Tuomas Porvali

2D-pelihahmon animointi käänteiskinematiikalla

Tietotekniikan pro gradu -tutkielma

28. joulukuuta 2020

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Tuomas Porvali

Yhteystiedot: thetupo26@gmail.com

Ohjaaja: Raino Mäkinen

Työn nimi: 2D pelihahmon animointi käänteiskinematiikalla

Title in English: Animating 2D Video Game Character with Inverse Kinematics

Työ: Pro gradu -tutkielma

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Sivumäärä: 48 + 2

Tiivistelmä: Käänteiskinematiikkaa hyödynnetään teollisuuden robotiikassa ja tietokoneanimaatioissa. Nykyään videopeleissä käänteiskinematiikkaa käytettään valikoidusti pelihahmojen jalkojen asettamiseen virtuaaliympäristön maaston mukaisesti tai kohteen osoittamiseen hahmon kädellä tai katseella. Animaatiot peleissä ovat ennalta tehtyjä, joten ne ovat staattisia ja käytöltään hyvin rajattuja. Käyttämällä käänteiskinematiikka voitaisiin ratkaista aiemmin mainitut ongelmat ja valmistaa hahmoanimaatiota, joka on dynaamisempaa ja interaktiivisempaa. Tässä pro gradussa selvitettiin käänteiskinematiikan hyödyntämistä ihmismäisen 2D pelihahmon animoimisessa ja valmistettiin prototyyppi, jossa pelihahmo animoidaan automaattisesti hyödyntäen vain ainoastaan käänteiskinematiikkaa. Animaatio on perustettu ihmisen biomekaniikkaan ja 12 animaation perusperiaatteen hyödyntämiseen. Prototyyppikomponentti kehitettiin Unity-pelimoottoriin käyttäen sen työkaluja ja jo olemassa olevaan pelikoodiin. Prototyypissä pelihahmo kykenee kävelemään ja juoksemaan tasaisia tasoja ja 45-asteen ramppeja ja portaita.

Avainsanat: tietokonegrafiikka, pelianimaatio, videopelit, käänteiskinematiikka

Abstract: Inverse kinematics is used in industry robotics and computer animation. In video games inverse kinematics is used to adjusting video game characters legs on the level of virtual surface terrain or pointing at their targets by their hand or gaze. Animations in games are pre-baked and therefore are static and have limited uses. By using inverse kinematics, earlier mentioned problems could be solved and would create more dynamic

and interactive character animations. This thesis examines use of inverse kinematics in animating human-like 2D game character and produces a prototype that exclusively uses inverse kinematics to animate character automatically. Animations are based on human biomechanics and use of 12 principles of animation. The produced prototype component is developed for Unity game engine using its tools and already existing game code. The prototype is capable to animate character's gait and run on even leveled terrain and 45-degree ramps and stairs.

Keywords: computer graphics, game animation, video games, kinematics, inverse kinematics

Termiluettelo

IK	Lyhenne sanasta Inverse Kinematics (käänteiskinematiikka).
FK	Lyhenne sanasta Forward Kinematics (suorakinematiikka).
CCD	Cyclic Coordinate Descent. Heuristinen käänteiskinematiikka algoritmi.
FABRIK	Forward and Backward Reaching Inverse Kinematics. Heuristinen käänteiskinematiikka-algoritmi.
C#	Ohjelmointikieli. Lausutaan “C sharp”.
Animaatio	Kuva kuvalta toteutettu illuusio hahmon liikkeestä sen katsojalle.
Biomekaniikka	Tieteenala, jossa tutkitaan ihmisen tai muun elollisen liikkumismekaniikoita.
Delta-aika	Kulunut aika edellisestä päivityskerrasta ohjelmassa.
Jacobin Matriisi	Matriisi, jolla lasketaan yhden muuttujan muutosvaikutus koko funktioon.
Juuriliike	(eng. root motion) Hahmoanimaatio, joka muuntaa sen juuripistettä. Muuttaa pelihahmon sijaintia tai kokoa virtuaaliympäristössä.
Juuripiste	(eng. root joint) kinematiikka ketjun alkuosa. Pelihahmon karteesinen piste sijainti virtuaaliympäristössä.
Kinematiikka	Liikkeiden tutkimista, ilman niiden fyysisiä ominaisuuksia tai ulkoisia vaikutuksia.
Kinematiikkamalli	Konkreettinen tai ohjelmallinen malli, jolla tutkitaan liikkeiden vaikutuksia isoon kokonaisuuteen.
Keyframe-animaatio	Työkalu, jolla tehdään animaatiota aikajanalle kuva kuvalta hyödyntäen suoraa kinematiikkaa asettaen hahmon asentoja tai käsin piirtäen hahmon asentoja.
Käänteiskinematiikka	Matemaattinen prosessi, jolla lasketaan kinematiikkaketjun nivelille kulmat, jotta sen pääte-efektori saadaan haluttuun pisteeseen karteesisessa avaruudessa.
Interpolointi	Animaatiotekniikka, jossa ohjelmallisesti rakennetaan animoitavan hahmon alku- ja tavoiteasennosta puuttuvat tilat.
Liukuhihnaliike/Luistelu	(eng. treadmill motion) Hahmoanimaatio, jolla ei ole

	vaikutusta juuripisteeseen. Ei muuta pelihahmon sijaintia tai kokoa virtuaaliympäristössä.
Limb	CCD algoritmiin perustuva käänteiskinematiikka ratkaisija-algoritmi. Suunniteltu kolminivellisille raajoille.
Motion Capture	Liikkeen kaappaus. Animaatiotekniikka, jossa näyttelijän liikkeet nauhoitettaan animoitavalle hahmolle.
Pelimoottori	Ohjelmisto, joka sisältää erilaisia komponentteja (grafiikan piirtäminen, fysiikan mallinnus, äänimikseri, skriptimoottori, jne.) videopelien rakentamiseen.
Pääte-efektori	Kinematiikan ketjun pääteosa.
Riggaaminen	(eng. rigging) Luiden asentaminen tahkoon.
Skeleton Animation	Animaatiotekniikka, jossa hahmolle rakennetaan animoitava luuhierarkia. Verrattavissa kinematiikkamalliin.
Sprite Sheet	Tiedosto, joka sisältää sarjan kuvia, joita yksikerrallaan nopeasti näyttämällä luo katsojalle animaation.
Suorakinematiikka	Prosessi, jossa asetetaan kinemaattisen ketjun nivelet haluttuun kulmaan riippumatta pääte-efektorin lopullisesta sijainnista.
Tahkosto	(eng. mesh) Vertekseillä koostettu 3D-geometria. Käytettään 3D-malleissa.
Unity	Pelimoottori.
UV-piste	Tekstuurin tai kuvan pikselikoordinaattitieto tahkostossa.
Valmisanimaatio	(eng. baked animation) Etukäteen, 3D-sovelluksessa tehty staattinen animaatio.
Vapausasteet	(eng. degrees of freedom) Kaikki muuttujat, jotka vaikuttavat kinematiikkaketjun asentoon.
Verteksi	(eng. vertex) 3D-koordinaatitieto, käytettään tahkostoissa ja tietokonegrafiikoissa.

Kuviot

Kuvio 1. Yksinkertainen kinematiikkaketju. 7
Kuvio 2. CCD-algoritmin yksi iteraatio 10
Kuvio 3. FABRIK-algoritmin yksi iteraatio. 12
Kuvio 4. Ihmisen kinematiikkamalli. 15
Kuvio 5. Yksinkertaistettu kinematiikkamalli sivusta katsottuna. 16
Kuvio 6. Pelihahmo osissa ja koottuna. 16
Kuvio 7. Hahmo rigattu aiemman luvun (kuvio 5.) kinematiikkamallin mukaisesti. 17
Kuvio 8. Tahkoston alueet ovat värjätty samalla värillä, kuin niitä manipuloiva luu. 18
Kuvio 9. Sprite Skin ja IK-manager2D -komponentit Unityssa. 19
Kuvio 10. Juuresta (root) alkava luuherarkia. 20
Kuvio 11. IK-ohjaus -komponentti Unityn editorissa. 21
Kuvio 12. Esimerkki Bezier-käyrä GeoGebra-laskimessa. 23
Kuvio 13. Bezier-käyrän pisteet hahmotettu kuviossa valkoisella ja keltaisella viivalla. 25
Kuvio 14. Kinematiikan mallin sovitus. 27
Kuvio 15. Hahmo kävelee mäkeä ylös ja jalkaterä on asetettu tason kaltevuuden mukaisesti. 27
Kuvio 16. Jalan liikerataa on korjattu tulevaa ylämäkeä varten. 28
Kuvio 17. Hahmon oikean jalan liikerata. Vihreä viiva on raycast-funktion säde. 30
Kuvio 18. Hahmo kävelee portaissa. 31
Kuvio 19. Hahmon oikean jalan liikerata takaperin kävelyssä. 32
Kuvio 20. Hahmo tasapainottelee reunalla. 32
Kuvio 21. Hahmo on tarrautunut reunaan ja ottaa tukea jaloilla. 33
Kuvio 22. Hahmon hyppimistoiminta. 34
Kuvio 23. Kolmijalkainen pelihahmo. Askeleet toimivat yllättävän hyvin ilman ylimääräistä työtä. 44

Taulukot

Taulukko 1. Pelihahmojen määrän vaikutus FPS:ään. 43

Sisältö

1 Johdanto 1

2 Tutkimuksen tavoite 3

2.1 Ihmisen kävelyanimaatio 4
2.2 2D-animaatio peleissä 4
2.3 Käänteiskinematiikka videopeleissä 5
3 Kinematiikka 7

3.1 Kinematiikka-algoritmit 8
3.2 Analyyttiset ratkaisijat 8
3.3 Numeeriset ratkaisijat 9
3.4 Heuristiikkakäänteiskinematiikka -algoritmit 9
3.5 Tekoälyn soveltaminen 13
4 Unity & 2D-käänteiskinematiikkapaketti 14

4.1 2D-käänteiskinematiikkapaketti 14
5 Hahmomalli 15

5.1 Kinematiikkamalli 15
5.2 Hahmomallin riggaus 16
6 Ohjelman komponentit 21

6.1 Hahmo-ohjaus 21
6.2 Käänteiskinematiikan ohjauskomponentti 22
7 Askeleiden toteutus 23

7.1 Askeleen laskeminen 23
7.1.1 Askeleen toiminta 25

7.2 Kinematiikkamallin sovitus ympäristöön 26
7.3 Interpolointi 28
8 Hahmon liikkumismekaniikat 30

8.1 Kävely ja juokseminen 30
8.2 Käsien heilunta ja osoittaminen 32
8.3 Muu toiminta 33
9 Animaatio ja perusperiaatteet 35

9.1 Disneyn 12 animaation perusperiaatetta 35
9.1.1 Venytys ja litistys 35
9.1.2 Ennakointi 35
9.1.3 Näyttämöllepano 36
9.1.4 Improvisoitu liike sekä analysoitu liike 36
9.1.5 Epäsynkronia 36
9.1.6 Hidastuminen ja kiihtyminen 36
9.1.7 Kaaret 37
9.1.8 Toissijainen liike 37
9.1.9 Rytmitys 37
9.1.10 Liioittelu 37

9.1.11 Anatomian ymmärrys 38
9.1.12 Hahmon kiinnostavuus 38
9.2 Virtuaalisensaatio peleissä 38
10 Kohdatut haasteet ja ongelmat 40
10.1 Vanhat tavat 40
10.2 Hahmon liikkuminen portaissa 40
10.3 Uuden teknologian ongelmat 41
11 Työn tulos 42
11.1 Suorituskyky 43
11.2 Jatkokehitys ja -työ 44
11.2.1 Laajennus 3D-animaatiohon ja pelianimaation ulkopuolelle 44
11.2.2 Askeleiden optimointi ja reaktiivinen liike 44
Lähteet 46
Liitteet 49

1 Johdanto

Käänteiskinematiikka on kinematiikkayhtälöiden käyttämistä kinematiikkaongelman ratkaisemiseksi, jossa kinematiikkaketjun nivelille lasketaan kulmat, jotta sen pääte-efektori saadaan siirrettyä haluttuun sijaintiin hyväksyttävässä asennossa. Käänteiskinematiikkaa käytettään laajalti tietokone grafiikoissa, videopeleissä ja teollisuuden robotiikassa. Videopeleissä käänteiskinematiikkaa käytetään fysiikka simulaatioissa ja apuvälineenä tietokoneanimaatioissa (Aristidou, Lasenby, Chrysanthou & Shamir, 2018).

Nykyään animaatiot videopeleissä ovat valmisanimaatioita (eng. baked animation), jotka ovat tehty etukäteen 3D-mallinnussovelluksissa tai animaatiosovelluksissa ja hyödyntävät ensisijaisesti ’skeleton animation’- ja ’keyframing’-tekniikkaa ja suorakinematiikkaa. Koska videopelit ovat hyvin interaktiivisia sovelluksia, jossa pelaaja virtuaalihahmolla reagoi virtuaaliympäristöön, niin hyvin animoidulla hahmolla saavutetaan miellyttävä vuorovaikutus pelaajan ja pelin välillä. Valmisanimaatiot ovat käytöltään ovat hyvin rajallisia interaktiivisuudeltaan (Ruuskanen, 2018). Ne ovat staattisia (rajallinen vuorovaikutus virtuaaliympäristöön) ja niitä ei voi käyttää fysiikkasimulaatioissa ja niistä on vaikea tehdä variaatioita pelin sisällä ja sen ajon aikana.

Yleensä peleissä käänteiskinematiikka käytetään pelihahmojen jalkojen asettamiseen tason pinnan mukaisesti pelihahmon paikallaanolon aikana tai käsien asettamaiseen tasoa vasten tai kohteen osoittamiseen. Mutta käänteiskinematiikkaa ei hyödynnetä, kun pelihahmot kävelevät tai juoksevat aktiivisesti virtuaaliympäristössä. Varsinkin portaat, mäet ja muut epätasaiset ja kompleksiset tasot, jotka hankaloittavat pelihahmojen kävelyä, käyttävät valmisanimaatioita, joissa ei huomioida käveltävän tason muotoa. Portaille ja mäille on yleensä tehty omat kävelyanimaatiot, jotka suunniteltu tietylle kaltevuudelle. Koska animaatiot ovat suunniteltu tietylle kaltevuudelle, ne myös rajoittavat virtuaaliympäristön suunnittelua. Tämän kaltaiset animaatiot useissa peleissä vaikuttavat hyvin konemaisilta, joka ilmenee voimakkaasti, kun hahmo vaihtaa porraskävelyanimaatiosta normaaliin kävelyanimaatioon, sen muutos ei ole sulava. Myös ns. vaikutelma pelihahmon luistelusta virtuaaliympäristössä, jossa pelihahmon jalkojen liike ja askeleet kävelyanimaatiossa ei

vastaa varsinaista liikettä suhteessa muun virtuaaliympäristön kanssa, on vielä yllättävän yleinen pelianimaatioissa tänä päivänä.

2 Tutkimuksen tavoite

Tutkimuksessa on tarkoitus selvittää käänteiskinematiikan käyttöä 2D-animaatiossa ja kehittää komponentti pelihahmon animoimiseksi. Tutkimus rajoittuu vain 2D-animaatioihin, mutta sovellus toteutetaan vektorilaskennalla mahdollista tulevan 3D-animaatiota varten. Käänteiskinematiikka-animaatiolla pyritään saavuttamaan parempaa interaktiivisuutta virtuaalihahmon ja -ympäristön kanssa.

