Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Taru Liukkonen

Parametriset animaatiot peleissä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

2.11.2020

Tekijä Otsikko Sivumäärä Aika	Taru Liukkonen Parametriset animaatiot peleissä 53 sivua 2.11.2020
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikka
Ammatillinen pääaine	Pelisovellukset
Ohjaaja	Lehtori Antti Laiho

Insinöörityön tarkoituksena oli tuottaa erityyppisiä parametrisoituja kolmannen persoonan animaatioita työn tilaajana olleen pelistudion peliprojektiin. Pääasiallisesti animaatioita oli kolmea erityyppistä: tähtäyksen sarjojen luomista, elehtiviä emote-animaatioita ja elokuvamaisia esittäviä kohtauksia pelin alfa-traileriin. Traileria varten valmistettiin neljä kohtausta, kun taas emote-animaatioita ja tähtäyssarjoja tehtiin lukuisia. Jokainen emote-animaatio ja tähtäyssarja poikkesi muista huomattavasti, joten toistoa ei tapahtunut.

Animaatiot luotiin suurimmalta osin itse konfiguroimalla animaatio käsittelemättömästä liikkeentunnistusdatasta aina pelimoottorille käsiteltäväksi animaatioksi ja sen sarjoiksi saakka. Yritys tarjosi pitkälle jalostetun nivelistön kontrolleineen, mutta silloin tällöin animaatioiden toiminnallisuus vaati nivelistön hienovaraista säätämistä animaation tarpeisiin.

Jokainen animaatio mahdollisine sarjoineen konfiguroitiin pelimoottoriin itsenäisesti. Konfiguroinnissa käytettiin Lua-koodikieltä ja yrityksen oman pelimoottorin työkaluja ja asetuksia. Tämä työvaihe sisälsi itsenäistä koodausta ja pelimoottorin hallintaa. Jokaista animaatiota muokattiin ja hiottiin lukuisia kertoja, jolloin muutokset tehtiin 3D-ohjelmassa ja pelimoottoriin vietiin uusi animaatio korvaamaan vanhan.

Lähes kaikki opinnäytetyössä työstetyt animaatiot päätyivät peliprojektin suljettuun alphaversioon pelaajien käyttöön muutamia aliversioita lukuun ottamatta. Eniten tätä opinnäytetyötä varten tehtiin erilaisia tähtäyssarjoja ja emote-animaatioita. Suljettua alfaversiota varten tuotettiin myös muutama elokuvamainen kohtaus suljetun alfaversion traileria varten.

Avainsanat	3D-animaatio, pelianimaatio, animaatiojärjestelmä

Author Title Number of Pages Date	Taru Liukkonen Parametric animations in games 53 pages 2 November 2020
Degree	Bachelor of Engineering
Degree Programmer	Information and Communications Technology
Professional Major	Game Applications
Instructor	Antti Laiho, Senior Lecturer

The purpose of this thesis was to produce multiple kind of parametric animations for the gameproject the thesis was ordered for. There were to be primarily three kind of animations: aiming sets, emote animations and some cutscenes for the game's alpha trailer.

In most cases, the animations were created modifying raw motion capture data. The company offered a highly developed rig system for the animations, but occasionally the functionality of the rig needed the to be slightly modified for the purpose of the animation.

Configuration included using Lua-code language and the tools inside the game engine. This work stage included independent coding and game engine management. Each animation was modified and polished several times.

Most of the animations that were produced for this thesis got included in the final game project, and the feedback given by the users was positive.

In conclusion, the topic of the thesis proved to be the one anticipated. Parametric animation is wide concept, and there are not as much information available about it as there are about sequence based or performing game animations. Especially the history of the topic is still not generally recorded field, even though parametric animations are widely used type of animation.

Keywords	3D-animation, gameanimation, animation system

Sisällys

Lyhenteet ja termit

LERP	Linear interpolation. Matemaattinen teoria pisteen paikan löytämiselle janalla kahden pisteen välillä.
SLERP	Spherical linear interpolation. Lineaarinen interpolaatio 4-ulotteisen yksikköpallon pinnalla.
Avainkehys	Animaatioleikkeessä yhden tai useamman nivelen translaation ja kierron tiedon tallenne haluttuna ajanhetkenä.
Emote	Pelianimaatioissa tunteita ja elehdintää kuvaava animaatiotyyppi.
Fps	Ruutunopeuden toisto sekunnissa.
IK	Inverse kinematics. Hahmon nivelistössä valitut nivelet seuraavat asetettua niveltä.
FK	Suoraan kinemaattinen nivelistö. Hahmon nivelet ovat yhteydessä vain seuraaviin osuuksiin, jolloin jokainen nivel tulee asetella vuoron perään.
Alfa	Pelijulkaisun varhaisin, usein suljettu vaihe.
Early access	Pelijulkaisun vaihe, jossa pelaajat pääsevät pelaamaan peliä sen kehitysvaiheessa joutumatta maksamaan pelistä täyttä hintaa.
MMO	Massively Multiplayer Online Game. Suuri monen pelaajan verkkopeli.

1 Johdanto

Pelaaja luultavasti harvoin tulee ajatelleeksi, kuinka montaa animaatiotasoa hän kullakin hetkellä käyttää tehdessään pelissä tuiki tavallisia asioita. Pelaaja saattaa ohjata hahmonsa esimerkiksi lähelle kirjahyllyä asetta pidellen, ottaa siitä ehkä laatikollisen luoteja, laittaa sen taskuunsa, ja tämä toiminto saa ehkä hahmon sanomaan jotain. Edes pelaajan kääntyminen kirjahyllyltä ei keskeytä keskustelua, ja ehkä pelaaja hahmoa seuraavalle hyllylle ohjatessaan päättää ladata pistoolinsa ja vilkaista ovelle, ettei ketään ole tulossa yllättämään häntä takaapäin. Näin lyhyessä ajassa pelaajan hahmon läpi on kulkenut jo suuri määrä hyvin erilaisia ja erityyppisiä animaatioita, jotka ovat toimineet keskenään yhdessä koodin avulla luodakseen mahdollisimman interaktiivisen kokemuksen pelaajan ja hahmon sekä hahmon ja sen ympäristön välillä.

Nykypäivänä pelit ovat interaktiivisempia kuin koskaan, ja pelistudiot johdattelevat tarinankerrontaa ja interaktiivisuutta eteenpäin uusilla ja entistä kekseliäämmillä keinoilla myös animaatioissa. Tavoitteena on sekä hahmon että tarinan sulava eteneminen pelaajan käsissä, ja tätä varten on luotu erilaisia hyvinkin laajoja animaatiojärjestelmiä, joilla pelihahmon ja pelaajan vuorovaikutuksesta saadaan saumattomampi. Tällaiset järjestelmät ovat tuoneet mukanaan aivan uusia keinoja kertoa tarinaa ilman erikseen kuvattuja elokuvan kaltaisia kohtauksia ja moninaistaneet pelaajan mahdollisuuksia tehdä asiat pelissä, niin kuin hän itse haluaa, milloin hän haluaa.

Insinöörityössä tarkastellaan parametrisoituja animaatioita peleissä, sitä, mihin niitä käytetään ja miksi ja mitkä ovat ne järjestelmät taustalla, jotka saavat aikaan nykypäivän interaktiivisen hahmon, josta ennen vain haaveiltiin. Parametriset animaatiot ovat hyödyiltään ja käyttömahdollisuuksiltaan hyvin moninaiset, ja ne ratkaisevat monta ongelmaa nykypäivän pelien animaatioiden saralla. Insinöörityön tarkoituksena on luoda työn tilaajan eli Frozenbyte Oy:n pelihahmolle erityyppisiä elehtiviä emote-animaatioita sekä mekaanisempia tähtäyssarjoja erityyppisille aseille ja työkaluille jokaiselle animaatiotilalle. Tähtäyssarjojen tasojen on pystyttävä toimimaan yhdessä niin, että hahmolla voi tähdätä vapaasti annettujen raja-arvojen sisässä eikä animaatiotasojen muutoksia tulisi voida havaita herkästi. Kasvoanimaatioita tässä opinnäytetyössä ei käsitellä, koska projektissa pelattavat hahmot ovat robotteja ilman minkäänlaisia kasvoja tai kasvonpiirteitä.

2 Parametriset animaatiot

2.1 Pelianimaation historia

Ensimmäiset tietokoneanimaatiot saatiin aikaan 1960-luvun lopussa ja 1970-luvun alussa, ja ne loi joukko Yhdysvaltain Utahin yliopiston tutkijoita ja taiteilijoita. Yksi varhaisimmista uraauurtavista 3D-animaation havainnollistuksista oli saman yliopiston vuonna 1972 tuottama Ed Catmullin ja Frederic Parken videotallenne ”Hand/Face”. Siinä oikeasta kädestä tietokoneelle mallinnettu käsi kokeilee erilaisia liikkeitä, kuten osoittaa sormellaan kameraa, ja naisen kasvoja jäljittelevä polygoneista muodostettu pää jäljittelee muutoksia kasvoissa nauhoitetun puheen mukaan (kuva 1). [1.]

Kuva 1. Polygonimallinnetut naisen kasvot animaatioesityksessä Hand/Face (1972) [2].

Animaatio saatiin aikaan keräämällä kasvojen ilmeistä koottua polygonista dataa ja ketjuttaen sitten muutokset toisiinsa lineaarisen interpolaation avulla [1]. Digitalisaatio oli vasta alkuvaiheessa, joten tieto siirrettiin analogisesti ja animaatio on kuvattu nauhalle asettamalla filmikamera tietokoneruudun eteen. Vaikka tällainen havainnollistava kuvamateriaali on nykyajan näkökulmasta primitiivisen näköistä, se oli silloin paljon aikaansa edellä. [3.]

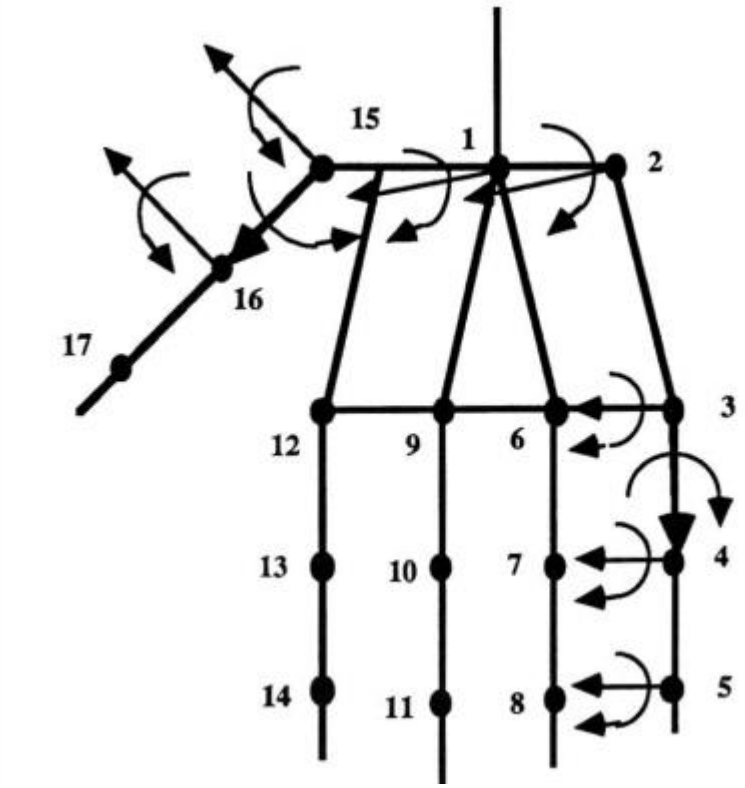
1970-luvun lopulla ensimmäiset kaupalliset yritykset tuoda 3D-animaatiota suurelle yleisölle tuottivat tulosta. Vuonna 1976 Ohion yliopistossa Tom DeFanti kehitti derivaattoihin pohjautuvan tietokonegrafiikkajärjestelmän, jonka tuloksia esiteltiin vuonna 1977 Starwars: Episode IV -elokuvassa. Kuvassa 2 nähdään elokuvan kohtaus, jossa C-3PO ja Chewbacca pelaavat peliä, jota voisi luonnehtia shakiksi 3D-hirviöillä. [1.]

Kuva 2. Star Wars Episode IV -elokuvassa nähdään kohtaus, jossa esitellään ensimmäistä kertaa tietokoneanimaatiota elokuvassa [3].

Tämä kohtaus sysäsi koko tietokoneanimaatioalaa liikkeelle, ja seuraavina vuosina tietokoneanimaatiota nähtiin jo useissa muissakin elokuvissa, sillä nyt sitä alettiin pitää uutena tulevaisuuden teknologiana, jolla saatiin aikaan asioita ja kohtauksia, joita perinteisin kuvauskeinon ei voitu tuottaa. [1.] Tämä oli innostavaa aikaa kaikille, jotka tutkimuslaitoksissa pohtivat, näkisikö tietokoneanimaatio koskaan kaupallista päivänvaloa [4].

1980-luvun alussa kehitettiin jälleen uusi tapa luoda 3D-animaatiota. Kokonaista objektia hallitsi parametri, ja sen arvoja muuttamalla mallia voitiin liikuttaa. Animaattori loi avainkehyksiä spesifioimalla sopivan sarjan parametrien arvoja annetussa ajassa. Avainkehysten välinen muutos laskettiin spline- tai kuutiointerpolaatiolla, ja näin saadut kuvat on jokainen yksitellen luotu interpoloiduista parametreista. Tätä tapaa kutsuttiin silloin parametriseksi avainkehysanimaatioksi tai avainmuutosanimaatioksi. [5; 6.]

Vuonna 1985 kehitettiin ensimmäiset nivelpohjaiset tavat, joilla liikuttaa hahmoa. Niissä 3D-mallinnetun polygonisen käden sisään asetettiin niin kutsuttuja niveliä kohtiin, joista sormet luonnollisimmin taipuisivat. Todellisuudessa nivelet ovat pisteitä 3D-ympäristössään ja niillä on selvä hierarkia ja tarkoin määritellyt rotaatiosuunnat (kuva 3). Näitä niveliä kiertämällä mallissa saatiin liikkumaan vain haluttu osa kerrallaan. [7.]



Kuva 3. Yksi varhaisimpia esitysmalleja vasemman käden nivelistön kulmille ja rotaatioille [6].

Tällainen nivelistö mahdollisti mallin osittaisen liikuttelun 3D-ympäristössään, kun aiemmissa tekniikoissa liikuttelu oli onnistunut hallitsemalla joko polynomeja tai kokonaista objektia kerrallaan. Nivelistö saattoi kattaa kokonaisen ihmistä jäljittelevän mallinnuksen, sormista silmien niveliin. Tämän idean kehittivät ja toteuttivat aikansa pioneerit Nadia Magnenat-Thalmann, Richard Laperriére ja Daniel Thalmann, ja tästä menettelytavasta on tullut hallitseva tapa liikuttaa lähes mitä tahansa objektia 3D-animaatiossa yhä tänä päivänäkin. [8.]

Algoritminen animaatiotekniikka kehitettiin hyödyntämään tätä uutta nivelteknologiaa. Sitä, kuten parametrista avainkehysanimaatiotakin, kutsuttiin useilla termeillä, koska ala

kehittyi jatkuvasti. Algoritmista animaatiotapaa kutsuttiin myös mallinnetuksi animaatioksi. [8.] Tällainen termistön vakiintumattomuus kertoo myös siitä, kuinka nopeasti ala kehittyi ja kuinka monta tutkimusta aiheesta julkaistiin hyvin lyhyen ajan sisällä.

Algoritmisessa animaatiotekniikassa objektien liike on esitetty listana arvoja, kuten siirtyminen ja kiertäminen. Jokainen muutos on parametrien ohjaama, ja nämä parametrit voivat muuttua riippuen annetusta fysiikan laista. Nivelten parametrien muuttujien ansiosta niveliä saateltiin nyt ohjata myös kinemaattisin tai dynaamisin laein. [5; 7.] Näistä pohjaa antavista keksinnöistä ja tekniikoista lopulta kehittyi kaikki tietokoneanimaatio.

Ensimmäinen 3D-polygoneja grafiikassaan käyttävä kaupalliseen tarkoitukseen tuotettu peli kehitettiin vuonna 1983 peliyhtiö Atarille (kuva 4) [9].

Kuva 4. I, Robot -pelissä pelaajan hahmo liikkuu palkeilla, tavoitteenaan väistellä esimerkiksi polygonipalloja ja ampua taustalla näkyvää kilpeä päästäkseen pelissä eteenpäin [10].

I, Robot oli Dave Threurerin kehittämä kolikkopeli, jossa koko pelin grafiikka oli kauttaaltaan ensimmäistä kertaa tasavarjostettuja 3D-polygoneja. Lisäksi se oli ensimmäinen peli, jossa pelaaja saattoi kontrolloida kameraa itsenäisesti. [9.]

3D-pelianimaatio kehittyi näistä alkuvaiheistaan tähän päivään asti, mutta tälle opinnäytetyölle keskeinen animaatiojärjestelmä kehitettiin vasta 2000-luvulla. Ennen kuin parametriset animaatiot kehitettiin, animaatiot peleissä olivat peräkkäisiä. Tämä tarkoitti, että vain yksi animaatio ohjasi hahmon liikettä ja seuraava animaatio saattoi alkaa vasta, kun edellinen oli päättynyt. [11.]

Esimerkiksi useat japanilaiset roolipelit, kuten Final Fantasy X (2001), pohjautuivat tällaiseen järjestelmään jakamalla vuorot taisteluissa pelaajan ja tekoälyn välille. Vuorossa sai käyttää yhden toimen, jota vastasi yksi animaatio, minkä jälkeen vuoro siirtyy yleensä viholliselle tai seuraavalle hahmolle (kuva 5).

Kuva 5. Final Fantasy X -pelin tyypillinen taistelukohtaus, jossa vuorot esitetään ruudun oikealla puolella nousevana pinona [12].

Tällainen tapa tuottaa animaatioita peliin tuo mukanaan tyypillisen tilanteen, jossa pelaaja ohjaa hahmon toimettomasta tilasta esimerkiksi juoksuun. Muutos johtuu pelaajasta, eikä animaattori peliä luodessaan pysty ennustamaan kohtaa, jossa muutos animaatiossa tapahtuu, joten siirtymät olivat usein kankeita ja hypähteleviä. [9.]

