och ändrar objektet/symbolinstansen i varje ram. Det här tillvägagångssättet kräver ett stadigt öga och mycket tålamod för att få en mjuk rörelse. En ”motion tween” kan göra det här automatiskt och endast två nyckelramar behöver användas. Säg exempelvis att ett lager har en nyckelram med ett objekt, i form av en bild, i sig. För att skapa en motion tween animation, konverteras först objektet till en symbol, grafisk eller filmklipp. Efter det lägger utvecklaren till en nyckelram längre fram i tidslinjen. Ju längre fram i tidslinjen den nyckelramen läggs, ju längre blir animationen. Den tillagda nyckelramen innehåller en symbol som är en instans av den första orginalsymbolen. Sedan placeras instanssymbolen på scenen där det är tänkt att animationen ska sluta. Till sist väljs egenskapen ”Tween motion” för orginalsymbolen och så är en ”motion tween” animation skapad.

- Interaktivitet

Det finns två sorters händelser i Flash: Filmhändelser och användarhändelser. En filmhändelse sker automatiskt under filmens gång, exempelvis att filmen stoppar vid en speciell ram. En användarhändelse är precis vad namnet säger, något som användaren gör, exempelvis trycker ned en musknapp. En åtgärd (action) kan kopplas till dessa händelser som bestämmer vad som ska ske när händelserna inträffar. Händelser och åtgärder sammankopplat ger interaktivitet. Exempel på interaktivitet i filmer skapade med Flash kan vara att användaren startar och stoppar filmen, startar och stoppar ljud, flyttar runt objekt på scenen osv.

När interaktivitet skapas, enkel eller komplex, används alltid åtgärdspanelen, Figur 2:13. I åtgärdspanelen syns också den existerande interaktiviteten.



Figur 2:13: Åtgärdspanelen

Ett val som kan göras i åtgärdspanelen är mellan normal mode eller expert mode. I normal mode behöver utvecklaren inte veta hur åtgärdsscriptkod skrivs. Här väljer utvecklaren istället ur en meny över åtgärder som kan användas och åtgärdsscriptkoden skrivs automatiskt. I expert mode kan koden skrivas fritt i textrutan som finns i åtgärdspanelen. Fördelen med det är att utvecklaren kan vara mer specifik och har själv full kontroll över åtgärdsscriptkoden.

Åtgärder kan skapas för en ram eller för ett objekt. En åtgärd kopplad till en ram bestämmer vad som ska ske när filmen spelar den ramen, händelsen kopplad till en ramåtgärd måste då tillhöra kategorin filmhändelse. Exempel på åtgärdsscriptkod för en ramåtgärd: stop(); - stoppar filmen, gotoAndStop("start"); - filmen hoppar till den namngivna ramen ”start” och filmen stoppas. En åtgärd för ett objekt gäller under hela objektets livstid (de ramar objektet finns på). En åtgärd för ett objekt kan kopplas till både en filmhändelse eller en användarehändelse. Ett exempel på en objektåtgärd för en filmhändelse kan vara att objektet på scenen ändrar position vid början på varje ram för objektet. Ett exempel på en objektåtgärd för en användarhändelse kan vara att ett objekt på scenen ändrar position när en musknapp är nedtryck. Alla objekt som det ska skapas åtgärder för måste först konverteras till symboler (filmklipp- eller knappbeteende). Efter det har gjorts kan objekten namnges och användas i åtgärdsscriptkoden.

3 Experimentell spelkonstruktion

Kapitlet är en redovisning av det arbete vi gjorde för att uppnå MÅL-2 och AP-2 och en del av det arbete vi gjorde för att uppnå MÅL-3 och AP-3, beskrivet i Tabell 1:1 och Tabell 1:2.

Målet med att göra det här experimentet var att vi skulle lära oss använda utvecklingsverktygen, eftersom vi aldrig hade använt dem tidigare. Vi funderade ett tag på vilket sorts spel vi skulle skapa. Det fick inte vara något för avancerat eftersom det då fanns risk att vi skulle fastna någonstans på vägen och därmed inte hinna prova på de delar vi behövde prova på.

3.1 Spelidé

Vi bestämde oss för att spelet behövde innehålla lite av de viktigaste beståndsdelarna i begreppet multimedia: Text, statisk grafik, animationer, interaktivitet och ljud.

Figur 3:1: Spelet

Spelet som visas i form av en ”skärmdump” i Figur 3:1, innehåller följande: En bakgrundsbild som täcker hela scenen, flytande vatten i form av en animation i nedre delen av

scenen, svävande moln i form av en animation i övre delen av scenen, ett flygplan som rör sig upp eller ned på scenen, beroende på vad användaren gör (interaktivitet). Slutligen en luftballong som rör sig från höger till vänster och som inte får krocka med planet. Lite information visas i början av spelet, hur spelet startas, hur flygplanet styrs mm. Idén är att om planet krockar med ballongen, vattnet eller molnen så är spelet slut och en ”Game over”-text visas på scenen.

3.2 Konstruktionen

Två experimentella spel skulle konstrueras, ett för Flash och ett för Shockwave. Vi bestämde oss för att försöka få spelen att likna varandra så mycket som möjligt, eftersom vi senare skulle göra en jämförelse av de båda. Vi gjorde så att vi skapade först ett element/objekt, exempelvis ritade molnen, i det ena programmet och sen i det andra. Efter det fortsatte vi med att skapa en nytt element/objekt i det programmet vi slutade med. Eftersom vi arbetade på det här sättet så kunde inte ett och samma program hela tiden dra fördel av att vi lärt oss något förfarande i det första programmet.

Storleken på den synliga delen av scenen (spelplanen) bestämde vi till 512x342 pixlar, som var en fördefinierad storlek i Director och som kändes lagom stor för att ge en bra spelkänsla. Vi bestämde också redan från början för att vi skulle sätta visningshastigheten till 30 ramar per sekund (fps) för att få mjuka och fina rörelser i animationerna.

En ganska viktig beståndsdel i en multimediaproduktion är ljud. Vi planerade att ha med några ljudsnuttar vid olika händelser. I Director var det här enkelt att göra på kort tid, men i Flash så satt vi ganska länge utan att få till något bra och därför uteslöt vi helt ljuddelen i projekten för att inte hamna i tidsbrist.

Vi beskriver nu hur vi skapade spelen i programmen, från enklare grafik i form av statiska bilder, till animation och interaktivitet. I det första avsnittet, 3.2.1, beskriver vi den statiska grafiken och *vad* som ska göras och *varför*. I de två avsnitten efter det, 3.2.2 och 3.2.3, beskriver vi *hur* vi gjorde den i Director och Flash. I avsnitt 3.2.4 *sammanfattar* vi de tre avsnitten vi skrivit om den statiska grafiken, genom att beskriva lite av den känsla vi fick av respektive program under konstruktionsarbetet. Vi nämner även några av de problem som uppstod och som i vissa fall ledde till en alternativ lösning jämfört med hur vi tänkt oss den från början. Efter det kommer fyra avsnitt, 3.2.5-3.2.8, där animationen beskrivs och fyra avsnitt, 3.2.9-3.2.12, med interaktiviteten, med samma förfarande som för den statiska grafiken.

3.2.1 Statisk grafik

Den statiska grafiken som ska skapas är en bakgrund som täcker hela scenen, moln, vatten, ett flygplan, en luftballong och lite informationstext. Kom ihåg sen tidigare att det vi ritar i Director sparas först som ett element i Cast innan det läggs in på scenen och blir ett objekt. I Flash ritar vi direkt på scenen och att då skapas ett objekt direkt.

- Bakgrunden

Först ska vi rita en bakgrund som täcker hela scenen. Vi har tänkt blanda två färger gradvis, ljusblått och vitt, för att få en ljusare ton i nedkanten av scenen. Det i ett försök att efterlikna en himmel.

- Moln och vatten

Eftersom molnen och vattnet senare ska animeras för att få känslan av att flygplanet flyger, beskrivs i avsnitt 3.2.5, så ska vi rita flera moln- och vattenelement.

- Flygplanet och luftballongen

Vi ska så bra vi kan försöka rita ett flygplan och en luftballong. Vi vill gärna ha med Hög-Datas logotyp någonstans i spelet och en lämplig plats för det här tycker vi är på luftballongen.

- Informationstext

Vi ska skapa text som senare ska användas för att beskriva för användaren i början av spelet vad det går ut på. Vi ska även skapa en ”game over” text som ska visas när flygplanet krockar med ballongen, molnen eller vattnet.

3.2.2 Statisk grafik: Shockwave

- Bakgrunden

Vi använde Vector Shape, för att skapa vektorgrafik. Där ritade vi en fyrkant stor nog att täcka hela scenen. Det blev nu ett element i Cast, som vi döpte till background och när vi lade in den på scenen så skapades automatiskt ett objekt som även visas i översta kanalen i Score, med namnet background. Under konstruktionens gång anpassade vi bakgrundsobjektet i kanalen så att det visas i filmens alla ramar.

- Moln och vatten

Fyra respektive tre nya moln- och vattenelement ritades och de lade sig automatisk i Cast, som användbara element precis som med bakgrunden. Vattenelementen ritades i Vector Shape med ritstiftet. Molnen däremot gick inte att göra där, eftersom de helt enkelt blev för fula med de ritverktyg som finns där. Vi ritade dem istället i Paint där det fanns fler ritverktyg, men med skillnaden att där skapas det bitmappgrafik istället för vektorgrafik.