Pelianimaation tulisi toimia Unity-pelimoottorissa (Unity) käyttäen sen työkaluja ja erikseen ladattavaa käänteiskinematiikkapakettia. 2D-käänteiskinematiikka on uutta Unity-pelimoottoriin. Työssä ohjelmointikielenä käytetään C#-kieltä ja Visual Studiota koodieditorina. Vaikka Unity jo sisältää keyframe-animaatiojärjestelmän, sillä ei voi tehdä animaatioita, johon tutkimuksessa pyritään. Unityn animaatiojärjestelmä perustuu keyframe-animaatioon, joten tässä tapauksessa on vaikeaa tehdä animaatiota ilman vaikutelmaa pelihahmon ‘luistelemisesta’. Keyframe-animaatio hyödyntää suoraa kinematiikkaan, jossa hahmon raajat asetetaan etukäteen haluttuun asentoon. Käänteiskinematiikkaa käytetään yleensä keyframe-animaatiossa aputyökaluna.

Animoitava pelihahmo on ihmismäinen. Sillä on jalkapari ja käsipari. Halutut animaatiot ovat: seisominen, käveleminen eteenpäin ja taaksepäin, juokseminen, hyppiminen, roikottaminen ja kyykistyminen. Joten käänteiskinematiikkaa tulee hyödyntää koko hahmon kehossa. Haluttu animaatiojärjestelmä tulisi toimia Unityssä ja kuuluisi selviytyä portaissa ja enintään 45-asteen kaltevan tason rampeissa. Hahmon animaation on oltava uskottava sen katsojalle ja pelihahmon liikkeessä ei luistelun vaikutelmaa. Täysin realistisiin tai todelliseen maailmaan mukautuvaan animaatioon ei ole tarkoitus pyrkiä, koska jopa nykyaikaisissa tietokoneissa se vaatii hyvin paljon laskentatehoa ja on aikaa vievää toteuttaa. Sen sijaan uskottavan animaation luomiseksi hyödynnetään Disneyn 12 perusanimaation sääntöä (Thomas & Johnston, 1981, s.47-69) ja tutkimuksia liittyen ihmisen anatomiaan.

Työssä valmistettava komponentti luo animaation automaattisesti eli käytetään vähän tai ei ollenkaan etukäteen valmistettua animaatiota. Myös animaatiotyökaluja ei käytetä kuten esim. liikkeen kaappausta (motion capture) tai Unityn animator -työkalua. Komponentti

sisältää säätömahdollisuuksia animaation vaihtelevuuden ja persoonallisuuden luomiseksi. Automaattisella animaatiolla voidaan vähentää vaadittavaa työmäärää ja komponentti on helposti uudelleen käytettävissä eri hahmoissa.

2.1 Ihmisen kävelyanimaatio

Ihmisen kävelyanimaation rakentaminen on tutkimallisesti haastavaa ja siihen sekoittuu useita muita aloja ohjelmistotekniikan lisäksi. Alat ja ihmiset, jotka ovat kiinnostuneet ihmisen kävelyanimaatiosta, ovat mm. robotiikan kehittäjät, urheilijat ja terveystiedon tutkijat. (Multon, France, Cani-Gascuel & Debunne, 1998).

Animaatio jaetaan laskentateholtaan kahteen käyttötarkoitukseen: Korkean laskennan tehon animaatioon, joka on lähes verrannollinen simulaatio aidosta ihmisen kävelystä ja matalan laskennan tehon animaatioon, jota käytettään peleissä ja sarjakuvamaisessa animaatiossa, jossa realismiin pyrkiminen ei ole tavoite (Multon, France, Cani-Gascuel & Debunne, 1998).

Yleisiä ongelmia kävelyanimaation rakentamisessa ovat sen uskottavuuden ylläpitäminen katsojalle, askelkontakti tasoissa, kävely epätasaisilla pinnoilla esim. kaltevat tasot ja portaat, liikkeen mukauttaminen eri nopeuksille ja sen personoiminen eri hahmoille sekä reagointi ulkoisille voimille ja hahmon taakan simulointi. Näiden ongelmien tärkeys vaihtelee halutun sovelluksen tavoitteiden mukaan. Koska tässä työssä käsitellään pelianimaatiota, työn tärkeät alueet ovat uskottavuus, askelkontakti, kävely erilaisissa tasoissa ja liikkeen mukauttaminen eri nopeuksille. Myös muita ratkaistavia ongelmia kävelyanimaatiossa ovat aiemmin mainittu hahmon liukuminen ja askeleiden läpäisy käveltävien tasojen läpi (Multon, France, Cani-Gascuel & Debunne, 1998).

2.2 2D-animaatio peleissä

Yleisin 2D-animaatio tekniikka peleissä on piirtää pelihahmon animaatio kuva kuvalta (eng. frame-by-frame), eli animaatio tehdään käsin ja on verrattavissa perinteiseen paperilla tehtyyn animaatioon. Tämänkaltainen animaatio paketoidaan ’Sprite Sheet’-kuvatiedostoon, joka toistetaan pelissä nopeasti kuva kerrallaan sen katsojalle (Rantala 2013).

Aiemmissa peliprojekteissa käytettiin vain ja ainoastaan valmisanimaatioita (eng. baked animation). Ihmishahmoihin siirryttäessä valmisanimaation rajat alkoivat ilmentyä konemaisina toiston takia sekä hahmon toimintaa ja liikeitä rajoittavina. Valmisanimaatiot ovat staattisia, eli niitä ei voi käyttää pelimoottorin fysiikkasimulaatiossa, mukauttaa eri nopeudelle ja niihin on vaikea yhdistää muita toimintoja esim. partikkeliefektejä (Rantala 2013). Myös haluttujen animaatioiden määrä projekteissa alkoi kasvaa, koska pelkästään kävelyanimaatiosta löytyy monta variaatiota. Esim. jos haluttaan hahmon kävelevän kaltevissa tasoissa ja portaissa, niin jokaiseen erilaiseen kaltevaan tasoon tarvitaan oma animaatio sekä ylöspäin, että alaspäin kävelemiseen ja eri kävely nopeuksille myös tarvitaan omat animaatiot. Valmisanimaation suurin etu on tietenkin, että pelihahmo toistaa sille annetun animaation tismalleen animaattorin työn mukaisesti sekä on toistettavissa ja uudelleenkäytettävissä helposti.

2.3 Käänteiskinematiikka videopeleissä

Suoraa kinematiikkaa käytettään pelianimaatiossa apuvälineenä haluttujen pelihahmon asentojen löytämiseen. Ohjelmallisesti tätä tekniikkaa käytetään rakentamalla sarja valmiita asentoja kinematiikkaketjuista, jossa siirtyminen asennosta toiseen interpoloimalla luo animaation. Tätä tekniikkaa kutsuttaan ’Skeleton animation’:ksi (Ruuskanen 2018).

Käänteiskinematiikkaa käytetään animaatioapuvälineenä esim. videopeleissä hahmonjalkojen asettamiseen tason pinnan mukaisesti, joka tapahtuu käyttämällä analyyttisiä menetelmiä. 2D-animaatioissa käänteiskinematiikkaa ei yleensä nähdä, mutta sen käyttäminen on mahdollista, jos pelihahmolle on rigattu kinematiikkamalli (Ruuskanen 2018).

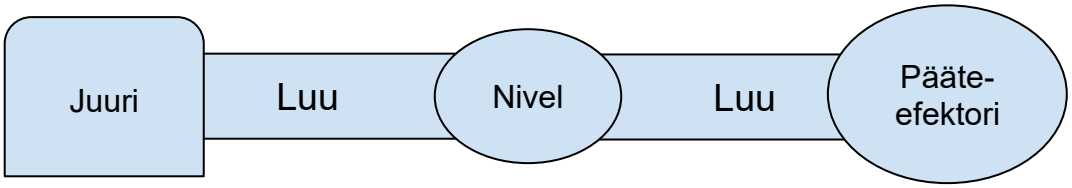
Toinen tapa luoda animaatiota käänteiskinematiikalla on proseduraalinen animaatio. Proseduraalinen animaatio rakentuu useista osista, mutta ensisijaisesti se simuloi pelihahmon liikkeittä, massaa ja ulkoisia voimia ja on siten enemmän fysiikkasimulaatiota kuin varsinaista hahmoanimointia. (Swink, 2017, s.158). Proseduraalisessa animaatiossa siis huomioidaan ominaisuuksia, joita ei tarvita kinematiikassa. Peleissä proseduraalista

animaatiota käytetään pelihahmon vaateiden, hiuksien ja räsynukkejen simulointiin animaatioissa ja muissa fysiikkasimulaatioissa. (Ruuskanen 2018)

3 Kinematiikka

Kinematiikka on klassisiin mekaanikoihin kuuluva liikeoppi, joka kuvaa kappaleen tai useista kappaleista koostuvia järjestelmien siirtymistä ja kiertoa riippumatta niiden koosta, massasta tai ulkoisista fysikaalisilta voimista.

Kinematiikassa useista kappaleista koostuvia järjestelmiä kutsuttaan kinematiikka ketjuiksi ja siitä suurempi järjestelmä on kinematiikkamalli. Kinematiikan ketjun osat koostuvat juuresta, nivelistä, luista ja pääte-efektorista. Näitä osia myös kutsutaan jäseniksi. Juuri tai kiintopiste on kinematiikka ketjun alku ja on yleensä staattinen tai liikkuu osana muuta kokonaisuutta. Kinematiikan loppupäätä kutsutaan pääte-efektoreiksi. ja ketjulla voi olla useita kuin yksi pääte-efektoreita.



Kuvio 1. Yksinkertainen kinematiikkaketju.

Kinematiikka ketjun rajoitteita mallinnetaan vapausasteilla ja globaalirajoitteilla. Vapausasteet (Degrees of Freedom) ovat koko järjestelmän kinematiikkaketjujen luiden ja nivelien kaikki mahdolliset muuttujat, joilla voidaan vaikuttaa pääte-efektorin lopulliseen sijaintiin. Vapausasteilla tarkoitetaan pääte-efektorin kykyä liikkua (translaatio) ja kiertää (rotaatio) X, Y ja Z-akseleissa. Vapausasteen arvo on kinematiikkaketjun nivelten kokonaismäärä vähennettynä yhdellä. Jos niveliin sisältyy rajoitteita, niin ne vähennetään aiemmin lasketusta kokonaismäärästä. Globaalirajoitteilla tarkoitetaan, että yhden nivelen liike vaikuttaa muiden nivelten rajoiteihin muuttaen niiden rajoitteita. Esim. biomekaniikassa ihmisen etusormen uloimman pään taivuttaminen liikuttamatta etusormen muita niveliä on vaikeaa. (Aristidou, Lasenby, Chrysanthou & Shamir, 2018).

3.1 Kinematiikka-algoritmit

Kinematiikkaketjua manipuloidaan suora- ja käänteiskinematiikka-algoritmeilla. Suorakinematiikka yksinkertaisesti on kinematiikan ketjun asennon laskeminen ennalta asetettujen nivelkulmien θ avulla. Lopputuloksena saadaan pääte-efektori johonkin sijaintiin karteesisessa koordinaatistossa.

$$f(\theta) = e,$$

missä funktio f mallintaa suorakinematiikka-algoritmia.

Käänteiskinematiikka on suorakinematiikan vastakohta. Se on matemaattinen prosessi, jossa lasketaan robotin tai animoitavalle hahmon kinematiikkaketjun nivelille kulmat, jotta saadaan sen pääte-efektori mahdollisimman tai tarpeeksi lähelle haluttuun pisteeseen karteesisessa avaruudessa ja vaihtoehtoisesti myös halutussa asennossa.

$$f^{-1}(e) = \theta,$$

Käänteiskinematiikkafunktio f on epälineaarinen ja f^{-1} ei ole yksikäsitteinen, Usein saadaan monia ratkaisuja, joista yleensä halutaan se, mikä luo sujuvimman liikkeen. Erilaisia käänteiskinematiikka algoritmeja on kehitetty, joilla haetaan haluttuja ratkaisumalleja varsinkin, kun käytetään rajoitteita nivelien yliliikkumisen välttämiseksi tai vähentämiseksi. Ratkaisutavat jaetaan analyyttisiin ratkaisuihin ja numeerisiin ratkaisuihin (Aristidou, Lasenby, Chrysanthou & Shamir, 2018).

3.2 Analyyttiset ratkaisijat

Analyyttiset ratkaisijat etsivät kaikki mahdolliset ratkaisut ja niihin voi sisällyttää globaaleja rajoitteita. Ratkaisijat laskevat suoran ratkaisun käyttäen matemaattisia geometrisia ja algebrallisia menetelmiä. Analyyttisia ratkaisuja käytettään robotiikassa, jossa tiedetään robotin rakenne ja sen työtila, joten jokaiselle erilaiselle kokoonpanolle voidaan johtaa omat ratkaisut tai yksittäiset käyttötapaukset, jos vapausasteen lukumäärä on yhtä suuri kuin kinematiikka ketjun nivelten määrä. Jos vapausasteiden lukumäärä on suurempi, kuin nivelliitosten määrä, niin ratkaisuja on äärettömästi ja analyyttista ratkaisua

ei ole, jolloin on hyödynnettävä numeerisia ratkaisijoita (Aristidou, Lasenby, Chrysanthou & Shamir, 2018).

3.3 Numeeriset ratkaisijat

Numeeriset ratkaisijat ovat. iteratiivisia, jossa ratkaisua etsitään toistamalla ratkaisija-algoritmia. Numeeriset ratkaisijat ovat hitaampia kuin analyyttiset ratkaisijat ja niitä käytetään, kun analyyttinen ratkaisu on vaikea tai mahdoton toteuttaa. Numeeristen ratkaisijoiden käyttöönotto on myös helpompaa. Tässä työssä päädyttiin käyttämään numeerisia heuristiikka-algoritmeja (Aristidou, Lasenby, Chrysanthou & Shamir, 2018).

Numerisilla ratkaisioilla etsitään tehtäville

$$F(\theta) = f(\theta) - e \qquad (*)$$

tai

$$minimoi \ \|F(\theta)\|^2 \qquad (**)$$

likiratkaisua. Numeeriset ratkaisijat ovat usein Newton tai kvasi-Newton tyyppisiä ja ne tarvitsevat funktion F osittaisderivaattoja θ:n komponenttien suhteen. Nämä osittaisderivaatat voidaan laskea käsin tai niitä voidaan approksimoida äärellisillä differensseillä. Myös tietokoneohjelmien automaattista derivointia voidaan hyödyntää (Haslinger & Mäkinen, 2003).

