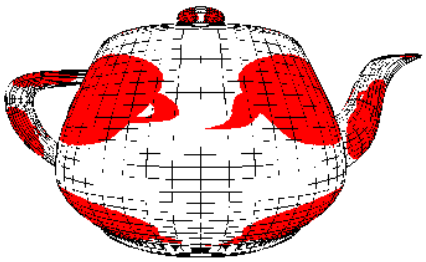
Rendering till textur

Vid t ex kub-mappning vore det bekvämt att kunna rita direkt till en textur och senare använda den som sådan. Det finns en ARB-utvidgning (begreppet beskrivs senare) *WGL_ARB_render_to_texture* som tillåter detta, som har stöd i Windows-miljö, men inte i Linux-miljö (jag gissar att det har med GLX att göra). Ev kan den senaste utvidgningen *ARB_pixel_buffer_object* (Dec 2004) hjälpa.

I figuren visas ändå en sådan effekt. Först har jag ritat upp en röd tekanna mot en textur. Texturen har sedan använts som textur för samma tekanna. För att vi verkligen skall se att föremålet är en tekanna har jag ritat kannan även som en trådmodell (p g a djupbuffertens arbetssätt syns inte alla delarna av trådarna (en lösning på det problemet beskrivs senare).



I Linux-miljö (RENDER_TEXTUR.c) kan effekten uppnås genom att vi ritar den röda tekannan i det osynliga bildminnet och rasterkopierar till en variabel, som sedan används som sista parameter i anropet av *glTexImage2D*. Anropen av *glReadPixels* och *glTexImage2D* kan kombineras till ett enda anrop `glCopyTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, 0, 0, X,Y, 0)`, där *X* och *Y* anger fönsterbredd och -höjd.

Animering

Syfte med dessa sidor: Att ge en liten inblick i ett jätteområde som dessutom är hårt knutet till kommersiell eller intern programvara. Hearn/Baker: kap 16, 584-597, Angel: 347-349(gamla), 435-437 (nya), Hill har mycket lite: 170-172. Möller: Inget.

Vad är animering?: Levandegöra. Förknippades ursprungligen med tecknad film. Eller dockfilm. Eller lerfilm. Nu datorframställd (Toy Story, A Bugs Life etc). Animeringen kan utspela sig i realtid eller vara inspelad. Vetenskaplig animering: någon process simuleras och visas grafiskt (såväl MATLAB som Mathematica klarar sådant). Animeringen kan ha interaktionsmöjligheter.

Många utmaningar: Fotorealism, stelkroppsrörelse, mänsklig rörelse, tygs interaktion med omgivningen, kollision, ansiktsuttryck, synkronisering av tal och munrörelse, natureffekter (eld, rök, moln etc).

Milstolpar: Disney (Musse Pigg 1928, Snövit 1933), Fleischer (Betty Boop 1930), östeuropeisk dockfilm, leranimering, datoranimering (Luxo jr 1986, Tin Toy 1988, Jurassic Park 1993, Toy Story 1995/Antz 1998/A Bugs Life 1998), BBC Dinosaurs 2000.

Datoranimering

Marknad: Reklamfilm, spelfilm, barnprogram.

Producenter av datorgenererad film: Pixar (Toy Story, Toy Story II), Pacific Data Images, Industrial Light & Magic (Jurassic Park), Aardman (Wallace & Gromit).

Exempel på programvara för animering: SoftImage, 3D Studio MAX, LightWave, Alias/Wavefront Maya, TrueSpace. Alla program för fotorealism brukar klara viss animering, gäller t ex POVRay (men modelleraren Moray ger inget eget stöd).

Vinst med datoranimering: 3D, kameraåkning (inkl korrekt perspektiv) lätt, bildritningen automatiseras, mellanbilderna kan genereras direkt, snabbare, billigare (?), mera detaljer kan tillåtas (texturer, massor av ljuskällor), mass-scener, effekter.

Resursbehov: 24 bilder per sekund, dvs 100 minuter film kräver 100x60x24 = 144000 bilder. Vid inspelad animering kan man lägga ner mycket tid per bild. Pixar uppgav 1996 1-10 tim. Med 1 tim/bild blir det en total datortid om 144000/24 = 6000 dygn, dvs 16.5 år. Inte konstigt att produktionen av Toy Story gjordes med ett kluster av ett hundratal datorer. Även minnesbehovet stort. Med 1 MB/bild (för litet för filmkvalitet men i överkant för video) blir det 144 GB.