- Flygplanet och luftballongen

Eftersom det inte går att rita helt fritt i Vector shape så övervägde vi på nytt att använda Paint, när vi skulle rita flygplanet. Men vi lyckades till slut genom att rita en oval ring och sedan dra ut noderna för att få något så när bra form. Vi nämnde tidigare att vi gärna ville ha med Hög-Datas logotyp någonstans i spelet och att vi tyckte att en lämplig plats för den skulle vara på luftballongen. Problemet med det här var att logotypen var i form av en bild i bitmappgrafik och det gick inte att ”klistra” in logotypens grafik i luftballongens, eftersom de inte är av samma grafiktyp. Det här ville vi göra eftersom vi senare skulle animera luftballongen och ville då inte ha flera element, eftersom det skulle försvåra animeringen. Vi löste det genom att kombinerade den luftballong vi ritade och logotypen till *ett* enda element, markera all grafik som skulle kombineras och göra den till en filmloop (beskriven i avsnitt 2.3.2).

- Informationstext

Texten skrevs i Textfönstret, beskrivet i avsnitt 2.3.1, och elementen lade sig i Cast. Texten lades senare in som objekt, när vi skapade interaktiviteten.

3.2.3 Statisk grafik: Flash

- Bakgrunden

Vi skapade först ett nytt lager som vi döpte till background. Därefter markerade vi den ram i background-lagret på tidslinjen där vi ville att det objekt som skulle ritas, först skulle visas i filmen, den första i det här fallet. Efter det ritade vi direkt på scenen en fyrkantig ruta som täckte hela scenen och fick automatiskt ett nytt objekt i den markerade ramen på bakgrundslagret, på tidslinjen. Precis som i Director anpassade vi under konstruktionens gång bakgrundsobjektet i bakgrundslagret så att det visas i filmens alla ramar.

- Moln och vatten

Fyra nya lager med ett molnobjekt i varje skapades. De ritades m.h.a. ritverktyget pensel (för att få tjockare linjer än med ritstiftet). Tre lager med ett vattenobjekt i varje skapade på samma sätt. Vilka ramar respektive moln och vatten skulle visas på bestämdes senare när vi animerade dem.

- Flygplanet och luftballongen

Vi skapade ett nytt lager för flygplanet och ritade återigen direkt på scenen med ritverktyget pensel. Problemet med luftballongen, som vi beskrev under Shockwave-avsnittet, löste vi här genom att markera all grafik som skulle kombineras och göra den till en enda filmklippssymbol (beskriven i avsnitt 2.4.2).

- Informationstext

Vi skapade först två lager, ett för informationstexten och ett för ”game over”-texten. Texten skrevs direkt på skärmen med textverktyget och lade sig som ett objekt i den ram och lager som markerats. Vilka ramar texten skulle visas på i filmen bestämdes senare när vi skapade interaktiviteten.

3.2.4 Statisk grafik: Sammanfattning

Det vi tyckte var allra viktigast i utvecklingen av grafik var att allt som skapades skulle vara av typen vektorgrafik tack vare dess stora fördelar. Att vi inte klarade att rita molnen i Director med något annat än bitmapp-grafik var därför ett misslyckande för oss. Svårigheterna att skapa lite mer avancerade figurer av vektorgrafik i Director har gjort att det i flera böcker rekommenderas att utvecklaren använder sig av externa program som t.ex. Adobe Illustrator[6][17]. Eftersom vi ritade i ett annat fönster än själva scenen i Director var det svårt att veta var gränserna för det som skulle ritas gick. Känslan som vi fått under konstruktionsarbetet är att Director är avsett för att användas med importerade, färdiga vektorbaserade figurer och Flash är avsett att kunna användas för att rita vektorbaserade figurer direkt i programmet.

3.2.5 Animation

- Moln och vatten

För att få en känsla av att flygplanet rör sig framåt och åt höger vill vi att molnen och vattnet ska röra sig från höger till vänster i bilden

- Evighetsloop av moln och vatten

När vi har animerat molnen och vattnet vill vi att de ska fortsätta sin animering i all evighet om ingen händelse sätter stopp för det, mer om det i interaktivitet-avsnittet. Vi vill med andra ord skapa en loop utan avbrott som gör att animeringen fortsätter från början igen vid tidslinjens slut.

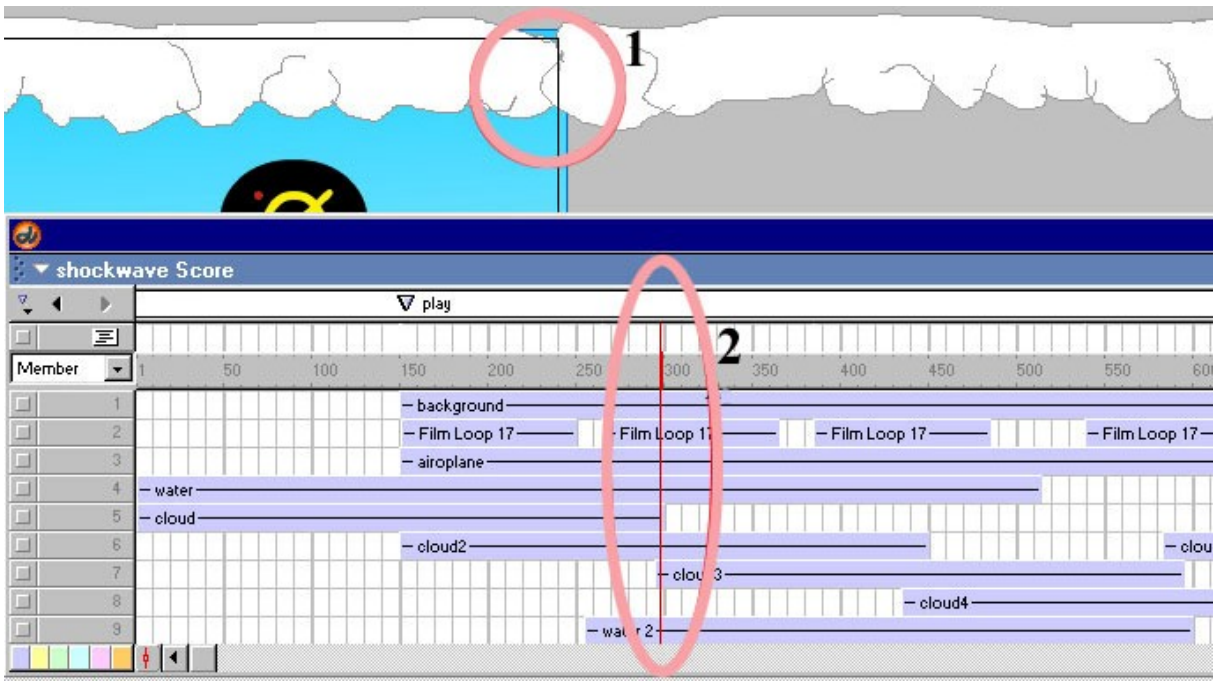
- Luftballongen

Luftballongen som vi tidigare ritat och monterat till en filmklippssymbol respektive en filmloop, animerar vi för att nå själva spelidén att flygplanet ska undvika föremål som det passerar längs färden. Luftballongen ska röra sig från höger till vänster över scenen för att ge känslan av att flygplanet passerar den, eller krockar med den.

3.2.6 Animation: Shockwave

- Moln och vatten

Vi började med att markera första ramen i tidslinjen och tog därefter ett av molnen vi skapat, från Cast och placerade på scenen. Vi placerade molnet på den position som vi ville att molnet skulle börja förflytta sig ifrån, alldeles utanför den synliga delen av scenen. Sedan utnyttjade vi Directors inbyggda animeringsmekanism genom att markera och dra i objektets animeringshandtag (beskriven i avsnitt 2.3.2). Den skapade rörelsebanan visade i vilken riktning molnet skulle röra sig för att slutligen stanna när molnets högra ände nått utanför scenens vänstra kant. För att animationen av molnen skulle anses som färdiga behövde vi bestämma med vilken hastighet molnen skulle förflytta sig från kant till kant. Det var ett viktigt beslut eftersom vi trodde att det inte skulle vara så enkelt att ändra på det senare när fler objekt placerats ut i spelet. Det här gjorde vi genom att dra i objektets högra ände i Score så att vi fick ett önskat antal ramar för animeringens livstid. Nu kunde vi skapa ytterligare molnobjekt från de tre andra molnelementen i Cast och placera ut dem i nya kanaler i Score. De här tre objekten animerades på samma sätt, och naturligtvis med samma hastighet som det första molnet. För att det inte skulle bli några luckor mellan molnen på scenen placerades vänster kant på det moln som skulle passera efter ett annat, lite ovanpå det föregående molnets högra ände, se markering ”1” i Figur 3:2. Det nästkommande molnets animation började med andra ord tidsmässigt i spelet i den ram där det föregående molnets högra ände är på väg in på scenen, se markering ”2” i Figur 3:2. På samma vis animerade vi sedan även vattnets rörelse.