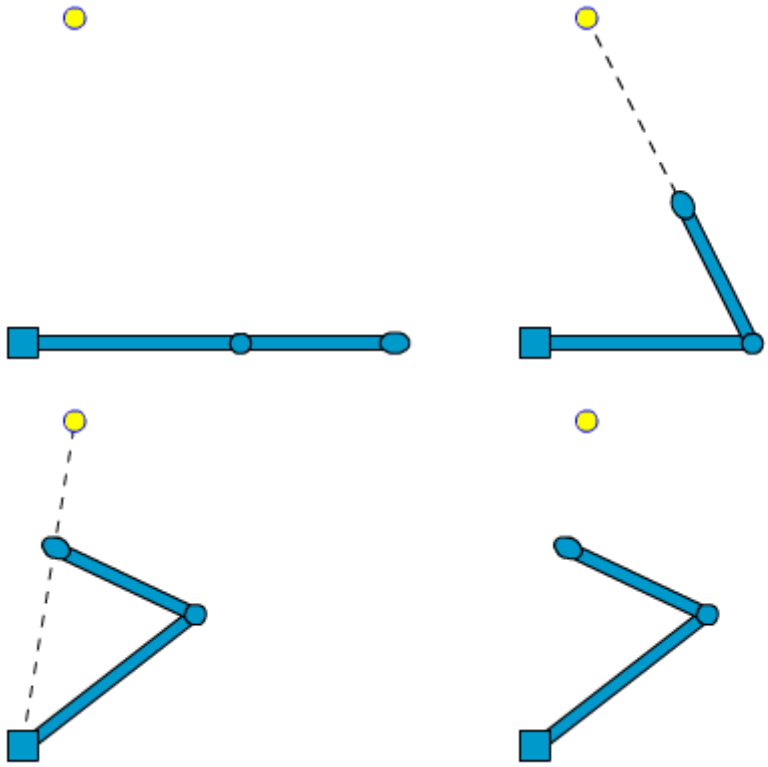
Formuloinnin (**) etuna on, että siihen voidaan lisätä rajoitteita. Koska tehtävän (*) ratkaisu ei yleensä ole yksikäsitteinen, niin lisäämällä rajoitteita tehtävään (**) voidaan sulkea pois sellaisia θ, jotka ovat epäfysikaalisia tai muuten ei-toivottuja.

3.4 Heuristiikkakäänteiskinematiikka -algoritmit

Vaikka aiemmin mainittiin analyyttisen menetelmien käyttöä pelihahmojen jalkojen asettamista tason pinnan mukaisesti, tässä työssä tutkitaan animaation rakentamista käänteiskinematiikalla, kun pelihahmo liikkuu aktiivisesti virtuaaliympäristössä. Koska pelit ovat käytöltään reaaliaikaisia interaktiivisia sovelluksia, jossa virtuaalihahmon raajojen rakenne, määrä, nopeudet ja virtuaaliympäristöt voivat olla mielivaltaisia, niin työn animaation toteuttamiseen käytetään laskenta teholtaan alhaisia heuristiikka

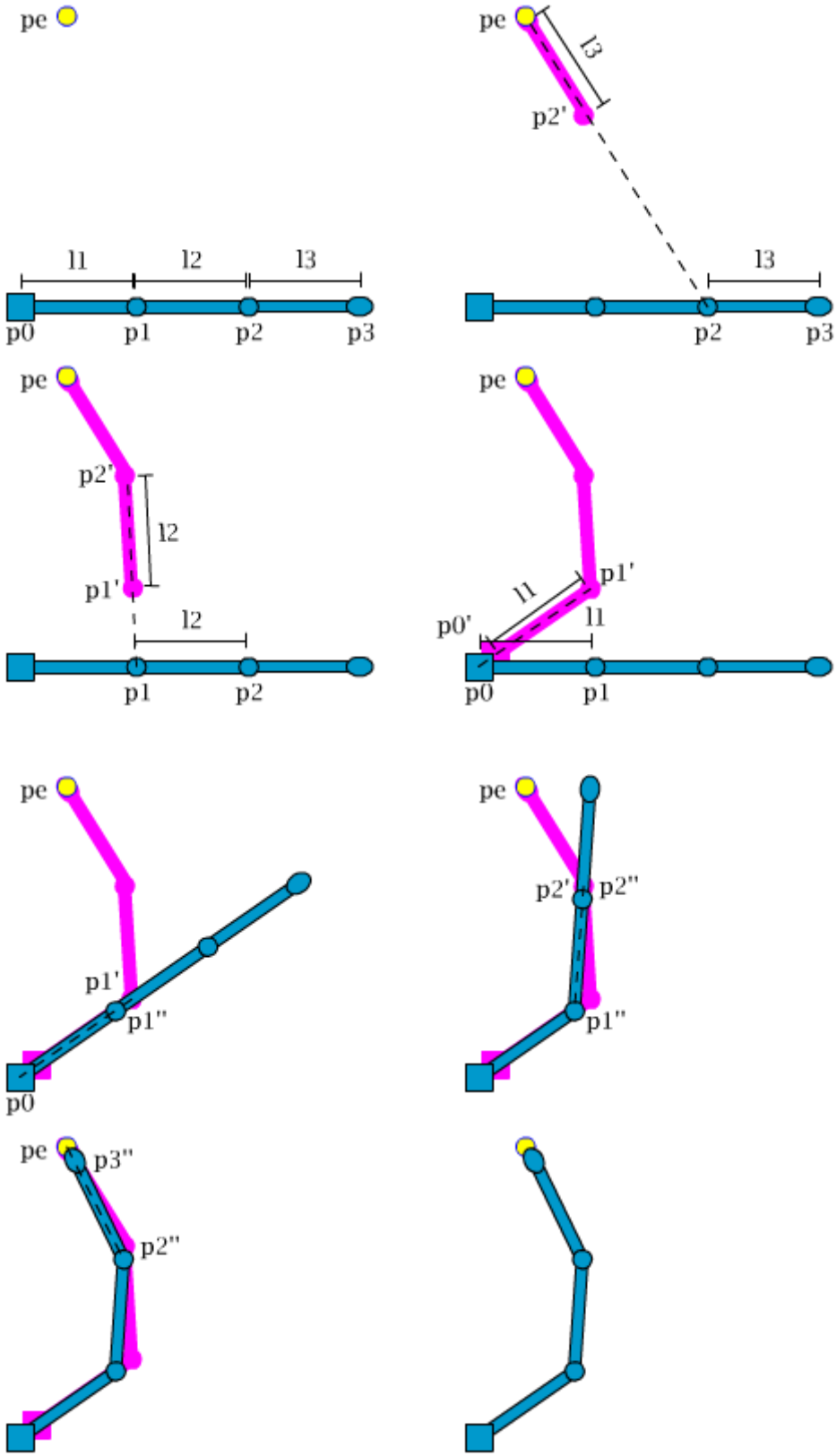
käänteiskinematiikka -algoritmeja. Heuristiikka käänteiskinematiikka -algoritmit ovat iteratiivisia, jotka vievät pääte-efektoria lähemmäs haluttua kohdetta iteraatio kerrallaan. Koska algoritmit eivät sisällä monimutkaisia laskufunktioita, ne laskevat raajan kulmat hyvin nopeasti, joten ne sopivat täydellisesti työn tavoitteisiin. Heuristiikka-algoritmien puute on, että ne eivät huomioi saman raajan yhteisiä nivelliikerajoja, joten ne päättyvät useasti biomekaanisesti ja tarkan anatomian kannalta ei-ihanteellisin asentoihin, joten nivelien rajoitteet on tehtävä käsin tai ohjelmallisesti yksitellen. Heuristiikka-algoritmeja on useita, joten tässä työssä käytettään niistä kolmea (Aristidou, Lasenby, Chrysanthou & Shamir, 2018).

CCD (Cyclic Coordinate Descent) on suosittu käänteiskinematiikka-algoritmi sen nopeuden takia ja sitä käytettään ihmismäisissä ja hyönteismäisissä malleissa. CCD:n idea on kohdistaa nivelet pääte-efektorin kera yksi kerrallaan kohti tavoitetta. Kohdistus alkaa pääte-efektorista ja jatkuu kinematiikkaketjun juureen asti. Prosessia toistetaan, kun pääte-efektori on tarpeeksi lähellä tai on tehty tarpeeksi iteraatioita. Pitkät raajat useilla nivelillä kietoutuvat kohteen ympärille. Raajan alkuasento vaikuttaa myös sen lopulliseen asentoon. Kun asento on iteroitu ohjelmallisesti, lasketaan siitä halutut nivelten kulmat.



Kuvio 2. CCD-algoritmin yksi iteraatio. Keltainen piste pääte-efektorin tavoite.

FABRIK (Forward and Backward Reaching Inverse Kinematics) on hitaampi kuin CCD ja tarkoitettu raajoille, jossa on useita niveliä, nivel rajoitteita ja pääte-efektoreita. FABRIK-algoritmissa lasketaan kohdistus ensin raajan päästä juureen ja takaisin sen päähän. FABRIK:ssa lasketaan lineaarialgebralla kinematiikkaketjun pääte-efektorista pisteet ja pituudet kinematiikanketjun luiden ja nivelten avulla juureen asti (ks. kuvio 3.). Laskettu juuri on tietenikin väärässä kohdassa, jolloin juuren oikeasta kohdasta lasketaan takaisin pääte-efektorin kohteeseen laskien nivelten aidot positiot aiemmin laskettujen pisteiden suunnan mukaisesti, jonka jälkeen on suoritettu yksi iteraatio. Kun iteraatioita on suoritettu tarpeeksi tai pääte-efektori on tarpeeksi lähellä, lasketaan nivelten kulmien asteet.



Kuvio 3. FABRIK-algoritmin yksi iteraatio. ’pe’ on pääte-efektorin tavoite.

Limb on CCD-algoritmiin perustuva enintään kaksiniveliselle raajalle tarkoitettu ratkaisija, eli ensisijaisesti käsivarsien tai jalan tapaisille raajoille. Limb-algoritmissa voi valita mihin suuntaa keskinivel taipuu.

3.5 Tekoälyn soveltaminen

Tieto-ohjautuvat algoritmit ovat uusi nouseva ala, jossa käytetään neuroverkoja ja tekoälyä käänteiskinematiikkaongelman ratkaisun löytämiseen. Elokuvissa ja tietokonepeleissä käytetyistä näyttelijöiden liikkeen kaappauksista rakennetaan animaatiokirjasto, josta neuroverkko tai tekoäly valitsee, yhdistää ja rakentaa tilanteelle ja kontekstille sopivan animaation (Aristidou, Lasenby, Chrysanthou & Shamir, 2018).

4 Unity & 2D-käänteiskinematiikkapaketti

Unity on Unity Technologies:n kehittämä alustariippumaton pelimoottori, johon on integroitu kehitysympäristö. Unityn voi ladata ilmaiseksi omaan henkilökohtaiseen käyttöön ilmaispelien tekemiseen. Kaupallisten tuotteiden kehittämiseen ja myymiseen tarvitaan lisenssi. Unity tukee kohdealustoina tietokoneiden lisäksi pelikonsoleita ja mobiililaiteitta ja sisältää tuet 3D- ja 2D-grafiikalle, fysiikkamoottorin ja skriptityökalun. Unity pelimoottoriin kuuluu myös animaatiokomponenttikokonaisuus, joka sisältää työkalut keyframe-animaatioille, eli suoraa kinematiikkaa, mutta työkalua ei käytettä tässä työssä. Unity käyttää C#-kieltä skriptikielenä ja ohjelmointiin tarvitaan oma koodieditori, joten tässä työssä käytettiin Unityn suosittelemaa Microsoftin Visual Studiota.

Unityyn on mahdollista ladata erikseen ns. esittely paketteja (preview packages), jotka ovat kehityksen alla olevia komponentteja ja työkaluja, jotka laajentavat Unityn ominaisuuksia. Eräs näistä paketeista on 2D-käänteiskinematiikkapaketti, jota käytettiin tässä työssä.

4.1 2D-käänteiskinematiikkapaketti

Käänteiskinematiikkakomponentit ja työkalut on ladattava erikseen. Pelihahmon riggaukseen sisältyvät työkalut ja algoritmit sisältyvät aiemmin mainittuun käänteiskinematiikkapakettiin. 2D-käänteiskinematiikka on suhteellisen uutta Unityssa, joten se sisälsi bugeja, jotka haittasivat työn toteuttamista. Paketti sisältää Limb, CCD ja FABRIK - heuristiikka käänteiskinematiikkaratkaisija-algoritmit. Kyseiset algoritmit ovat selitetty tarkemmin luvussa 3.4. Algoritmien lisäksi paketti sisältää kaksi työkalua: Bone Editor hahmon riggaamiseen ja Geometry & Weight Editor tahkoston luomiseksi kuvan geometrian mukaan ja verteksi painojen asettaminen kuvan muodon muuttamiseksi ja. tarvittavan skriptirajapinnan omien skriptien kirjoittamiseen. (Unity, 2018).

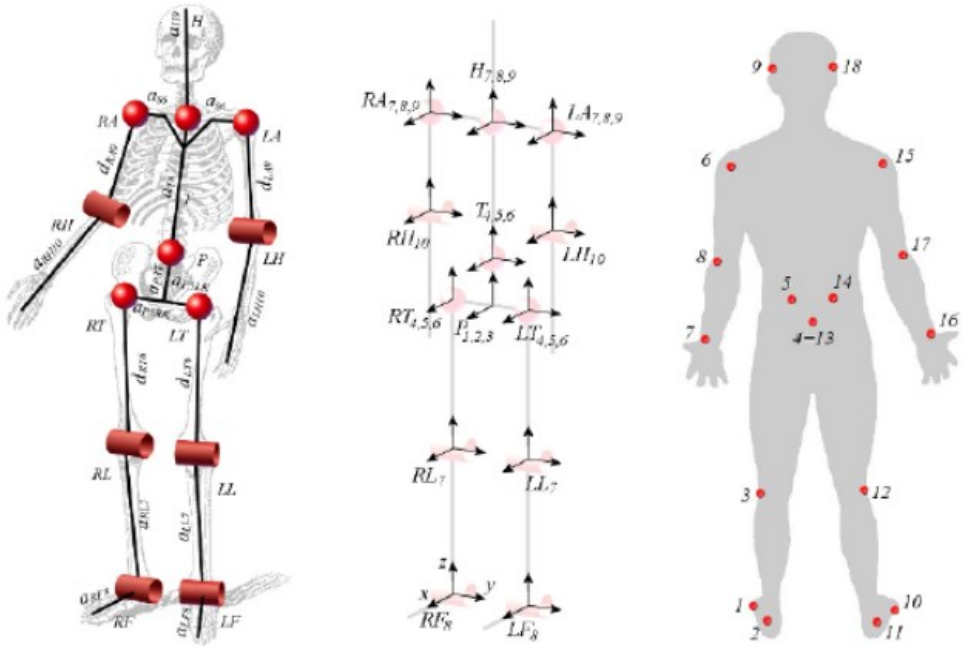
5 Hahmomalli

Jotta kinematiikkaa ja käänteiskinematiikkaa voidaan käyttää pelihahmossa, se on oltava suunniteltu sitä varten ja siihen tarvitaan sopiva kinematiikkamalli (Multon, France, Cani-Gascuel & Debunne, 1998). Työssä hahmomalli rigattiin Unityn käänteiskinematiikkapaktin sisältävillä työkaluilla.

5.1 Kinematiikkamalli

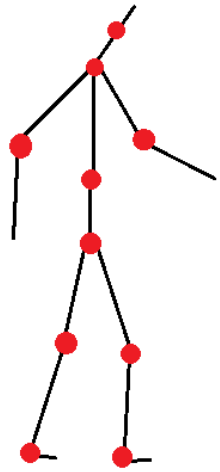
Ihmisen kehon ja sen liikkeiden toteuttamiseen tarvitaan kinematiikkamalli, joka on yksinkertainen malli animoitavasta tai simuloitavasta hahmosta. Yleensä ihmisestä mallinnetaan koko kehon sijaan vain luuranko, jonka päälle rakennetaan yksityiskohtaisempi lihasrakenne tai muu simulaatiomalli, jota animaattori manipuloi luurangon luiden avulla (Multon, France, Cani-Gascuel & Debunne, 1998).