Traditionell animering (tecknad film)

1. Skissat bildmanus (eng. story-board), en bild per scen, typiskt 2-10 sekunder.
2. Ljud
3. Bakgrund
4. De rörliga objekten ritas bild för bild på plastark (celluid), eng. cel animation[1]
5. Stel rörelse: Arket rörs över bakgrunden och avfotograferas.
   Normal rörelse: Arken byts successivt ut.

Man brukar säga att resultatet blir 2 1/2D (personerna är platta vid rörelse men kan finnas på olika djup). Med datoranimering blir det lättare att arbeta i 3D.

Bilderna var av två slag - huvudbilder (eller nyckelbilder; eng. key-frame) och mellanbilder (eng. in-betweens). De tillverkades av i huvudsak två grupper av animatörer med olika lön: chefsanimatörer och “inbetweeners”. Sedan fanns det en ännu lägre grupp som fyllde i färger etc. Huvudbilderna görs för ytterlighetslägen (t ex sänkt arm resp höjd).

Stop-motion animering

I vanlig film vill man ibland ha med levande varelser som inte finns (jätteapor, rymdvarelser, dinosaurier). En teknik som använts sedan 1900-talets början är stop-motion animering. Den innebär att en konstgjord ledad modell i naturlig eller förminskad storlek byggs och förses med passande kroppsvolym. Man tar sedan en bild i taget och ändrar på ett eller annat sätt i modellen mellan tagningarna. Ett känt exempel är King Kong (1933). Ett mindre känt Lost World (1925), som hade dinosauriescener. Dinosaurieinslagen i Jurassic Park (1993) var tänkta att göras med denna teknik och man byggde mekaniska fullskalemodeller. Men under arbetets gång fick datoranimering ersätta flera av scenerna. Dock användes rörelsen (styrd av stop-motion-animatörer) hos en del mekaniska modeller som indata till animeringsprogrammet

1. Cell animation är något annat. Se t ex `http://vcell.ndsu/nodak.edu/animations/home.htm`.

Disneys animeringsregler

Som belysning av att animering är en konst snarare än en teknik.

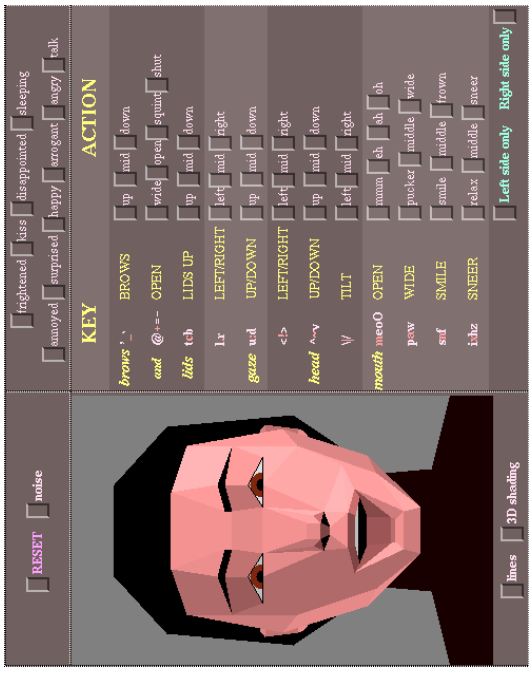
Hämtade från John Lasseter: Principles of Traditional Animation Applied to 3D Computer Animation, SIGGRAPH'87, där han exemplifierar med bl a sin egen film Luxo jr.

John Lasseter var animatör hos Walt Disney och gick senare till Pixar och gjorde en serie omtalade datorgenererade kortfilmer (bl a Luxo jr och Tin Toy). Regisserat Toy Story.

1. **Squash and Stretch** ("tryck ihop och sträck ut"): De flesta objekt deformeras under rörelse.
2. **Timing** ("timing"): Se till att saker och ting tar lagom tid, varken mer eller mindre.
3. **Anticipation** ("förväntan"): Varje handling föregås av en inledande fas. T ex om man skall ta ett föremål börjar man med att rikta blicken mot det.
4. **Staging** ("scensättning"): Se till att åskådaren har blicken där handlingen sker. Koncentrera handlingen till en sak i taget.
5. **Follow Through and Overlapping Action**: Handlingar har alltid en avslutande fas, t ex svänger en ben litet efter det att man sparkat iväg en boll.
6. **Straight Ahead Action and Pose-To-Pose Action**:
7. **Slow In and Out**:
8. **Arcs**: Rörelse längs krökta kurvor snarare än rätlinjigt.
9. **Exaggeration**: Överdriv utan att det blir orealistiskt så fattar åskådaren lättare.
10. **Appeal**: Gör det trevligt och nöjsamt.