Figur 3:2: Överlappande molnobjekt i Directors utvecklingsmiljö

- Evighetsloop av moln och vatten

Vi gjorde ett ramscript, beskrivet i avsnitt 2.3.2, för den ram där det *sista* molnets (och vattnets) högra ände är på väg in på scenen. I det här scriptet skrev vi kod för att få filmen att ”hoppa” tillbaka till den ram där *första* molnets högra ände är på väg in på scenen. För att enklare kunna hålla reda på till vilken ram på tidslinjen som filmen skulle ”hoppa” tillbaka till så skapade vi en markering med namnet ”play” som vi sedan placerade ovanför ramen. Nu kunde vi hänvisa till det namnet i scriptet istället för ramnumret. Anledningen till att filmen inte hoppar från sista ramen för sista molnobjektet tillbaka till allra första ramen är att då skulle det *sista* molnet först försvinna ut helt från scenen åt vänster för att sedan låta det *första* molnet komma in på scenen från högerkanten. För att undvika det beteendet även när spelet startar så skrev vi ett script för den första ramen som gör att filmen hoppar direkt till en ”start”-markering som skapats för ramen alldeles före "play"-markeringen.

I det här läget trodde vi att vi var klara med den här funktionaliteten, men så fort vi testade och tittade lite närmare på övergången mellan första och sista molnobjekten upptäckte vi det uppenbara. Eftersom inte första och sista objektet hade exakt samma form så syntes övergången väldigt tydligt. Lösningen till det här var helt enkelt att duplicera de första moln- och vattenobjekten, med animering och allt, genom att kopiera alla ramar i kanalen som innefattade dem. Därefter klistrade vi in dem sist och flyttade naturligtvis också med scripten som får filmen att hoppa tillbaks.

- Luftballongen

Animationen skapades på samma sätt som för molnen och vattnet. För att få luftballongen att dyka upp ganska ofta så skapades flera objekt genom att kopiera det första flera gånger precis som vi gjorde när vi duplicerade moln- och vattenobjekten. Sedan ändrade vi varje luftballongobjekt så att animeringarna startade på olika positioner i höjdled. De olika animerade luftballongobjekten delades sedan upp med ojämna mellanrum i en kanal för att få oregelbundna tidsluckor mellan luftballongernas framträdanden på scenen, se objekten ”Film Loop 17” i Figur 3:2.

3.2.7 Animation: Flash

- Moln och vatten

Här använde vi exakt samma metod som i Director, se avsnitt 3.2.6, för att få rätt hastigheten på molnen, och för att inte få några luckor mellan dem på scenen. För att få molnen att röra sig över scenen i en viss riktning så konverterade vi molnobjekten till en symbol med filmklippsbeteende och sedan bestämde vi de objektens start- och slutposition i animationen, precis som i Shockwavespelet. Med början på första ramen i tidslinjen så skapades en motion-tween, beskrivet i avsnitt 2.4.2, som sträckte sig lika många ramar som livstiden för animationen skulle vara. Även i Flash så animerades sedan vattnet på exakt samma sätt som molnen.

- Evighetsloop av moln och vatten

Gjordes på samma sätt som i Director, se avsnitt 3.2.6. Med de skillnader att i Director skapades en markering för ramen. Här gav vi istället *ramarna* namnen ”loop” respektive "start".

- Luftballongen

Gjordes på samma sätt som i Director, se avsnitt 3.2.6, med den skillnaden att själva rörelsen skapades med motion-tween och inte med animationshandtaget.

3.2.8 Animation: Sammanfattning

Vi nämnde tidigare att det går att skapa ett antal olika sorters animationer. Men de vi skapade handlade enbart om att förflytta ett objekt från en position till en annan, under en viss tidsperiod. I Director gjordes det här genom att dra i objektets högra ände i kanalen i Score för att bestämma animeringens livstid. Förflyttningen av objektets position från start- till slutposition bestämdes m.h.a. animeringshandtaget. I Flash skapades en nyckelram med objektet (animeringens startposition) och ytterligare en nyckelram längre fram (animeringens

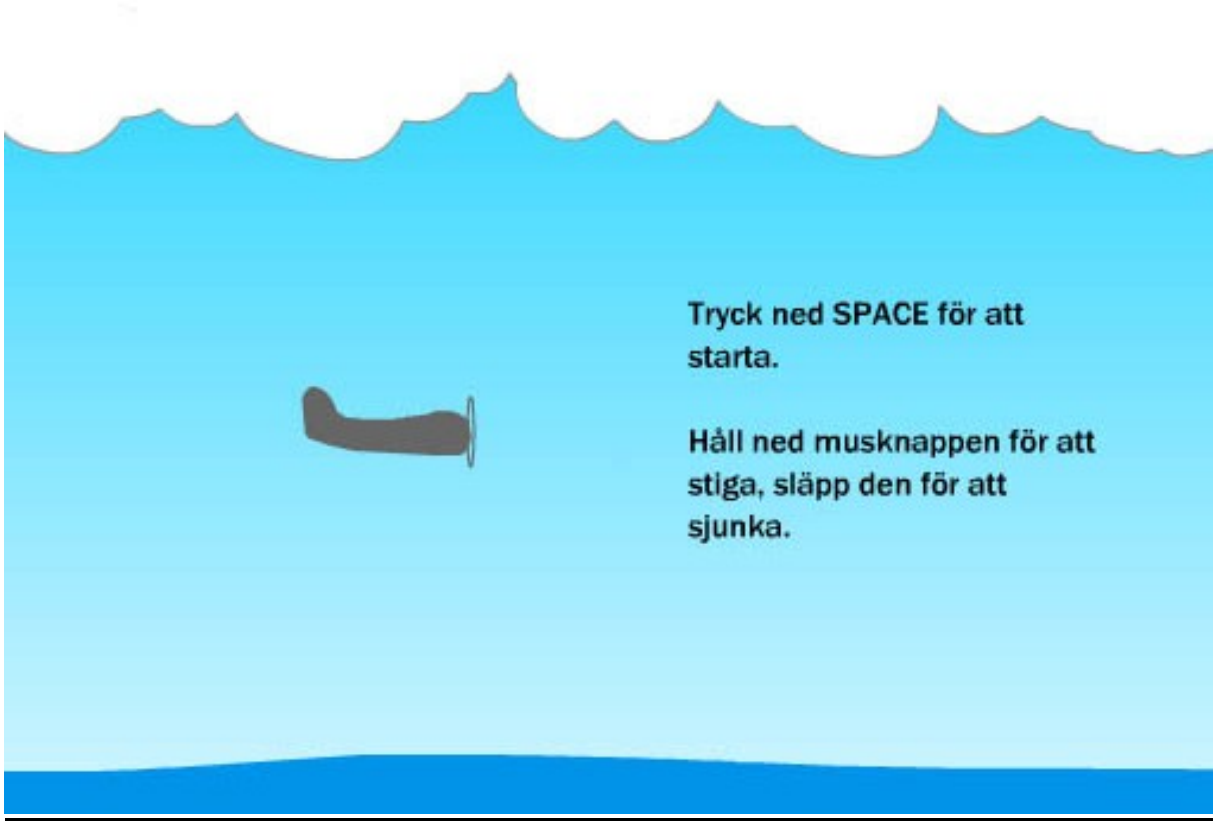
slutposition) på tidslinjen. Det som skapade problem här var att den första nyckelramen (startposition) skulle först konverteras till en symbol innan den andra nyckelramen (slutposition) skapades. Gjordes inte det här så blev inte den andra nyckelramens objekt en instans av den första nyckelramens objekt och då visade det sig senare att det inte gick att göra en motion-tween. Proceduren var alltså lite mer omständig i Flash än i Director och kändes inte lika intuitiv.

3.2.9 Interaktivitet

Vi har bestämt att vi vill ha följande interaktivitet i spelet: Flygplanet stiger om användaren trycker ner vänster musknapp på scenen och sjunker om den är uppsläppt. Spelet är slut om flygplanet krockar med ballongen, störtar i vattnet eller flyger för högt bland molnen. När en krock skett visas en ”game over”-text och användaren kan trycka på Space-tangenten för att starta spelet på nytt.

- Startbilden

Vi börjar med att skapa en startbild i spelet. Med det menas att vi vill ha en första bild där spelet inte rullar (allt står still). Här ska flygplanet vara mitt på skärmen och den instruktionstext vi skapade i statisk grafik avsnittet ska visas.



Figur 3:3: Startbild av spelet

Användaren ska m.h.a. instruktionstexten få veta att spelet startas genom att trycka på Space-tangenten och att planet stiger om han/hon trycker på vänster musknapp på scenen och sjunker om den är uppsläppt, se Figur 3:3.

- Spelstart

Vi vill att spelet/filmen ska startas igen, från att ha stoppats på startbilden, genom att användaren trycker på Space-tangenten.

- Styrningen av planet

Efter att användaren har tryckt på Space-tangenten så ska spelet starta. Nu ska flygplanet stiga om användaren trycker ned vänster musknapp någonstans på scenen och sjunka om den är uppsläppt.

- Krock

För att fånga upp händelsen av att två föremål på scenen korsar varandras väg (krockar) så kommer vi att behöva skriva någon scriptkod som håller koll på objektens koordinater på scenen. Den här scriptkoden ska kunna ”se” om flygplanet har krockat med luftballongen, molnet eller vattnet. Sker det så ska filmen stoppas och visa den ”game over” text vi skapade i avsnittet ”statisk grafik”.