Työn pelihahmon kinematiikkamalli on mukailtu Koritnikin esittämän mallin (kuvio 4.) mukaisesti. Kinematiikkamallia tarvitaan tässä työssä aiemmin mainittujen animaatioiden toteuttamiseen käänteiskinematiikalla (Koritnik, 2010).



Kuvio 4. Ihmisen kinematiikkamalli.

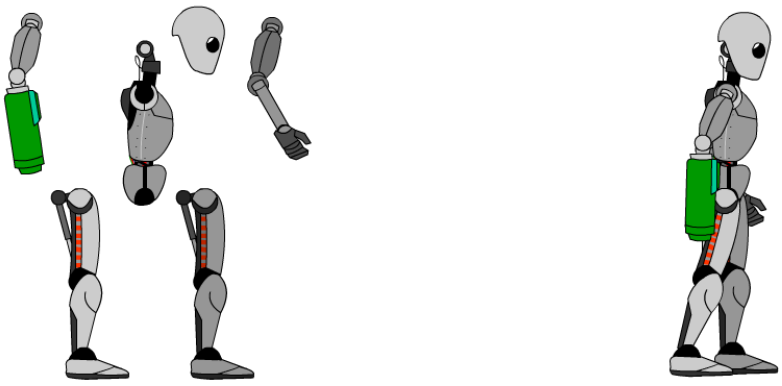
Koska työ rajoittuu sivulta katsottuun 2D-animaatiohon, tehtiin mallista yksinkertaisempi versio, jossa hahmon lantion lonkkaluut ja olkapäät ovat yhdistetty yhdeksi niveliksi (kuvio 5).



Kuvio 5. Yksinkertaistettu kinematiikkamalli sivusta katsottuna.

5.2 Hahmomallin riggaus

Käänteiskinematiikan käyttämiseksi työssä piirretty 2D-hahmomalli pilkotaan osiin kinematiikkamallin riggaamista varten. Pilkkominen tapahtuu jakamalla hahmon raajat ja keho omiin osiin, joiden sisälle sitten riggataan kinematiikkaketjut (Aristidou, Lasenby, Chrysanthou & Shamir, 2018). Työssä käytetty hahmomalli (kuvio 6.) on suunniteltu etukäteen toteutusta varten.

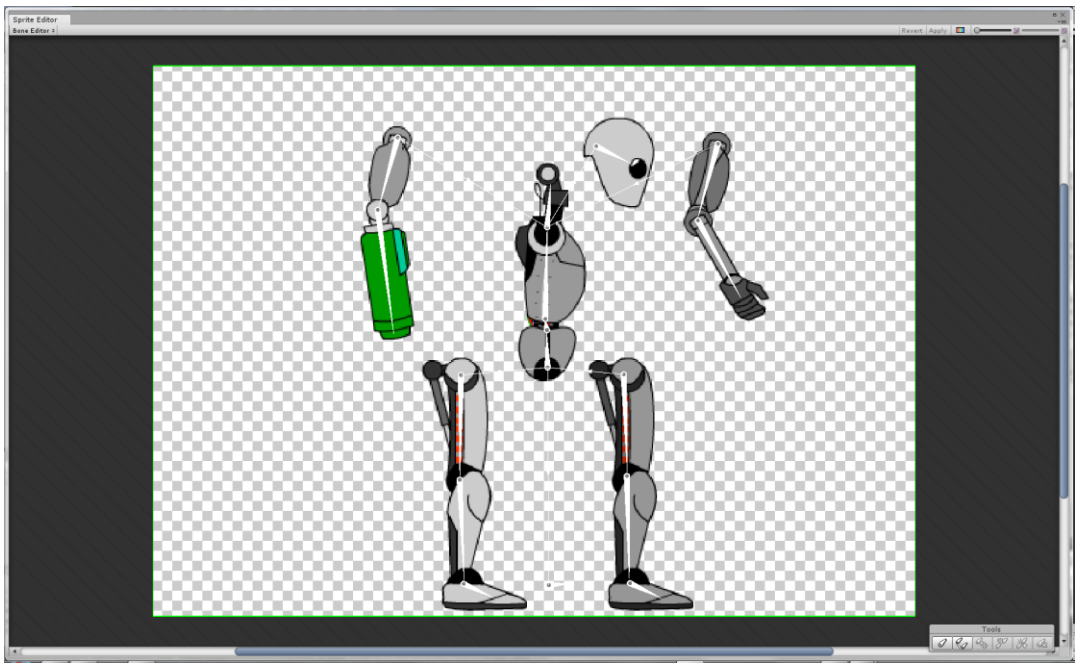


Kuvio 6. Pelihahmo osissa ja koottuna.

Pelihahmo on piirretty Flash ohjelmalla ja viimeistelty GIMP kuvankäsittelysovelluksella. Hahmomallin osat on jaettu vasempaan ja oikeaan jalkaan ja käsivarteen, kehoon ja päähän

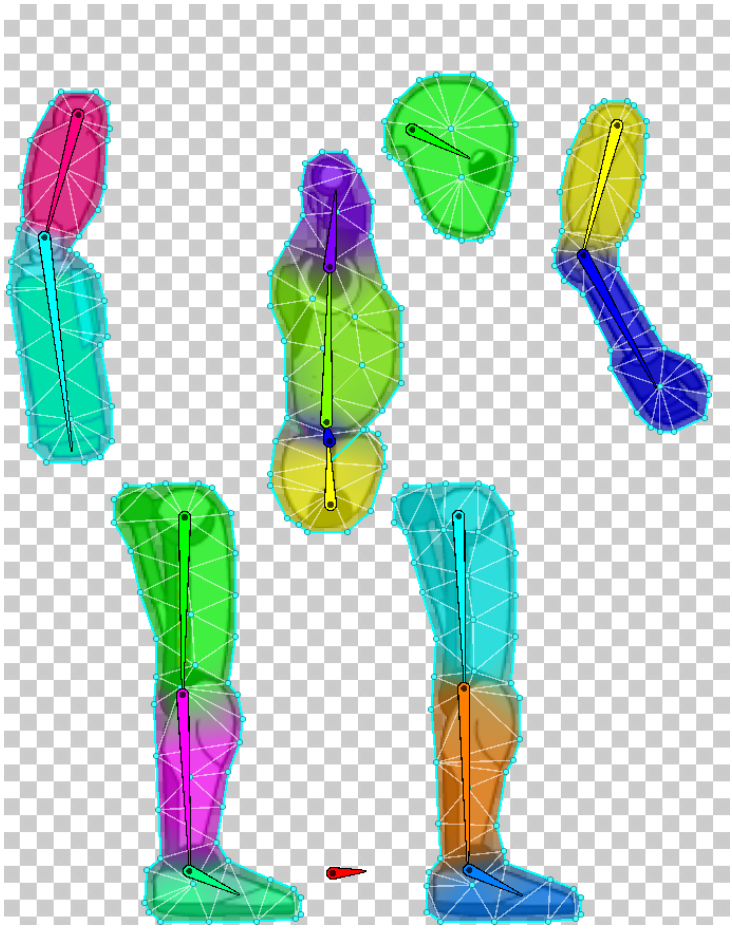
(Chung, 1999). Pelihahmo kootaan ja riggataan Unityssa. Luiden ja kinematiikan ketjun osien asentamista hahmomalliin kutsutaan riggaamiseksi (Joensuu, 2016).

Riggaaminen alkaa asentamalla ensin juuri tai ’root’. Juuri toimii muiden luiden ja kinematiikan ketjujen alkuna. Juuri sijoitetaan yleensä hahmon jalkojen väliin tai lantioon, josta ylöspäin rakennetaan luuhierarkia, joka muodostaa hahmon luurangon ja kinematiikkamallin (Hinton-Jones, 2018).



Kuvio 7. Hahmo rigattu aiemman luvun (kuvio 5.) kinematiikkamallin mukaisesti.

Jotta hahmomallin osia ja sen grafiikkaa voidaan manipuloida luilla, tarvitaan tahkosto (eng. mesh). Unityn käänteiskinematiikkapaketti sisältää työkalut tahkoston valmistamiseen. Tahkosto muodostuu verteksi-pisteistä, jotka yhdistämällä hahmografiikkaan voidaan sitten tahkoston verteksejä muuttamalla samalla muuttaa kuvan UV-pistekoordinaatteja. UV-pisteet ovat kuvan pikseleiden koordinaattipisteet. Kun luita siirretään tai kierretään, niin myös samalla muutetaan niihin liitettyjä tahkoston verteksejä ja kuvapikseleitä verrannollisesti saman verran.



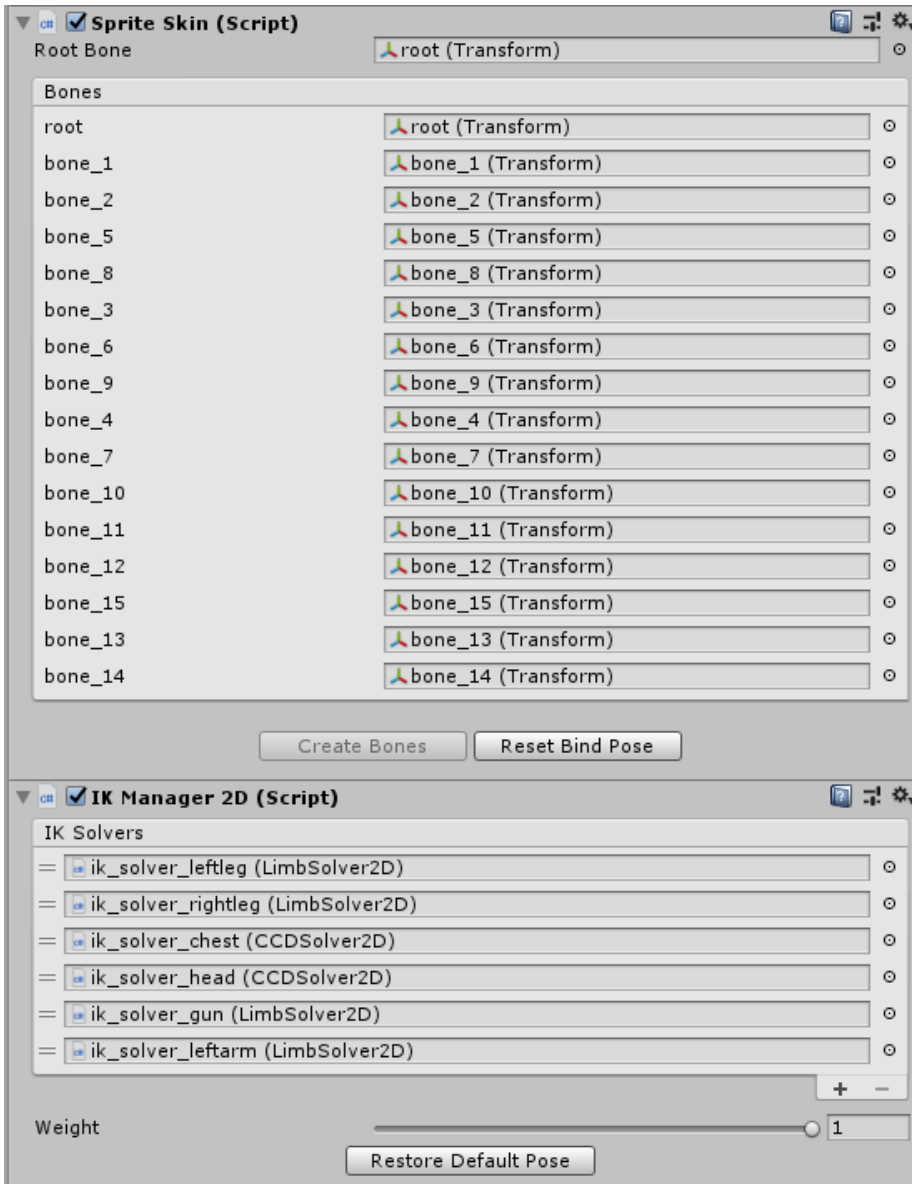
Kuvio 8. Tahkoston alueet ovat värjätty samalla värillä, kuin niitä manipuloiva luu.

Unityssa peliobjektien kaikki ominaisuudet ja toiminnot koostuvat ja rakentuvat komponenteista. Jotta käänteiskinematiikkaa voidaan käyttää, pelihahmoon pitää asettaa siihen tarvittavat komponentit (Unity, 2019).

Unityssa ennen pääte-efektoreiden asettamista on luottava ohjelmallinen luuranko. Tämä tapahtuu asettamalla hahmoon käänteiskinematiikkapaketin sisältämä ‘Sprite Skin’ -komponentti, joka automaattisesti noutaa hahmosta aiemmin rigatut luut ja rakentaa luurangon hierarkian pelihahmolle (Hinton-Jones, 2018).

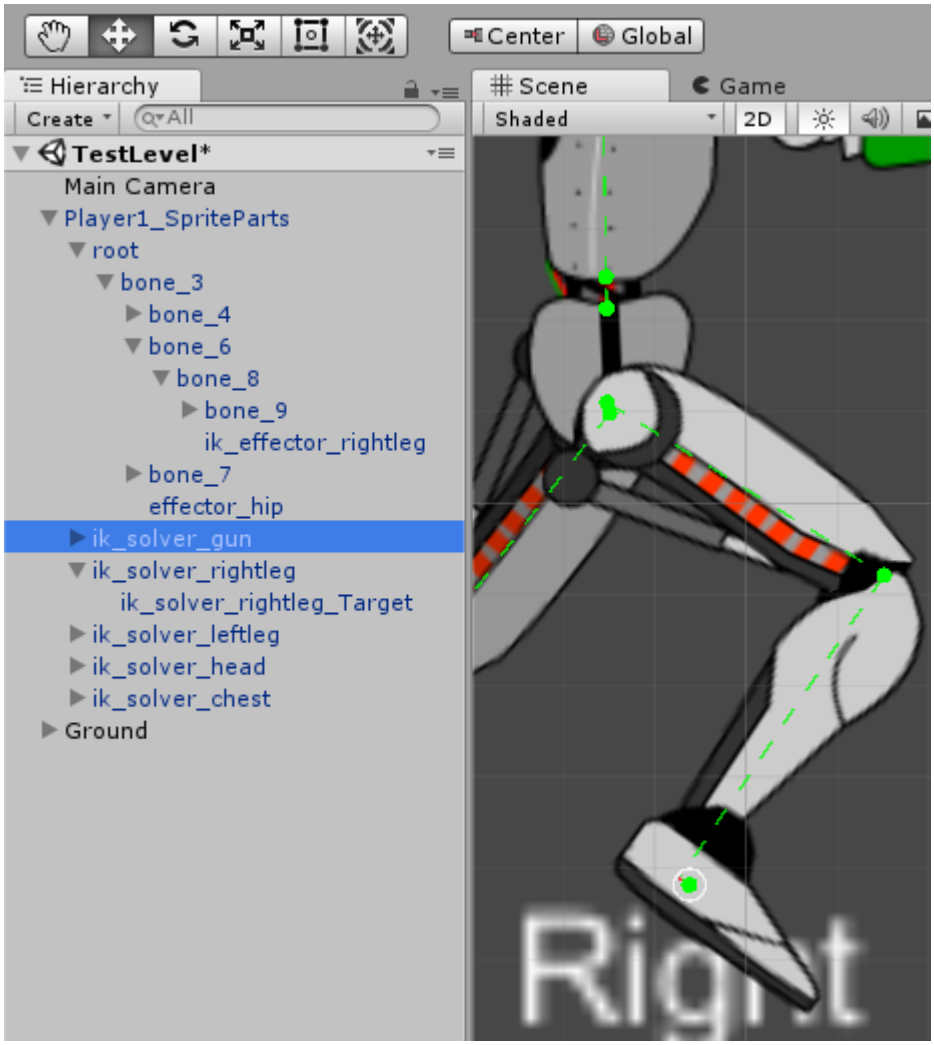
Seuraavaksi sijoitettaan IK-manager2D -komponentti, jolla asetetaan ja hallinnoidaan käänteiskinematiikkaratkaisija-algoritmeja. Ratkaisija-algoritmit tarvitsevat pääte-efektorit, jotka sijoitettaan raajojen päihin. Esim. jalan pääte-efektori sijoitetaan nilkkaan, koska jalkapohjan halutaan toimivan jalan pääte-efektorina, josta algoritmilla luuhierarkiaa alaspäin valitaan jalan vastaavat sääri- ja reisiluut. Kun nämä valinnat on tehty, on luotu

kinematiikkaketju, jossa lantio on kinematiikkaketjun juuri ja jalkapohja pääte-efektori. Ratkaisija-komponentti sisältää valitun raajan ratkaisualgoritmin, jolla lasketaan kinematiikkaketjun nivelten kulmat. Työn pelihahmo sisältää kuusi kinematiikkaketjua: jalat, kädet, selkä ja pää. Jalat ja kädet käyttävät Limb-algoritmia, kun taas selkä ja pää käyttävät CCD-algoritmia (Hinton-Jones, 2018).