Ansiktsanimering

Ett intressant område (eng. facial animation) med aktivitet. Även av intresse för enklare bildtelefoni (ansiktstextur överförs en gång!)

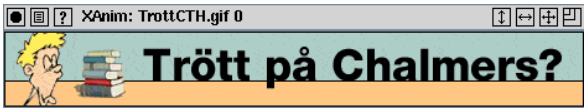


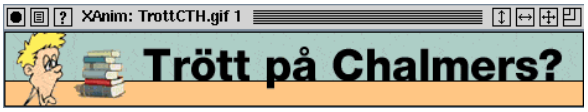
Bilden hämtad från `http://mrl.nyu.edu/~perlin/facedemo/`. Systemet skrivet helt i JavaScript och Java.

GIF-animering

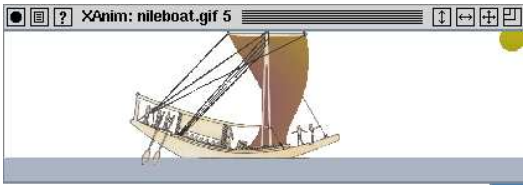
GIF är ett format för lagring av bilder. Det tillåter att man staplar ett antal bilder på varandra som återges en efter en. Används ofta för enkel webb-animering. Webbläsarna, t ex Mozilla, har inbyggt stöd. Fungerar bra för mindre bilder. Enkla medel kan ge påtaglig effekt.

Exempel: Blinkande person. Två inledningsbilder ur en sekvens.





Exempel: Seglande nilbåt



Flash-animering

Flash är också ett format för enklare grafikanimering via nätet. Läsaren Flash Player (insticksprogram till t ex Mozilla) är fri, medan produktionsprogrammet Flash kostar. Används på vissa av Chalmers officiella sidor. Borde finnas installerat tillsammans med Mozilla hos oss enligt StuDAT-specifikationen, men tycks inte göra det. Ute finns åtminstone version 7, men vi har vad jag kan se bara version 5 (Sun-miljön) respektive version 6 (2004 års Linux-miljön). I version 7 kan bl a ett C/Java-liknande skriptspråk användas, vilket gör att riktig 3D-grafik kan åstadkommas. SVG (Scalable Vector Graphics) är ett format, som jag tror har ambitionen att tävla med Flash. Kommer från WWW-konsortiet W3C. Vår Mozilla har stöd för SVG.

Exempel: Hoppande gubbe som rör sig mot en rörlig bakgrund.

Prova själv: Flytta dig till mappen `$DG/BILDER/FLASH`. Skriv `flashp6` (kollad 2005) och öppna någon av filerna `Cyber-Calculator-weltin.swf` och `Xmas-Elf-steele.swf` (ovanstående). Man får en del felutskrifter som beror på att jag inte installerat flashplayer enligt alla konstens regler. Filerna borde - men det går inte nu - också kunna betittas med en webbläsare.

Tidslinjeanimering

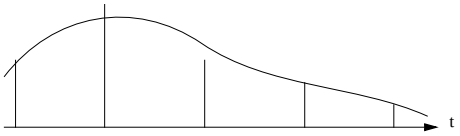
Låt oss anta att en scen består av N st objekt. Ett vanligt användarsnitt ser ut ungefär så här:

Objekt	Bild 1	Bild 2	Bild 3	
1				
2				
N				

I princip skulle rutorna innehålla aktuella data (positioner och ev hastigheter m m) för objekten och fyllas i manuellt. Men i praktiken manipuleras objekten i stället med t ex musens hjälp. Det är svårt att få naturlig rörelse.

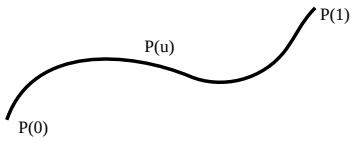
I de avancerade programmen finns mer sofistikerade sätt att styra rörelserna. Liksom vid traditionell animering används huvudbilder (borde väl nu heta huvudscener) för vilka fullständiga data finns (inkl hastigheter direkt eller indirekt). Variationerna modelleras med kurvor (B-splines eller motsv).