3.2.10 Interaktivitet: Shockwave

- Startbilden

Att få filmen att stanna på ett visst ramnummer löstes m.h.a. scriptprogrammering. Vi skrev ramscriptskoden: go to "start", för filmens första ram. Det som händer då är att filmen ”hoppar” från första ramen till ramen med markeringen ”start”, vilket innehåller alla objekt som ska visas på första bilden. Eftersom det inte fanns någon funktion i Director som hoppar till en ram och stoppar spelningen av filmen fick vi först göra hoppet i första ramen och sedan stoppa filmen i ramen med markeringen ”start”. Där skrev vi ramscriptskoden: go to the frame, som gör att ramen ”loopas” fram och tillbaka, vilket innebär att filmen har stoppats.

- Spelstart

Filmen/spelet startar alltså om Space-tangenten trycks ned. Vi skapade den här funktionaliteten m.h.a. ett ramscript för ramen med ”start”-markeringen, Figur 3:4.

```
if the keyPressed = SPACE then
    play "play"
```

Figur 3:4: Lingoscript för spelstart

Scriptet säger att om Space-tangenten trycks ned så "hoppa" till ramen med markeringen "play", dvs. filmen startas.

- Styrningen av planet

Vi skrev först scriptkoden i flygplanselementet, vilket borde vara det naturliga, men det visade sig då att det inte gick att ”läsa av” nedtryckningen av musknappen annat än när den skedde *på* flygplanet. Det vill säga när användaren klickade direkt på flygplanet. Vi löste det här genom att skapa ett nytt element. En transparent rektangel som täcker hela scenen och som då ska kunna läsa av mushändelser överallt på hela scenen. Vi drog elementet från Cast till en ny kanal och ändrade livstiden för objektet till samma som för flygplanet. Det här nya objektet/symbolen kallar vi i fortsättning av beskrivningen för ”knappen”. Scriptkoden för ”knappen” ser ut som i Figur 3:5.

```
Global pressed
Global move
on MouseDown
        pressed = true
end
on MouseUp
        pressed = false
end
on Enterframe
        if move = TRUE then
                if pressed = true then
                        sprite(3).locV = sprite(3).locV - 3
                else
                        sprite(3).locV = sprite(3).locV + 4
                end if
        end if
end
```

Figur 3:5: Lingoscript för manövrering av flygplanet

Funktionen ”on MouseDown” exekverar koden i sin funktionskropp när musknappen är nedtryckt. Den består av en rad med en global variabel ”pressed” som sätts till ”true”. Funktionen ”on MouseUp” exekverar koden i sin funktionskropp när musknappen inte är nedtryckt. Den globala variabeln ”pressed” sätts då till ”false”. Funktionen EnterFrame exekverar koden i sin funktionskropp varje gång en ny ram under objektets livstid har påbörjats. Där kollas först om den globala variabeln ”move” har värdet ”true”. Den är ”true” om spelet startats och användaren inte fått ”game over”. Den här variabelns värde ändras om planet krockar med molnet, vattnet eller ballongen och förklaras under ”krock” på sidan 36. Efter det kontrolleras om musknappen är nedtryckt. Är den det så minskas värdet för flygplanets y-koordinat med 3 pixlar, vilket medför att flygplanet stiger lite vertikalt på

scenen. Är den inte nedtryckt ökas värdet för flygplanets y-koordinat med 4 pixlar medför att flygplanet sjunker lite vertikalt på scenen.

- Krock

I det här spelet är det hela tiden flygplansobjektet som, genom att korsa (krocka) något av moln-, vatten- eller luftballongsobjekten kan ”se” om en krock har skett. Därför skrev vi scriptkoden vi pratade om i avsnitt 3.2.9, i form av ett objektscript för flygplanet. Objektscriptet för flygplanet ser ut som i Figur 3:6.

```
on exitFrame
    if (sprite 3 intersects 2) or (the locV of sprite 3 < 20) or (the locV of sprite 3  > 310) then
        if the keyPressed = SPACE then
            go to frame 1
        end if

        sprite(13).visible = TRUE

        go to the frame
    end if
end
```

Figur 3:6: Lingoscript för krock mellan objekt

Funktionen ”on exitFrame” exekverar koden i sin funktionskropp varje gång en ny ram har avslutats. I den första kontrollsatsen kontrolleras om flygplansobjektets y-koordinat passerar ett bestämt y-koordinat uppåt eller nedåt, vilket då betyder det att händelserna "krock med vatten" eller "krock med moln" har inträffat. Det gick bra för vattnen- och molnobjektens y-koordinat, eftersom de inte förändras under spelets gång. Det gäller dock inte för händelsen "krock med luftballong" eftersom luftballongen inte har ”fasta” koordinater. Det här löste vi genom att i samma kontrollsats som ovan kontrollera om flygplansobjektet och luftballongobjektet ”träffar” varandra (sprite 3 intersects 2).

Om någon av dessa är sanna visualiseras objektet med texten ”game over” och filmen "hoppar" till samma ram som spelhuvudet står på (go to the frame). Att ”hoppa” till samma ram betyder att objekten på scenen är helt stilla och koden loopas igenom om och om igen. Så här skulle filmen/spelet hålla på i all evighet om inte kodraden funnits som säger att filmen ska hoppa till ram nummer ett om Space-knappen trycks ner (i ram nummer ”1” dirigeras filmen vidare till ”start”-markeringen)

3.2.11 Interaktivitet: Flash

- Startbilden

Här skrev vi åtgärdsscriptskoden: gotoAndStop(”start”), för filmens första ram. Precis som i Shockwave så ”hoppar” då filmen från första ramen till ramen med namnet ”start”. Men här

stoppas även filmen, vilket innebar att vi inte behövde skriva någon åtgärdsscriptskod som stoppar filmen för ramen med namnet ”start”.

- Spelstart

Vi skrev följande åtgärdsscriptkod för ”start”-ramen (se Figur 3:7).

```
on(keyPress "<Space>")
{
    play();
}
```

Figur 3:7: Actionscript för spelstart

- Styrningen av planet

Skapades genom att skriva åtgärdsscriptkod för flygplansobjektet. Som vi beskrev tidigare fungerade det här inte i Director eftersom musknappshändelserna inte registrerades någon annanstans än i det objektet där de var definierade. Men i Flash visade det sig att det gick att registrera mushändelserna överallt på scenen, även fast de var definierade i flygplansobjektet. Flygplanet är konverterat till en filmklippssymbol för att kunna skriva åtgärdsscriptkod för den, beskrivet i avsnitt 2.4.2. Åtgärdsscriptkoden ser ut som i Figur 3:8.

```
onClipEvent (mouseUp) {
    status = "ner";
}
onClipEvent (mouseDown) {
    status = "upp";
}
onClipEvent (enterFrame)
{
    if(move == "true")
    {
        if(status == "upp")
            _y -= 3;
        else
            _y += 4;
    }
}
```

Figur 3:8: Actionscript för manövrering av flygplanet

Funktionen onClipEvent läser av en händelse under hela objektets livslängd. Händelsen som ska läsas av är inparametern till funktionen, i de här fallen om musknappen är nedtryckt

eller släppt. Den globala variabeln ”status” sätts till värdet ”ner” om musknappen släppts och ”upp” om den är intryckt. När funktionen onClipEvent har inparametern ”enterFrame” så exekverar koden i dess funktionskropp varje gång en ny ram under objektets livstid har påbörjats. I funktionskroppen kontrolleras först om den globala variabeln ”move” har värdet ”true”. Den är ”true” om spelet startats och användaren inte fått ”game over”. Den här variabelns värde ändras om planet krockar med molnet, vattnet eller ballongen och förklaras under ”krock”. Efter det kontrolleras om musknappen är nedtryckt. Är den det minskas värdet för flygplanets y-koordinat med 3 pixlar, vilket medför att flygplanet stiger lite vertikalt på scenen. Är den inte nedtryckt ökas värdet för flygplanets y-koordinat med 4 pixlar, vilket medför att flygplanet sjunker lite vertikalt på scenen.

- Krock

Som vi sa när vi tidigare i det här avsnittet, när vi beskrev ”Krock” för Shockwave, så är det hela tiden flygplansobjektet som, genom att korsa (krocka) något av moln-, vatten- eller luftballongsobjekten kan ”se” om en krock har skett. Därför har vi även här skrivit åtgärdsscriptkoden för flygplansobjektet. Åtgärdsscriptkoden för flygplansobjektet visas i Figur 3:9.

```
onClipEvent (enterFrame)
{
	x = _root.plane._x - _root.balloon._x;
	y = _root.plane._y - _root.balloon._y;
	if( (x > -30 && x < 100) && (y > -50 && y < 70) || _y > 330 || _y < 30  )
	{
		_root.gameover._visible = TRUE;
		_root.stop();
		move = "false";
	}
	else
		move = "true";
}
on(keyPress "<Space>")
{
	if(move == "false")
	_root.gotoAndStop("start");
}
```

Figur 3:9: Actionscript för krock mellan objekt

I Flash finns ingen funktion som håller koll på om objekt träffar eller passerar varandra. För att göra samma sak i Flash var vi tvungna att kontrollera differensen mellan flygplanets och luftballongens x-koordinat och y-koordinat och sedan prova oss fram till hur liten differens vi kunde tillåta innan vi skulle anse att en krock inträffat.