Kuvio 9. Sprite Skin ja IK-manager2D -komponentit Unityssa.

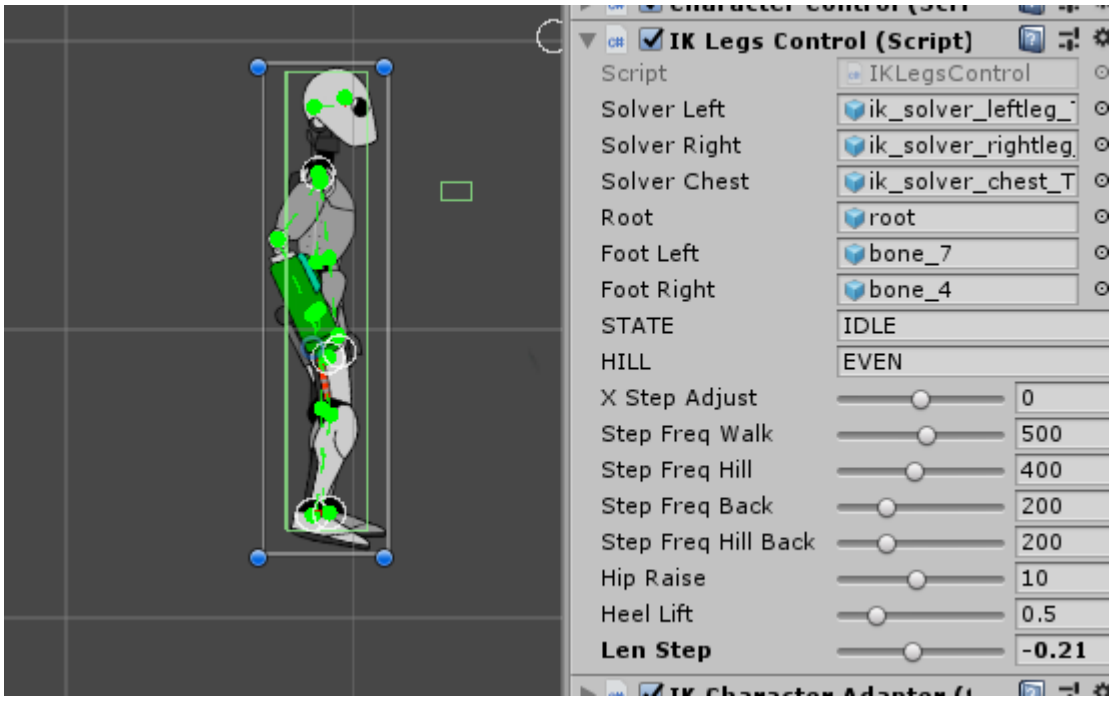
Kuvion 10. pelihahmo ‘Player1_SpriteParts’ sisältää luuhierarkian alkaen ‘rootista’ ja ratkaisijat (ik_solver). Luut ‘bone_6’ ja ‘bone_8’ ovat hahmon oikean jalan reisiluu ja sääriluu ja niiden päässä on oikean jalan pääte-efektori (ik_effector_rightleg).



Kuvio 10. Juuresta (root) alkava luuherarkia.

6 Ohjelman komponentit

Käytetty ohjelmakoodi rakentuu tässä työssä tehdystä koodista ja jo olemassa olevasta koodista. Unityssa kaikki peliobjektien ominaisuudet ja toiminnot koostuvat ja rakentuvat komponenteista. Myös tarvittava ohjelmointi on tehty omiin skriptikomponentteihin.



Kuvio 11. Jalkojen IK-ohjaus -komponentti Unityn editorissa.

6.1 Hahmo-ohjaus

Hahmo-ohjauskomponentin on kehitetty aikaisemmin ennen tätä työtä ja toimii yhdessä Unityn CharacterController-komponentin kanssa. Hahmo-ohjaus sisältää tarvittavat algoritmit osuma tunnistukselle virtuaaliympäristön kanssa, hahmon fysiikan mallinnukselle (esim. nopeus, kiihtyvyys ja painovoima) ja pelaajan syötteen käsittelylle, jotta se voi mahdollistaa pelihahmon liikkuvuuden virtuaalimaailmassa. Tässä työssä kehitettävä käänteiskinematiikan ohjauskomponentti (Lyhyesti IK-ohjaus) on yhteensopiva nykyisen pelihahmon hahmo-ohjaus -komponentin kanssa.

6.2 Käänteiskinematiikan ohjauskomponentti

Käänteiskinematiikka ohjauskomponentti (lyhyesti IK-ohjaus) on tässä työssä toteutettava komponentti, joka ohjaa pelihahmon jalkojen pääte-efektoreiden haluttuja tavoite sijainteja, kun pelihahmo on liikkeessä. Komponentin toimintaa on kuvattu vuokaaviolla liitteessä 1. Riippuvuuksien vähentämiseksi ohjauskomponenttien välille on kehitetty välikomponentti, jonka kautta kutsutaan funktioita, joiden sisällyttäminen itse käänteiskinematiikkaohjauskomponenttiin tai hahmo-ohjauskomponenttiin ei ole mieltä ja sillä noudetaan hahmon nopeus ja suunta hahmo-ohjauskomponentilta.

Komponentti sisältää säätömahdollisuuksia askeleiden personoimaiseksi (askeleen korkeus, lepoaskeleen päätössijainti ja lantion nousu) ja askeltiheyden asettamiseksi, kun hahmo liikkuu eteen- tai taaksepäin tasaisilla pinnoilla tai mäissä.

Käsien osoittamiselle ja heilumiselle tehtiin omat komponentit, joiden käyttöä voidaan vaihdella helposti sen omalla ohjauskomponentilla tarvittavan tilanteen mukaan. Askeleiden toiminta on tarkemmin selitetty luvussa 7.

7 Askeleiden toteutus

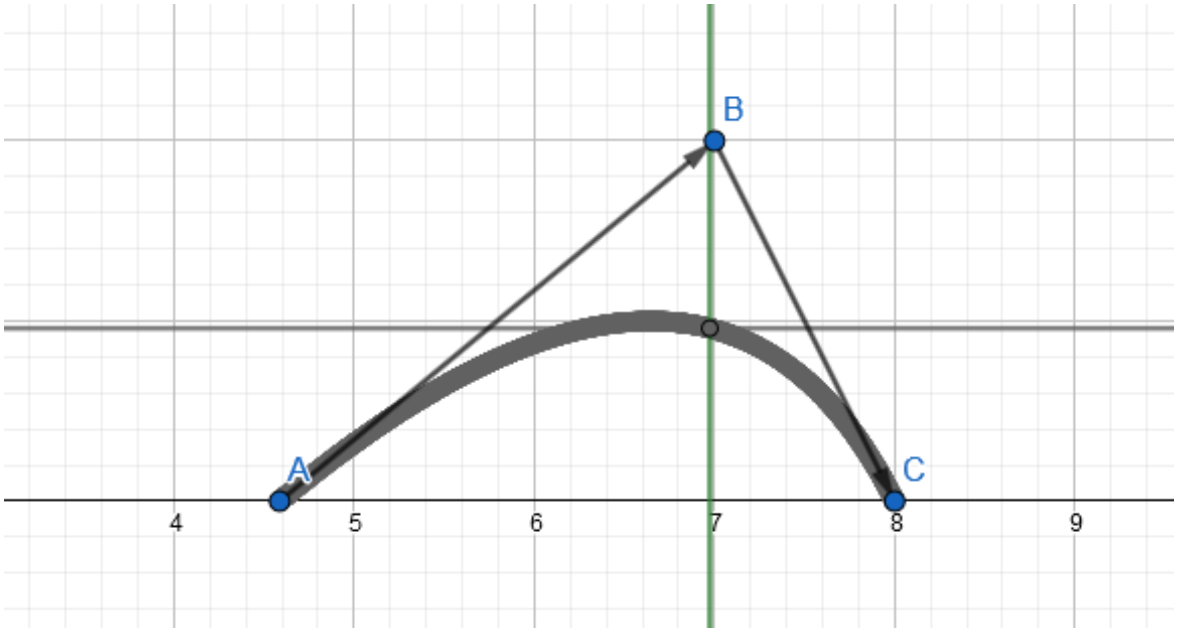
Chungin ja Hahnin (Chung & Hahn, 1999) tekemä tutkimus esittää ratkaisupohjan jalan heiluriliikkeen laskemiselle. Askeleen liikerata perustuu Bezier-käyrän hyödyntämiseen. Bezier-käyrän avulla voidaan laskea jalan pääte-efektorin liikerata jalan heilumisen aikana. Pääte-efektorin siirto kaaressa animoidaan interpoloimalla. Työn kaikki matematiikkalaskut tehtiin lineaarialgebralla. Jalkojen positio ja liikkeen suunta lasketaan vektoreilla ja matriiseilla.

7.1 Askeleen laskeminen

Bezier-käyrää käytetään ensisijaisesti vektorigrafiikassa ja 3D-mallinnuksissa. Bezier-käyrä koostuu vähintään kahden pisteen välistä piirretystä käyrästä, jonka kaarevuutta voidaan muuttaa kontrollipisteillä. Bezier-käyrä on muotoa:

$$B(t) = (1-t)^n P_0 + \binom{n}{1}(1-t)^{n-1} t P_1 + \cdots + \binom{n}{n-1}(1-t)t^{n-1}P_{n-1} + t^n P_n,$$

Missä $P_0, P_1, \ldots, P_n$ ovat kontrollipisteitä ja $\binom{n}{i}$ ovat binomikertoimia.



Kuvio 12. Esimerkki Bezier-käyrä GeoGebra-laskimessa.

Esimerkiksi tapauksessa $\mathrm{n} = 2$ Bezier-käyrän on muotoa:

$$\mathrm{B}(t) = (1-t)^2\mathrm{A} + 2(1-t)\mathrm{tB} + t^2\mathrm{C},$$

(ks. kuvio 12).

Käyttämällä kolmea pistettä A, B ja C muodostettaan jalan heilumisliike. A on askeleen aloituspiste, B:llä ohjataan askeleen nousua ja C on askeleen arvioitu laskeutumispiste.

Jalan liikeradan laskeminen alkaa, kun hahmo on liikkeessä. Bezier-käyrän alkupiste A asetetaan askeljalan nykyiseen positioon. Askeleen laskeutumispiste C lasketaan hahmon eteen ja kontrollipiste B suhteutetaan loppupisteen taakse.

Askeleen muutos lasketaan seuraavasti:

$$\mathrm{p} = \mathrm{v} * s\mathrm{f},$$

Missä p on liikemäärä, v on hahmon kehon nopeus ja sf on askeltiheys.
Liikemäärällä voidaan sitten laskea askeleen muutos.

$$\Delta s = \Delta t * \mathrm{sd} * \mathrm{p},$$
$$\mathrm{s} = \mathrm{s} + \Delta s,$$

Missä Δs on askeleen muutos kaaressa, Δt on kulunut aika edellisestä päivityskerrasta, sd on askeleen suuntamatriisi ja p on liikemäärä. Toisessa kaavassa s on uusi positio askeleen nykyisen position ja muutoksen summana. s:n arvoalue on 0–1 ja sillä lasketaan askeleen sen hetkinen positio Bezier-kaaressa.

$$\mathrm{B}(s) = S(x, y),$$

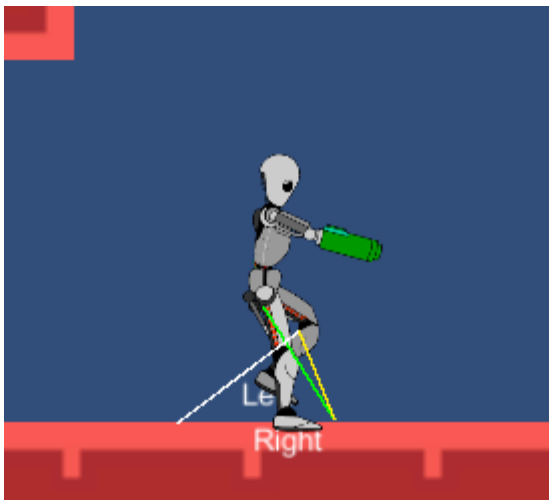
Askeltiheys on animaattorin asetettava käsin (Bruderlin & Calvert, 1989). Askeltiheydellä määritetään askelevaihtojen tiheys. Liian pieni askeltiheys ja jalat liikkuvat silmää nopeammin. Liian suuri arvo ja hahmo näyttää liitävän. Kun sopiva askeltiheys on löydetty kävelemiselle, voidaan sitä sitten mukailla ohjelmallisesti muille kävely toiminnoille esim.

juokseminen, hiipiminen, takaperin kävely jne. Laskettu liikemäärä skaalataan delta-ajalla ja lisätään askeleen positioon. Kun askel on päätepisteessä tai Bezier-käyrän lopussa, vaihdetaan tukijalka ja alustetaan liikemäärä seuraavalle askeleelle. Tukijalka pysyy hahmon liikkeen aikana paikallaan. Askeleen siirto uuteen positioon pelinajon aikana lasketaan interpoloimalla, jolla vältetään askeleen nykiminen, jos Bezier-käyrä tai käveltävätaso muuttuu nopeasti hahmon alla. Taaksepäin kävelyssä algoritmi on lähes samanlainen, pistettä A käytettään askeleen loppupisteenä ja pistettä C askeleen alkupisteenä.

Kävelyn aikana hahmon lantio nousee. Lantion nousu kävelyn aikana lasketaan seuraavalla kaavalla:

$$\frac{-4(askeleenpositio-0.5)^2+0.25}{nousu[0.1-1]},$$

Askeleen positio on sen nykyinen sijainti aiemmin mainitussa Bezier-käyrässä ja nousulla säädetään lantion nousun huippuarvo.



Kuvio 13. Bezier-käyrän pisteet hahmotettu kuviossa valkoisella ja keltaisella viivalla.