Vuonna 2007 pelistudio Valve kehitti ja julkaisi ensimmäisen persoonan ammuntapelin Team Fortress 2, jossa pelihahmoa ohjasivat ensimmäistä kertaa kokonaan, osittain tai häivytettynä koodin säätelemät animaatiot (kuva 6) [8].

Kuva 6. Pelistudio Valven Team Fortress 2 -pelissä pelihahmon animaatioita säädeltiin ensimmäistä kertaa kokonaan, osittain tai häivytetysti koodilla [14].

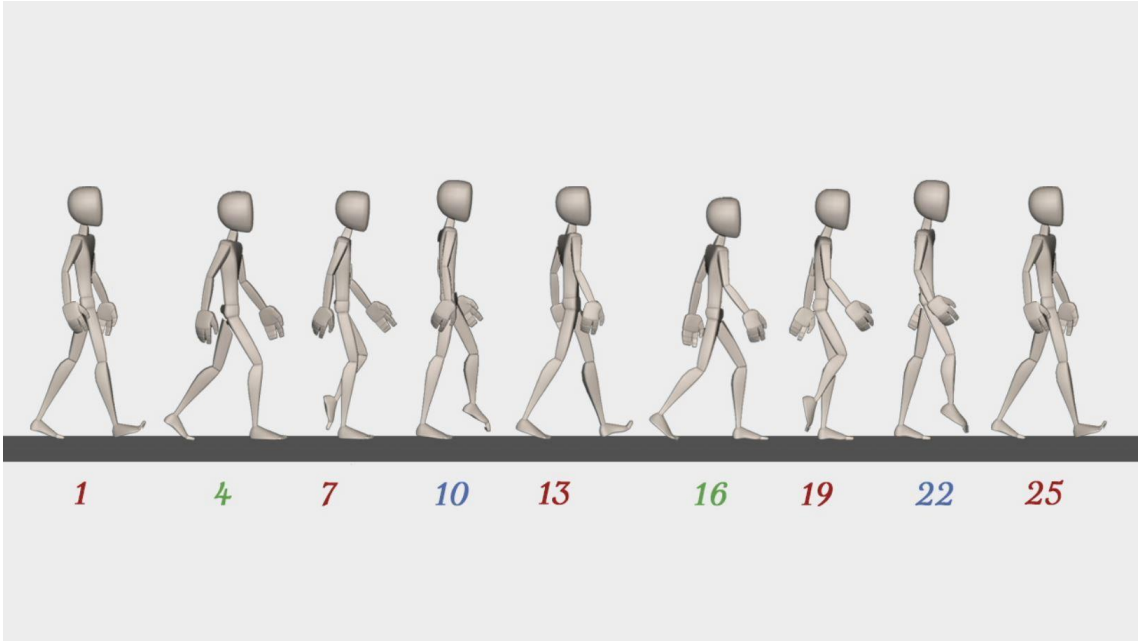
Tällainen animaatiojärjestelmä korjasi monta pelialan animaatio-ongelmaa kerralla, sillä nyt jokaisesta animaatiosta ei tarvinnut tallentaa montaa variaatiota. Riitti, että aseen latausanimaatiosta oli olemassa seisovan hahmon versio. Hahmon juostessa ja ladatessa asetta latausanimaation jalkojen osuus mykistettiin niin, että ne jatkoivat juoksuanimaation toistamista, kun ylävartalosta tarvittava osa toisti latausanimaatiota, jonka lisäksi vartaloon häivytettiin juoksuanimaation hytkyvää liikettä tietty prosentuaalinen osuus. Tällaista juoksevan latauksen animaatiota ei ollut tallennettu erikseen peliin, vaan animaation tilalle oli annettu koodissa ehdot ja tiettyjen animaatioiden osuudet, ja lopputulos oli tarkkaan määritelty sekoitus halutuista animaatioista. Tällainen järjestelmä paitsi vapautti paljon muistia, myös korosti hahmon ja pelaajan välistä vuorovaikutusta.

2.2 Perusanimaatiotyypit peleissä

Tässä luvussa esitellään perusanimaatiotyyppejä peleissä 3D-hahmolle. Luvussa esiteltäviä animaatiotyyppejä parametrisoidaan nykyään useimmissa tilanteissa jollain tavalla, mutta ennen parametrisoitujen animaatioiden kehittämistä nämä olivat pääasialliset tavat, joilla pelihahmoa animoitiin.

Syklit

Syklit (cycles) ovat yleisimmin käytetty animaatiomuoto peleissä. Niissä animaation ensimmäisen ja viimeisen avainkehyksen tulee olla identtiset, jotta saavutettaisiin näennäisesti saumaton animaatiosilmukka. Sitä voidaan toistaa loputtomasti, ja sen tulisi näyttää mahdollisimman luonnolliselta toistuessaan. Animaatio ei katkea, ennen kuin toinen animaatio keskeyttää sen. Tyypillisin esimerkki idle, eli paikoillaan toimettomana seisominen, löytyy lähes jokaisesta pelistä. Yhtä yleiset syklit ovat esimerkiksi kävely- ja juoksuanimaatiot. Niitä voidaan toistaa peräkkäin loputtomasti ja animaatio näyttää silti yhtenäiseltä (kuva 7). [15.]

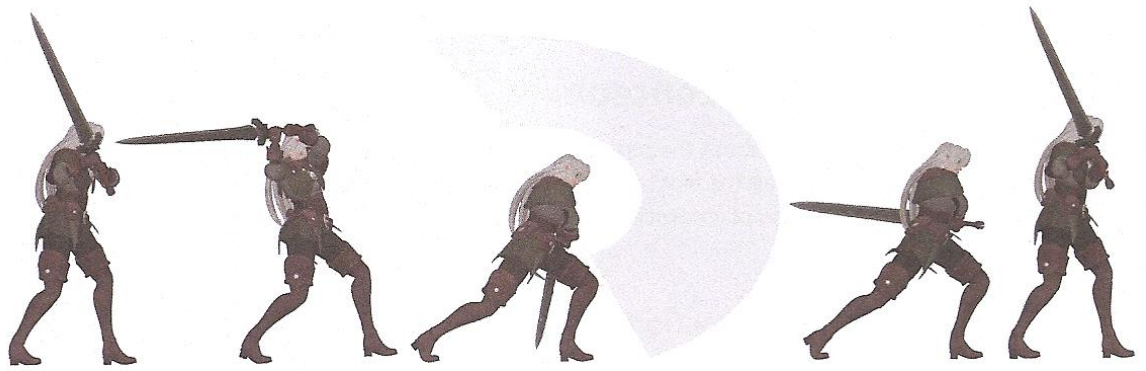


Kuva 7. Tyypillinen kävelyn animaatiosykli 3D-hahmolle [16].

Hyvä pelianimaatiosykli toimii niin, ettei pelaaja edes välttämättä huomaa sen tapahtuvan. Tämä vaatii sen, että animaatiokäyrien tangenttien tulee animaation alussa ja lopussa olla identtiset ja animaation on oltava yhtenäinen nopeudeltaan. Nopeita ja muusta animaatiosta erottuvia liikkeitä tulee myös välttää, sillä ne saattavat herkästi rikkoa pelaajan illuusion hengittävästä ja elävästä hahmosta, jos syklin yksi liike selvästi eroaa muista ja antaa animaatioon selvästi erottuvan kohdan. [15.]

Lineaariset toiminnot

Lineaarinen toiminto (linear actions) johtuu usein pelaajan antamasta käskystä. Sillä on selvästi muista animaatioista erottuva alku ja loppu, eikä sen pehmennyksistä tulisi välittää liikaa, vaan animaation tulisi keskittyä enemmän mielenkiintoisiin siirtymiin ja voimannäyttöön. Tämä johtuu siitä, että tällaiset animaatiot on usein sidottu kiinni ohjaimen tärkeimpiin nappuloihin, ja näin ollen pelaaja on sitoutunut ajatukseen, että liike tapahtuu usein suoraan eteenpäin eikä sitä voi keskeyttää, ja tiedostanut, että animaatio usein poikkeaa edellisestä reilusti ja palaa joskus jopa uuteen animaatiotilaan (kuva 8). [15.]



Kuva 8. Tyypillinen pelihahmon hyökkäysanimaatio sivulta [15].

Tällaisia animaatioita ovat esimerkiksi hyppy- ja hyökkäysanimaatiot, ja niiden painopiste tulisi olla pikemminkin selkeissä ja nopeissa siirtymisissä liikkeeseen ja siitä ulos, jotta vuorovaikutus hahmon ja pelaajan välillä säilyisi hyvänä [15].

Siirtymät

Siirtymät (transitions) ovat usein pelimekaniikallisesti tarpeettomia ja ylimääräisiä animaatioita. Ne luovat silti uudenlaista ilmettä ja energiaa hahmojen siirtymisiin animaatioista toiseen ja antavat animaattoreille taiteellista vapautta, kun halutaan näyttää, miten hahmo siirtyy liikkeestä toiseen. [15.]

Siirtymät ovat usein lyhyitä animaatioita, joissa jalat pidetään pääosin maassa, ja niissä halutaan esittää kunnollista painon vaihtoa esimerkiksi jalalta toiselle. Tyypillisesti peleissä pelaajan annettua käskyn liikkumisesta idle keskeytyy ja juoksu alkaa. Siirtymät tarjoavat tähän väliin lyhyen animaation, jossa hahmo valmistellaan juoksuun esimerkiksi kallistamalla hahmoa kohti annettua suuntaa, valmiina nostamaan nopeuttaan. Mikäli näin tehdään, voidaan samalla tehdä transitio jokaiseen suuntaan (yleensä 45 asteen välein). Tämä luonnollisesti kasvattaa animaatioiden määrää tuntuvasti, ja yleensä vain suurimpien budjettien peleillä on varaa lisätä hahmon liikkuvuuden sulavuutta näin. [15.]

2.3 Parametristen animaatioiden toteutustavat

Sana ”parametrinen” viittaa matematiikassa ja tietotekniikassa funktioon eli toiminnallisuuteen, jolle välitetään tietoa eli ns. argumentteja, jotka säätelevät funktion tuloksia.

Parametrinen animaatio peleissä nykypäivänä tarkoittaa animaatiota, jossa alkuperäiseen animaatioon lisätään yksi tai useampia animaatioita, ja lopputulos on näiden yhdistelmä. Yhdistelmän lopputulos on aina haluttu, ja tarkoin suunniteltu ja koodattu. Parametriset animaatiot lisäävät vuorovaikutusta pelihahmon ja pelaajan välillä ja poistavat rajoja pelaajan ja virtuaalisen maailman väliltä.

Kun kaikista perinteisin animointitapa pelihahmolla on ollut toistaa yksi animaatio kerrallaan ja sitten mahdollisesti häivyttää animaatio seuraavaan tarvittaessa, useimmat pelit nykypäivänä toistavat useita animaatiota samanaikaisesti tai johtavat yhteen animaatioon liukuvia arvoja, kuten nopeuden muutosta.

Monen animaation samanaikainen häivyttäminen yhteen on edellytys, kun halutaan jatkuva illuusio elävästä ja luonnollisesti liikkuvasta hahmosta ja vetäytyä pois animaatioista, joissa yksi animaatio johtaa koko hahmoa. Nykypäivänä elävän ja hengittävän pelihahmon takana pyörii monimutkainen järjestelmä erilaisia animaatioita ja paljon koodia, jotta haluttu hienovarainen sulavuus pelihahmon animaatioissa voidaan saavuttaa. [15.]

Pääasiallinen syy useiden animaatioiden häivyttämiseen yhdeksi on halu suorittaa monia erilaisia toimia pelissä, riippuen annetuista parametreista. Esimerkiksi jos havaitaan, että pelihahmo kulkee mäkeä ylös tai alas, voidaan hahmon jalkojen liikkumiseen asettaa versio, joka on nimenomaan luotu ylämäkeen astelulle kävellen tai juosten. Tämä luo illuusion, jossa hahmo olisi tietoinen ympäristöstään ja reagoi sen mukaisesti eikä töyssy tai kuoppa maassa juuri vaikuta sen etenemiseen ympäristössään. Tämä juurruttaa pelihahmoa tiukemmin pelin maailmaan. [15.]

Muita yleisiä parametrejä voi olla esimerkiksi sarja vaikuttavia häivytysarvoja monien animaatioiden välillä, kuten esimerkiksi juoksemisen nopeus tai hypyn korkeus. Mitä nopeampi juoksu, sitä tiukempi kääntyminen tai pidempi ja korkeampi hyppy. [15.] Tätä kaikkea ohjailee koodi hahmon takana. Pääasiallinen tavoite on tehdä hahmon liikkeistä realistisempia, ja nykypäivänä tällaista säädeltyä animaatiojärjestelmää pidetään usein paljon kauniimpana kuin perinteisiä tyylejä toistaa yksi animaatio kerrallaan ilman minkäänlaisia parametrejä.

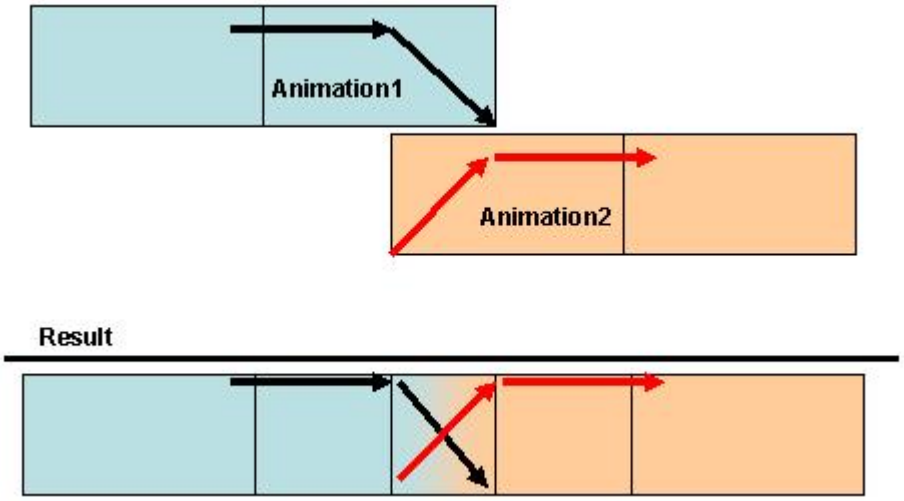
Nykyajan peleissä pelkästään animaatioilla, joita yksinkertaisesti vain häivytetään toisiinsa, ei enää saavuteta nykypäivän vaatimuksia sulavasti liikkuvasta, luonnollisesta ja ympäristöönsä reagoivasta hahmosta. Esimerkiksi ammuskelupeleissä, joissa jokainen sekunti merkitsee, hahmojen on vastattava heti annettuun käskyyn. Tämä pätee myös vaikka edellinen käsky, usein liikkuminen osoitettuun suuntaan, olisi yhä voimassa. Pelin tyylistä riippuen hahmo tekee usein monta asiaa kerralla pelaajan ohjaamana. Se saattaa esimerkiksi juosta ja ladata asetta samalla, kun seuraa katseellaan seuraava ammuttavaa kohdetta, ja hypätä ilmaan kesken latauksen.

Pelaajan ja hahmon välistä vuorovaikutusta on luotu helpottamaan joukko parametrisoituja animaatioita, joiden tyypit ja käyttötarkoitukset ovat varsin vaihtelevat. Niitä voidaan toistaa päällekkäin, mutta kaikille niille on yhteistä, ettei pelaaja pysty itse toistamaan

niitä. Tämä johtuu siitä, että ne on luotu pikemminkin helpottamaan ja tapauskohtaisesti nopeuttamaan pelaajan ja hahmon välistä vuorovaikuttamista ja joissain tapauksissa viemään tarinankerrontaa eteenpäin esimerkiksi animaatioilla, jotka reagoivat esimerkiksi pelimaailman sen hetkiseen säähän tai tilanteeseen.

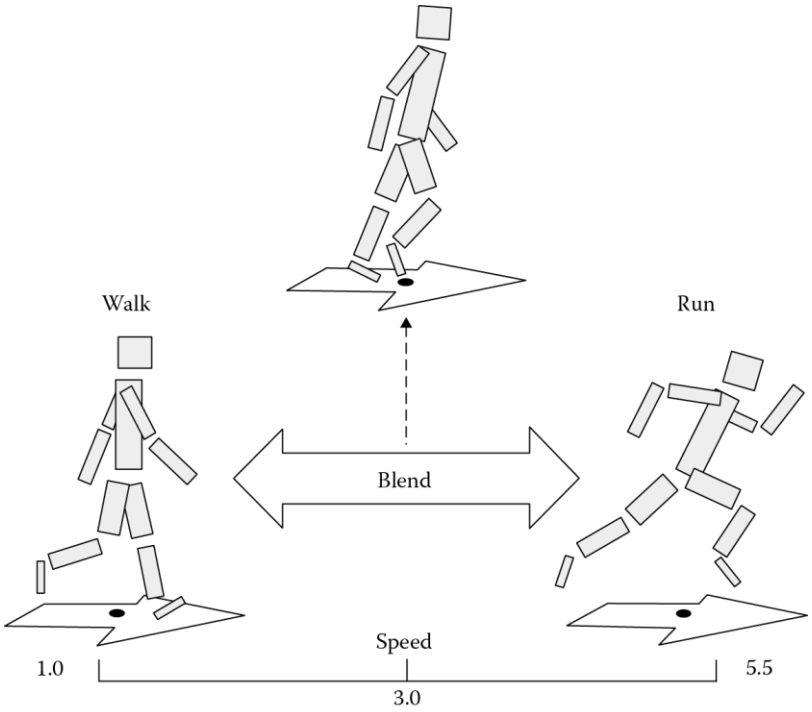
Ristihäivyttäminen

Tavallinen ristihäivytys (cross blending) saadaan pelissä aikaan lineaarisella interpolaatiolla. Ristihäivyttäminen on yleensä kahden animaation hetkellinen yhdistelmä, jossa alkuperäinen animaatio sulautuu toiseen ja häilyy pois ja jäljelle jää animaatio, johon alkuperäistä yhdistettiin (kuva 9). [15.]