Att filmen stoppas betyder inte att spelet stoppas utan spelet snarare gör en paus på den ramen som visas för tillfället och koden i scriptet exekveras färdigt. Eftersom koden i Flash inte läses igenom hela tiden som filmen är stoppad var vi tvungna att använda en flagga i form av en variabel för att tala om att en krock inträffat. Längre ned i koden i samma script har en händelsehanterare implementerats för händelsen att Space-knappen trycks ned. Skulle det inträffa och flaggans tillstånd visar att en krock inträffat hoppar filmen tillbaka till ramen med namnet "start" och stoppar filmen igen.

3.2.12 Interaktivitet: Sammanfattning

Styrningen av flygplanet skapades på olika sätt i de två programmen. I Flash skrev vi scriptkoden i flygplansobjektet, medan vi i Director skrev den i ”knappen”. Till att börja med fick vi ingen registrering av mushändelserna med ”knappen” i Director. Anledningen till det var att när vi skapade ”knappen” och egenskaperna för den blev den inte heltäckande, (täcker hela scenen) och genomskinlig, som vi hade avsett, utan det blev bara en kantlinje runt scenen. Länge trodde vi att det var fel på scriptkoden när inget hände med flygplanet, men när vi väl förstod orsaken till problemet var det lätt att ordna upp.

När vi gjorde scriptet för att planet skulle krocka med molnen använde vi i Director först funktionen för att kontrollera om ett objekt träffar ett annat objekt en bit in i objektet och inte alldeles på ytterkanten. Det här tyckte vi var snyggare för då försvinner flygplanet in i molnen en bit innan krocken sker. Problemet med det här var att det gick att flyga mellan två molnobjekt utan att utlösa en krockhändelse och så ville vi inte ha det. Därför använde vi samma lösning som i Flash, där flygplanets y-koordinat kontrolleras, beskrivet i avsnitt 3.2.9.

4 Jämförelse och rekommendation

Kapitlet är en redovisning av det arbete vi gjorde för att uppnå MÅL-4 och AP-4, beskrivet i Tabell 1:1 och Tabell 1:2.

För att komma fram till en rekommendation av vilket program/utvecklingsverktyg som Hög-Data bör använda sig av, för att vidareutveckla deras webbsida, har vi jämfört de erfarenheter och slutsatser som vi kommit fram till i de olika utvecklingsfaserna under den experimentella spelkonstruktionen. Vi har också jämfört några av de ingående verktygen där likheter funnits i båda programmen samt programmen i helhet, där vi sett till om någon av utvecklingsmiljöerna saknar ett lämpligt verktyg för en viss konstruktion. Vidare jämför vi olika kostnader som tjänsteförmedlaren, och i viss mån användaren, får i samband med utveckling, eller användning av Shockwave- eller Flash applikationer. Efter det sammanfattas alla jämförelser vi har gjort i en uppstaplad, mer direkt och sammanfattande jämförelse. Sist görs en rekommendation där vi ger våra slutgiltiga åsikter om vilket program vi tycker lämpar sig bäst.

4.1 Utgångspunkt

Den övergripande jämförelsen bygger på flera mindre jämförelser som vi gjort utifrån de erfarenheter och slutsatser vi dragit under arbetet med att beskriva delarna i kapitel två och tre. Att de på Hög-Data inte alls jobbat med programmen tidigare, eller i alla fall inte i någon större utsträckning, gjorde att vi tyckte det var lämpligt att ha med jämförelse för olika kostnader i samband med utveckling. Kostnader för inlärning och utveckling är ju sådana kostnader som är svåra att bedöma om ingen på företaget har någon större erfarenhet av programmen. Än svårare blir det då naturligtvis att bedöma vilken som är kostnadseffektivast av två så pass lika program som Director och Flash. En fingervisning om de här kostnaderna har vi fått i och med att vi själva inte hade några direkta förkunskaper i att utveckla multimedia i någon av utvecklingsmiljöerna.

Utgångspunkten för spelkonstruktionen, som jämförelsen bygger mycket på, har varit att hålla en så låg kostnad som möjligt. För konstruktionsarbetet är det framför allt systemresurer som vi kunnat påverka och då främst genom att så strikt som möjligt hålla oss till vektorgrafik. Kan utvecklaren också klara sig utan att använda några externa program för att

skapa grafik, animationer osv. som vi gjort, så kan kostnaderna för inköp och inlärning hållas nere.

De programversioner vi använt för den experimentella spelkonstruktionen är Director MX och Flash MX för Windows-plattform.

4.2 Funktionalitet

Flash och Director har liknande koncept för sina applikationer. Båda är interaktiva filmer organiserade i bildrutor som spelas upp i enlighet med definierade händelser och åtgärder. Utvecklingen av filmerna går till på ungefär samma sätt. Först skapas en filmfil där utvecklaren anger egenskaper som t.ex. scenens storlek, fps etc. och därefter skapar och/eller importerar utvecklaren element som sedan arrangeras på scenen och utefter en tidslinje. För interaktivitet och navigering lägger utvecklaren till script och slutligen publiceras filmen.

Skillnaderna på användningsområden som programmen riktar in sig på är, enligt företaget Macromedia själva, att du bör använda Flash när du vill skapa vektorbaserade animeringar, lättviktig video och funktionsrika Internet-tillämpningar för handburna enheter såväl som för webben. Director bör du använda när du vill skapa utökningsbart multimediainnehåll som använder video, inklusive avancerade 3D-spel och utbildningstillämpningar som kan implementeras på flera medier, som Internet, CD-ROM, DVD mm[23].

4.2.1 Utvecklingsmiljön

Det finns en del skillnader men också en hel del likheter i Flash och Directors utvecklingsmiljöer. Exempelvis har de båda en scen som fyller ungefär samma syfte i båda programmen. Enda skillnaden är att i Flash sker allt skapande direkt på scenen medan det i Director ofta sker i något annat fönster som t.ex. fönstren Paint, Vector Shape mm. I både Flash och Director har scenen samma namn, Stage, men det finns andra likheter i programmen som inte är lika uppenbara p.g.a. att de inte använder sig av samma terminologi för fönster/verktyg som används i samma syfte. Ett tydligt exempel på det är tidslinjen, i Flash, som kallas för Score i Director. I Director har istället en annan komponent som ingår i Score fått namnet Timeline, se nr: 3 i Figur 2:2. Vi kan inte riktigt identifiera oss med uttrycket Score utan tycker att Timeline passar bättre, kanske av den anledningen att andra program som vi kommit i kontakt med också har en tidslinje liknande dem i Director och Flash, där den också kallas just Timeline. Exempel på det är Adobe Premiere. Anledningen till att Director har de lite udda namnen Score, Cast etc. är att de ska passa in i den metafor som Director bygger sin utvecklingsmiljö på, se avsnitt 2.3.

Förutom själva namnet på tidslinjen så finns några skillnader. De utrymmen i Score där objekten är placerade kallas för kanaler i Director. De här kanalerna behöver inte skapas av utvecklaren utan de finns redan där och det kan bara finnas ett objekt i varje kanal vid samma tidpunkt. På det här viset fungerar kanalerna ungefär som olika lager som skiljer objekten åt för att utvecklaren ska komma åt dem enklare i efterhand. I Flash har problemet med att hålla isär olika objekt lösts just med lager. Men eftersom det går att skapa alla objekt i samma lager så är skillnaden mot Director att utvecklaren själv bestämmer när och om ett nytt lager behövs, och det finns ingen begränsning på antalet lager annat än när minnesutrymmet på datorn tar slut. Dessutom kan utvecklaren enkelt gruppera lagren så att de får en trädstruktur liknande principen för de flesta filhanterare. Det här illustreras av nr:4 "Water" i Figur 2.11. Utvecklaren *behöver* alltså inte skapa ett nytt lager för varje objekt, men det rekommenderas för att underlätta hanteringen av dem.

Det är lätt att slarva med att strukturera objekten, vilket ger Directors lösning en fördel eftersom det i viss mån görs automatiskt av programmet. Nackdelen är dock att utvecklaren själv inte får bestämma hur den strukturen ska se ut. I Flash finns inga förbestämda lager för specialeffekter som det gör i Director, utan där får utvecklaren skapa vanliga lager precis som vanligt. Det här gör att hanteringen av objekten automatiskt blir ganska bra i Director medan Flash istället ger utvecklaren en större valfrihet att själv strukturera objekten.

Den uppfattning vi fick när vi gjorde spelkonstruktionen var att hanteringen av objekten var något enklare i kanalerna i Score än i lagren i Tidslinjen. Vår uppfattning är dock att det här mycket väl kan förändras vid större projekt än vårt, där mycket fler lager/kanaler behövs och när man som utvecklare fått mer erfarenhet av att strukturera lagren. Men den erfarenhet som vi gjort, är att Director har en fördel vid den här hanteringen, som är en viktig del av animeringen i framför allt Flash.