7.1.1 Askeleen toiminta

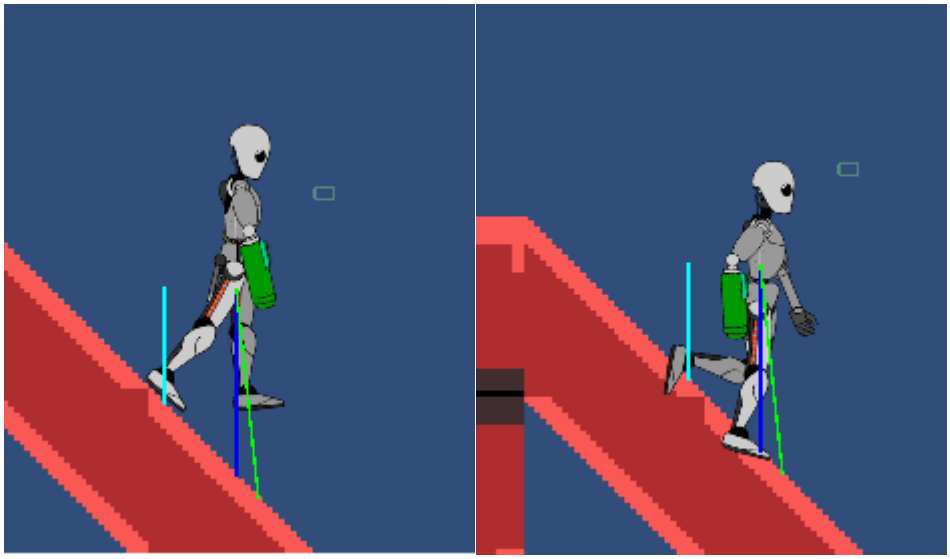
Koska animaatio on rajoittunut sivulta seurattuun 2D-avaruuteen, kävelysuunnalle on vain eteenpäin ja taaksepäin suunta, niin suorituskyvyn kannalta skaalausmatriisi on paras vaihtoehto liikkeen suunnan laskemiseksi. Kiertomatriisin liitäminen olisi tulevaisuuden ja uudelleen käytettävyyden kannalta parempi vaihtoehto varsinkin, jos järjestelmää aiotaan

käyttää 3D-avaruudessa. Jotta skaalausmatriisia voi käyttää, ensin laskettava hahmon translaatio origon suhteen. Jos translaatiota ei tehdä, niin skaalausmatriisi lasketaan suhteessa hahmon nykyisen sijainnin mukaan origon sijaan, jolloin ei välttämättä saada haluttua suuntaa. Tätä laskua käytettään askeleiden lopullisen laskeutumispisteen laskemiseksi (Satran, 2018).

Kun pelihahmo liikkuu eteenpäin, sen askeleelle etsitään laskeutumispiste. Askeleen haussa on käytetty kirjallisuutta liittyen ihmisen anatomiaan, josta prototyyppiin on mallinnettu ihmisen keskiverto kävelyaskeleen pituus, joka etsitään hahmon edestä raycast-funktiolla ampumalla raycast-säde alaspäin. Ihminen ottaa ylämäkeä noustessa pidempiä askeleita ja alamäkeä laskeutuessa lyhyempiä askeleita, joten askeleen laskeutumispistettä etsitään ylämäessä kauempaa ja alamäessä lähempää (Psarras, Mertyri & Tsaklis, 2006). Myös hahmon nopeus on otettu huomioon, joten nopeuden mukaan askelta etsitään hieman kauempaa. Pysähtymisessä käytetään lyhyttä askeleen etsimistä. Takaperin kävelyssä käytetään samaa lyhyttä askel etsimistä kuin, edellä paitsi, että askeleen etsimisen alku peilataan käyttämällä skaalausmatriisia.

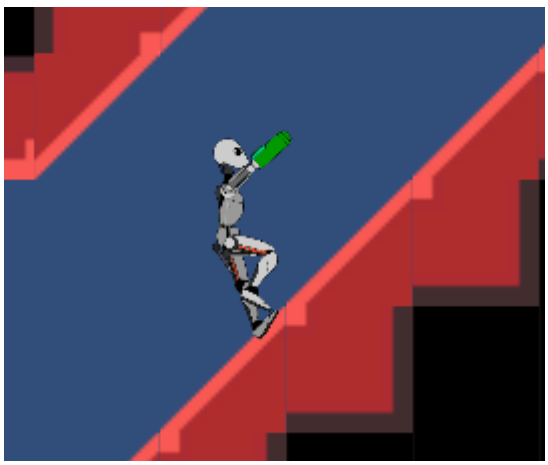
7.2 Kinematiikkamallin sovitus ympäristöön

Epätasaisissa tasojen pinnoissa hahmon kinematiikkamallia on nostettava ja laskettava tason mukaisesti, jotta hahmon jalat ylettyvät koskettamaan tasoa ja jotta hahmo ei ole kyyryssä. Tämä tapahtuu nostamalla ja laskemalla kinematiikkamallin juuripistettä. Vasemmalla kuviossa 14. juurta ei ole laskettu ja oikeassa on. Laskun syvyys etsitään löytämällä hahmon alta syvin pudotus ja laskemalla tarvittava erotus pinnan ja juuren välillä.



Kuvio 14. Kinematiikan mallin sovitus. Vasemmassa kuvassa pudotusta ei ole tehty.

Jalkaterä asetettaan pinnan mukaisesti laskemalla sen alhaalla olevan pinnan normaalivektori. Normaali vektorin ja pystysuoran vektorin väliltä lasketaan kulmavektori, joka lisätään jalan oletus kiertoon. Kierrot Unityssa lasketaan kvarternioilla. Tosin kyyristymisessä käytettään samaa algoritmia jalkaterän pitämiseksi tason pinnan mukaisesti, jolloin jalkaterä on biomekaanisesti ei-ihanteellisessa asennossa johtuen puutteellisesta kinematiikkamallista, jossa huomioidaan jalan kantaluu ja varpaiden luut yhtenä kokonaisuutena.



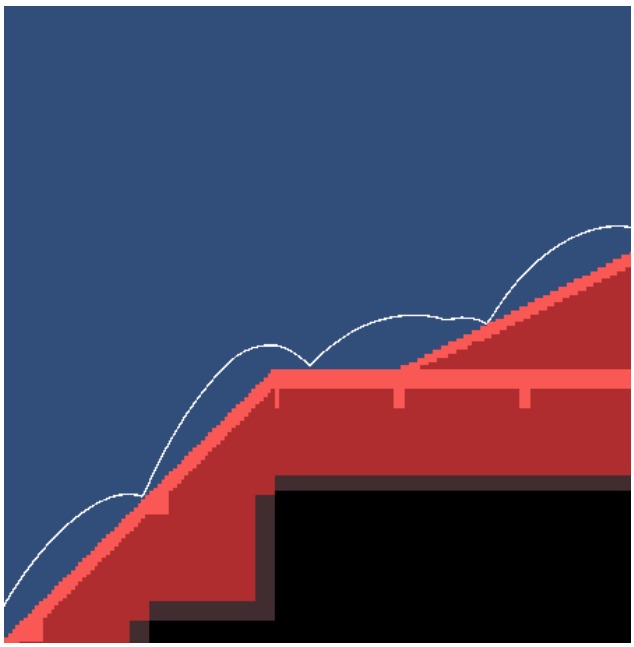
Kuvio 15. Hahmo kävelee mäkeä ylös ja jalkaterä on asetettu tason kaltevuuden mukaisesti.

7.3 Interpolointi

Animaatiossa ’inbetweening’ on prosessi, jossa animaattori piirtää animoitavan hahmon alkuasennon ja loppuasennon välistä puuttuvat hahmon liikkeiden tilat. Tietokoneanimaatiossa ’inbetweeningiä’ kutsutaan interpoloinniksi, jossa tietokoneohjelma tekee kyseisen työn. Interpolointi on välttämätöntä animaatiossa, koska ilman sitä hahmon liikkeissä ilmenee nykimistä. (Burtnyk & Wein 1976)

Työn aikana havaittiin pelihahmon liikkeiden nykimistä. Jos virtuaaliympäristö ei ole staattinen (esim. sisältää liikkuvia tai keinuvia tasoja) tai vastaan tulee kalteva taso, niin aiemmin laskettu Bezier-käyrä askeleen liikeradalle pitää laskea uudelleen. Tämä näkyi pelin aikana pelihahmon jalan nykimisenä, kun sen jalka siirtyy vanhasta liikeradasta uuteen liikerataan välittömästi.

Kun askeleen liikerata muuttuu kävelyn aikana, niin lineaarisella interpoloinnilla voidaan laskea jalan pääte-efektorin siirto vanhasta liikeradasta uuteen liikerataan, ilman häiritsevää nykimistä (Multon, France, Cani-Gascuel & Debunne, 1998).



Kuvio 16. Jalan liikerataa on korjattu tulevaa ylämäkeä varten.

Interpoloinnille löytyi muutakin käyttöä. Esim. Kun hahmo lähestyy reunaa, se on animoitu tasapanoitelemaan reunan takana. Mutta kun hahmo ylittää reunan, se napsahtaa saman tien oletusasentoon. Lineaarisella interpoloinnilla voidaan hahmottaa tila, jossa hahmo korjaa asentoaan reunan ylittämisen jälkeen. Muita interpolointi käyttökohteita löytyi hahmon hyppimis- ja laskeutumistiloihin siirtymiseen ja palautumiseen normaaliin tilaan sulavasti (Multon, France, Cani-Gascuel & Debunne, 1998).

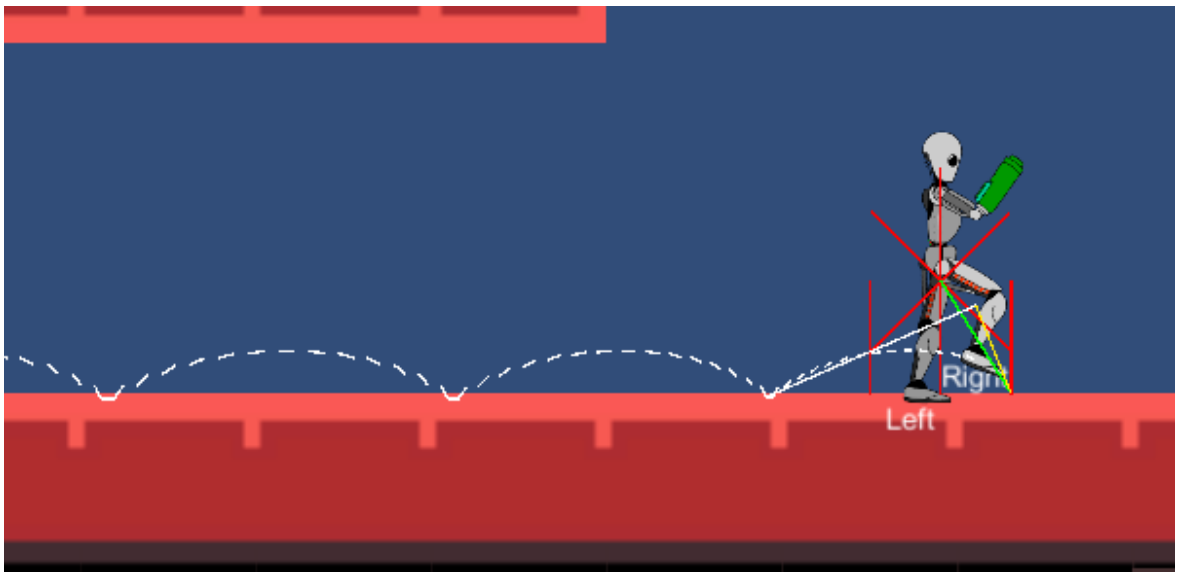
8 Hahmon liikkumismekaniikat

Tässä luvussa kuvataan komponentin toimintaa. Valmistettu komponentti kykenee seuraaviin luvussa 2 aiemmin mainittuihin tavoite toimintoihin. Tässä työssä tutustuttiin biomekaniikkaan ja ihmisanatomiaan hahmon liikkeiden toteuttamiseksi. Biomekaniikka on biologisten järjestelmien tutkimista mekaniikan avulla.

8.1 Kävely ja juokseminen

Kävely ja juokseminen ovat ihmisen yleisin liikkumisen muoto ja kävelyn ja juoksemisen välissä on eroja. Kävelyn aikana tukijalka (jalka, joka pysyy maassa askeleen aikana) on jäykkä ja askeljalka nouse kaaressa sen ohi. Kun ihminen juoksee, juoksuaskel nostaa ihmistä enemmän ilmaan kuin kävelyaskel ja iskeytyy voimalla laskeutuessaan maahan (Cappellini, 2006).

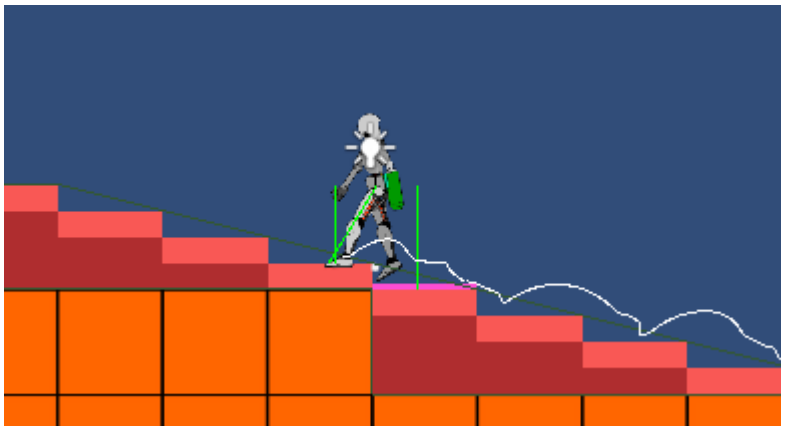
Pelissä pelihahmo alkaa kävelemään, kun se vastaanottaa syötteen peliohjaimesta tai siihen vaikuttaa ulkoinen voima. Peliohjaimena toimii konsoliohjaimen analoginen ohjaussauva tai näppäimistön painikkeet. Hahmo kävelynopeus on verrannollinen sen koko kehon nopeuteen, jolla on kiinteä kiihtyvyys ja hidastuvuus. Konsoliohjaimella pelihahmon rajanopeus riippuu ohjainsauvan asennosta. Kun pelihahmo saavuttaa kehon maksiminopeuden, niin sen askeleet pitenevät juoksuun ja lantion nousee ylemmäs askeleen aikana. Askeleen laskeutumispaikka löydetään raycast-funktiolla, joka ampuu säteen alaspäin maahan hahmon edessä lantion korkeudesta (ks. kuvio 17).



Kuvio 17. Hahmon oikean jalan liikerata. Vihreä viiva on raycast-funktion säde.

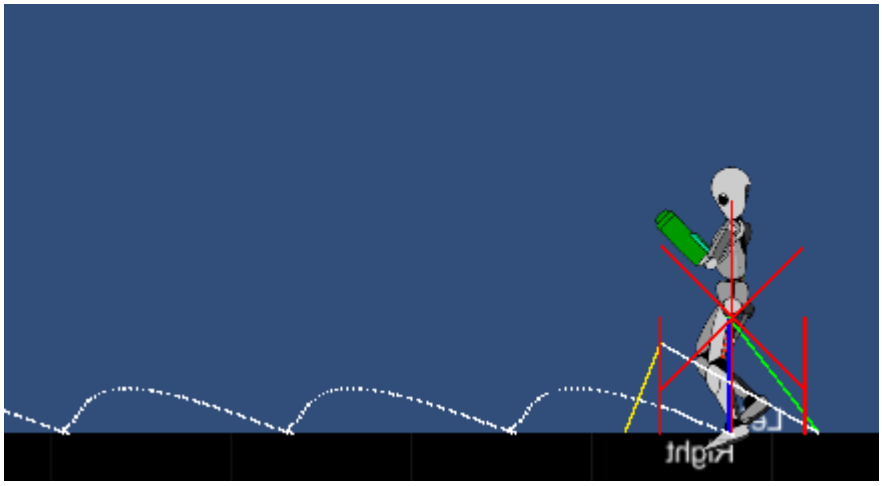
Ihminen ottaa pidempiä askeleita ylämäessä ja lyhyempiä askeleita alamäessä pitääkseen kehon keskinopeuden vakiona (Psarras, 2016). Mäen havaitseminen tapahtuu hahmon ohjaus -komponentissa laskemalla sen alla olevan tason normaali ja vertaamalla sen välistä kulmaa y-akseliin. Jos laskettu normaalivektori samansuuntainen y-akselin kanssa, hahmon tasaisella pinnalla.