Kuva 9. Ristihäivytyksessä kaksi animaatiota yhdistyy hetkellisesti, kunnes alkuperäinen on häivyttynyt kokonaan pois [17].

Yleisimmin tällaista ristihäivytystä käytetään peleissä, kun halutaan nostaa hahmon nopeutta kävelystä juoksuun. Kuvassa 10 kuvataan hahmon nopeudenmuutosta ja sitä, kuinka muutoksen keskellä hahmon tulisi vaikuttaa siltä, kuin se hölkkäisi. [15; 18.]

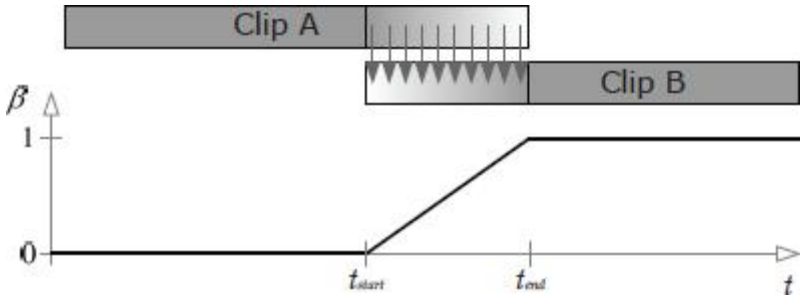


Kuva 10. Yksinkertainen esimerkki kahden animaation hetkellisestä sulautumisesta toisiinsa [18].

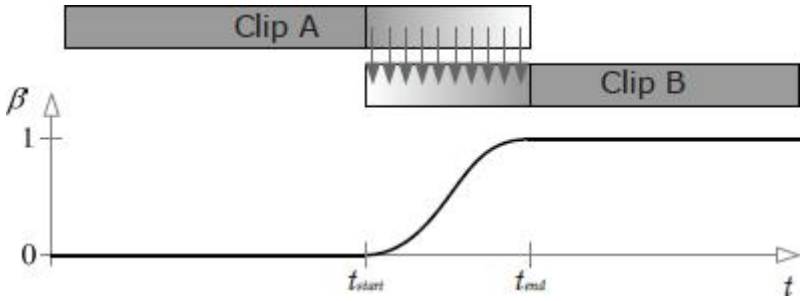
Jotta tällainen ristihäivytys näyttäisi mahdollisimman sulavalta, tulisi molempien animaatioiden olla täysin syklisiä ja karkeasti samankaltaisia käsien ja jalkojen asentojen osalta. Jos näin ei ole, ristihäivytys usein näyttää luonnottomalta [19].

Pehmeä muutos

Pehmeä muutos (smooth transition) on ristihäivyttämisen edistyneempi muoto, jossa ristihäivytystä pyritään pehmentämään ennestään. Kuvassa 11 nähdään, että koska tavallinen ristihäivytys käyttää interpoloimiseen lineaarista interpolaatiota, on muutos animaatioissa suoraviivainen eikä pehmennä animaation muutosta lainkaan. Kuvassa 12 interpolaatiota on muunneltu, jolloin muutoskäyrä etenee pehmeämmässä kaaressa.



Kuva 11. Lineaarinen interpolaatio ei tarjoa pehmennystä animaatiomuutokseen [19].

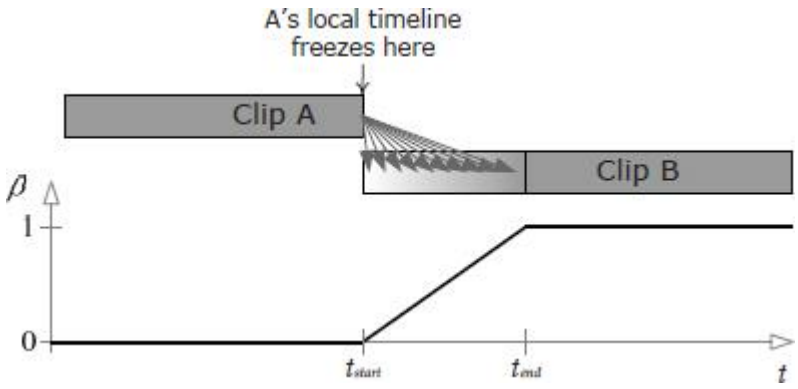


Kuva 12. Kun ristihäivytyksestä halutaan luoda pehmeämpi versio, voidaan käyttää Bezier-interpolaatiota [19].

Pehmeässä muutoksessa menettelytapa on sama kuin ristihäivytyksessä, mutta lineaarisen interpolaation sijaan käytetään Bezier-interpolaatiota. Se lisää animaatiomuutokseen tangentteja, jotka pehmentävät muutosta. [19.]

Jähmeä muutos

Jähmeä muutos (frozen transition) on ristihäivyttämisen kaltainen tekniikka, jossa ensimmäisenä toistettu animaatio pysäytetään hetkellä, jolloin seuraavaa animaatiota aletaan toistaa. Näin ollen animaatio A:n asento on jähmettynyt paikoilleen ja animaatio B ottaa sitä haltuun ajan edetessä. [19.]

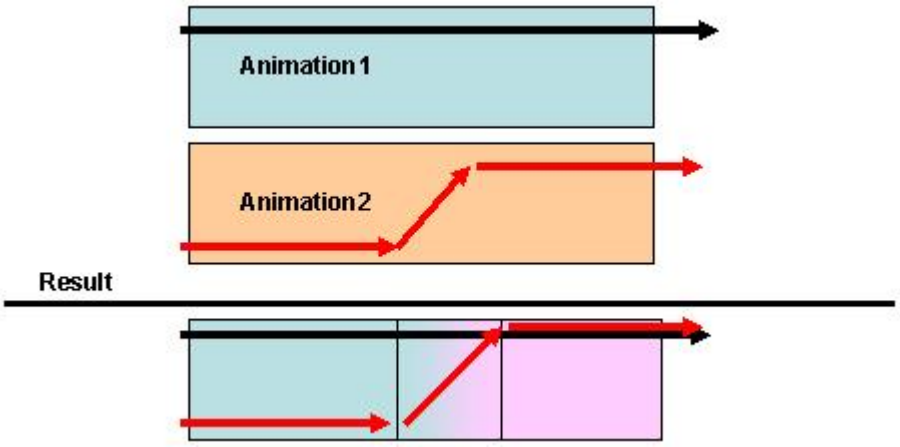


Kuva 13. Jähmeässä muutoksessa animaatio A:n aika pysäytetään, ja animaatio B ottaa sen asentoa haltuunsa ajan edetessä. [25]

Tällainen transitionaalinen sekoitus toimii hyvin, kun kaksi animaatiota ovat täysin toisistaan poikkeavat eikä niitä voi synkronoida, kuten ne tulee olla pehmeässä muutoksessa. [25]

Additiivinen animaatio

Additiivisessa eli lisäävässä animaatiossa (additive animation) toistettavaan animaatioon lisätään toinen animaatio ja näiden yhdistelmää toistetaan, kunnes uusi animaatiotila korvaa nykyisen tilan (kuva 14).



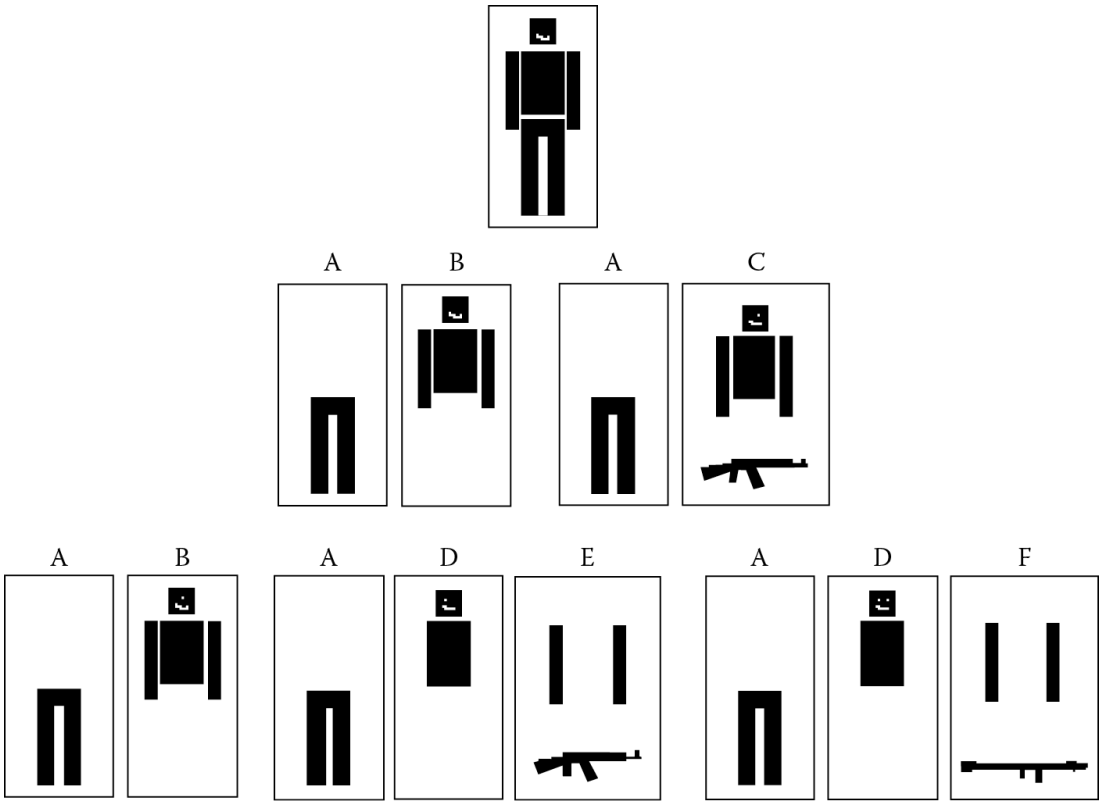
Kuva 14. Additiivisessa animaatiossa kaksi animaatiota yhdistyy, kunnes uusi animaatiotila korvaa nykyisen [17].

Additiivinen animaatio voi olla täysin sama animaatio lähes kaikille hahmoille pelissä, ja se usein halutaan lisätä muiden animaatioiden päälle kaikkina aikoina. Yksi yleisimmistä esimerkeistä on animaatio, joka luo hahmolle illuusion hengittämisestä. [15.] Tätä samaa animaatiota käytetään vahvennettuna tai laimennettuna usein kaikille humanoideille pelissä. Animaatiosta voidaan tehdä myös versiot, joissa hahmo selvästi on hengästynyt tai ponnistelee loukkaantuneena hengittämisensä kanssa, ja se aktivoidaan, kun hahmo on saanut tietyn määrän tai enemmän vahinkoa esimerkiksi tulitaistelussa. [18;20.] Näin ollen esimerkiksi juoksuanimaatio on lähes sama, mutta usein selkeästi raskaampi kertomaan pelaajalle ilman erillistä ilmoitusta, että hahmo on saanut vahinkoa eikä siksi enää liiku kuten yleensä. Alkuperäiseen animaatiotilaan palataan vasta, kun pelaaja on esimerkiksi käyttänyt ensiapupakkauksen loukkaantuneella hahmollaan, jolloin se usein siirtyy alkuperäiseen tilaansa ja jatkaa matkaansa nyt normaalisti.

Additiivisia animaatioita käytetään myös, kun halutaan luoda hahmolle tähtäyksen animaatiojärjestelmä, jolla hahmo voi vapaasti tähdätä tietyllä toimintasäteellä ympäriinsä, ennen kuin jalat reagoivat ja kääntävät hahmoa ohjattuun suuntaan. Mitä useampia väliasentoja hahmolle luodaan, sitä sulavampi ja tarkempi tähtäysanimaatioiden sarjasta tulee. [18.] Tällaiset tähtäävät animaatiot eivät usein poikkea paljoa alkuperäisen animaation asennosta uusissa tähtäysasennoissaan, vaan tavoitteena on pikemminkin asetella esimerkiksi hahmon kyynärpäät, lantio ja pää niin, että transitio tähtäysten välillä näyttäisi mahdollisimman luonnolliselta eikä ylimääräistä hypähtelyä asentojen välillä olisi.

Osittainen animaatio

Osittaisessa animaatiossa (partial animation) osa hahmon raajoista naamioidaan eli ns. maskataan piiloon. Tämä tarkoittaa sitä, että animaatiota toistetaan yleensä vain tietylle pelihahmon nivelistön osuudelle. Järjestelmä mahdollistaa esimerkiksi sen, että hahmo voi kantaa tai ladata asetta samalla, kun se juoksee. Näin vartalon ylempi osa toistaa latausanimaatiota ja alempi juoksun animaatiota (kuva 15). [15.]



Kuva 15. Osittainen animaatio luodaan usein asettamalla hahmolle ns. maskeja, jotka määrittävät, mitä osia hahmosta tietty animaatio saa käyttää, ja kaikki muu jää huomioitta [18].

Käytännössä tällainen järjestelmä toteutetaan pelimoottorissa lähes täysin koodilla. Kuvan 15 hahmon mustat osat on aktivoitu animaation käyttöön, ja kaikki muu toistaa yleensä animaatiota, joka jää toistettavan animaation alle [18].

Pelistudio Naughty Dogin animaatiopäällikkö Almudena Soria Sancho kannusti vuonna 2017 GDC (Game Developers Conference) animaatiotilaisuudessa alan animaattoreita käyttämään osittaisia animaatioita puskemaan tarinankerrontaa ja hahmonkehitystä pienillä, piilotetuilla ja reagoivilla animaatioilla, joiden ei ole tarkoitus keskeyttää pelin kulkua. Niissä hahmoille voidaan antaa omia uniikkeja animaatioitaan esimerkiksi reagoimaan kulloiseenkin säähän. Kuvassa 16 esitellään The Last of Us -pelin pelihahmo Ellien osittaista animaatiosarjaa, jossa pienieleisessä hiustensipaisuanimaatiossa jalat on maskattu piiloon. [21.]

Kuva 16. The Last of Us -peli (2013) Käytti osittaisia animaatioita usein pieniin animaatioihin lisätäkseen hahmon elävöittämistä [21].

Osittainen animaatio mahdollisti Ellien kohdalla sen, että hahmo saattoi toistaa animaation riippumatta siitä, kävelikö vai seisoiko hahmo paikoillaan animaation alkaessa. Tällaiset pienet ja pelimekaniikan kannalta turhat animaatiot ovat yksi nykypäivän peliteollisuuden keino lisätä hahmojen uskottavuutta ja tunnesidettä pelaajan kanssa sekä kuljettaa tarinaa eteenpäin hienovaraisin keinoin.

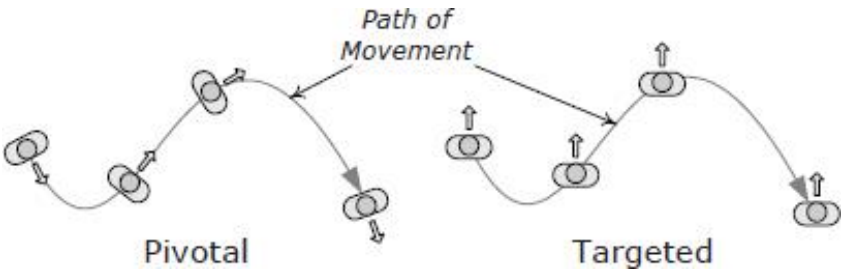
Höyhenanimaatio

Höyhenanimaatio (feather animation) häivyttää animaatiota haluttuun suuntaan, yleensä ylös tai alas. Höyhenanimaatio aiheuttaa sen, että animaatio on näkyvämpi höyhenen alapäässä, kuten käsissä tai selkärangan yläosissa. [15.] Esimerkiksi animaatiossa, jossa hahmo ampuu aseella juostessaan, alavartalo käyttää juoksuanimaatiota ja animaation paino vähenee ylöspäin eikä enää hallitse ylävartaloa, joka tähtää ja ampuu [22]. Tämä on vaihtoehtoinen tapa erottaa animaatiot toisistaan, kun halutaan toistaa useaa animaatiota kerrallaan pehmeämmin kuin osittaisella animaatiolla.

Ohjattu liikkuminen

Kohdennettu eli ohjattu liikkuminen (directional locomotion) pyrkii etsimään erilaisia keinoja ohjata hahmon animaatioita, kun ollaan kulkemassa tiettyyn suuntaan. Oikea liikkumisanimaatio löydetään usein yhdistämällä kuljetun suunnan läheisiä animaatioita toisiinsa lineaarisen interpolaatiohäivytyksen avulla. [19.]

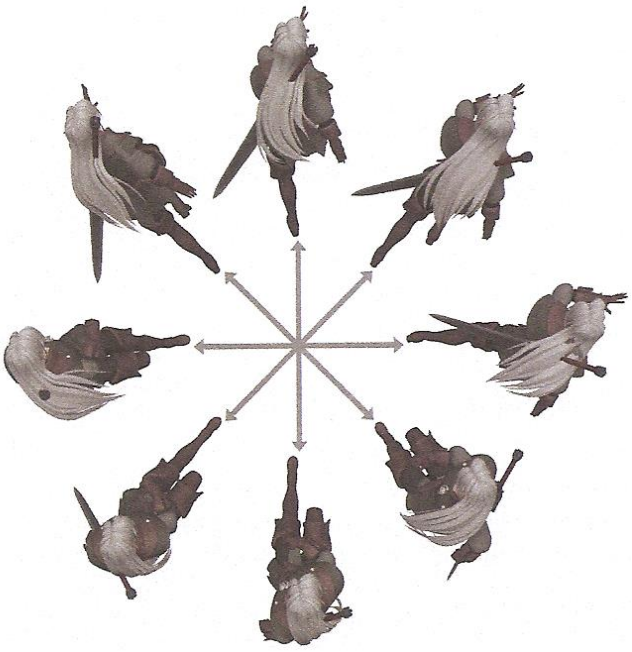
Kun ihminen kävelee tai juoksee, hän saattaa vaihtaa kulkunsa suuntaa kahdella tavalla. Niin kutsutussa napakääntymisessä, englanniksi pivotal movement, ihminen kääntää koko kehonsa kulkemaansa suuntaan. Toinen tapa on pitää katse tietyssä pisteessä, mutta liikkua esimerkiksi sivuttain. Tällaista liikkumista kutsutaan peleissä englanninkielisellä termillä strafing, ja se tarkoittaa kohdennettua liikkumista. Kuvassa 17 esitellään pelihahmon kehon suunnan ero napakääntymisellä ja kohdennetulla liikkumisella. [19.]