Tidslinje



Stelkroppsrörelse

I enklaste fallen handlar det om att flytta ett föremål längs en kurva eller rotera det kring någon punkt eller axel. Om kurvan är given på parameterform
P = P(u) , 0≤u≤1, med P(0) och P(1) givna

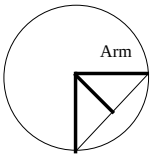


är det frestande att generera t ex 9 st mellanbilder baserade på positionerna P(0.1), P(0.2), ..., P(0.9) med tanken att rörelsen då blir jämn. Men det är inte självklart att parametern u har med kurvlängd att göra, varför någon form av reparametrisering kan behövas. Dessutom skall rörelsen i allmänhet inte vara jämn. Rörelsen skall kanske först accelerera från vila och på slutet bromsas. Detta kan naturligtvis i vissa fall vara inbyggt i parameterframställningen genom att u helt enkelt står för tiden.

Enkla fall: Fritt fall, en kanonkulas rörelse, en studsande boll.

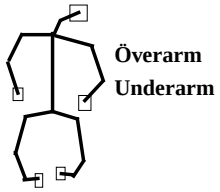
Objekt som deformeras: Om vi har en parameterframställning för deformation, så kan detta hanteras på samma sätt. T ex en boll som som krymper eller förvandlas till en ellipsoid (låt radien resp halvaxlarna vara parametrar).

Linjär interpolation: Är sällan det rätta sättet. Kan ge oönskade effekter, t ex en arm som sänks från horisontellt till vertikalt läge.

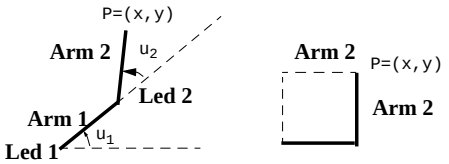


Framåtkinematik

Objektskelett: Levande varelser är inte stela objekt. Förenklat kan man se dem som uppbyggda av ben (kallade armar i figurerna), som är hopkopplade med leder och senare kläs med (ev deformerbara) volymer. För studiet och beskrivningen av rörelsen kan skelettet duga.



För att belysa problemets karaktär låt oss se på ett objekt med bara två leder och ben och med enbart två frihetsgrader (i verkligheten upp till sex frihetsgrader per led i 3D; tre rotationer och tre translationer). Totalt sägs en människokropp behöva 200 frihetsgrader.



Vid framåtkinematik utgår man från förändringar i den interna strukturen - i vårt fall de två vinklarna u_1 och u_2 - och ser hur dessa påverkar ett ändläge P=(x,y). Vi ser av högra figuren att det kan finnas flera arrangemang som ger samma ändläge. Om vi för enkelhets skull låter armarna ha samma längd:

$$x = L(\cos(u_1) + \cos(u_1 + u_2))$$
$$y = L(\sin(u_1) + \sin(u_1 + u_2))$$

Kinematik, invers kinematik

Vid invers kinematik utgår man i stället från en eller flera ändpunkter och vill bestämma de vinklar (etc) som behövs för att nå dessa. Det blir alltså fråga om att lösa ett högeligen olinjärt ekvationssystem F(**u**)=**x**, där nu **u** och **x** är vektorer. Det kan finnas 0, 1 eller flera lösningar.

I vårt exempel kan man med litet möda lösa ut u_1 och u_2 uttryckta i x och y, t ex

$$u_2 = \arccos\left(\frac{x^2 + y^2 - 2L^2}{2L^2}\right),\ u_1 = \arctan\left(\frac{-x\sin u_2 + y(1+\cos u_2)}{y\sin u_2 + x(1+\cos u_2)}\right)$$

men detta är givetvis inte möjligt i allmännare fall. I praktiken måste man använda någon approximativ metod, t ex Newtons metod (eller ta till något annat trick/approximation)

$$J(\mathbf{u}^{(k)})\mathbf{u}^{(k+1)} = J(\mathbf{u}^{(k)})\mathbf{u}^{(k)} + \mathbf{x} - F(\mathbf{u}^{(k)})$$

där J är Jacobimatrisen (även kallad funktionalmatrisen). För varje iterationssteg har man alltså att lösa ett linjärt ekvationssystem. Men det kan uppstå en del problem.

Invers kinematik är en stor sak även inom robotteknik.

Fysikbaserad kinematik

Kinematik är en ofullständig modell. Man kan göra en matematisk modell för objektet, som tar hänsyn till massan hos de olika delarna och friktion och tröghet i lederna, liksom ev pålagda krafter. Rörelsen styrs av lagen kraften=massanxaccelerationen. En fysikalisk princip säger att rörelsen sker så att arbetet är minimalt. Man förvandlar alltså rörelseproblemet till ett matematiskt optimeringsproblem innehållande en differentialekvation
mx”(t)=F(t).
Det är hanterbart.