Portaat toimivat samalla tavalla kuin mäet. Ohjauskomponentti tulkitsee ne kaltevina tasoina ilman porrasaskeleita pitääkseen hahmon liikkeen sulavana. Askeleiden laskeutuminen tosin lasketaan normaalisti porrasaskeleihin.



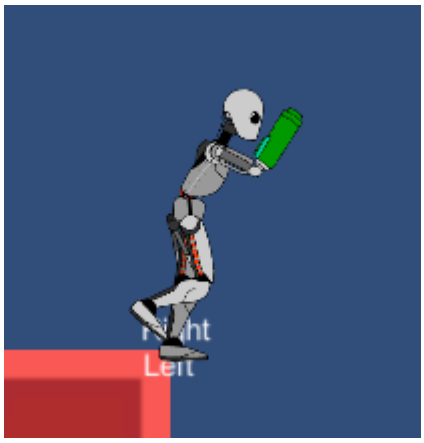
Kuvio 18. Hahmo kävelee portaissa.

Pysähtyminen tapahtuu, kun pelihahmo ei vastaanota ulkoisia syötteitä tai ei ole enää ulkoisten voimien vaikutuksessa. Pysähtymisessä pelihahmo hidastuu ja askeleet lyhenevät, kun viimeinen askel otettu ja hahmo on pysähtynyt, hahmo ottaa vielä yhden ns. ‘lepoon askeleen’, jonka jälkeen hahmon jalat ovat vierekkäin ja hahmo on siirtynyt lepotilaan. Takaperin kävelyssä pysähtymisen toiminta on identtinen. Hahmon taaksepäin kävelyssä sen mekaniikat samat paitsi askelsuunta on eri.



Kuvio 19. Hahmon oikean jalan liikerata takaperin kävelyssä.

Kun hahmo lähestyy reunaa, sen askeleet eivät mene reunan yli ja hahmon kehon keskipiste pysyy reunan takana. Tässä käytetään viimeisintä askeleen hyväksyttävissä olevaa laskeutumispistettä. Kun hahmon keskimassa on reunan yli, hahmo ylittää ja putoaa reunalta.



Kuvio 20. Hahmo tasapainottelee reunalla.

8.2 Käsien heilunta ja osoittaminen

Useissa anatomiaa käsittelevissä artikkeleissa käsien heilunta kävelyn aikana on jätetty kokonaan pois, koska niiden välinen yhteys on epäselvä, joten kinematiikka tutkimuksissa yleensä mallinnetaan vain ihmisen jalat, lantio ja torso (Meyns, 2013). Meynsin tutkimuksessa on tutkittu käsien heiluntaa, joten hänen työstään on johdettu käden heilunta tähän työhön.

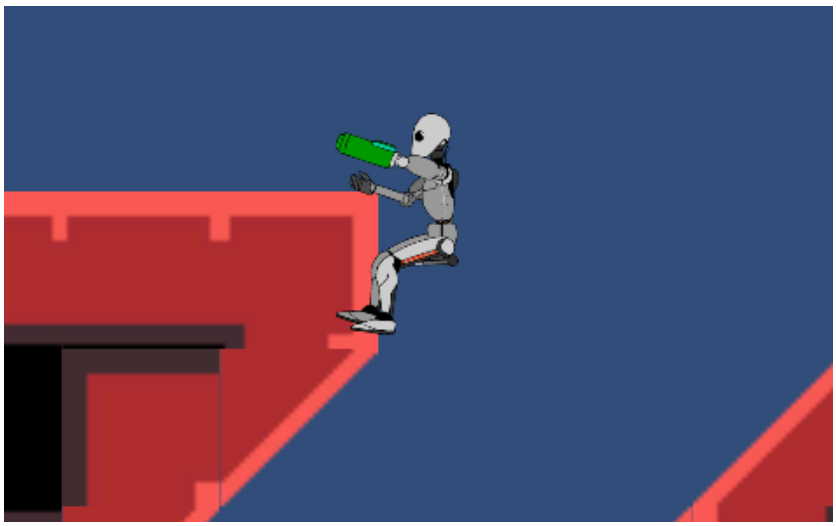
Pelissä kävelyn aikana hahmon kädet heiluvat suorana ja nopeuden kasvaessa juoksuun nousevat koukkuun. Levossa hahmon kädet ovat suorina hahmon lantion vieressä.

Putoamisessa kädet nousevat lähes olkapäiden tasolle. Käsien nousu on laskettu ylöspäin avautuvalla paraabelilla.

Osoittaminen on vain käänteiskinematiikan algoritmin iteratiivisen komponentin hyödyntämistä, koska algoritmi pyrkii samaan pääte-efektorin, jolla osoitettaan, tarpeeksi lähelle osoitettavaa kohdetta.

8.3 Muu toiminta

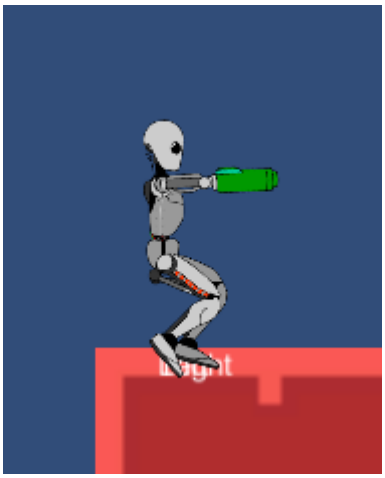
Hahmo tarttuu reunaan, jos se on putoamassa alaspäin ja on tarpeeksi lähellä reunaa. ja jaloille on paikka, mistä ottaa tukea.



Kuvio 21. Hahmo on tarrautunut reunaan ja ottaa tukea jaloilla.

Ihminen hyppää kyyristymällä ensin ja sitten nousee kyyrystä nopeasti ponnahtaen ilmaan. Pelissä hyppääminen on ajoitettu toiminto, jossa ensin hahmo kyyristyy ennen hyppäämistä. Kyyristyminen on toteutettu laskemalla hahmon kinematiikkamallin juuripistettä ja asettamalla sille normaalipositio, mihin juuripiste palaa takaisin ja maksimisyvyys, joka on syvin kyyristyminen, minkä hahmo voi tehdä. Kun ajoitettu aika on kulunut, pelihahmo hyppää ilmaan sille asetetulla voimalla.

Ihminen pehmentää putoamistaan laskeutumisessa joustamalla polviaan ja alaselkää (Zelik, 2013). Putoamisessa pelihahmo on animoitu koukistamaan polviaan. Koukistuksen syvyys riippuu putoamisnopeudesta ja tätä arvoa käytetään myös kyyristymissyötteessä.



Kyykistyminen

Ilmassa

Kuvio 22. Hahmon hyppimistoiminta.

9 Animaatio ja perusperiaatteet

Uskottavan pelihahmo animaation saavuttamiseksi sovelletaan Disney:n 12 animaation perusperiaatetta (Thomas & Johnston, 1981, s.47-69) ja niiden soveltamiseen hyödynnetään John Lassertin kirjoitusta. Vaikka John Lasseterin paperi käsittelee 2D-animaation periaatteiden soveltamista 3D-animaatioihin, tässä työssä käytetyt matemaattiset laskut ja mekaniikat ovat hyvin lähellä 3D-toteutusta, joten hänen paperistansa saatava hyöty on arvokas (Lassester. 1987).

9.1 Disneyn 12 animaation perusperiaatetta

Periaatteet ovat työkaluja ja auttavat selkeän ja uskottavan animaation valmistamiseen. Ne kehitettiin alun perin 1930-luvulla Disneyn animaatio piirtäjien kouluttamiseen ja ne koottiin kirjaan, joka julkaistiin vuonna 1981. Saarvan opinnäytetyö sisältää suomennokset ja kuvaukset termeille (Saarva, 2015).

9.1.1 Venytys ja litistys

Ehkä tärkein sääntö animaatiossa on venytys ja litistys. Kun ihminen liikkuu sen lihakset litistyvät ja venyvät liikkeen mukaisesti. Animaatiossa tärkeintä on muistaa, että hahmon koko tai sen volyymi ei muutu. Venytyksessä hahmo pidentyy ja samalla kaventuu ja litistyksissä hahmo lyhentyy ja leventyy. Mutta hahmon ei tarvitse muuttaa muotoansa litistymisen tai venytyksen takia. On tärkeää muistaa, että hahmolla on massa. Tässä työssä pelihahmo on robotti, joten esim. hyppiessä ja laskeutuessa hahmon ei tarvitse venyä tai litistyä. Pelkästään sen polvien voimakas joustaminen riittää luomaan vaikutelman litistymisestä, koska hahmon siluetin pituus lyhenee ja sen leveys vaakasuunnassa kasvaa. (Lassester, 1987).

9.1.2 Ennakointi

Ennakointi tarkoittaa animoitavan hahmon valmistumista liikkumiseen ja reagointia vastaliikeisiin (Saarva, 2015). Pelianimaatiossa ennakointi tärkeä alue, koska hyvin toteutettuna se parantaa pelaajaan ja pelihahmon välistä vuorovaikutusta, koska sen liikkeissä on massan tuntua (lisää luvussa 9.2). Huonosti toteutettuna pelaaja ei välttämättä ymmärrä hahmon toimintaa tai hänelle syntyy vaikutelma hahmosta, joka ei reagoi peliohjaimen syötteisiin välittämättömästi tai uskottavasti. Tässä työssä esim. pelihahmo

valmistautuu hyppäämiseen menemällä kyyryyn ja ennen liikkeelle lähtöä nojaa hieman eteenpäin. Myös kohteen seuraaminen pelihahmon katseella tai osoittaminen kädellä on esimerkki ennakoinnista.

9.1.3 Näyttämöllepano

Näyttämöllepano tarkoittaa, että pelihahmo erottuu muusta ympäristöstä ja sen toiminta selkeästi nähtävissä sen katsojalle (Saarva, 2015). Peleissä näyttämöllepano tarkoittaa hallittua ja ennakoitavaa kameran ohjausta. (Swink, 2017, s.157)

9.1.4 Improvisoitu liike sekä analysoitu liike

Kaksi eri tapaa animoida liike. Improvisoidulla liikkeellä tarkoitetaan, että tiedetään vain animoitavan hahmon alku- ja lopputila. Analysoidulla liikkeellä tarkoitetaan, että tiedetään tasan tarkkaan hahmon liikkeen tila annetulla hetkellä (Saarva, 2015). Valmisanimaatio voisi olla synonyymi analysoidulle liikkeelle.

Tämä työn animaatiojärjestelmä uniikki muihin pelianimaatioihin verrattuna, koska animaatio on rakennettu ohjelmallisesti ja rakennetaan pelin ajon aikana muuttujista interpoloimalla, joten se luo improvisoitua animaatiota ja sen työkalut ovat rajallisia animaation tekemiseen. Myöskään mitään ’motion capturea’ tai liikkeen kaappausta ei ole käytetty työssä.

9.1.5 Epäsynkronia

Ihmisen eri kehon osat liikkuvat toiminnan aikana itsenäisesti. Epäsynkronia auttaa hahmon kehonkielen rakentamisessa. Esim. kun hahmo lakkaa kävelemästä molemmat jalat eivät pysähdy yhtä aikaa ja hahmo korjaa asentoaan pysähdyksen aikana.
Myös hahmon eri osat liikkuvat eri suuntiin. Kädet liikkuvat eri lailla jalkoihin verrattuna kävelyn ja hypyn aikana. Esim. kävellessä vasen käsi heiluu hahmon eteen, kun taas vasen jalka jää sen taakse.

9.1.6 Hidastuminen ja kiihtyminen

Hahmo ei saavuta tavoite nopeutta saman tien vaan sen kiihdyttävä ensin (Saarva, 2015). Samoin hahmon on noustava kyyrystä hypätäkseen. Myöskään pysähtyminen ei välitön

toiminta vaan hahmon on hidastuttava ennen pysähtymistä. Kiihtyvyyttä ja hidastumista visualisoidaan pelissä lyhyillä askeleilla ennen kuin hahmo saavuttaa maksiminopeuden.

9.1.7 Kaaret

Hahmojen liikkeet ovat harvoin suoraviivaisia ja niihin lähes aina sisältyy kaari. Liioittelemalla liikettä kaarella parantaa katsojan ymmärrystä hahmon liikkeestä (Saarva, 2015). Peleissä pelihahmon kaareva liike näyttää ja tuntuu paremmalta ja luonnollisemmalta (Swink, 2017, s.306). Tätä perusperiaatetta sovelletaan työssä nostamalla Bezier-kaaren kontrollipisteettä enemmän pelihahmon juostessa luoden voimakkaan kaaren askeleen liikkeestä.

9.1.8 Toissijainen liike

Hahmon muu toiminta esim. kävelyn aikana (Saarva, 2015). Voi olla käsien heiluminen tai kohteen seuraaminen katseella tai sen osoittaminen kädellä. Tässä työssä jalkojen, käsien ja katseen toiminta on jaettu omiin komponentteihin, joiden toimintaa voidaan ohjata riippumatta toisistaan.

9.1.9 Rytmitys

Liikkeiden ajoitus ja ulkoisten voimien vaikutus liikkeeseen. Vaihteleva nopeus, kiihtyvyys ja hidastuminen. Muutamalla rytmitystä muuttaa katsojan kokemusta hahmoa seuratessa, nostaen katsojan mielenkiintoa hahmoa kohtaan (Saarva, 2015). Nämä ominaisuudet ovat mallinnettu hahmon muuttuvassa askeltiheydessä, kun hahmo kävelee tai juoksee.

9.1.10 Liioittelu

Hahmon liikkeen liioittelu auttaa katsojaa ymmärtämään hahmon tilaa. Ilman liioittelua hahmon liike vaikuttaa tylsältä ja mekaaniselta. Esimerkiksi omassa työssäni hahmon askeleissa askelkorkeus on hyvin korkea. Myös kyykistyminen ennen hyppäämistä ja laskeutumisen aikana on liioiteltu. Myös hahmon pitkät raajat auttavat sen toiminnan ymmärtämisessä.

9.1.11 Anatomian ymmärrys

Tämä periaate tunnetaan myös nimellä ”piirtämisen taito” (Saarva, 2015). Vaikka tässä työssä ei käsitellä piirtämistä, keskeisenä tarkoituksena on ymmärtää animoitavan hahmon anatomia ja sen kehon keskipiste (Lasseter, 1987).

Ihminen nojaa kävellessä ylämäessä eteenpäin ja alamäessä taaksepäin, näitä piirteitä on mallinnettu pelihahmossa. Aiemmin mainittu hahmon toiminta, tasapainottelu tasojen reunoilla, vaatii ymmärrystä hahmon keskipisteestä animoidessa.