Kuva 17. Pelihahmon tyypillinen napakääntyminen ja strafing eli sivuttain liikkuminen [19].

Napakääntymiseen ei välttämättä lisätä suuntaa antavia animaatioita lainkaan, vaan hahmon kääntyminen saatetaan aiheuttaa esimerkiksi kiertämällä koko hahmoa hallitsevaa kontrolleria kohti haluttua suuntaa [15].

Sen sijaan kohdennetusti liikutettavalle hahmolle tarvitaan parhaimmillaan jopa kahdeksan eri animaatiovariaatioita suunniksi, joihin hahmo voi kulkea. Näin pyritään saavuttamaan mahdollisimman luonnolliset liikeradat hahmon liikehdinnälle. Kuvassa 18 esitellään hahmon animaation keskeisiä variaatioita, kun hahmo ohjataan mihin tahansa suuntaan 45 asteen välein. [15.]



Kuva 18. Tyypillinen esimerkki siitä, kuinka hahmon tulisi lähteä liikkeelle 45 asteen kulman välein [15].

Kohdennettua kulkemista käytetään erityisesti ampumapeleissä, ja niiden animaatiovariaatiot ovat usein haastavia hallita. Ongelmia syntyy usein jalkojen maahan osumisen rytmityksen kanssa animaatioita häivyttäessä, kun jalkojen tulisi osua maahan edes karkeasti samaan aikaan ristihäivyttämisen sääntöjen vuoksi. Suurin haaste kohdennetussa kulkemisessa kuitenkin tulee vastaan kehon kierron hallinnassa, kun halutaan ohjata hahmoa ensin suoraan eteenpäin ja sitten heti taaksepäin. [15.]

2.4 Parametristen animaatioiden hyödyt

Koodin ohjaama animaatiojärjestelmä, jossa animaatiot on pidetty erillään ja jossa niitä käytetään päällekkäin tai limittäin toistensa kanssa, vapauttaa paljon muistia, kun jokaista variaatioita ei ole tarvinnut tallentaa erikseen. Tämä ei kuitenkaan tarkoita, että nykypeleissä olisi vähemmän animaatiota kuin ennen. Järjestelmä päinvastoin antaa taiteellista vapautta lisätä erilaisia animaatioita hahmoille, kuten ympäristöön, säähän tai varustuksiin reagoimista. [23.] Parametrisista animaatioista onkin tullut alan normi, ja järjestelmät animaatioiden hallinnan takana ovat suurimmilla pelistudioilla massiiviset.

Kuvassa 19 esitellään eläimen kaltaisen robotin kääntymistä ohjaileva parametrisoitu animaatiojärjestelmä.

Kuva 19. Robottialligaattorin animaatiovaihejärjestelmä robotin kääntymisen vaiheille Horizon Zero Dawn -pelin kehittäjätyökalun näkymästä [20].

Kuvassa 19 pelistudio Guerilla Gamesin animaatiojohtaja esittelee GDC-luennollaan vuonna 2018 Horizon Zero Dawn -pelin animaatiojärjestelmiä ja antaa esimerkkinä kuvan robottialligaattorin animaatiovaihejärjestelmän. Kuvassa nähtävä järjestelmä ohjailee ainoastaan robottihahmon kääntymistä, mutta on silti mittasuhteiltaan jo poikkeuksellisen massiivinen. Siniset palkit kuvaavat robotin eri osien liikettä ja vaiheita sen kääntyessä. [20.]

Pelisuunnittelijalle tällainen koodiin perustuva animaatiojärjestelmä avaa kaikki mahdollisuudet elävöittää hahmoa ja viedä tarinaa eteenpäin samalla kun pelaaja pelaa peliä jopa keskeytyksettä. Tällainen systeemi ei ulotu pelkästään pelihahmoon, vaan mahdollistaa jokaisen animoitavan objektin animaatioiden parametrisen animoinnin.

Pelaajalle tämä tarkoittaa interaktiivisempaa kokemusta pelissä. Hänen toiminnallisuutensa vapautetaan, ja hahmolla on mahdollista esimerkiksi kävellä tai hölkätä, kun pelaaja niin haluaa. Pelaaja voi halutessaan samaan aikaan esimerkiksi hypätä ja ladata

aseensa tai napata juoksun aikana esineen tasolta, jonka ohittaa. Parametriset animaatiot mahdollistavat parhaimmillaan mahdollisimman katkeamattoman ja luonnollisen pelielämyksen, kun animaatiot eivät enää keskeytä muita toimia, jos järjestelmä rakennetaan tukemaan pelaajan saumatonta yhteyttä peliin.

3 Animaation matemaattinen teoriapohja

Tässä luvussa käydään kevyesti läpi animaation matemaattisesti läpikäymän prosessin vaiheet 3D-ohjelmasta pelimoottorissa toistettavaksi animaatioksi ja esitellään kaksi yleisintä interpolaatiotapaa animaatioissa.

Minkä tahansa 3D-objektin animaatio on pohjimmiltaan sarja kierto- ja siirtymäarvoja pisteillä, jotka puolestaan liikuttavat polygoneja, tai tarkemmin niiden kulmapisteitä (vertices), antaen illuusion liikkuvasta tai elävästä objektista. Käytännössä animaatiot luodaan 3D-ohjelmassa manipuloimalla animoidulle objektille annettujen nivelten transitio- ja kiertoakseleita. 3D-ohjelmassa näitä paikan ja kierron tallennuksia animaation aikajanalla kutsutaan avainkehyksiksi, ja ne sisältävät kaiken informaation nivelen tilasta tallennuksen kohdassa. [15.] Kuvassa 20 on tyypillinen 3D-ohjelman aikajana, jossa punaiset viivat kuvaavat animoitavalle objektille annettuja avainkehyksiä tiettynä ajankohtana.

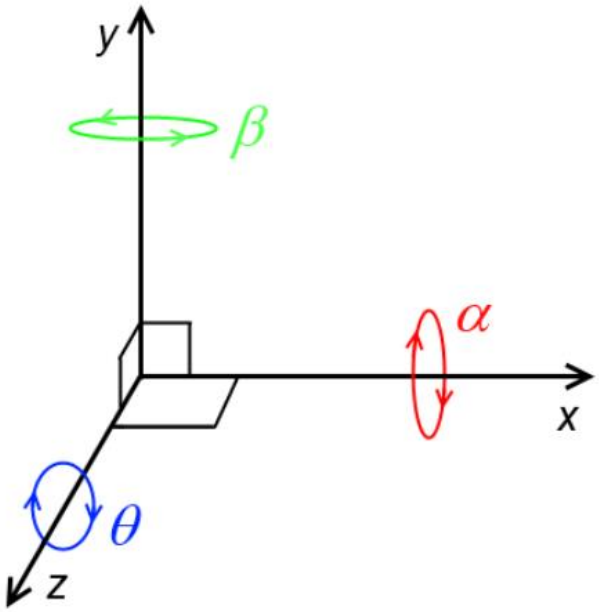
Kuva 20. Aikajananäkymä Maya-ohjelmassa. Punaiset viivat kuvaavat nivelen kierron ja paikan tallennuspaikkoja aikajanalla.

Animaatio-ohjelmasta riippumatta avainkehykset ovat animaatioiden luonnissa normi, ja niiden juuret johtavat aina Walt Disneyn aikoihin saakka. [24].

3.1 Euler-kulmat

Eulerin kulmat esittävät jäykän kappaleen asentoa kolmiulotteisessa tilassa kolmen yleisesti käytössä olevan koordinaatistoakselin (x, y ja z) avulla. Asento esitetään kulmina jokaisen akselin mukaan, jolloin kiertoakselien suomenkielisinä niminä käytetään joskus

kallistusta (x), pituuskallistusta (z) ja suuntakulmaa (y). Matemaattisissa malleissa näitä kulmamerkkejä voidaan merkitä myös α, β, θ. Kappaleen asento maailmassaan voidaan esittää näiden kolmen kiertymäarvon avulla yhtenä esitysmuotona. [25.] Kuvassa 21 osoitetaan kappaleen asennon suuntavektoreita ja akseleiden kiertokulmia.



Kuva 21. Kappaleen asennon mahdolliset suunta-akselit ja niiden kiertosuunnat [24].

Tästä käännöksestä johtuu kuitenkin myös animaattorille monia haasteita aiheuttava gimbal lock. Se tapahtuu, kun animoitu objekti, kuten hahmon sisäinen nivel, käännetään yhdellä akselillaan niin, että kaksi muuta akselia kohtaavat arvoissaan, mikä tekee niistä käyttökelvottomat. [15.] Gimbal lockin havaitsee helposti, sillä hetkellisesti animoitavan hahmon raaja saattaa käyttäytyä hyvin arvaamattomasti etsiessään sille seuraavaksi asetetut arvot. Esimerkiksi jos käännetään x-akselia 90 astetta, y-akseli luhistuu z-akseliin. Tämä estää y-akselin kiertämisen, koska siitä ja z-akselista on tullut identtiset.

Avainkehykset on luotu animaattoreille kuvaamaan Euler-kulmien arvoja tietyssä ajan kohdassa helpottamaan työskentelyä. Kun animaatio siirretään animaatio-ohjelmasta pelimoottoriin, se kääntää Eulerit ja lukee animaatiot kvaternioina, interpoloiden arvot jokaisen ruudun välillä. [15.]

3.2 Kvaterniot

Kvaternio (quaternions) on abstrakti neliulotteinen kompleksiluku, jolla voidaan kuvata jäykän kappaleen asento kolmiulotteisessa tilassa. Se on vaihtoehtoinen tapa esittää kulmia matemaattisesti kätevässä ja kompaktissa muodossa. Sen pieni koko on suuri matemaattinen etu pelimoottoreille. Kvaternioita voidaan interpoloida ja jopa kertoa keskenään, ja siksi se on yleisin tapa kuvata kiertoa pelimoottoreissa. [15; 19; 25; 26.]

Kvaternio koostuu kolmesta imaginääriosasta i, j ja k sekä neljästä reaaliluvusta w, x, y, ja z. Se noudattaa muotoa

$$w + xi + yj + zk$$

Neljän ulottuvuuden vuoksi kvaterniot eivät kärsi gimbal lock-ongelmasta, mutta niitä on haastavaa havainnollistaa kuvainnollisesti [25].

Kvaternion formulointi antaa sille kaksi suurta etua akselien ja kulmien esitysmuotoon nähden. Ensinnäkin se estää kiertojen ketjuuntumisen ja lisäykset suoraan pisteisiin ja vektoreihin. Toiseksi se mahdollistaa kiertojen olevan helposti interpoloitavissa joko lineaarisella interpolaatiolla (LERP) tai lineaarisella interpolaatiolla 4-ulotteisen yksikköpallon pinnalla (SLERP). [19.]

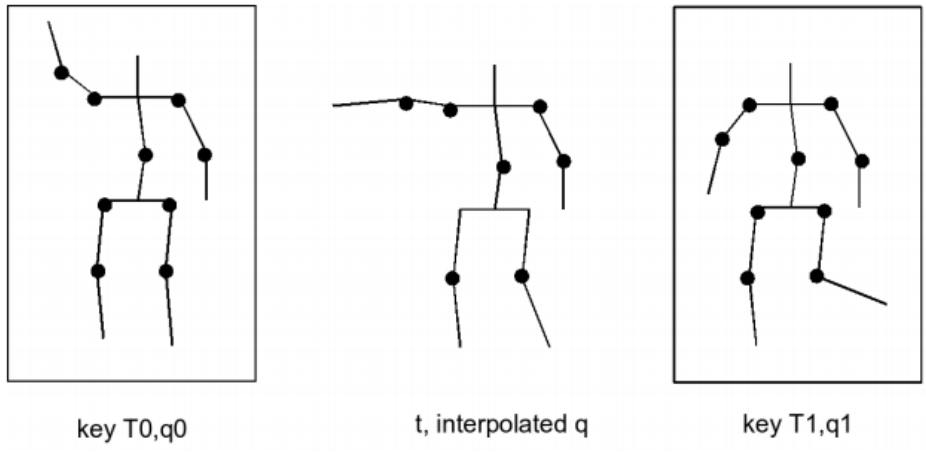
3.3 SRT-kiertomatriisit

Kun kvaterniot voivat esittää ainoastaan rotaation, 3 × 3 -matriisi voi esittää mielivaltaisen affiinimuunnoksen (rotaatio, translaatio, skaalaus). Kun yhdistetään rotaatiokvaternio, translaatiovektori ja skaala, saadaan 3 × 3 -matriisiesitystapa affiinimuunnoksesta. [1; 19.] Tätä kutsutaan joskus pelialalla SRT-transformaatioksi tai SQT:ksi [19]. SRT-transformaatiomatriiseja käytetään laajalti tietokoneanimaatiossa niiden pienen koon ja interpoloitavuuden vuoksi. Translaatio ja skaala voidaan interpoloida LERP:llä ja kvaterniot SLERP:llä. [25] Näitä matriiseja voidaan myös kertoa muilla transformaatiomatriiseilla tuottamaan sekoitettuja muunnoksia, ja silti ne säilyttävät kompaktin 3 × 3 -muotonsa. [1.]

3.4 Interpolaatio pelihahmon animaatiossa

Interpolaatio tarkoittaa väliarvojen laskemista jo tunnettujen arvojen välille [26]. Kvaternioiden avulla rotaatiot ovat interpoloitavissa kuten vektorit ja pisteetkin [19]. Riippumatta avainkehyksistä, joita animaatio-ohjelmassa käytetään, animaatio viedään peliin jokaista avainkehystä myöten projektin ruutunopeudella, joka yleensä peleissä on 30 fps. Pelimoottorissa matriiseiksi muutetut avainkehykset saavat uudet arvot ajan jokainen hetki animaatiota ajettaessa. Näiden arvojen muutos lasketaan yleensä lineaarisella interpolaatiolla. [15; 27.]

Interpolaatiomenetelmiä on erilaisia, mutta yleisesti käytetään lineaarista interpolaatiota, joka on interpolaatioista kevyin ja tuottaa suoraviivaista liikettä ilman pehmennyksiä [26; 27]. Kuvassa 22 esitellään yksinkertaisen tikku-ukon nivelten avulla lineaarista interpolaatiota, jossa siirrytään avainkehyksestä seuraavaan ja väliin muodostuva asento on kahden avainkehyksen täydellinen sekoitus.



Kuva 22. Hahmon luusto on avainkehystetty ajassa 0 ja 1. Keskimmäinen asento kuvaa interpoloitua tilannetta annetun ajan keskikohdassa [28].

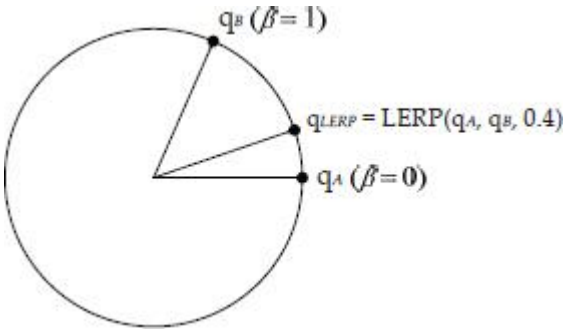
LERP on alan pelimoottoreissa yleisimmin käytetty tapa interpoloida animaatiota avainkehysten välillä, sillä se on laskennallisesti kevyt eikä vie näin ollen paljoa prosessointitehoa. Lineaarinen interpolointi laskee suorimman tien pisteestä seuraavaan, ilman pehmennystä animaatiokäyrään [27.]

Kierron lineaarinen interpolaatio

Kierron interpoloimisella on monia tärkeitä käyttötarkoituksia esimerkiksi animaatioon, dynamiikkaan ja kameran järjestelmiin pelimoottorissa [24].

Kuvassa 23 havainnollistetaan interpoloitua kiertoa kahden kvaternion välillä. Nivelelle on annettu kaksi kvaternioarvoa q_A ja q_B esittämään kappaleen kiertoa. Lineaarisella interpolaatiolla voidaan löytää q_{LERP}, joka on β 0,4 niiden väliltä.

$$q_{LERP} = LERP\ (q_A, q_B, \beta) = \frac{(1-\beta)q_A + \beta q_B}{|(1-\beta)q_A + \ \beta q_B|}$$



Kuva 23. Lineaarinen interpolaatio (LERP) kahden kvaternion q_A ja q_B välillä. [25]

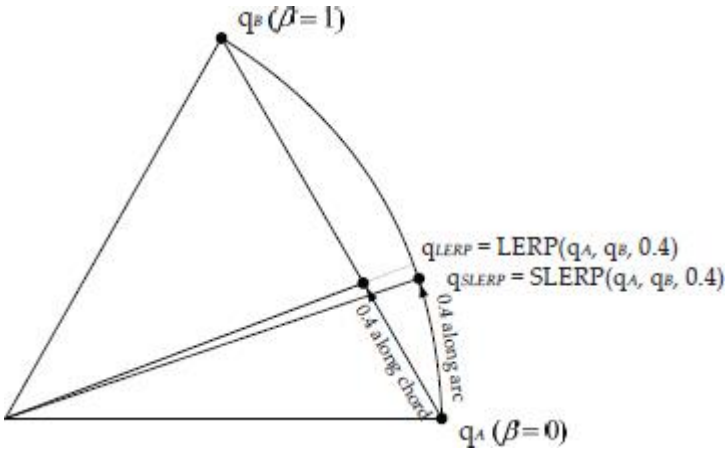
On kuitenkin otettava huomioon, että tulos interpoloiduista kvaternioista on uudelleen normalisoitava lopuksi. Tämä on tärkeää, sillä LERP-operaatio ei tallenna vektorin pituutta. [19.]

Geometrisesti $q_{LERP} = LERP\ (q_A, q_B, \beta)$ on kvaternio, jonka orientaatio on β-prosentuaali matkalla orientaatiosta A orientaatio B:hen, kuten kuvassa 23 esitetään selkeyden vuoksi 2-ulotteisena. Matemaattisesti LERP-operaatio saa tuloksenaan painotetun keskiarvon kahdesta kvaterniosta, painoilla (1 − β) ja β. On huomattava, että nämä kaksi painoa yhteenlaskettuina ovat aina 1. [19.]