Verklighetsbaserad kinematik

En tidig ide vid animering var att hämta rörelsedata från verkliga varelser. I början av 1900-talet uppfanns rotoskopet, som innebar att man ritade av förinspelat material. Denna teknik skall ha använts för animeringen av Snövit (men inte dvärgarna) i Disneys Snövit och de sju dvärgarna (1933). Senare har man använt sensorer placerade på ett antal strategiska ställen på en kropp. I Jurassic Park användes denna teknik i ett par fall, men man utgick från de mekaniska dockor som från början skulle använts genomgående. Dockorna användes alltså för rörelsen. Hud och liknande var även i dessa fall datorgenererad.

Datorgenererade objekt i reell värld eller vice versa

Handlar om att placera ett datorgenererat objekt i en en filmad värld eller en filmad varelse i en datorgenererad värld. Detta är huvudsakligen en filmteknisk fråga.

Exempel (`BILDER/GENEVA.mov`): En artificiell Marilyn Monroe vandrande utmed Geneve-sjön.

Morfing

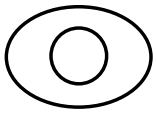
Problem: Låt ett objekt övergå i ett annat (formförändring). Effekten blir mest intressant när de två objekten är väsentligt olika, t ex en bil blir en tiger (se Hearn/Baker). Man kan tänka sig processen utspelad i 2D eller 3D. Effekten gjordes populär med filmer som Terminator II (199x) och Indiana Jones (199x). Kan ses som en form av animering.

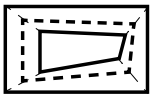
Parametrisk morfing: Om objekten kan beskrivas med en och samma ekvation men med olika parametervärden kan vi lätt åstadkomma övergången. T ex om en ellips

$$\frac{x^2}{a^2}+\frac{y^2}{b^2} = 1$$

Morfing, forts

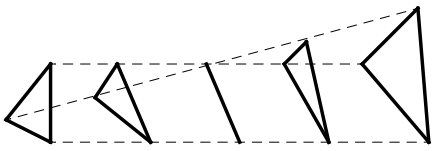
med a=3 och b=2 skall omformas till en cirkel med radien 1, så kan vi bilda en följd av ellipser genom linjär interpolation mellan a=3 och a=1, b=2 och b=1.

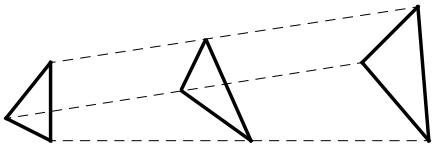




Liknande om en rektangel skall omformas till en fyrhörning

För att parametrisk morfing med linjär interpolation skall ha en chans att lyckas måste hörnen paras ihop lämpligt. Jfr följande två figurer med identiska start- och slutobjekt. I första figuren urartar förloppet, vilket är mindre lyckat.

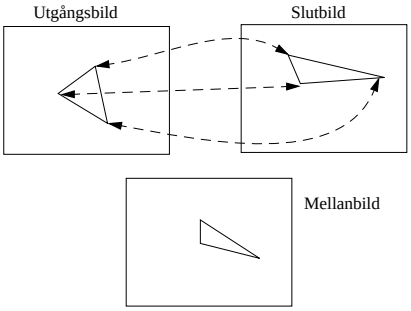




Bildmorfing

Låt oss se på ett specialfall där det gäller att få en bild att övergå i en annan under viss tidsperiod. Det enklaste sättet är att bara tona över från den ena bilden till den andra (eng. cross-dissolve, sv. övertona). Detta är ett gammalt trick i filmindustrin (jag kommer ihåg det från filmatiseringen av R.L. Stevensons Dr Jekyll och Mr Hyde med en ung Spencer Tracey). Men då framgår inte en eventuell formförändringen så väl.

Vi beskriver ett möjligt sätt. Om det gäller två ansikten så är idén att även om näsorna är olika placerade och stora så skall näsa övergå i näsa mjukt.



En mellanbild skapas så här:

1. Trianglarna interpoleras från motsvarande i utgångs- och slutbilderna.
2. Fyll i en punkt (x,y) = (u,v) i barycentriska koordinater (se kommande avsnitt om Beräkningsgeometri) enligt $I(u,v) = (1-t)I_{start}(u,v) + tI_{slut}(u,v)$

Vi får kontinuitet längs kanterna men inte mer. Professionella program arbetar annorlunda.