Båda programmen har sina egna scriptspråk för att skapa interaktivitet, Lingo och ActionScript. Scriptspråken har en hel del skillnader, som beskrivs i avsnitt 4.2.3, men båda programmen har ett separat utvecklingsfönster, för att skriva kod i, med en del likheter. I Director heter det helt enkelt Scriptfönstret medan det i Flash kallas Åtgärdspanelen. När utvecklaren vill skriva in sin kod i Åtgärdspanelen så finns två val, normal mode och expert mode, beskrivna i avsnitt 2.4.2. En fördel med normal mode är att det förhindrar utvecklaren från att använda felaktig syntax, eftersom det inte går att skriva direkt i fönstret. Expert mode påminner mest om Directors Scriptfönster eftersom koden skrivs in direkt i panelen. I Scriptfönstret skriver utvecklaren alltid direkt i fönstret, beskrivet i avsnitt 2.3.1, men har också alltid tillgång till en ordlista som ger ungefär samma hjälp som menysystemet i normal

mode i Åtgärdspanelen. Scriptfönstret har alltså bara ett läge, med i stort sett en kombination av funktionaliteten i Åtgärdspanelens två lägen, som vi tycker räcker gott och väl. Det här är en fördel för Director eftersom vi tycker att det inte är nödvändigt att ha två olika lägen som utvecklaren måste växla mellan för att kunna utnyttja alla fördelar i scriptspråken. I Flash-litteraturen[7] rekommenderas just att växla mellan expert- och normal mode under utvecklingen.

För redigering av bitmapp-bilder i Flash finns bara några enklare verktyg medan Director har verktyg liknande dem i vanliga bildbehandlingsprogram. Saknaden över den funktionaliteten är dock inte speciellt stor eftersom bitmapp-bilder är bäst att undvika helt. Fotografier går det naturligtvis att importera och visa precis som i Director, men det går inte rita nya element som bitmappgrafik. Det görs istället med fördel i form av vektorgrafik.

Verktygspanelen i Flash har många användbara verktyg eftersom den är placerad i direkt anslutning till scenen där all utveckling i Flash sker. I Director däremot har verktygspaletten (Tool Palette) inte lika många verktyg. Det beror framför allt på att det mesta av utvecklingen sker i separata fönster med egna verktygspaletter, så utvecklaren behöver dem inte i anslutning till scenen. Att varje fönster, för att t.ex. rita, har egna verktyg för att rita samma figurer fast med olika grafikformat är en nackdel med Directors utvecklingsmiljö. Verktygspaletten borde kanske gälla för utvecklingen i alla fönster, där verktyg som är specifika för ett grafikformat visualiseras enbart när det fönstret är öppet. Fördelen med att all utveckling sker på scenen, som i Flash, är att alla verktyg finns nära och lättåtkomliga.

4.2.2 Utveckling

En nackdel för Director är att utvecklaren bara kan ångra en gjord handling åt gången. I Flash finns fler ångra steg än vad vi behövde någon gång under vår experimentella spelkonstruktion. De flesta program brukar åtminstone ge utvecklaren den möjligheten att själv ställa in antal ångra-steg, men hur vi än letade efter det menyvalet i Director fann vi det inte.

När det gäller att skapa grafik för multimedia som ska visas på webben är vektorgrafik att föredra framför bitmappgrafik, beskrivet i avsnitt 2.1.2. Tack vare den enkelhet som Flash verktyg medför för utvecklaren tycker vi att Flash är betydligt bättre än Director för det ändamålet. Vid riktigt avancerade grafiska kreationer räcker kanske inte Flash utvecklingsverktyg heller till, men är ändå betydligt bättre än Director, som i flera skrifter t.o.m. beskrivs som ett program där utvecklaren *bör* importera färdiga vektoriserade figurer skapade i fristående program.

Det var enklare att använda ritverktygen i Flash jämfört med i Director, men när vi skulle skapa animeringar av de ritade figurerna fick vi det motsatta förhållandet. Vi beskrev i avsnitt 2.4.2 att i Flash måste de ritade objekten konverteras till symboler för att gå att animera. Den här nackdelen finns inte i Director. Utvecklaren kan där animera objekten direkt med animeringsmekanismen, utan att behöva välja rätt ”typ” när objektet skapas. Proceduren för den enklaste formen av animation, att förflytta ett objekt från en position till en annan, under en viss tidsperiod, upplevde vi som mer omständig i Flash än i Director, eftersom den utgjorde fler steg. Under vår experimentella konstruktion lyckades vi inte heller alltid med att skapa en sådan animering vid första försöket i Flash, trots att vi enligt vår uppfattning följt ett korrekt förfarande. Orsaken var troligen att vi ändå inte lyckats få rätt ordningsföljd bland de många stegen.

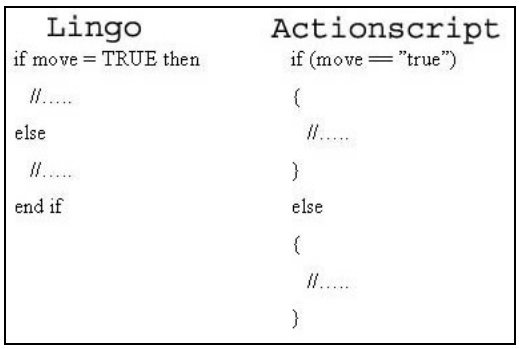
Den funktionalitet som särskiljer Flash och Director från många andra utvecklingsverktyg för webbsidor, som t.ex. Dreamweaver och FrontPage, är dess scriptspråk som tillför förmågan att skapa interaktivitet[6]. Här jämför vi enbart programmens förmåga att skapa den interaktiviteten och scriptspråken jämförs närmare i avsnitt 4.2.3. Det som vi kände som en nackdel med Director var att när vi skulle skriva scriptkod för att skapa interaktivitet var det svårt att veta vilken typ av script som skulle skapas. Det eftersom programmet har hela fyra olika sätt att skriva dem på beroende på vilket objekt som interaktiviteten ska gälla. De sätten beskrivs i avsnitt 2.3.2. I Flash finns det två sätt, som beskrivs i avsnitt 2.4.2, och det var aldrig några tveksamheter om vilket som skulle väljas.

Ljud är en viktig del av begreppet multimedia, därför är vårt misslyckande med att få ljud på ett riktigt sätt en stor nackdel för Flash, men vi hade heller ingen möjlighet att lägga ner lika mycket tid som vi gjort på andra delar i konstruktionen för att senare inte hamna i tidsnöd. Vi kanske inte kan göra någon rättvis bedömning av svårighetsgraden i att skapa händelseljud i Flash, men vi kan i alla fall påstå att det är enkelt i Director.

4.2.3 Scriptspråken

I kapitel 3 beskrev vi bl.a. hur vi skapade interaktivitet i den experimentella spelkonstruktionen. Vi visade flera exempel på scriptkod skrivet i Lingo och Actionscript. Det som syns och märks först och som är den tydligaste skillnaden mellan språken är den mellan deras syntax. Actionscript härstammar från Javascript[18] som i sin tur är besläktad med C++, vilket också syns på dess syntax. Vart Lingo härstammar ifrån är lite mer oklart. I sin födelse påminde det lite om programspråket BASIC[18], men mycket har förändrats sedan dess. Både Lingo och Actionscript är dock högnivåspråk. Ett högnivåspråks syntax påminner mer om ett

naturligt språk, om än ett aningen formellt och korthugget sådant. Lingosyntax är på en högre nivå än Actionscriptsyntax och Lingo liknar då mer ett naturligt språk än vad Actionscript gör. Figur 4:1 illustrerar detta.



Figur 4:1: Lingo- och Actionscript

Givetvis finns det fler jämförelser att göra språken emellan än bara deras syntax, men det är en hel rapport i sig att skriva om alla dem. En av de mest intressanta jämförelserna borde vara prestanda. Här ”vinner” Lingo[18]. Vi testade detta genom att skriva ett script i både Lingo och Actionscript, som hittar alla primtal mellan 1 och 1000. För Actionscript tog det 15 sekunder att hitta alla och för Lingo 0,33 sekunder. Alltså ungefär 45 gånger snabbare.

Den personliga åsikt som vi har om scriptspråken är att den syntax som används i ActionScript passar oss bättre eftersom den liknar de programspråk som vi använt mest och koden känns mer strukturerad när måsvingar används för att markera olika program-block.

4.3 Kostnad för tjänsteförmedlare och utvecklare

Med kostnad menar vi inte bara priset för att köpa in programmen, utan även den kostnad i tid det tar för utvecklaren att lära sig att utveckla något i programmen som är användbart eller lönsamt för företaget. Kostnad för ett företag är också den tid som utvecklaren måste lägga ner för att utveckla en produkt som företaget sedan kan ha nytta av, antingen direkt genom att produkten kan säljas eller mer långsiktigt i exempelvis reklamsammanhang. En annan kostnad är kraven på de systemresurser som behövs. Tabell 4:1 innehåller inköpspriset för fullversionen av programmen och hur många användare som har deras plug-in.

Program	Pris	Användare av plug-in	Datum
Flash MX	5136 SEK	525,356,058	1 maj 2003
Director MX	12 344 SEK	373,890,497	1 maj 2003

Tabell 4:1: Pris och statistik[19][24][25]

4.3.1 Inlärning

På webben finns det mycket material som kan underlätta inlärningen. Mest material hittade vi för Flash i form av tutorials, guider osv. Förklaringen till det här torde vara det höga priset för Director och att Flash helt inriktar sig på webblösningar. Director har å andra sidan fördelen av att ha ett mycket mer omfattande, inbyggt hjälpsystem än Flash.