Kierron kaartuva interpolaatio

Lineaarisen interpolaation ongelmaksi muodostuu lopulta, ettei se ota huomioon rotaation kvaternioiden olevan vain pisteitä 4-ulotteisella yksikköpallolla. LERP interpoloi tehokkaasti jänteen mukaan 4-ulotteisen pallon läpi, eikä pitkin sen pintaa kaareutuen. Tämä aiheuttaa nivelen kiertämisen nopeudessa epätasaisuutta. Nivelen kääntymisen voi havaita olevan hitaampi käännöksen alku- ja loppupäässä ja nopeampi keskellä. [19.]

Ongelman ratkaisuun voidaan käyttää kierron kaartuvaa menetelmää interpolaatiossa. Kvaternioita interpoloidaan sineillä ja kosineilla 4-ulotteisen yksikköpallon pinnalla lineaarisesti vektorien välisen kulman suhteen. [1; 19.] Tämä saa jänteen kaareutumaan pisteiden välillä. Kuvassa 24 kuvataan SLERP pitkin 4-ulotteisen yksikköpallon pintaa. [19.]



Kuva 24. SLERP pitkin 4-ulotteisen yksikköpallon pintaa [19].

Tällä menetelmällä nivelen kierto saa yhtenäisen nopeuden kääntyessään. Se on kuitenkin laskennallisesti huomattavasti raskaampi suoritus kuin LERP [19].

SLERP:n kaava on samansuuntainen kuin LERP:n kaava, mutta painot (1 − β) ja β on korvattu painoilla w_p ja w_q ,jotka sisältävät kahden kvaternion sinikulmat [19].

$$SLERP\ (p, q, \beta) = \ w_p p \ + \ w_q p,$$

jossa

$$w_p \ = \frac{sin(1-\beta)\theta}{sin\ \theta}$$
$$w_q \ = \frac{sin\ \beta\theta}{sin\ \theta}$$

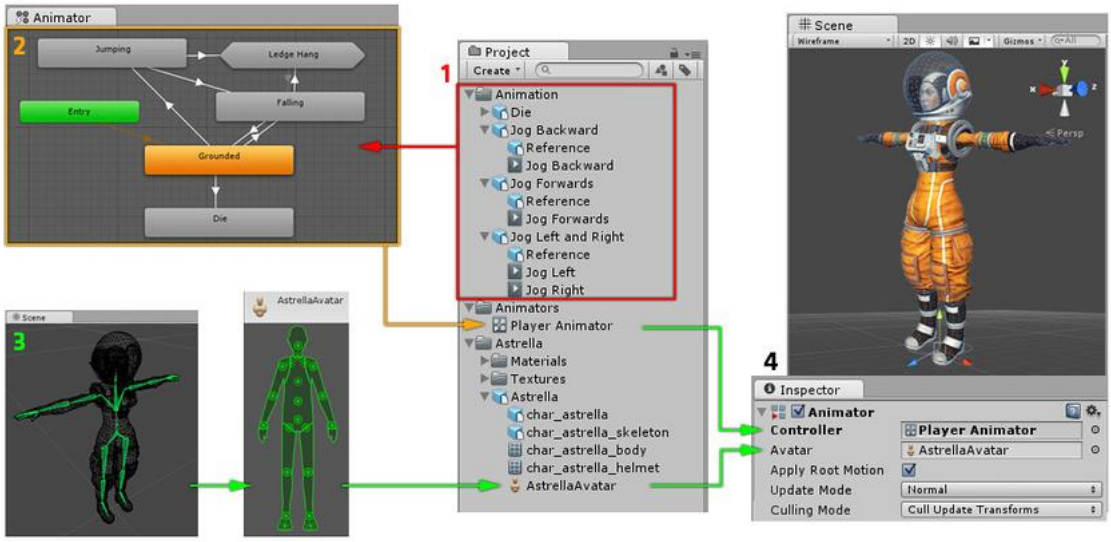
Kun häivytetään useampaa kuin kahta kvaterniota SLERP:llä, on laskutoimitusten järjestyksellä huomattava merkitys. Jos interpoloidaan kvaternio p q:lla ja sitten r:llä, saadaan eri lopputulos kuin häivyttämällä p r:ään ja sitten q:hun. Tämän tärkeys käy ilmi, kun aletaan häivyttää esimerkiksi juoksua kävelyanimaatioon. [29.]

Nykypäivän pelianimaattorin ei päivittäisessä työssään tarvitse perehtyä yllä esiteltyyn matematiikan teoriaan lainkaan, ellei hän päätä erikoistua teknisempään animaatiopuoleen, jossa vaaditaan edistynyttä matematiikkaa ja koodausta vaikkapa erilaisten ominaisuuksien luontiin. Tällainen työ ei usein enää ole esittävää animaatiota, vaan usein se liittyy esimerkiksi animaattorien uusien työkalujen luontiin tai esimerkiksi animoitavien dynaamisten objektien kehitykseen, kun halutaan luoda objekteja, jotka esimerkiksi toimivat yhteistyössä pelin fysiikkamoottorin kanssa.

4 Unityn animaatiojärjestelmä

Pelimoottoreissa animaatiot tallennetaan pelimoottoriin leikkeinä, joissa yksi animaatio vastaa yhtä leikettä [15; 30]. Unity-pelimoottorissa animaatioleikkeet on organisoitu tietynlaiseen jäsenneltyyn kulkukaavioon, jota kutsutaan animaatiokontrolleriksi. Se toimii koodin avulla ns. tilakoneena, joka säätelee, mitä animaatiota pelihahmolla milloinkin

toistetaan, ja milloin animaation tulisi vaihtua tai häivyttyä toiseen animaatioon. Kuvassa 25 esitellään Unityn animaatiojärjestelmän sisäisiä suhteita toisiinsa.



Kuva 25. Unityn animaatiojärjestelmän eri näkymiä. Nuolet merkitsevät niiden suhteita toisiinsa. [30.]

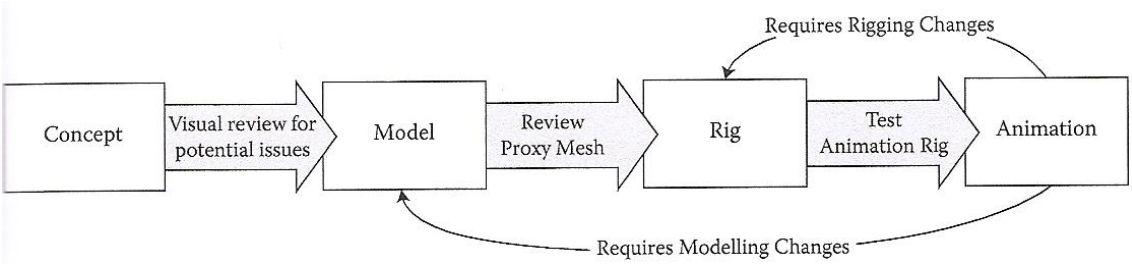
Kuten sivuilla 20 ja 21 mainittiin, tällaiset animaatiovaihejärjestelmät saattavat paisua massiivisiksi suurimmilla pelistudioilla. Kuitenkin yksinkertaisimmillaan ne voivat sisältää jopa vain kaksi animaatioleikettä. [30.]

4.1 Pohjaprosessi ulkoisessa 3D-ohjelmassa

Pelialan käytäntö on, että pelihahmo luodaan ensin pelimoottorin ulkopuolisilla ohjelmilla ja prosessi saattaa edetä usein animaatioksi saakka, kunnes huomataan, että alkuperäistä mallinnusta tarvitsee muuttaa, jolloin palataan hetkeksi alkupisteeseen.

Hahmomallintaja saattaa luoda hahmonsa esimerkiksi dynaamista topologiaa hyödyntävällä ohjelmalla kuten Zbrushilla, jossa hahmomallin tiedostokoosta ei vielä tarvitse välittää. Tämän jälkeen hahmolle tehdään monessa vaiheessa sen tekstuurit, nivelistö, kontrollit ja lopuksi hahmon polygonit eli sen ”iho” kiinnitetään seuraamaan kontrollien avulla nivelistöä. Kaavakuvassa 26 osoitetaan suositeltu työnkulkuprosessi, jossa

hahmomalli käy läpi testausvaiheita ja palaa usein edellisiin työvaiheisiin, jotta tarvittavat muutokset mallissa tai nivelistössä voitaisiin toteuttaa. [15.]



Kuva 26. Kaava esittää suositellun pelihahmon prosessin työnkululle [15].

Pelialan vakiintuneet kaupalliset ohjelmat näille työprosesseille ovat Autodesk 3ds Max ja Autodesk Maya [15], mutta pienemmät studiot käyttävät myös ilmaista Blenderiä.

Animoitavasta hahmosta tuodaan pelimoottoriin sen ns. mesh, eli polygoninen malli, joka on käytännössä sen visuaalinen muoto. Lisäksi tuodaan hahmon sisäinen nivelistö. Nivelistöä liikuttava data tuodaan erikseen, ja yhdessä nämä kolme koodin avulla näyttäytyvät pelaajalle lopulta liikkuvana pelihahmona. [15.]

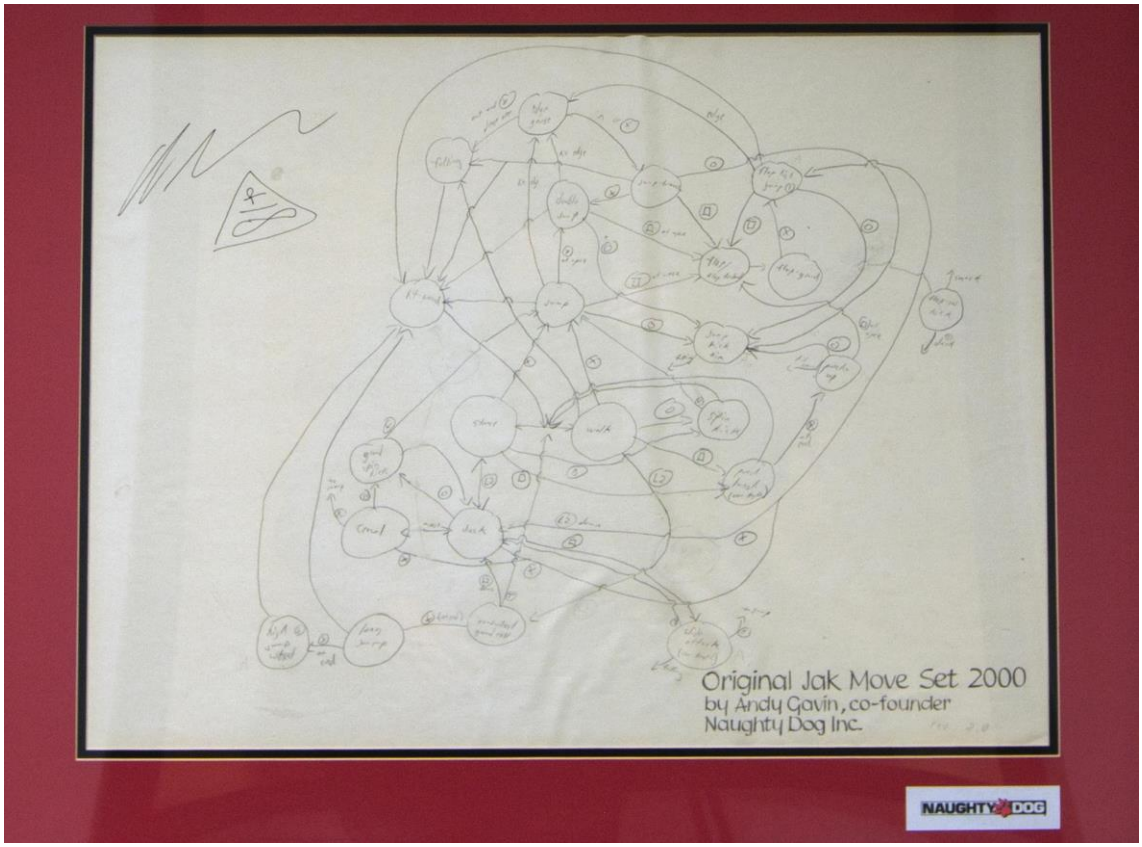
Tuodut animaatiotiedostotyypit vaihtelevat alalla, mutta yleisimmin käytetään fbx-formatointia. Sitä käyttävät mm. Unity ja Unreal. Samalla kerralla useamman animaation tuominen pelimoottoriin on alalla normaali käytäntö, sillä se nopeuttaa animaattorin työprosessia huomattavasti esimerkiksi testatessa ja hioessa animaatioita tai sen sarjoja. [15.]

4.2 Tilakone

Unity-pelimoottorissa animaatioleikkeet on organisoitu tietynlaiseen jäsenneltyyn kulkukaavioon, jota kutsutaan animaatiokontrolleriksi. Se toimii koodin avulla ns. tilakoneena, joka säätelee, mitä animaatiota, esimerkiksi pelihahmossa, milloinkin toistetaan ja milloin animaation tulisi vaihtua tai häivyttyä toiseen animaatioon. Yksi animaatiotila voi käsittää esimerkiksi seisovan tähtäyksen ja sen kaikki tähtäyssuuntaukset. [30.]

Tilakoneen lyhyt historia

Perinteisesti tilakonetta alettiin luonnostella peleihin piirtämällä paperille ympyröiden sisään hahmon ns. ydinanimaatiot ja suunnittelemalla sitten viivoilla, mitkä animaatiot olivat suoraan linkissä toisiinsa. Tämä helpotti paitsi tarvittavien animaatioiden listaamista, myös tarkensi, millaisia transitioanimaatioita mahdollisesti tarvittiin animaatioiden välille pehmentämään muutosta animaatioista toiseen. Tämän jälkeen animaattori tai pelisuunnittelija yhdessä koodaajan kanssa loi animaation tilakoneen ja sen tarvittavat animaatiotilat hahmolle. Kuvassa 27 on valokuva, jossa silloinen pelitalon perustaja on kaavaillut paperille pelin päähahmon tilakoneen Jak & Daxter -peliin. [15.]



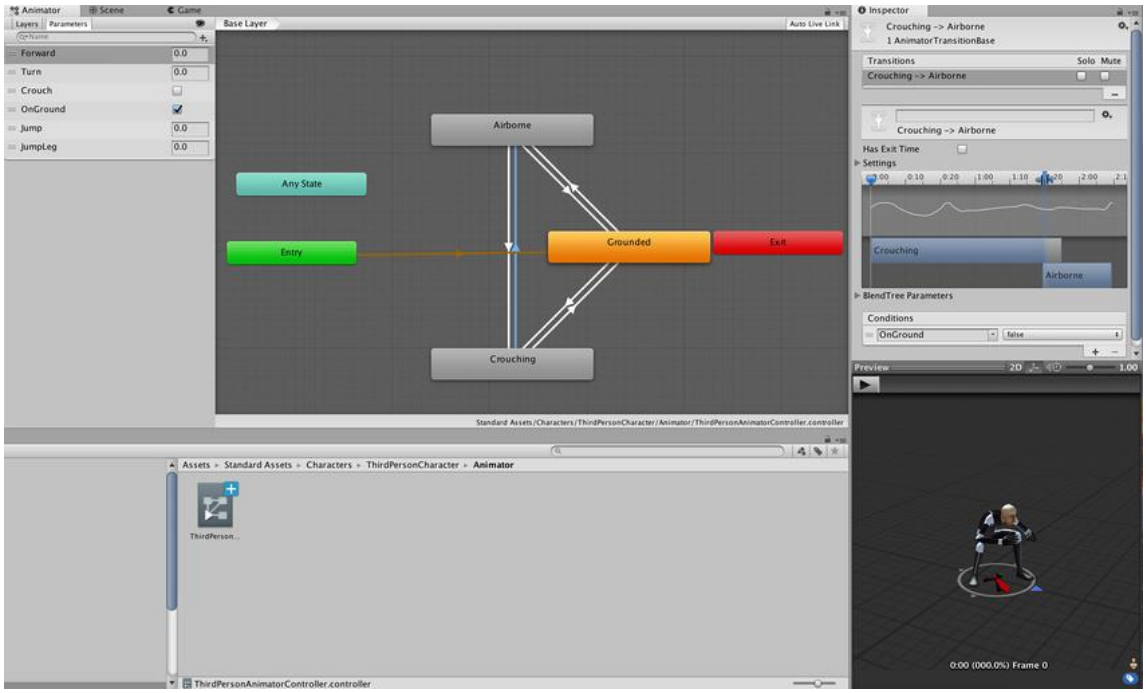
Kuva 27. Alkuperäinen Jak & Daxterin animaatioiden kulkukaavio vuodelta 2000 [31].

Kulkukaavio näyttää ensisilmäykseltä massiiviselta, mutta siitä huolimatta on muistettava, ettei parametrisia animaatioita ole vielä tässä vaiheessa edes täysin kehitetty, joten tämän hahmon kulkukaaviot mahtuvat vielä yhdelle suurelle paperille.

Tekniikan kehittyessä myös yksittäisen pelihahmon animaatiokaaviot kasvoivat monimutkaisemmiksi, ja pian niistä muodostui satoja animaatioita sisältäviä verkostoja, joita alkoi olla vaikea hallita. Hallintaa helpottamaan pelimoottoreihin kehitettiin graafiset käyttöliittymät animaatioiden kulkukaavioiden hallintaan, ja nykypäivänä pelialalla animaattori hoitaa itse tilakoneen toiminnan hahmon animaatioiden kanssa työskennellessään ilman koodaajan apua. Nykypäivän tilakoneissa voi muokata esimerkiksi animaation häivytystä seuraavaan ja liittää tai poistaa animaatiolinkityksiä toisistaan. Nykypäivän animaattorin tulee myös ymmärtää, miten koodi tilakoneiston takana toimii, jotta päästään käsiksi animaation jokaiseen osa-alueeseen, kun animaattorin tarvitsee muokata animaatiota ja esimerkiksi ns. tapahtuman laukaisuja (event) pelissä. [15.]

Tilakone Unityssa

Kun animaatiot ensi kertaa tuodaan Unityyn, ne asetellaan pelimoottorin animaationäkymään ja niiden väleille voidaan asettaa nuolina näkyviä linkkejä, jotka toimivat tilamuunnoksina animaatioiden välillä, kuten kuvassa 28 esitetään. [30.]