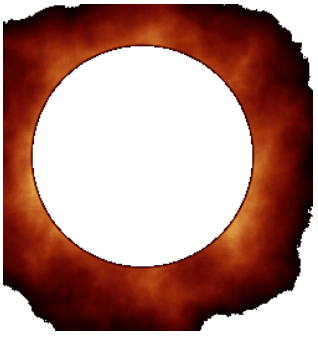
Tekniker vid återgivning i realtid

1. Dubbelbuffring är nödvändig.
2. Tidsstyrning. Förloppet skall gå lika fort oberoende av dator. Se avsnitt 17 i OpenGL-häftet.
3. Vid resursbrist hoppa över bilder eller sänk bildkvalitén.
4. Rasterkopiera i stället för att rita om en komplicerad bakgrund.
5. Använd sprite-teknik, dvs lägg små objekt ovanpå annat.

Effekter, naturfenomen

För detta finns ett stort antal tekniker. Här kommer början på en uppräkning.

1. Partikelsystem. Mer om detta separat.
2. Lösning med brus. I fallet Perlin-brus utökar man bara med en dimension för tiden, dvs 2D-fallet övergår i 3D och 3D i 4D. Knappast realtidsmetod. Bilden visar eldsflammor kring ett klot (svart i ursprungsfilen). Finns som `Flames500.gif` och körs med t ex någon webbläsare Hämtad från Perlins `www.noisemachine.com`..

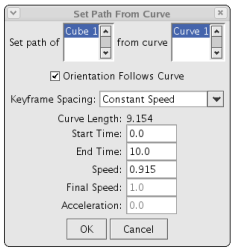


3. Plakat (bill-boarding). Tas upp senare.
4. Visa förloppet som en förinspelad GIF-animering (eller motsv).

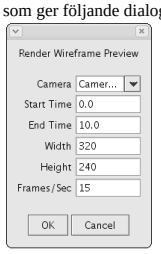
Animering i Art of Illusion 1(6)

Vi vill belysa hur datoranimering kan gå till praktiskt och väljer i år att arbeta i *Art Of Illusion*. För detaljer hänvisas du som vanligt till manual och handledningar (tutorials). I *AoI* finns tre huvudsätt: rörelsen definierad av bana, rörelsen definierad av ett antal situationer och rörelsen definierad av formler.

Exempel 1 (`OH_AOI1.aoi`): Vi börjar med att rita kurvan med något av de två kurv-verktygen (avsluta kurvan med ENTER). Sedan skapar vi en kub med kubverktyget (kan placeras var som helst; det är mittpunkten som kommer att följa kurvan). Det är praktiskt att reducera storleken på AOI-fönstret, så att ingen del döljs av annat, t ex systemmenyn.
Vi ser till att både kurva och kub är valda och väljer **Animation**/**Set Path from Curve**, som ger en dialog.

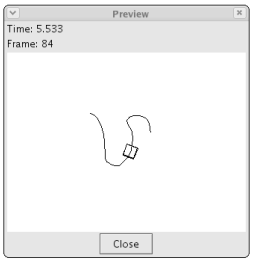


Vi ändrar **End Time** till 10.0. Trycker på **OK** och begär en förhandsvisning med **Animation**/**Preview Animation**, som ger följande dialog.

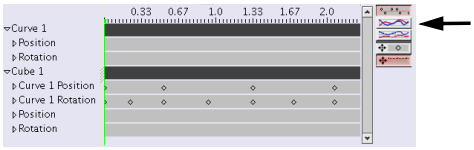


Vi trycker på **OK** efter ha granskat värdena. Detta gör att det dyker upp ett fönster som visar animationen.

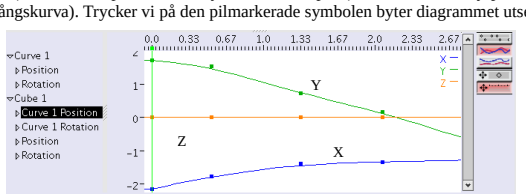
Animering i Art of Illusion 2(6)



Vi tar fram animeringens partitur (eng. score) med **Animation**/**Show Score**.



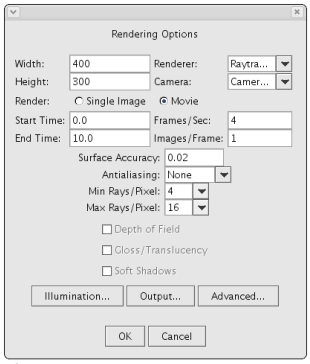
Överst finns en tidslinje med sekunder som enhet. Nedtill markerar små romber (eng. diamonds) vid vilka tidpunkter som nyckelbilder skapas (dessa motsvarar styrpunkter hos vår utgångskurva). Trycker vi på den pilmarkerade symbolen byter diagrammet utseende.