Storleken och omfattningen på programmens utvecklingsmiljö skiljer sig en del åt. Vi beskriver i avsnitt 2.3 att det går att skapa en mängd olika sorters multimediaproduktioner med Director, medan Flash huvudsakligen riktar in sig på produktioner för webben, se avsnitt 2.4. I vår experimentella konstruktion skulle vi visserligen endast skapa en produktion för webben, men eftersom Directors utvecklingsmiljö innehåller fler verktyg så kändes den mer svårövergriplig, vilket försvårade inlärningen av Director. Därmed inte sagt att vi tycker att något utav programmen har en *dålig* utvecklingsmiljö. Tvärtom tycker vi att både Flashs och Directors utvecklingsmiljö är intuitiva.

Något av det första som nybörjaren stöter på i sin inlärning av de båda programmen är all terminologi. Director bygger nästan hela sin utvecklingsmiljö på en metafor. Delarna i miljön är regissörens verktyg (Cast, Score osv.). Det här underlättar förmodligen i ett tidigt stadie, men det dröjde inte länge innan vi tyckte det hela kändes lite krystat. Även Flash använder sig av denna terminologistruktur, men inte helt och hållet. Där blandas metafortermerna med andra kanske lite mer naturligare termer.

En annan sak som försvårade inlärningen var hanteringen av objekten i kanalerna respektive lagren (ändra storlek, förflytta mm.). Först provade vi oss fram i de båda programmen. Det visade sig då att i Director kunde vi sköta all hantering direkt, utan någon ytterligare hjälp. I Flash fastnade vi då och då, vilket medförde att vi fick söka hjälp i den inbyggda hjälpmanualen eller på webben.

Efter att ha övervägt de nackdelar som försvårade inlärningen och de fördelar som fanns för att underlätta den tycker vi att Flashs nackdelar var större och Directors mindre. Fördelarna programmen emellan var ungefär lika stora. Därför kommer vi till slutsatsen att Director hade en något lägre inlärningströskel, men att vi i slutet av konstruktionsarbetet ändå kände att vi kunde Flash nästan lika bra.

4.3.2 Utveckling

Vi har tidigare jämfört olika delar i utvecklingen och där även nämnt att vi i några fall inte lyckats med att skapa ett visst beteende eller egenskap i spelet med programmet Flash. Anledningen till att vi inte lyckades skapa ljud vid olika händelser i Flash berodde delvis på

att vi ville undvika att hamna i tidsnöd. Det innebär att vi måste anse det vara en nackdel, tidsmässigt, för utvecklingen i Flash eftersom det inte tog lång tid alls att hantera det i Director. Även objekthanteringen i tidslinjen i Flash var besvärligare än den i Directors Score, vilket ledde till att utvecklingen i Flash tog lite längre tid för att skapa animeringar. Det gällde framför allt i början av vår experimentella konstruktion och kan härledas till den något högre inlärningströskel som Flash har. Därför anser vi inte att skillnaden i tid för att skapa animeringar är på långt när så stor som skillnaden i tid för att skapa oregelbundna element i vektorgrafik, till Flash fördel. Det under förutsättning att inga externa program används för att skapa eller redigera de ingående elementen.

För att skapa interaktivitet kan vi inte säga att vi upplevde någon större skillnad i tid mellan Director och Flash trots flera skillnader både i utvecklingsfönstren för script och scriptspråkens syntax.

4.3.3 Systemresurser

”Flash är ett multimediaprogram som utvecklats speciellt för webben”, skrev vi i avsnitt 1.1. Något som de flesta som någon gång har skapat webbsidor eller andra webbproduktioner vet är att filstorleken är en väsentlig faktor. En webbsida med exempelvis för stora bilder gör kanske så att den som surfar in på sidan tröttnar på att vänta på att sidan ska laddas färdigt och lämnar den. Eftersom Director inte är skapat helt för webben, beskrevs i avsnitt 2.3, vilket Flash och dess utvecklingsverktyg är, faller det sig ganska naturligt att det som skapas i Flash har en mindre filstorlek än det som skapas i Director[22]. De filer som skapades i vår experimentella spelkonstruktion i Director och Flash tyder på samma sak. Den skapade filmen i Director för visning i Shockwave player har en filstorlek på 32 kB och den skapade filmen i Flash för visning i Flash player har en filstorlek på 28 kB. Nu kanske inte det här låter som någon större skillnad med tanke på hur små de här filerna är. Men konstruktionen/projektet i sig innehöll inte speciellt många objekt, så det är en skillnad till fördel för Flash.

Director kräver en maskin med en Pentium II processor och Flash en maskin med Pentium processor. Flash kräver en maskin med 64 MB RAM (internminne)[21] och Director en maskin med 128 MB RAM[20]. En liten nackdel för Director är att det har något högre hårdvarukrav än Flash.

4.4 Sammanfattning

Genom de jämförelser som vi gjort över programmens viktigare enskilda delar har vi kunnat skapa oss en helhetsbild av vilket program som vi uppfattat är det bästa utvecklingsverktyget

för multimediaproduktioner, för webben. Båda programmen har flera fördelar gentemot det andra programmet och de vi kommit fram till under våra jämförelser är framför allt de här. I texten nedan listar vi upp respektive programs fördelar, först Flash och sedan Director.

Fördelarna med Flash gentemot Director:

- Närheten till alla verktyg i utvecklingsmiljön, eftersom all utveckling sker i samma fönster.
- Terminologin i Flash är lite mer beskrivande för motsvarande verktyg/fönster.
- I utvecklingsarbetet är möjligheterna att själv strukturera tidslinjens lager för objekthantering en fördel för Flash.
- Vid utveckling i Flash kan utvecklaren enkelt göra om flera steg bakåt i processen genom att bara trycka på ångra-knappen.
- En mycket stor fördel för Flash är dess överlägsenhet när det gäller att skapa statisk grafik.
- För att skapa interaktivitet så är fördelen för Flash att det är lätt att veta vilken typ av script som ska skrivas.
- ActionScripts bättre strukturerade syntax föll oss, som programmerat en del C++ tidigare, bättre i smaken.
- Kostnadsmässigt är Flash en fördel, dels för att inköpspriset är lägre och dels för att utvecklingen går snabbare om mycket vektorgrafik ska skapas. Flash kräver också mindre systemresurser än Director.
- Fler användare har Flash player än Shockwave player på sina datorer och Flash player är betydligt mindre i filstorlek.

Fördelarna som Director har gentemot Flash:

- Den automatiska uppdelningen av objekt i olika kanaler är bra, framför allt innan utvecklaren förstått vikten av att ha en bra struktur över de olika objekten.
- Den enkla hanteringen av objekten i kanalerna utefter tidslinjen.
- Det tydligare och enklare förfarandet att skapa animationer av den statiska grafiken i Director, är en stor fördel.
- Egenskapsfönstrets visningsläge lista, som framför allt kan underlätta för utvecklaren vid skrivandet av Lingoscript.
- Att snabbt och enkelt lägga till händelseljud, något som vi inte lyckades med alls i Flash.

- Fönstret för att skriva scriptkod i Director tycker vi är något mer lätthanterligt än Flash motsvarighet, främst tack vare att all funktionalitet alltid finns tillgänglig.
- Scriptspråket Lingos bättre prestanda kan vara en fördel även om vi inte märkte av skillnaden direkt i de konstruerade spelen.
- Kostnadsmässigt är fördelen den att det går något fortare att lära sig grunderna för att utveckla i Director, tack vare dess bättre och mer lättförståeliga mekanism för att skapa animeringar. Director har också en bättre inbyggd hjälpmanual.
- En viktig fördel för Director är att vi faktiskt inte misslyckades med att utföra någon uppgift i den experimentella konstruktionen.
- Även om det ligger utanför ramen för vårt projekt så är en fördel med Director att programmet har en hel del funktionalitet för att utveckla för andra medier än just webben.

Vid en första anblick kan det tyckas att programmen är likvärdiga eftersom de har ungefär lika många fördelar. Det är dock inte fallet, eftersom några av fördelarna kommer från större skillnader mellan programmen än andra och några fördelar ligger på lite viktigare punkter. Det här gör att fördelarna slutligen väger över åt ett av programmen, se avsnitt 4.5. För utvecklingsmiljön väger de punkterna tyngst som har med programmets förmåga att skapa de element som är absolut nödvändiga även för den mest grundläggande multimedia-produktion. Däribland finns förmågan att skapa animation men också den kanske allra viktigaste förmågan, att skapa statisk grafik. Den kanske viktigaste fördelen bland kostnadsjämförelserna är förmågan att hålla storleken på filerna nere eftersom det är extra kritiskt vid webbproduktion.

Vi antydde i avsnitt 1.2 att det skulle gå att konstruera en fräck, intresseväckande och användarvänlig hemsida med hjälp av utvecklingsverktygen. Den biten som faller under utvecklingsverktygens ansvar för en sådan konstruktion anser vi inte är några större problem för varken Director eller Flash.

4.5 Rekommendation

Med bakgrund av den sammanfattande jämförelsen i avsnitt 4.4 har vi kommit fram till den rekommendation som Hög-Data förhoppningsvis kan få användning av.

Det program som har något bättre verktyg i sin utvecklingsmiljö, är kostnadseffektivast för tjänsteförmedlaren och utvecklaren och är minst systemresurskrävande vid utveckling och användning av multimediaproduktioner för webben, är Flash. Det som vägt tyngst är de olika kostnadsaspekterna och det faktum att Flash, mycket mer än Director, är ett program som det går att rita vektoriserad grafik i, på ett enkelt och intuitivt sätt, med ett gott resultat.