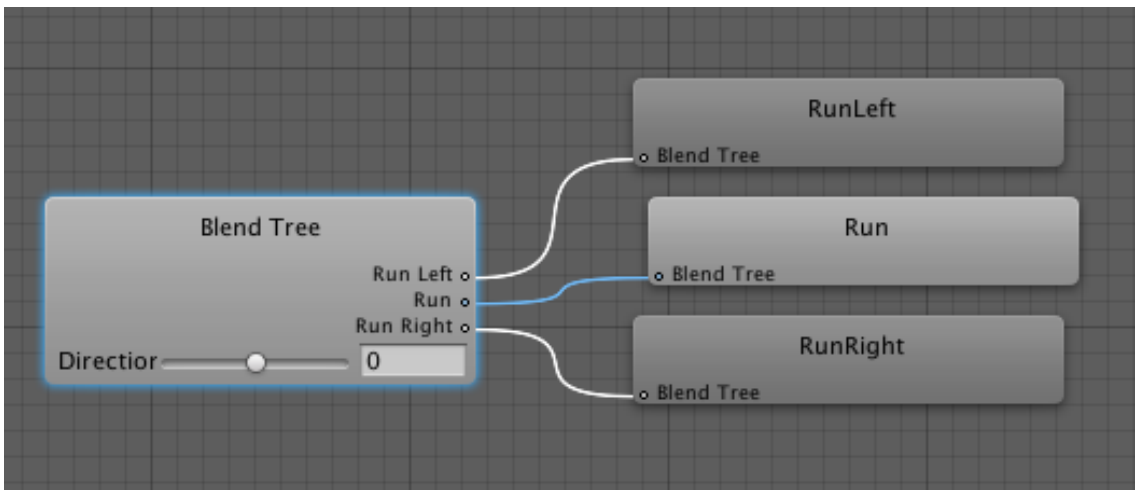
Kuva 28. Tyypillinen animaatiotilakone Unityssa [30].

Unityssa animaatiokontrolleri sallii animaatioiden järjestelyn ja hallinnan pelihahmolle tai animoitavalle objektille. Kontrollerilla on viittaukset kaikkiin animaatioleikkeisiin, ja se hallitsee ja ohjaa kaikkia animaatiotiloja ja niiden välisiä muutoksia käyttäen tilakonetta, joka taas esittää animaatioiden väliset suhteet visuaalisesti, mikä helpottaa niiden hallintaa käyttäjälle. Tilakone on myös mahdollista koodata ilman visuaalista kulkukaaviota, mutta usein käytetään niiden yhdistelmää. [30.]

4.3 Animaatioiden häivytys Unityssa

Siinä, missä transitiot tilakoneen sisällä häivyttävät kaksi animaatiota toisiinsa hetkellisesti, jotta seuraava voi jatkua ilman edellistä animaatiota, voidaan häivytyspuulla (blend tree) sekoittaa monta animaatiota keskenään niin, että ne toistuvat hahmossa samanaikaisesti. Häivytyspuu on tilakoneen erityistyyppinen tila. Sitä, kuinka suurella painolla animaatiot tullaan näkemään hahmon animaatioissa, voidaan säädellä animaatiokontrollerin parametreilla. Jotta lopputulos olisi mahdollisimman miellyttävä, tulisi animaatioiden olla suunnilleen samankaltaisia liikkeissään ja ajassaan. [30.]

Kuvassa 29 on esimerkkejä karkeasti samankaltaisesta animaatiosta, jotka voidaan häivytyspuun avulla sekoittaa yhdeksi. Kävelyn ja juoksun välinen häivytys on tyypillisin tapaesimerkki häivytyspuun käytöstä.



Kuva 29. Häivytyspuu, jolla on kolme alisolmua, joista jokaisen animaatio on karkeasti samantyyppinen häivytettävään animaatioon [30].

Jotta tällainen menettelytapa toimisi, liikkeen tulee tapahtua samassa kohdassa normalisoidussa ajassa. Esimerkiksi juoksuanimaatio voidaan rytmittää niin, että jalan koskетukset maahan tapahtuisivat samoissa pisteissä normalisoidussa ajassa, esimerkiksi kohdissa 0.0f ja 0.5f. Animaation pituuksilla tässä ei ole väliä, kun käytetään normalisoitua aikaa. Häivytyspuu käyttää lineaarista interpolaatiota animaatioiden sekoittamiseen. [30.]

4.4 Interpolaatio Unityssa

Kuten luvussa 2.3 on mainittu, on parametrisia animaatioita monia erilaisia, ja niitä voidaan tuottaa peliin monella eri tavalla. Jokaista tarkoitusta varten on sopiva, tai useita sopivia menetelmiä, mutta kaikkea taustalla hallitsevat animaatiokontrolleri ja koodi. Luvussa 3 esiteltiin lineaarisesta interpolaatiosta, joka interpoloi animaatiossa kahden avainkehyksen sarjat toisiinsa. Koodissa nämä esitellyt matemaattiset mallit on määritelty erilaisissa matemaattisissa kirjastoissa, kuten C++:ssa Mathf.Lerp, joka pystyy laskemaan lineaarisen interpolaation. [30.]

Esimerkkikoodissa 1 tuotetaan kahden animaation välinen ristihäivytys lineaarisella interpolaatiolla, jossa a on aloitusarvo ja b lopetusarvo ja t interpolaatioarvo, jota voidaan kasvattaa 0:n ja 1:n välillä. [30].

```
public static float Lerp(float a, float b, float t);
```

Esimerkkikoodi 1. C++-koodiesimerkki lineaarisesta interpolaatiosta [30].

Esimerkkikoodissa 2 interpoloidaan jälleen lineaarisesti a b:hen t:n aikana. Ajan parametri t on asetettu janalle [0, 1]. Kun t = 0, koodi palauttaa a:n. Kun t = 1, se palauttaa b:n, ja kohdassa 0.5 se palauttaa a:n ja b:n keskiarvon. [30.]

```
using UnityEngine;

public class Example : MonoBehaviour
{
    // animate the game object from -1 to +1 and back
    public float minimum = -1.0F;
    public float maximum =  1.0F;

    // starting value for the Lerp
    static float t = 0.0f;
```

```
    void Update()
    {
        // animate the position of the game object...
        transform.position = new Vector3(Mathf.Lerp(minimum, maximum, t), 0,
0);

        // .. and increase the t interpolater
        t += 0.5f * Time.deltaTime;

        // now check if the interpolator has reached 1.0
        // and swap maximum and minimum so game object moves
        // in the opposite direction.
        if (t > 1.0f)
        {
            float temp = maximum;
            maximum = minimum;
            minimum = temp;
            t = 0.0f;
        }
    }
}
```

Esimerkkikoodi 2. Lineaarinen interpolaatio a:sta b:hen t:n aikana [30].

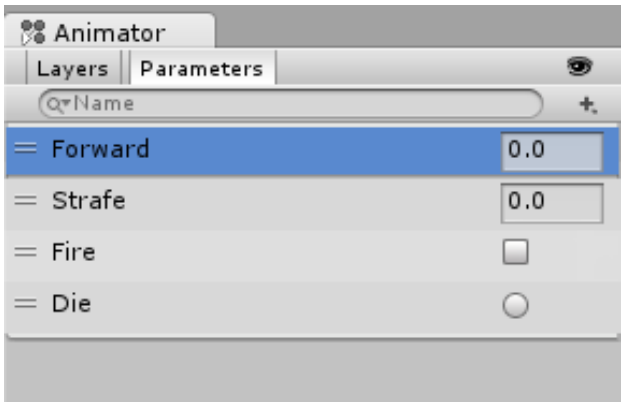
Toinen vastaavanlainen tapa tuottaa kahden animaation välinen ristihäivytys on käyttää suoraan Unityn tarjoamaa interpolaatiosuoritinta (esimerkkikoodi 3) [30].

```
public void Blend(string animation, float targetWeight = 1.0F, float
fadeLength = 0.3F);
```

Esimerkkikoodi 3. Ristihäivytys Unityn tarjoamalla interpolaatiosuorittimella [30].

4.5 Animaatioparametrit Unityssa

Animaatioparametrit ovat muuttujia, jotka määritellään animaattorinäkymässä. Niitä voidaan määrittää ja säädellä myös koodissa. Tällä tavalla koodi voi kontrolloida tai vaikuttaa tilakoneen työnkulkuun. Koodilla voidaan esimerkiksi asettaa käyttöliittymän puolelle animaationäkymään parametrin manipulaattoreita, joiden avulla animaatioon voidaan vaikuttaa käyttöliittymän puoleltakin. Kuvassa 30 animaattori ikkunaan on asetettu koodissa parametreja, joita voidaan säätää käyttöliittymän puolelta. Manipulaattorityyppejä on neljä erilaista: kokoluku (int), liukuluku (float), boolean (bool) ja triggeri (trigger). [30.]



Kuva 30. Animaation parametrejä voidaan säätää käyttöliittymän puolella koodiin asetetuilla luokilla [30].

Kuvan 30 kaltainen kontrolleri helpottaa työnkulkua, kun objektin toimintojen arvoja voidaan muokata pelinäkymän viereisessä tilassa ilman koodin avaamista.

5 Parametrisoituja animaatioita Starbase-peliin

Starbase on avaruuteen sijoittuva massiivinen monen pelaajan verkkopeli (Massively Multiplayer Online Game), jonka pääpainona on avaruusaluksien ja -asemien suunnittelu ja rakentaminen, tutkimusmatkailu sekä taistelu. Kuvassa 31 on Starbasen yksi tämänhetkisistä kansikuvista ja pelin virallinen logo. [32.]

Kuva 31. Starbasen pelijuliste ja virallinen pelin logo [32].

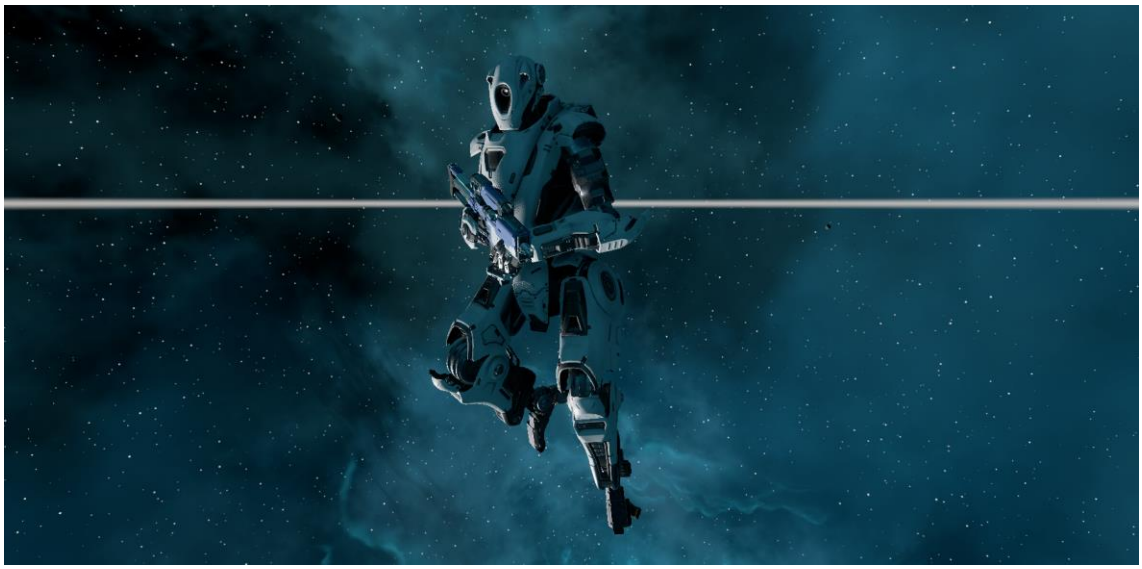
Tässä luvussa selostetaan insinöörityönä tehty parametristen animaatioiden tuottaminen Frozenbyten Starbase-peliin. Insinöörityötä tehdessä Starbase on pian siirtymässä seuraavaan vaiheeseen suljetun alfan tilasta.

Tavoitteet ja vaatimukset

Starbasen perusperiaate animaatioissa on välittää mahdollisimman paljon informaatiota muille pelaajille pelaajan toimista niin taistelussa kuin tavallisessakin kanssakäymisessä ja vapauttaa pelaajan toiminnallisuutta aina mahdollisuuksien mukaan hyödyntäen parametrista animaatiojärjestelmää, joka hahmon toiminnan taakse on rakennettu. Kaikille aseille ja työkaluille on olemassa omat siirtymään, tähtäykseen ja käyttöön liittyvät animaationsa jokaiselle mahdollistetulle animaatiotilalle. Tällainen animaatiojärjestelmä mahdollistaa toiminnan tunnistamisen jo hyvän matkan päästä. Pelaajan esimerkiksi kantaessa liekinheitintä, poikkeavat aseelle omistetut animaatiosarjat muista sarjoista selvästi, mikä helpottaa toiminnan tunnistamista. Tämä luo kehyksen koko Starbasen animaatiojärjestelmälle. [34.]

Näin laajan ja toiminnallisuutta korostavan animaatiotilaverkoston käytön sulavoittamiseksi on luotu myös huomattava määrä siirtymäanimaatioita. Siirtymäanimaatiot mahdollistavat tehokkaan ja mahdollisimman luonnollisen näköisen aseen ja animaatiotilan vaihdon. Mahdolliset rajoitukset aseiden ja työkalujen käyttöön syntyvät ainoastaan pelaajan senhetkisestä olinpaikasta ja tämän henkilökohtaisesta aseiden ja työkalujen valikoimasta. [34.]

Pelaaja voi esimerkiksi aluksellaan valita aseekseen konekiväärin ja madaltaa kulkunsa ryömimiseksi, kun taas lennettäessä mahdollisuutta ryömimiselle ei ole, jos pelaaja ei löydä itselleen laskeuduttavaa alustaa, mutta konekiväärin hän voi silti aina halutessaan nostaa esiin. Pelaaja voi siis tilastaan riippumatta kantaa mitä tahansa asetta tai työkalua, jota hän kantaa valikoimassaan, mutta mahdollistetut tavat liikkua määrittelee ympäristö, jossa pelaaja on. Kuvassa 32 pelaajan hahmo leijuu avaruudessa ampuma-asetta kantaen. [34.]



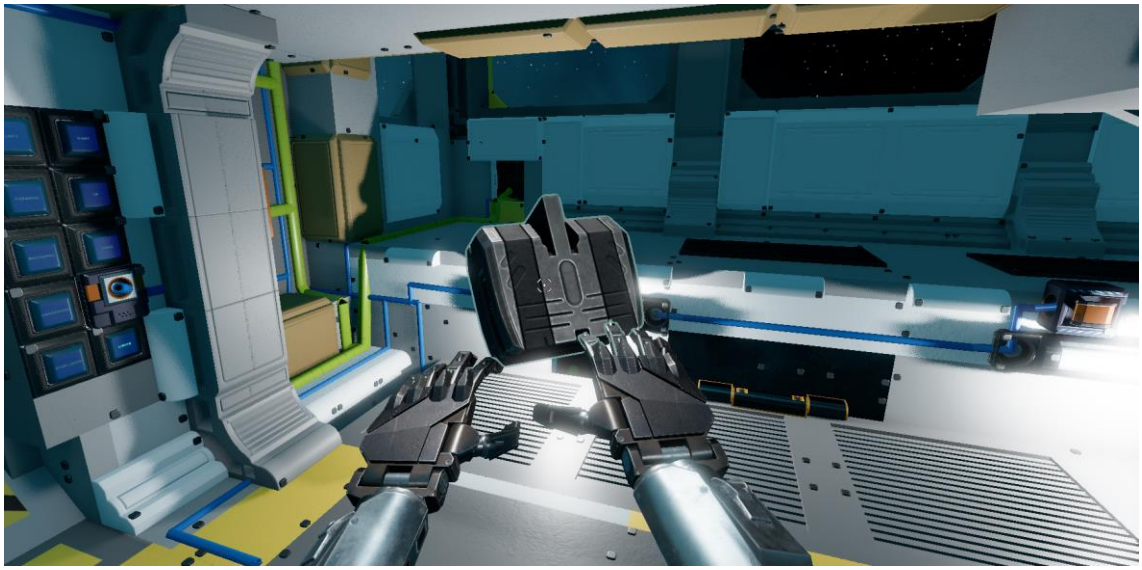
Kuva 32. Pelaajan hahmo leijuu avaruudessa konekivääriä kantaen [33].

Jokainen animaatiotila pyrkii aina säilyttämään pelaajan ohjaaman suunnan, vaikka animaatiotila toimintaa suorittaessa vaihtuisikin. Esimerkiksi ammuttaessa animaatiotila on ensin tähtäys, josta se nousee ampumisen ajaksi ampuvaan tilaan toistaen ampumisen animaatiota. Tämän jälkeen se palaa takaisin tähtäävään tilaan säilyttäen aina pelaajan osoittaman suunnan ja tähtäyksen asteet. Tällainen tähtäykseen perustuva hahmon asennon hallinta on tyypillistä ja erittäin tarpeellista Starbasen kaltaisille peleille, joissa taistelun ja rakentamisen tähtäyksen käyttö on keskeinen osa pelikokemusta. [34.]

Useimmat elehtimiseen käytettävät eli emote-tyyppiset animaatiot on suljetun alfan aikaan asetettu alkamaan ja loppumaan aseettoman seisomisen tilasta. Nämä kaksi pääasiallista animaatiotyyppiä voidaan maskata esimerkiksi kyyristyneeseen tai istuvaan tilaan mahdollisimman sopivaksi. Kaikkia emoteja tulee pystyä käyttämään, oli kannossa sitten minkä tahansa tyyppinen ase tai työkalu. Tätä pyritään rajoittamaan ainoastaan animaatiotilalla, ja tavoitteena on mahdollistaa pelaajien mahdollisimman saumaton kanssakäyminen emote-tyyppisiä animaatioita apuna käyttäen. [34.]