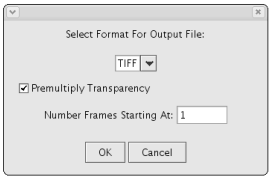
Kurvorna visar hur kubens position varierar med tiden (tidslinjen överst). Det finns en kurva för var och en av de tre koordinaterna X, Y och Z. Vi ser att kurvan för Z-koordinaten är z = 0, vilket naturligtvis beror på att vår bankurva är 2-dimensionell. Y avtar som väntat inledningsvis, medan X ökar svagt. Med den understa symbolen till höger kan vi flytta diagrammet. De fyrkantiga punkterna (som motsvarar styrpunkterna för den

Animering i Art of Illusion 3(6)

ursprungliga kurvan) fungerar som nyckelbilder. Dessa kan flyttas genom att vi väljer den näst understa symbolen till höger och sedan använder MK1 och drar. De kan flyttas i både tidsled (horisontellt) och rumsled (vertikalt). En flyttning kan i det här fallet leda till att vi hamnar utanför den avsedda banan. En punkt kan också redigeras genom att den markeras och man väljer **Animation**/**Edit Keyframe**. MK3 flyttar diagrammet. Kurvorna är lokala polynom (splines), dvs ändringar brer inte ut sig.
Nästa steg är att göra den slutliga animeringen, för vilket vi använder
Scene/**Render Scene**.

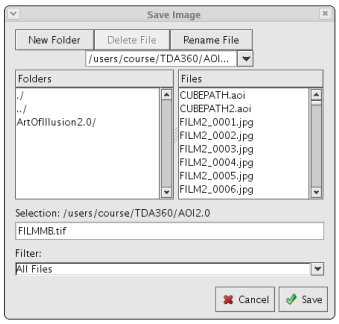


Vi markerar **Movie** och fyller i sluttid samt önskat antal bilder per sekund och får efter OK-et en ny fråga



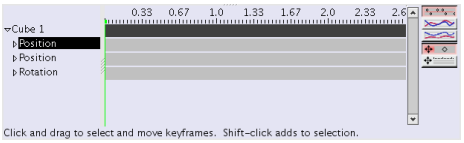
Animering i Art of Illusion 4(6)

Jag ändrar bildformatet till TIFF (PNG hade också gått bra). Sedan dyker det upp en fildialog.



Javas fildialoger brukar vara långsamma vid musklick. Själv använder jag ENTER-tangenten efter att gjort en markering.
Nu skapas ett antal bilder kallade `FILMMB0001.tif` till och med `FILMMB0040.tif` (alla med svart bakgrund även om jag väljer transparens). De får sedan fogas ihop till en film med något lämpligt program (se nedan).

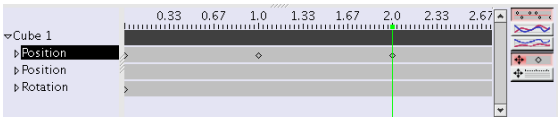
Exempel 2 (`OH_AOI1.aoi`): Återigen animerar vi en kub i rörelse. Denna gång genom att vi anger tre punkter för banan. Vi skapar en liten kub med kubverktyget och placerar den (med MK1) i övre vänstra delen av Front-fönstret. Ser till att kuben är vald. Med **Animation**/**Add Track To Selected Objects**/**Position**/**XYZ** (**One Track**) skapar vi ett spår (eng. track). Vi begär visning av spåret med **Animation**/**Show Score**.



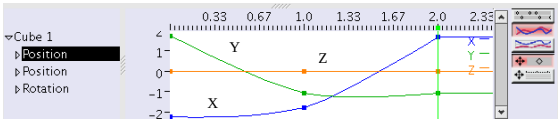
Överst syns tidslinjen. Tidslinjemarkören (ett grönt streck med en klump upptill) finns vid tiden 0. Vi vill stoppa in en nyckelbild vid denna tid, vilket görs med

Animering i Art of Illusion 5(6)

Animation/Keyframe Modified Tracks of Selected Objects. Denna nyckelbild utgörs av kuben i sitt utgångsläge. Vi vill också skapa nyckelbilder vid t = 1 och t = 2. Med MK1 flyttas markörhandtaget (det gäller att pricka det rätt) till t = 1. Vi väljer flyttningsverktyget högst upp i AoI-fönstret. Sedan flyttar vi med MK1 kuben nedåt i Front-fönstret. Sedan **Animation/Keyframe Modified** .... igen. Vi gör motsvarande för t = 2, men flyttar denna gång kuben åt höger.