Skulle vi utveckla för något annat medium än webben som t.ex. CD-ROM, DVD etc. som inte är lika kritisk med avseende på storleken på filer och säkerhetsrisker så skulle valet bli lite svårare och troligen väga över till Directors fördel, särskilt om tjänsteförmedlaren/utvecklaren inte tycker att det är någon nackdel att använda andra utvecklingsverktyg för att skapa eller redigera element.

5 Sammanfattning

Ingen av oss kunde någonting om vare sig Shockwave eller Flash innan vi fick uppdraget av Hög-Data. Det enda vi visste var att de båda var plug-in till webbläsare för att visa upp multimedia på webbsidor och att företaget Macromedia stod bakom båda utvecklingsverktygen. Det här har inneburit att vi kunnat vara objektiva i vår bedömning av hur användarvänliga deras utvecklingsmiljöer är. De erfarenheter och iakttagelser som de olika utvecklingsfaserna under konstruktionsarbetet gav oss var viktiga för att kunna göra en övergripande jämförelse mellan programmen.

I den första fasen av vårt examensarbete skulle vi läsa in oss på ämnet och lära oss grunderna för hur verktygen i programmen används. Det var en bra mjukstart där vi gick igenom diverse lektioner (tutorials) och läste guider på nätet. Vi fann även några bra filmer på Macromedias hemsida, som visade några enkla exempel som vi testade. För att förkunskaperna skulle bli ungefär lika stora för båda programmen försökte vi finna likvärdiga guider etc. för dem. Förutom några guider på nätet och företaget Macromedias egna så gick vi igenom exempel ur bl.a. böckerna "The web wizard´s guide…"-serien som fanns för både Shockwave och Flash. När vi kände att vi hade provat tillräckligt många exempel började vi med nästa fas, den experimentella spelkonstruktionen.

Vi var lite osäkra på hur den experimentella spelkonstruktionen skulle fungera som grund till att lära oss programmen. Anledningen var att vi inte visste om vi i konstruktionen skulle använda tillräckligt mycket av programmens verktyg för att få en något så när heltäckande inlärning. Nu efteråt kan vi säga att det fungerade mycket bra. Vi fastnade flera gånger när vi använde vissa verktyg och det medförde att vi fick prova alternativa lösningar vilket innebar att vi hela tiden lärde oss nya saker. Alla delar, grafik, animation, interaktivitet, vi skapade i den experimentella spelkonstruktionen dokumenterades detaljerat direkt efter vi skapat dem. Vi är nöjda med att vi gjorde det, eftersom vi annars aldrig hade kommit ihåg alla detaljer när vi senare skrev kapitlet om den experimentella spelkonstruktionen.

Vi hittade en massa information och åsikter om programmen på webben. Åsikterna skiljde sig åt på flera punkter och det här var en anledning till att vi bestämde oss för att grunda vår jämförelse främst på våra egna erfarenheter. Från början var det tänkt att vi skulle göra en förbättrad version av spelet vi gjorde i den experimentella spelkonstruktionen. Anledningen till det var att vi ville att vårt examensarbete skulle vara en blandning av ett

konstruktionsarbete och ett utredningsarbete. Men under tidens gång märkte vi mer och mer att arbetet med att tydliggöra vad vi utfört i den experimentella spelkonstruktionen samt att göra jämförelsen så tydlig som möjligt i rapporten tog lite mer tid än vi beräknat. Det var också en tanke från Hög-Datas sida att konstruktionsarbetet inte skulle prioriteras lika högt som utredningsarbetet och att vi därför inte skulle försaka något i jämförelseutredningen för att hinna göra ett spel åt dem. När vi satte upp vår tidplan hade vi t.ex. inte tänkt på att kontrollera när det var helgdagar vilket gjorde att cirka en dags försening egentligen blev en vecka på tidplanen. Vi upptäckte det här relativt tidigt, så vi hade ev. kunnat jobba ikapp den tiden, men eftersom vi ändå inte hade avsatt så mycket tid i tidplanen till att färdigställa spelet så tyckte vi det var bättre att avstå från det och inte stressa igenom jämförelsen.

Vi nämnde tidigare att vi kom lite fel i tidplanen på slutet, men annars måste vi säga att den stämde förvånansvärt bra och det var bara de två veckor som vi tänkt lägga på att färdigställa spelet som vi istället lade på jämförelsen. Hur tidsåtgången i mantimmar fördelades mellan de olika arbetsmomenten beskrivs i Tabell 5:1.

	tim	**Beskrivning**
AP-1	50	Inläsning som vi gjorde för att lära oss grunderna i inledningen av examensarbetet
AP-2	130	Den experimentella konstruktionen tog den här tiden i ren konstruktion plus lite tid för dokumentation som gjordes under arbetets gång
AP-3	70	Parallellt med ovanstående arbetsmomenten gjorde vi en del datainsamling för att få material till delar av rapporten.
Dok.	130	Efter den experimentella spelkonstruktionen gjorde vi en beskrivning av programmen och utvecklingsmiljöerna samt viktiga moment vid utveckling.
Dok.	30	I det här skedet så behövde vi renskriva de tidigare anteckningarna över den experimentella konstruktionen, för att få en tydligare bild av de erfarenheter och iakttagelser vi gjort och som skulle utkristalliseras i jämförelsekapitlet.
AP-4 /Dok.	200	Här gjordes jämförelsen, och eftersom den nästan fullständigt bygger på tidigare moment så skulle den kunna hamna under dokumentation.
AP-5	0	Som vi nämnt i texten så uteslöts det här momentet för att säkerställa en så bra jämförelse och rapport som möjligt
S:a	610	Ovanstående timmar plus några timmar för planering av examensarbetet och iordningställande av arbetsplats gjorde att vi hamnade på den här summan.

Tabell 5:1: Tidsfördelningen mellan arbetspaketen

Referenser

[1] Funet, *Arpanet*, http://www.nic.funet.fi/index/FUNET/history/internet/se/arpanet.html, 2003-05-01

[2] Funet, *World Wide Web*, http://www.nic.funet.fi/index/FUNET/history/internet/se/www.html, 2003-05-01

[3] Metamatrix, *Experience the history of the web*, http://dejavu.org/, 2003-05-01

[4] Netscape, *Macromedia and Netscape combine technologies to bring true multimedia to the Internet*, http://wp.netscape.com/newsref/pr/newsrelease28.html, 2003-05-01

[5] Microsoft, *Microsoft and Macromedia Deliver Shockwave and ActiveX To Millions of Web Customers and Developers*, http://www.microsoft.com/presspass/press/1996/jun96/mmediapr.asp, 2003-05-01

[6] James G Lengel, *The web wizard´s guide to Shockwave*, Addison-Wesley´s, 2002

[7] Michael R. Kay, *The web wizard´s guide to Flash*, Addison-Wesley´s, 2002

[8] Bacon, Martin and Nyquist, *Director 7 and Lingo Bible*, IDG Books Worldwide, 1999

[9] Jesmond Lewis, *Director Tutorials*, http://www.herts.ac.uk/lis/mmedia/directortutorial/, 2003-05-05

[10] Stephen Wilson, *How-to Guide for Using Shockwave to Deliver Web Multimedia* http://userwww.sfsu.edu/~infoarts/technical/howto/wilson.shockwave.html, 2003-05-05

[11] Paula Sanders, *Review of Macromedia Director 8.5*, http://www.perpetualvisions.com/articles-and-graphics/review-macromedia-director85.htm, 2003-05-05

[12] Debbie Berg, *DEB Tip*, http://www.flawebdeb.com/augdebtip.htm, 2003-05-05

[13] Instructional Technology Lab, *Introduction to Flash*, http://www.uic.edu/depts/adn/seminars/flashintro, 2003-05-05

[14] Jonas Ahlberg, *Vad är Multimedia*, http://www.jonasweb.nu/multimedia/mum_vad.html, 2003-05-05

[15] Macromedia, *Programmet Director MX:s inbyggda hjälpmanual*, 2003-01-01

[16] Kristoffer Eidhagen, Jesper Ek, Anna Wetterström, *Flash 5 - Handboken*, Pagina, 2001

[17] Bo Berggren-Bergius, Georg Ryttman, *Multimedia med Director 8.0/8.5*, Studentlitteratur, 2002

[18] Gary Rosenzweig, *Comparing Lingo and Actionscript*, http://www.director-online.com/buildArticle.php?id=967, 2003-05-05

[19] Macromedia, *Köp från Macromedia*, http://www.macromedia.com/se/buy/, 2003-05-05

[20] Macromedia, *Director MX System reguirements*, http://www.macromedia.com/software/director/productinfo/sysreq/, 2003-05-05

[21] Macromedia, *Flash MX System requirements*, http://www.macromedia.com/software/flash/productinfo/systemreqs/, 2003-05-05

[22] Streaming Media World, *Flash and Shockwave, What's the Difference*, http://smw.internet.com/video/tutor/dirflash/index4.html, 2003-05-05

[23] Macromedia, *Resurscentral för Macromedia Flash och Director*, http://www.macromedia.com/se/software/director/resources/integration/, 2003-05-05

[24] Macromedia, *Worldwide users of Macromedia Shockwave Player*, http://www.macromedia.com/software/player_census/shockwaveplayer/, 2003-05-13

[25] Macromedia, *Worldwide users of Macromedia Flash Player*, http://www.macromedia.com/software/player_census/flashplayer/, 2003-05-13