Starbasessa pelaaja voi myös ohjata hahmoaan joko ensimmäisen tai kolmannen persoonan näkökulmasta. Koska kyseessä on MMO, ensimmäisessä persoonassa pelaavan pelaajan hahmon tulisi toistaa muille pelaajille pelihahmon vastaavia kolmannen persoonan animaatioita. Tämä tarkoittaa, että jokaisesta animaatiotilasta on oltava

olemassa kaksi täysin erilaista versiota vastaamaan pelaajan valitsemaa näkökulmaa. Kuvassa 33 pelaaja on siirtynyt ensimmäisen persoonan näkymään. [34.]



Kuva 33. Pelaaja on siirtynyt ensimmäisen persoonan näkymään. Halutessaan pelaaja toi pelata peliä käyttäen ainoastaan ensimmäisen persoonan näkymää [33].

Starbase käyttää lähes kaikkien kolmannen persoonan animaatioidensa pohjana liikkeentunnistusteknologiaa. Siinä näyttelijän liikkeitä kuvataan sensoreilla, joiden data lopulta viedään animaattorille 3D-ohjelmaan. Animaattorin tehtävä on muokata data sellaiseen muotoon, että siitä voidaan tallentaa ja viedä eteenpäin pelimoottorille sopiva tiedostomuoto. Liikkeentunnistusdata on usein vielä hyvin käyttökelvotonta sellaisenaan, joten animaattorin pääasiallinen tehtävä onkin jalostaa siitä toimiva, mielenkiintoinen ja sulava animaatio tai animaatiosarja, ennen kuin se voidaan konfiguroida pelimoottoriin ja aloittaa testaus ja hiominen. [34.]

5.1 Kerrostetut animaatiot tähtäykselle

Starbasen kerrostettujen animaatioiden pohjana toimii aina jokaiselle aseelle tai työkalulle määritelty nollapisteen asento. Nollapisteen asennossa hahmo tähtää suoraan eteenpäin. Pelihahmolle on luotu animaatiojärjestelmä, jonka parametrien ansiosta se voi tähdätä mahdollisimman laajalle sekä vertikaalisesti että horisontaalisesti. Pää ja ase seuraavat osoitettua suuntaa tähtäävässä tilassa. [34.]

Tähtäysanimaatioiden kerrokset ovat käytännössä kahden identtisen avainkehyksen pituisia animaatioita, jotka luodaan pohja-animaation päälle 3D-ohjelmassa. Pohja-animaationa käytetään liikkeentunnistustekniikalla kuvattua syklinä toistuvaa animaatiota, jossa hahmo seisoo neutraalisti paikoillaan ja näyttäisi hengittävän. Kun uusi animaatiokerros aktivoidaan, hahmo tähtää asetettuun suuntaan, liikkuen kuitenkin yhä kuten pohja-animaatiossaankin. Pohja-animaation tulee olla tähtäysanimaatioissa melko neutraali, jotta vältyttäisiin silmiin pistäviltä hyppäyksiltä animaatioissa kerroksia parametrisoitaessa. [34.]

Kun tällainen animaatiosarja lopulta konfiguroidaan pelimoottoriin, voi pelaaja esimerkiksi tähdätä aseellaan ääriasentoon oikealle 89,9°, mutta vasta kun tähtäys saavuttaa 90°:n kulman, hahmo liikkuu alavartalollaan muutaman kymmentä astetta osoitettuun suuntaan suoristaen asentoaan luonnollisemmaksi. Kuvassa 34 hahmo on juuri saavuttanut 89.9°:n kulman tähtäyksessään, ja vaihtaa seuraavaksi asentoa askeltaen paikoillaan, kunnes vartalon kierto ei enää ole ääriasennossaan. Yläruumis tähtää silti yhä pelaajan osoittamaan suuntaan, eikä ole siirtynyt tähtäyksestään lainkaan. [34.]

Kuva 34. Hahmo saavuttaa tähtäyksessään 90°:n ääriasennon ja vaihtaa seuraavaksi asentoaan alaruumiillaan [33].

Näin ollen hahmolla on oltava sopivat äärianimaatiot molemmille puolille 45°:n ja 90°:n tähtäyksille, sillä transitio nollapisteestä 90°:n kulmaan saattaisi aiheuttaa 45°:n kohdalla epätoivottavia asentoja esimerkiksi kyynärpäiden kontrollien kanssa, jos käytetään IK-

systeemiä (inverse kinematics), jossa esimerkiksi ranteelle tai kannettavalle aseelle määritelty nivel johtaa koko käsivarren liikettä ja jossa kyynärpää on asetettu ns. polevektoriin. Siinä ranteen yhä ohjatessa käsivartta kyynärpäätä voidaan johdatella erillisellä vedettävän kaltaisella kontrollilla. Tästä syystä transitiolle on haluttu useimmissa tapauksissa antaa välivaiheessa ääriasento 45°:n kohdalle, jotta animaatiosarjan sulavuutta pystyttäisiin hallitsemaan paremmin transitioissa. [34.]

Tähtäyskulmille 0°, 45 ° ja 90° on olemassa myös erilliset tähtäysversiot ylös ja alas, ja lisäksi niiden on transitioitava keskenään sulavasti. Jos esimerkiksi tähtädään suoraan ylös 90° ja kierretään tähtäystä oikealle 89,9°, hahmon on pystyttävä sulavasti transitioimaan animaatiotasosta aim_up-tasoon aim_right45_up90 ja siitä aina aim_right90_up90-tasoon asti pitäytyen näennäisesti samassa asennossa koko transition ajan. [34.]

Näin ollen lähes jokaisella työkalulla ja aseella niiden jokaisella tilalla on pohja-animaation eteenpäin tähtäävän kerroksen lisäksi seuraavaksi luetellut animaatiokerrokset:

- aim_up
- aim_down
- aim_right45
- aim_right90
- aim_right45_up90
- aim_right90_up90
- aim_right45_down90
- aim_right90_down90
- aim_left45
- aim_left90
- aim_left45_up90
- aim_left90_up90
- aim_left45_down90
- aim_left90_down90.

Tällainen vähittäinen asentojen muutos lisää luonnollisesti testausvaiheen työtä. Ihanteellinen lopputulos on saavutettu, kun hahmo saadaan liikkumaan näennäisen vaivattomasti ja tähtäämään jokaiseen haluttuun suuntaan ilman havaittavia poikkeamia animaatiosarjassa. [34.]

Aseen tähtäykseen liittyvät haasteet ovat toinen tyypillinen osa-alue, johon animaattorin tulee käyttää erityistä huolellisuutta, jotta transitiot pysyisivät mahdollisimman sulavina.

Mutta koska pelimekaniikallisesti on tärkeämpää, että aseen tähtäys osoittaa oikeaan suuntaan, on hahmon luonnollinen asento toissijainen muutamassa harvassa äärita-pauksessa. Kuvassa 35 näytetään hahmo kantamassa raketinheitintä, kun tähdätään suoraan eteen ja kun tähdätään oikealle ja suoraan ylös. Radikaali muutos hahmon asennossa ja hahmon asennon hieman luonnoton asento ovat välttämättömät, kun ha-lutaan säilyttää alkuperäinen asento jokaiseen tähtäyksen suuntaan. [34.]

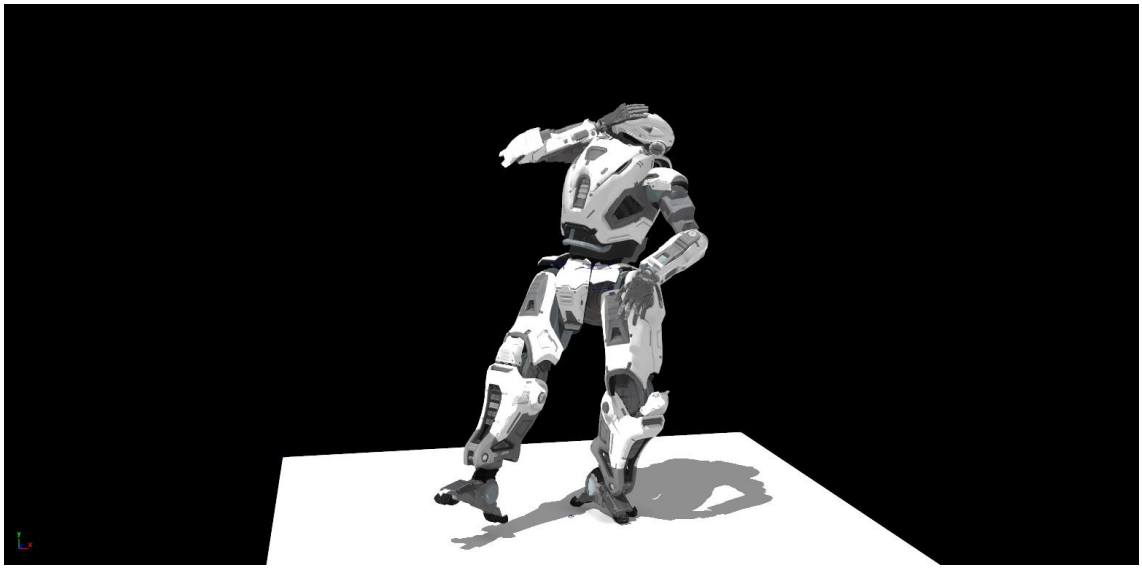
Kuva 35. Hahmon tähdätessä raketinheittimellä tulee koko kehon kääntyä tähtäyksen mukana, sillä ase lepää hahmon hartialla [33].

Edellä esitetty tapaus on harvinainen ääriesimerkki, joka tulee vastaan usein peleissä, joissa tähtäyksen suunta on pelimekaniikallisesti tärkeä. Tällainen hieman luonnoton asento on kuitenkin välttämätön, kun halutaan säilyttää alkuperäinen asento jokaiseen tähtäyksen suuntaan. Animaattorin työ onkin aina mahdollisuuksien mukaan helpottaa hahmon asennon luontevuutta aina, kun se on mahdollista. [34.]

5.2 Emote-animaatiot

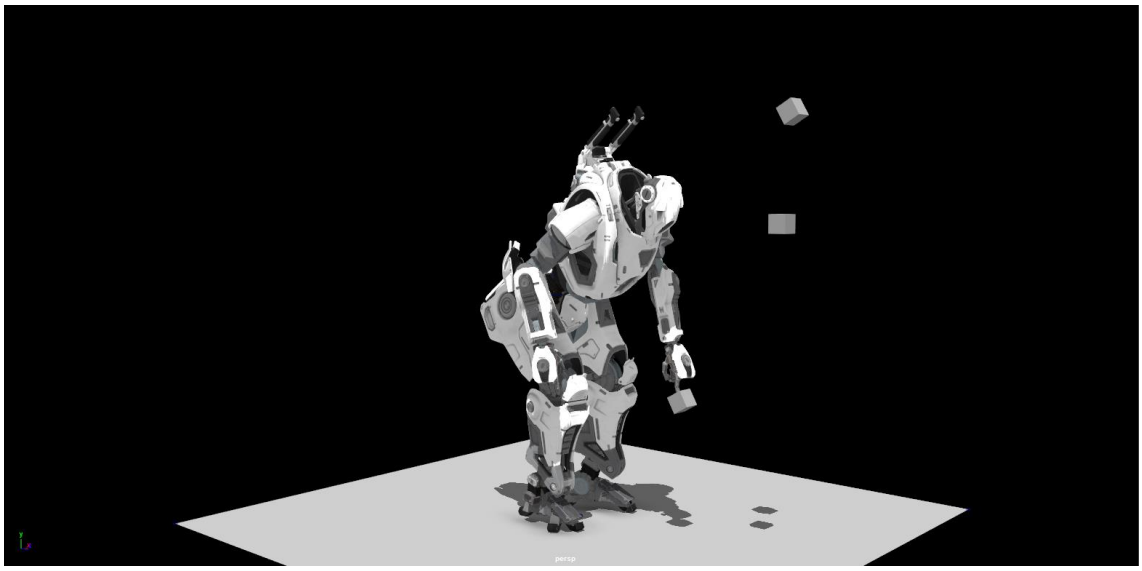
Edellisessä luvussa esiteltiin, kuinka Starbasen animaatiot pohjautuvat liikkeentunnis-tusdataan, vaikka pohja-animaatio tähtäysanimaatioissa on melko neutraali. Elehdintä-animaatioissa puolestaan animaation tulee olla näyttävä, mielenkiintoinen ja hyvin infor-matiivinen. Starbasessa pelihahmoilla ei ole kasvoja tai ihmismäistä ääntä, joten erilaiset

elehdinnät on osoitettava vahvasti keholla. Näin ollen eleiden informatiivisuuden tarve luo kehykset Starbasen emote-animaatioille, sillä eleen viestin on välityttävä toisille pelaajille niin, ettei pelaajille jää epäselväksi, mitä toinen pelaaja eleellään tarkoittaa. Kuvassa 36 pelaajan hahmo on juuri paiskannut kätensä kypärälleen mahdollisimman näyttävästi elehtiäkseen ehkä kykenemättömyyttään ymmärtää, mitä kanssapelaaja juuri teki. [34.]



Kuva 36. Hahmon tulee emote-tyyppisissä animaatioissa elehtiä kehollaan selvästi, mitä haluaa ilmaista.

Tällaisilla animaatioilla vain harvoilla on animaatiokerroksia, koska usein emote-tyyppiset animaatiot käsittävät koko kehon liikehdinnän. Tietyntyyppisille emoteille kuitenkin on Starbasessa sen suljetun alfan aikaan jo luotu animaatiokerroksia. Esimerkiksi sormen painallusta kuvaavalla emotella on parametrit animaation ylös ja alas tähtääviin versioihin (kuva 37). [34.]



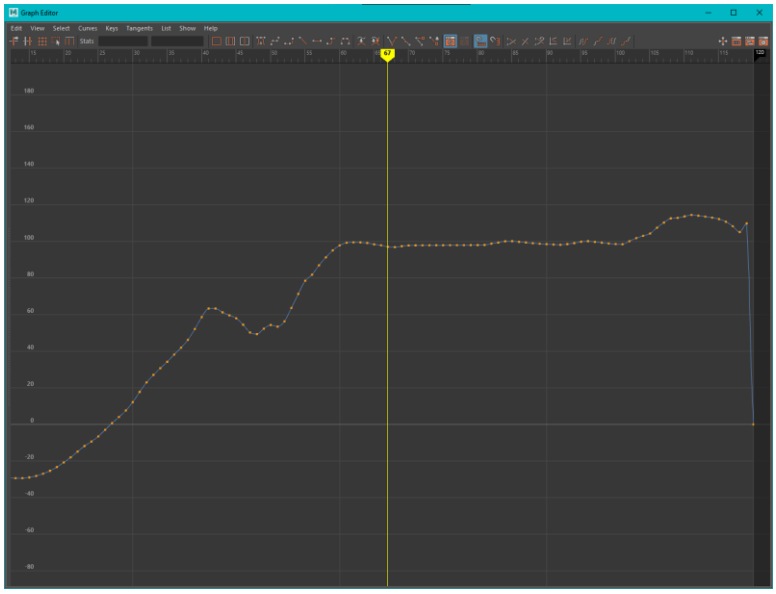
Kuva 37. Starbasessa emote_poke on parametrisoitu, ja sillä voi tähdätä ylös tai alas [33].

Kuten aikaisemmin mainittiin, Starbasen emote-animaatiot alkavat ja loppuvat aina aseettomasta versiosta siitä tilasta, jossa hahmo sillä hetkellä on. Tämä niin kutsuttu aseeton idle on yksi pelissä luultavasti yleisimmin toistuvista animaatioista, joten se on valittu emote-animaatioiden toistamisen alku- ja loppuasennoksi. [34.]

Käytännössä emote-animaatioiden kohdalla tämä tarkoittaa yksinkertaisesti sitä, että mikä tahansa animaatiotila transitioi tilaansa vastaavaan aseettoman tilan ensimmäiseen avainkehykseen, joka on samalla emote-animaation ensimmäinen avainkehys. Emote-animaatio toistetaan, ja animaatio päättyy jälleen aseettoman idlen avainkehykseen, minkä jälkeen hahmo transitioi itsensä takaisin tilaan, jossa emote-animaation käsky annettiin. Starbasessa jos hahmo on esimerkiksi ryömivässä tilassa ase kädessä, ase ja koko hahmo, tai käytännössä vain sen yläruumis, transitioivat ryömimistilan aseettomaan tilaan ja toistavat siitä emote-animaation palaten sitten jälleen asetta kantavaan ryömimistilaan. [34.]

Tämän tyypin animaatioissa on tärkeää, että teki hahmo millaisen animaation vain, se palaa aina takaisin määrättyyn asentoonsa, jotta siirtymä seuraavaan tilaan olisi mahdollisimman sulava. Yksi animaattorin tärkeimmistä tehtävistä emote-animaatioissa on varmistaa, että siirtymä aseettoman idlen avaikehyksestä animaatioon on paitsi sulava, myös tarpeeksi nopea ja mahdollisimman luonnollinen. [34.]

Äärimmäisissä esimerkeissä emote-animaatioita tehdessä liikkeentunnistusnäyttelijä on saattanut esimerkiksi hypätä ilmaan ja palata seisomaan huomattavan matkan päässä alkuperäisestä asemastaan. Näin ollen siirtymä takaisin samaan pisteeseen on mahdotonta luonnollisesti. näissä tapauksissa animaattorin on pidettävä huolta, että koko animaation aikana paikan muutokset eivät kasva liian suuriksi, jotta häivytys alkuperäiseen asentoon voitaisiin toteuttaa mahdollisimman luonnollisesti. Kuvassa 38 nähdään, kuinka hahmon asema 3D-ohjelman koordinaatistoa tarkasteltaessa on muuttunut huomattavasti eikä transitiota voida enää muovata luonnolliseksi ilman perustavanlaatuista muutosta animaation liikehdintään. [34.]



Kuva 38. Z-transitiokäyrästä voidaan havaita, että hahmon liikeratoja on muokattava huomattavasti, jotta luonnollinen siirtymä alkuperäiseen animaatiotilaan voidaan saavuttaa, sillä animaation toiseksi viimeisellä avainkehyksellä on korkea suuri arvo viimeiseen verrattuna.

Starbasen animaatiojärjestelmä varmistaa myös, että emote-animaation voi toistaa asetta kannettaessa. Tätä varten hahmon rintaan on kiinnitetty 3D-ohjelmassa nivel, wpn_loc, johon ase palaa animaation ajaksi. (Kuva 39.) [34.]