Vi ser nyckelbildssymbolerna. Genom att klicka på den andra av symbolerna till höger får vi förloppet i form av kurvor.

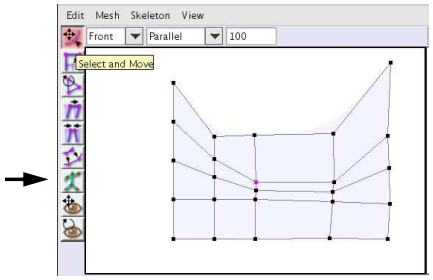


I Front-fönstret är x-axeln horisontell och y-axeln vertikal, som betyder att den inledande vertikala rörelsen minskar y och lämnar x (i stort sett) oförändrad, vilket bekräftas av kurvorna. Med **Animation/Preview Animation** kan vi som tidigare se hur animeringen tar sig ut: kuben faller under den första sekunden och drar sig sedan åt höger. Rörelsen avslutas vid t = 2, men animeringen fortsätter ytterligare 8 sekunder.

Kinematik i Art of Illusion 1(3)

Även för detta ett mycket enkelt exempel (`OH_AOI4.aoi`): som belyser arbetssättet. Realistiska exempel blir pillriga. Beträffande viss terminologi se tidigare OH. I *AoI* utgår man från ett splinenät (eng. spline mesh), till vilket man kopplar ett skelett (eng. skeleton). Skelettet består - oberoende av vilka benämningar som vi använt tidigare - av leder (eng. joint) (ibland kanske kallade knutar) och ben (eng. bone). Vi använder splineverktyget (mellersta i vänstra kolumnen) och skapar med MK1 ett standardnät i xy-planet. Påbörjar redigering av nätet med **Object/Edit Object**.

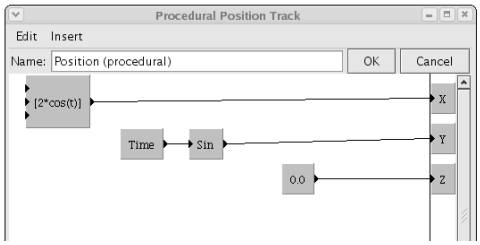
Jag flyttar med MK1 om styrpunkterna i Front-delen så att blir något i stil med



Nu skall skelettet in, vilket fixas med skelettverktyget på plats 3 nedifrån. Jag skapar ett ben genom att först markera nedre krysset (första leden) i figuren med CTRL-MK1 och sedan det övre skära krysset (andra leden) med CTRL-MK1. Nu ser det ut så här:

Animering i Art of Illusion 6(6)

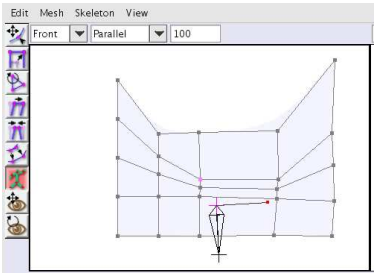
Exempel 3 (`OH_AOI3.aoi`): Det finns ett tredje sätt i *AoI* att beskriva en animering. Det utgår ifrån att man har en formel för rörelsen, vilket är vanligt i t ex simuleringssammanhang. Om vi vill att en sfär ska snurra i en elliptisk bana med halvaxlarna 2 resp 1 runt origo i xy-planet, kan vi göra som följer. Skapa en sfär och se till att den är vald. Välj **Animation/Add Track To Selected Objects**. Tag fram partituret med **Animation/Show Score**. I spåret se till att **Position (Procedural)** är vald. Vi vill redigera den valda egenskapen och tar till **Animation/Edit Track**. Redigeringen görs med samma blockteknik som vi använde för procedurtexturer. Efter redigeringen ser det ut så här.



Och vi kan betitta animeringen. Några kurvor i partiret tycks inte genereras när man arbetar med formler.

Vi har avsiktligt valt enkla objekt i dessa tre exempel och valt att arbeta med positioner. Man kan dessutom beskriva bl a rotationer.

Kinematik i Art of Illusion 2(3)



Vi ser ett rött handtag kopplat till den övre leden. Med det (MK1) kan vi vrida benet; än så länge utan att nätet påverkas. Den första punkten kommer att fungera som rot för vårt skelett.

Vi vill lägga till ytterligare ben. Ett ben läggs alltid till den skära punkten (man gör en punkt skär genom att klicka på den med MK1). Jag lägger en knut något högre upp med CTRL-MK1 och en till höger med CTRL-MK1. För att kunna lägga till den vänstra leden i figuren nedan gör jag först utgångspunkten skär med MK1 (om den inte redan är det).