Kuva 39. Aseen paikka siirtyy käsistä automaattisesti hahmon rinnalle emote-animaatiota toistettaessa, ellei toisin ole asetettu [33].

5.3 Konfiguroiminen pelimoottoriin ja testaus

Sanotaan, että pelianimaattorin työ on vasta puoliksi tehty, kun animaatio on niin valmis, että se voidaan ensi kertaa tuoda pelimoottoriin. Nykypäivän pelianimaattorin tehtäviin kuuluu myös konfiguroida animaatio pelimoottorissa toimivaksi kokonaisuudeksi ilman koodaajaa. [34.]

Starbase-projektissa tämä tarkoittaa sitä, että animaattori tuo animaationsa, tai sen sarjat, pelimoottoriin, ja mahdollistaa niiden ominaisuudet pelimoottorin sisäisillä työkaluilla. Toimiakseen animaatiot tarvitsevat myös koodipuolta, joten animaattorin seuraava tehtävä onkin luoda järjestelmä animaation taakse mahdollistamaan sen käyttö, kun esimerkiksi tietyt ehdot täyttyvät. Kuvassa 40 on esimerkki pelihahmosta, joka tähtää kohti läheistä nappulaa, joka aktivoi erillisen animaatiosarjan, jossa hahmon käsi nousee osoittavaan asentoon. Tässä uudessa tilassaan se pystyy osoittamaan sormellaan nappulaa ja painamaan sitä aktivoiden käsivarrenpainallusanimaation. [34.]

Kuva 40. Hahmon animaatiojärjestelmä on luotu niin, että tähtäämällä nappulaan hahmo aktivoi uuden animaatiosarjan, jossa hahmo pitää kättään ja sormeaan koholla kohti nappulaa [33].

Pelianimaattorin tehtävänä on varmistaa, että animaation kaikki osa-alueet toimivat, mutta yhtä tärkeää on tutkia. miltä animaatio näyttää ympäristössään. Lähes kaikissa tapauksissa animaatiosta löytyy vielä paljon kohtia, joihin keskittyä, ja prosessiin kuuluu, että animaatiota hiotaan 3D-ohjelman puolella, kunnes siihen ollaan tyytyväisiä. Pelimoottorin animaatiokompressoinnin vuoksi animaatio myös usein poikkeaa hieman alkuperäisestä, ja tasoittaa esimerkiksi pienimmät tärähtelyt animaatiokäyristä pois. [34.]

6 Tulokset

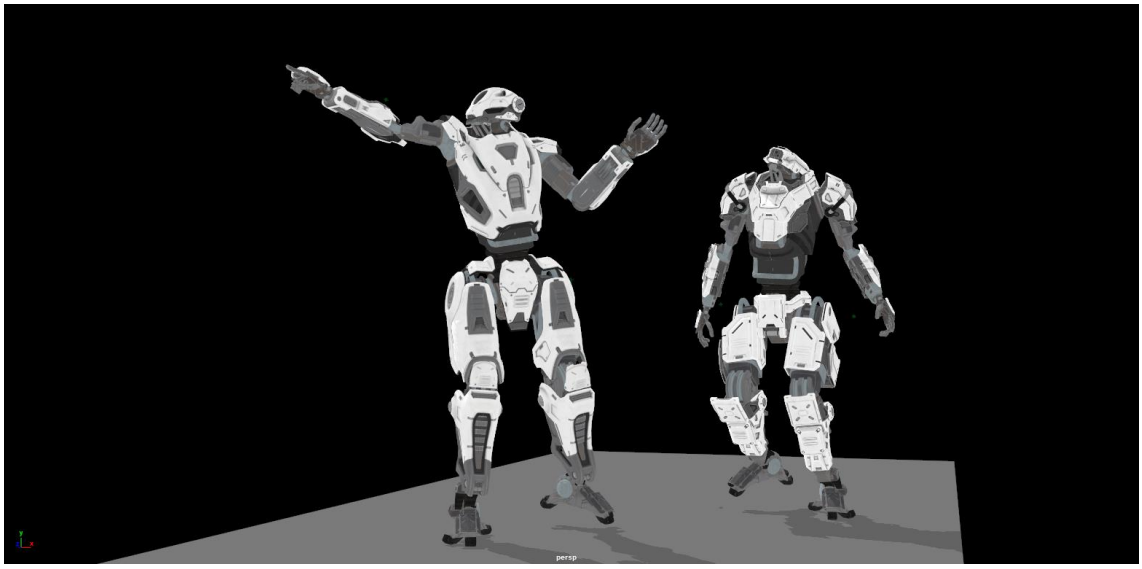
Koska Starbase on MMO-tyyppinen peli, sitä varten tuli tuottaa monentyyppisiä ja eri logiikalla toimivia animaatioita ja animaatiosarjoja, mutta lopulta kaikki Starbasen animaatiot ovat jollain tavalla parametrisoituja, mikä tarkoittaa, että niiden ominaisuuksia voidaan säädellä koodin avulla.

Animaatiopuolen tavoitteena oli tuottaa valmiiksi kaikki tarvittavat animaatiosarjat suurimmalle osalle pelaajille mahdollistetuista aseista ja työkaluista sekä aseettoman tilan hiotut animaatiosarjat. Emote-animaatioita varten kuvattiin liikkeentunnistustekniikalla huomattava määrä hyvin erilaisia ottoja, joista jalostettiin kirjava joukko erityyppisiä emoteanimaatioita. Starbasen emote-animaatioiden on tarkoitus olla hyväntuulisia ja

näyttäviä eleitä. MMO:n kaltaisen pelin pelaajat usein olettavat tietyntyyppisten emoteanimaatioiden olemassaoloa, mutta yleisimpien ja oletetuimpien emoteanimaatioiden lisäksi tiimi tuotti myös lukuisia harvinaisempia ja yllätyksellisempiä animaatioita, kuten leikkimielisiä potkuja, tansseja ja jopa pari temppua, joissa käytettiin hyväksi hahmon sisäisen nivelistön robottimaisia ominaisuuksia.

Tavoitteeseen päästiin, ja kaikki oli valmiina, kun ensimmäiset pelaajat saivat sähköpostitse kutsun liittyä peliin ensi kertaa. Pelissä lähes koko pelitiimi oli heitä vastassa. Pelaajat oppivat hyvin pian avaamaan näkymän, jossa erilaiset käytettävissä olevat emoteanimaatiot on listattu. Animaatiotiimi sai näin ollen pian suoraa palautetta animaatioista. Yksi pelaajista kommentoi emoteanimaatioita englanniksi: ”Kuka ikinä valitsi ja suunnitteli nämä emoteanimaatiot, hän teki oikeita valintoja.” Kaiken kaikkiaan animaatiot saivat hyvää palautetta pelaajilta jo ensimmäisinä pelitunteina.

Lukuisten erilaisten animaatiotyyppien lisäksi pelin suljettua alfaa varten tuli valmistella myös lyhyitä elokuvamaisia kohtauksia. Traileria varten kuvattiin liikkeentunnistusteknologialla useita erilaisia kohtauksia, ja lopulta videoon valikoitui neljä tarinaa johdattelevaa keskeistä kohtausta. Kuvassa 41 esitellään trailer_city-kohtauksesta avainkehys 3D-animaatio-ohjelmassa, ja kuvassa 42 sama kohtaus on esillä valmiissa trailerissa. Opinnäytetyötä tehtäessä traileri on nähtävissä esimerkiksi YouTubessa.



Kuva 41. Starbasen alfa-trailerin kohtaus 3D-ohjelmassa [33].

Kuva 42. Sama kohtaus valmiissa Starbase alfa-trailerissa [32].

Trailerin vastaanotto oli hyvin positiivinen, ja se lisäsi välittömästi keskustelua tulevan suljetun alfan ympärillä. Se nosti välittömästi käyttäjämääriä mm. Starbasen virallisella Discord-kanavalla. Mainittakoon, että opinnäytetyötä tehtäessä kaikki trailerissa nähtävät emoteanimaatiot ja suuri osa tähtäävistä animaatiosarjoista on uudistettu tulevaa seuraavaa julkaisuvaihetta kohti. Tämä on tyypillinen vaihe pelianimaatioalan luonnollista kehitysprosessista, jossa ensimmäisiä versioita usein muovataan tai uusitaan vielä kehityksen loppupäässäkin asti.

Julkaisun seuraavaa vaihetta varten lukuisia animaatioita muokattiin, lisättiin ja korvattiin uusilla. Esimerkiksi aseeton idle-animaatiosarja muutettiin täysin, ja näin ollen emote-animaatiot alkavat uudistetusta poseerauksesta ja loppuvat siihen. Useita uusia emote-animaatioita lisättiin, ja julkaisun seuraava vaihe sisälsi jälleen uusia työkaluja ja aseita, joista kaikille oli tehty omat uudet animaatiosarjansa niiden jokaiselle eri animaatiotasolle. Tavoitteena oli edelleen luoda pelaajille informatiivisia animaatiosarjoja sekä mielenkiintoisia ja selkeästi elehtiviä emote-animaatioita.

7 Yhteenveto

Insinöörityön tarkoituksena oli syventää tekijän omaa tietotaitoa pelianimaation alalla, mutta aivan heti ei ollut selvää, miten laaja aihe parametriset animaatiot on. Parametriset animaatiot paljastuivat hyvin laajaksi aiheeksi, ja ne käsittävät valtavasti erilaisia animaatiotekniikoita. Pelkästään erilaisista logiikoista muodostaa parametrisia animaatioita olisi saanut tehtyä oman opinnäytetyönsä.

Insinöörityötä varten tuotettiin suuri määrä hyvin erityyppisiä parametrisia animaatioita, joista jokainen lopputarkastelussa poikkeaa esimerkiksi toimintalogiikaltaan muista tuotetuista animaatioista ja animaatiosarjoista. Määrällisesti eniten tuotettiin mekaanisia tähtäyssarjoja ja niiden toimintalogiikoita. Tämä johtuu Starbase-pelin ideologiasta vapauttaa pelaajan toiminnallisuus animaatiotilasta riippumatta.

Toinen suuri osa-alue oli emote-animaatiot, sillä esittävän animaation työstäminen, myös liikkeentunnistusdataa käytettäessä, on aikaa vievää. Usein liikkeentunnistusdatasta muokattava elehdintä saa prosessissa vielä monia muutoksia animaatioon, ja monessa tapauksessa uusia osuuksia animaatioon tuli työstää käsin, ilman liikkeentunnistusdataa.

Parametriset animaatiot ovat kiistämättä välttämättömiä nykypeliteollisuudessa, ja ne kehittyvät jatkuvasti. Vuodesta toiseen suuret pelistudiot esittelevät uusia animaatioiden innovaatioitaan, ja varsinkin viime vuosina kehitys on ollut huimaa. Teknologian kehittyessä myös muistin määrä kasvaa, mikä tarkoittaa myös suurempaa lohkoa mahdollisille animaatioille peleissä.

Kaksi asiaa tuskin muuttuu. Peliteollisuus tuottaa nykyään hyvin kaunista ja luonnollista animaatiota ja muokkaa kuluttajan odotuksia, mitä tulee pelin animaatioiden tasoon. Entisajan tökeröt animaatiot ja niiden jäykät siirtymiset ovat menneisyyttä. Tähän liittyen nykypelaajat myös olettavat, että hahmo kykenee samaan aikaan toistamaan useita animaatioita kerralla ja reagoimaan ympäristöönsä aina vain interaktiivisemmin. On selvää, että jo tästä syystä pelistudioiden on rakennettava ja huollettava animaatiojärjestelmänsä hahmojen takana mahdollisimman joustavaksi ja responsiiviseksi, jotta kuluttajan odotukset toteutuvat.

Lähteet

1 Parent, Rick. 2007. Computer Animation: Algorithms and Techniques. 2. painos. E-kirja. Elsevier Science & Technology.

2 Catmull, Ed. 1972. Hand/Face. Verkkoaineisto. <https://www.youtube.com/watch?v=fAhyBfLFyNA&t=186s> Luettu 28.6.2020.

3 Tähtien sota: Episodi IV – Uusi toivo. 1977. Lucasfilm Ltd.

4 Kinney, Charles Jr. 3-D Animation: History & Definition. Verkkoaineisto. <https://study.com/academy/lesson/3-d-animation-history-definition.html> Luettu 19.7.2020.

5 Magnenat-Thalmann Nadia & Thalmann Daniel. 1990. Computer Animation: Theory and Practice. E-kirja. Springer-Verlag.

6 Magnenat-Thalmann, Nadia & Laperriére, Richard & Thalmann, Daniel. 1988. Joint-Dependent Local Deformations for Hand Animation and Object Grasping. Verkkoaineisto. <https://graphicsinterface.org/wp-content/uploads/gi1988-4.pdf> MiraLab, HEC/IRO, University of Montreal.

7 Magnenat-Thalmann Nadia & Thalmann Daniel. 1989. State of the art in Computer Animation: Proceedings of Computer Animation ’89. E-kirja. Springer-Verlag Tokyo 1989.

8 Tosiyasu L. Kunii. 1986. Advanced Computer Graphics: Proceedings of Computer Graphics Tokyo ’86. E-kirja. Tokyo: Springer-Verlag.

9 Peddie, Jon. 2013. The History of Visual Magic in Computers: How Beautiful Images are Made in CAD, 3D, VR and AR. E-kirja. Springer-Verlag.

10 Old Classic Retro Gaming. 2015. Arcade Game: I, Robot (1983 Atari). Verkkoaineisto. <https://www.youtube.com/watch?v=EHkwdvfXHJc&t=2130s> Luettu 22.7.2020.

11 Cooper, Jonathan. 2019. Game anim: video game animation explained. Verkkoaineisto. <https://www.gameanim.com/2005/06/19/blending-the-future-of-non-linear-animation/> Luettu 15.7.2020.

12 Williams, Leah. 2019. Playing Final Fantasy X for the first time in 2019 is fascinating and frustrating. Verkkoaineisto. <https://www.theaureview.com/games/playing-final-fantasy-x-for-the-first-time-in-2019-is-fascinating-and-frustrating/> The Au Rewiev. Luettu 5.9.2020.

13 Call, Joshua; Whitlock, Katie; Voorhees, Gerald A. 2012. Guns, Grenades, and Grunts: First-Person Shooter Games. E-kirja. Continuum International Publishing Group.

14 Team Fortress 2. Verkkoaineisto. Valve Inc. <https://store.steampowered.com/app/440/Team_Fortress_2/?l=finnish> Luettu 5.9.2020.

15 Cooper, Jonathan. 2019. Game anim: video game animation explained. CRC Press Taylor & Francis Group.

16 Walk Cycle Animation Blueprint: A how to tutorial. 2017. Verkkoaineisto. Rusty Animator. <https://rustyanimator.com/walk-cycle-animation/> Luettu 5.9.2020.

17 Edsall, Jerry. 2003. Animation Blending: Achieving Inverse Kinematics and More. Verkkoaineisto. <https://www.gamasutra.com/view/feature/131863/animation_blending_achieving_.php?page=2> Luettu 1.7.2020.

18 Rabin, Steven. 2015. Game AI Pro 2: Collected Wisdom of Game AI Professionals. E-kirja. A K Peters/CRC Press.

19 Gregory, Jason. 2018. Game Engine Architecture. 3.painos. E-kirja. A K Peters/CRC Press.

20 Oud, Richard. 2019. Animation Bootcamp: Bringing Life to the Machines of Horizon Zero Dawn. Verkkoaineisto. <https://www.youtube.com/watch?v=50mIKBNACU> GDC. Luettu 14.7.2020.

21 Almudena, Soria. 2017. Animation Bootcamp: Tricks of the Trade. Verkkoaineisto. <https://www.youtube.com/watch?v=VIgHW4ddHLc> GDC Vault. Luettu 10.7.2020.

22 Kenwright, Ben. 2011. A Beginners Guide to Dual-Quaternions What They Are, How They Work, and How to Use Them for 3D Character Hierarchies. E-kirja. School of Computing Science, Newcastle University.

23 McIntosh, Travis. Animation and Player Control in Uncharted: Drake's Fortune and Uncharted II: Among Thieves. Verkkoaineisto. <https://www.gdcvault.com/play/1012300/Animation-and-Player-Control-in>GDC Vault. Luettu 15.7.2020.

24 Williams, Richard. 2001. The Animator's Survival Kit Expanded edition. Faber & Faber.

25 Toura, Antti & Pettersson, Rasmus. 2012. Reaaliaikaisten Kalman- ja Madgwick-suodinten vertailu kolmiulotteisessa tilapaikannuksessa. Opinnäytetyö. Metropolia Ammattikorkeakoulu. Theseu-tietokanta.

26 Tykkälä, Tommi. 2015. 3D animaatio: liikekäyrät ja interpolointi. Verkkoaineisto. DocPlayer. <https://docplayer.fi/4680616-3d-animaatio-liikekayrat-ja-interpolointi-tommi-tykkala.html> Luettu 23.7.2020.

27 Kuperberg, Marcia. 2012. Guide to Computer Animation. E-kirja. Focal Press.

28 Faure, Francois. Introduction to Parametric Interpolation for Computer Animation. Verkkoaineisto. Evasion-LJK. <https://team.inria.fr/imagine/files/2014/06/interpolation1.pdf> Luettu 19.7.2020.

29 Van Verth, James M. & Bishop, Lars M. 2016. Essential Mathematics for Games and Interactive Applications. E-kirja. CRC Press, Taylor & Francis Group.

30 Unity Manual. 2019. Verkkoaineisto. Unity Inc. <https://docs.unity3d.com/Manual/UnityManual.html> Luettu 3.8.2020.

31 Original Jak Moveset. 2016. Verkkoaineisto. Naughty Dog Inc. <https://twitter.com/Naughty_Dog/status/685314776562577409/photo/1> Luettu 28.7.2020.

32 Starbase. 2020. Verkkoaineisto. Frozenbyte Oy. <https://store.steampowered.com/app/454120/Starbase/> Luettu 18.8.2020.

33 Starbase. 2020. Frozenbyte Oy.

34 Laine, Jaakko. 2020. Pelianimaattori. Frozenbyte Oy, Helsinki. Sähköpostiviesti. 22.9.2020.