9.1.12 Hahmon kiinnostavuus

Työn pelihahmon on väriltään tylsän harmaa, joten tätä ominaisuutta tulen kehittämään tulevaisuudessa, mutta on myös tärkeä tehdä hahmon liikkumisesta sen katsojalle nautittavan näköistä.

9.2 Virtuaalisensaatio peleissä

Animaation perusperiaatteiden lisäksi työssä tutustuttiin Steve Swinkin kuvailemaan virtuaalisensaatioon ja animaation rooliin videopeleissä. Steve Swink kuvaa virtuaalisensaatiota, joka johdetaan ensisijaisesti pelaajan syötteestä. Jos pelaajalta ei tule syötettä, niin pelihahmolla ei ole ärsykettä liikkua. Syötteen lopputulos on oltava ennakoitava ja pelaaja odottaa pelihahmon reagoivan välittömästi peliohjaimen napin painallukseen (Swink, 2017, s.297-306).

Swink kuvaa animaatiota osana harmoniaa, joka koostuu animaation lisäksi ääniefekteistä ja pelaajan syötteestä. Harmoniasta kehittyy pelaajalle mielihyvän tunne, joka luo vaikutelman pelaajalle virtuaaliympäristön interaktiivisuuden fyysisyydestä. Tässä harmoniassa animaation ei kuulu häiritä pelin pelattavuutta ja pelihahmon toiminta on selkeästi kommunikoitu animaatiolla pelaajalle. Hän painottaa virtuaalisensaatiossa 12 animaation perusperiaatteen tärkeyttä harmonian saavuttamiseksi. Näistä hän korostaa venytystä ja litistystä. Jos animaatio sisältää venytystä ja litistymistä, niin se on automaattisesti parempaa, kuin animaatio, joka ei sisällä venytystä ja litistymistä. Parhaat virtuaalisensaatiot liioittelevat liikeitä korostaakseen erilaisia interaktioita ja niiden tärkeyttä pelaajalle. Työn prototyypin palautteen parantamiseksi voidaan vielä hyödyntää

ääni- ja partikkeliefektejä. Esim. kun pelihahmo kävelee, askeleista kuuluu niiden ääni ja juostessa nostavat pölyä ilmaan (Swink, 2017, s.155-158).

Swink kertoo, että pelihahmosta pitää löytyä useita tapahtumia ja toimintoja yhden saman ohjaimen painikkeen painalluksen takana ja saman animaation toistaminen useita kertoja huonontaa harmonian tunnetta. Tässä työssä hahmon kävelyn nopeutta voi säädellä ohjauksella ja nopeus ja käveltävän tason kaltevuus vaikuttaa hahmon kävelyanimaatioon. Myös työssäni otettiin riski ja lisättiin pieni viive ennen kuin hahmo hyppää saadakseen hahmon kyyristymään ennen hyppyä. Koska hahmo ei hyppää heti napin painalluksen jälkeen, on tärkeää, että kyyristyminen hyppyä varten on nopea ja eikä estä muita hahmon toimintoja, joten hahmon on kyettävä hyppäämään liikkeestä ja seisaltaan. Toinen merkittävä animaatio sensaation parantamiseksi on hahmon tilan siirtyminen tilasta toiseen sulavasti. Esim. Pysähtyminen ja putoaminen. (Swink, 2017, s.298).

Laadukas animaatio peleissä auttaa pelin kehittäjää luomaan vaikutelman virtuaalimaailman fyysisyydestä ja uskottavuudesta, jossa pelihahmo on osa sitä. Esim. tässä työssä hahmon liukuminen tai luvussa 1 mainittu vaikutelma luistelusta ja askeleiden synkronisointi kehon nopeuteen on hävittänyt liukumisen vaikutelman kokonaan tuoden vahvan tunteen pelihahmon fyysisyydestä.

10 Kohdatut haasteet ja ongelmat

Työn toteuttaminen ei ollut avain ongelmatonta. Haasteita ilmeni jo valmiissa ja uudessa teknologiassa.

10.1 Vanhat tavat

Koska hahmon ohjauskomponentti oli aikaisempaa tehtyä koodia, se sisälsi ohjelmointivirheitä, jotka löytyivät vasta, kun aloitettiin kirjoittamaan käänteiskinematiikka komponenttia ja tukea peliohjaimelle. Viat liittyivät hahmon kiihtyvyyteen ja sen rajanopeuteen. Nämä viat estivät oikean askeltiheyden asettamisen käänteiskinematiikkakomponentissa.

Seuraava haaste liittyi vanhoihin toteutus tapoihin. Saatua hahmon jalat liikkeelle huomattiin, että kääntämällä hahmon toiseen suuntaan, ei kääntänyt sen rigattuja luita. Syy tähän oli, että vanha koodi käänsi pelihahmon grafiikan ympäri, mutta ei pelihahmon sisältämiä komponentteja. Tätä ratkaisua käytettiin, koska kääntämällä hahmon grafiikkaa, ei koskettu sen osuma tunnistukseen, joiden manipulointi pelin ajon aikana on riskialtista. Ratkaisu tähän ongelmaa löytyi implementoimalla muunnosmatriisi, jolla pystyttiin kääntämään koko pelihahmo ympäri rikkomatta sen osia.

10.2 Hahmon liikkuminen portaissa

Portaat osoittautuivat hyvin haastavaksi ongelmaksi. Käänteiskinematiikkakomponentti kykeni löytämään askelpaikat erinomaisesti, mutta muut ongelmat johtuivat hahmon ohjauskomponentista. Siirtyminen portaan askeleista ylöspäin tai alaspäin ei ollut sulavaa. Hahmon nousu seuraavalle askeleelle oli välitön, kun taas hahmo putoaa alaspäin portaita alas mentäessä. Tämä ongelma aiheutti sivuvaikutusten pelin kamerassa, joka ilmeni kuvan edestakaisin nykimisenä pystyakselilla. Vasta lineaarisen interpoloinnin implementoinnin ja muutokset ohjauskomponentin geometrian tulkitsemiseen jälkeen hahmo alkoi liikkua portaissa halutun tavoitteiden mukaisesti. Muutos geometrian tulkintaan oli, että portaat tulkitaan samanlailla kuin mäet, mutta askeleet osuvat vielä portaille normaalisti.

10.3 Uuden teknologian ongelmat

Unityn 2D-käänteiskinematiikkapaketti saapui saataville vuoden 2018 lopussa ja sisälsi useita ongelmia. Yleisin niistä oli paketin käytön epävakaus, joka pysäytti Unityn toiminnan ohjelmallisesti tai kaatoi koko sovelluksen. Jo työn aikana tulleet päivitykset ovat korjanneet pakettia ja ovat tehneet siitä paljon vakaaman.

Luiden asettaminen hahmon riggauksen aikana aiheutti päänsärkyä, koska tallennuksen aikana luut siirtyivät pois niille asetetusta paikasta tehden aikaisemman työn turhaksi. Tähän ongelmaan kirjoituksen aikana ei ole vielä tullut korjausta, mutta keino ongelman kiertämiseksi on löytynyt: Työn tallentaminen yhden luun asettamisen tai sen siirron jälkeen. Tämä ei ole miellyttävä ratkaisu, koska se rikkoo työn kulun sujuvuuden ja on aikaa vievää.

Myös paketin dokumentaatio on puutteellinen ja kattaa vain käyttöönoton. Tarkkoja kuvauksia ominaisuuksista ja paketin komponenteista ja niiden esimerkeistä ei ole, joten paketin ominaisuuksien hyödyntäminen vaatii mielikuvituksen käyttöä.

11 Työn tulos

Tässä tutkielmassa ja työssä tuotettiin prototyyppi, joka automaattisesti käänteiskinematiikalla luo animaatiota hahmon liikkumisesta virtuaaliympäristössä. Animaatio on rakennettu ihmisen anatomia ja animaation perusperiaatteiden mukaan. Verrattuna aikaisempiin peliprojekteihin ja käsin tehtyyn animaatioon, prototyyppi tekee animaatiota automaattisesti nopeammin ja laadukkaammin. Ajan säästö on merkittävä. Komponentti vapauttaa valtavasti animoijan aikaa muihin animaatiotöihin.

Käytetyistä algoritmeista Limb-algoritmi sopi parhaiten jalkojen ja käsivarsien animoimiseen ja oli yksinkertaisin käyttää. FABRIK-algoritmi puolestaan sopi erinomaisesti hyvin pitkille raajoille, jotka sisältävät useita niveliä ja pääte-efektoreita, kun taas CCD-algoritmi kiertää raajan kohteen ympärille, joten lyhyet raajat toimivat CCD-algoritmilla parhaiten. Työssä käytettiin enimmäkseen Limb-algoritmia ja CCD-algoritmia. Pelkästään käänteiskinematiikan algoritmien käyttäminen ei riittänyt animaation tekemiseksi, varsinkin, kun pelihahmon tilan muuttuminen ilmeni hahmon liikkeen nykimisenä. Liikkeen yhdistäminen lineaarisen interpolointiin täydensi käänteiskinematiikasta syntyviä puutteita ja paransi animaation perusperiaatteiden saavuttamista.

Pelihahmon piirtämisen lisäksi työläin vaihe on valmistaa tahkosto pelihahmolle. Varsinkin, kun työkalut sisältävät bugeja niiden käytön vaikeuttamiseksi. Itse prototyyppi on helppo ottaa käyttöön. Riggauksen valmistuttua hahmolle asetettaan pääte-efektorit ja lisätään käänteiskinematiikka algoritmiratkaisijat. Prototyyppiin sitten lisätään viitaukset pääte-efektoreihin animaation luomiseksi. Hahmon kinematiikan juuripiste on vielä lopuksi hienosäädettävä kohdilleen, jotta hahmon jalat asettuvat tason pinnoille halutulla tavalla.

Uskon, että tämänkaltaisten animaatiojärjestelmien käyttö kasvaa myös 3D-peleissä ja 3D-animaaitoelokuviss kanssa sekä muissa tehtävissä, joissa hahmon kävelyanimaation luominen käsin on työlästä ja aikaa vievää.

11.1 Suorituskyky

Koska tässä työssä toteutettu animaatio on laskentateholtaan vaativampaa, kuin staattisen valmisanimaation, tehtiin rasitustestaus, jossa selvitin kuinka paljon prosessoritehoa useamman pelihahmon ajaminen vie.

Unity sisältää ’profiler’ -työkalun, jolla voi mitata pelin suorituskykyä. Pelin suorituskykyä mitataan FPS (Frames Per Second) -arvolla, eli kuinka monta kuvaa pelimoottori kykenee piirtämään yhden sekunnin aikana. Useat modernit pelit pyrkivät 60 FPS arvoon, mutta myös 30 FPS arvo on hyväksyttävissä ei-toimintapeleissä. Kaikki suorituskykytestit ajettiin Unityn ’developer build’ -asetuksilla ja FPS:ää mitattiin ’profiler’-työkalulla ja pelin sisäisellä FPS-laskurilla. Suorituskyky mittauksen ajon aikana pelihahmot ovat aluksi paikallaan muutaman sekunnin, ennen kuin niitä alettiin liikutella, jolloin mittausarvo otettiin. Yhdellä pelihahmolla saavutettiin 1000 FPS arvo, mikä on suurin arvo, jonka profile-työkalu voi mitata, joten yksi pelihahmo ei merkittävästi aiheuttanut suurta rasitusta suorituskykyyn.

Pelihahmojen määrä	**Profiler FPS**	**Pelin sisäinen FPS**
1	1000	60
10	n. 500	60
20	n. 200	60
30	n. 150	60
40	n. 100	60
50	n. 100	60

Taulukko 1. Pelihahmojen määrän vaikutus FPS:ään.

50 pelihahmoa stabiililla 60 FPS:llä on enemmän kuin tarpeeksi muihin peliprojektin tarkoituksiin, varsinkin kun joiden hahmojen liikeitä ei tarvitse simuloida, koska ne ovat pelaajan näkökentän tai simuloitavan alueen ulkopuolella.

11.2 Jatkokehitys ja -työ

Prototyyppi tällä hetkellä toimii vain Unity-pelimoottorissa, joten eräs jatkotyö olisi tehdä prototyypistä alustariippumaton. Nykyistä prototyyppiä voi laajentaa uusilla ominaisuuksilla.

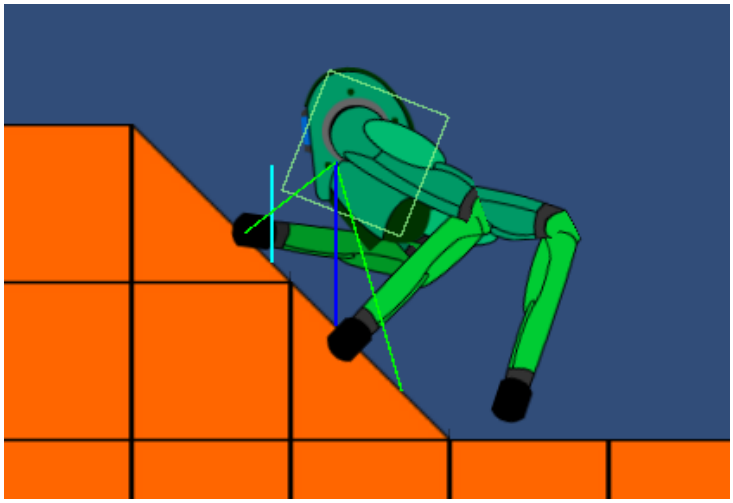
11.2.1 Laajennus 3D-animaatiohon ja pelianimaation ulkopuolelle

Käänteiskinematiikka ja hahmojen riggaus on jo ollut olemassa 3D animaatioihin useita vuosikymmeniä ja on ollut käytössä peleissä 3D-peligraafikan yleistettyä. Työ on toteutettu vektorilaskennalla, joten niiden laskuista puuttuu Z-akseli ja nykyiseen muunnosmatriisiin tulisi lisätä kiertomatriisi hahmon askeleiden suunnan laskemiseksi. Tällä hetkellä ei myöskään ole mahdollista animoida lantion tai olkapäiden heiluttamista kävelyn aikana, joten ne tulisi lisätä osaksi hahmon animaatiota.

11.2.2 Askeleiden optimointi ja reaktiivinen liike

Tällä hetkellä prototyyppi etsii askeleita jatkuvasti hahmon edestä, kun hahmo on liikkeessä. Eräs optimointi ratkaisu on laskea etukäteen askeleen lopullinen sijainti, kun hahmon nopeus on vakio.

Komponentti on suunniteltu ensisijaisesti ihmismäisiin hahmoihin, joten sen laajentaminen, vaikka nelijalkaisiin eläimiin toisi uusia mahdollisuuksia.



Kuvio 23. Kolmijalkainen pelihahmo. Askeleet toimivat yllättävän hyvin ilman ylimääräistä työtä.

Myös muita uusia ominaisuuksia olisi reaktiivinen järjestelmä, joka simuloi ulkoisia voimia hahmoon. Esim. kova tuulen puuska tai tönäisy ja vielä taakan simulointi animaatiojärjestelmässä kasvattaisi sen dynaamisuutta ja uskottavuutta (Harold & Dimitri, 2001), (Komura, Lueng, & Kuffner, 2004).

Lähteet

Aristidou, A., Lasenby, J., Chrysanthou, Y. & Shamir, A. (2018). Inverse Kinematics Techniques in Computer Graphics A Survey. *Computer Graphics forum* 37, 35-58. doi:10.1111/cgf.13310

Haslinger, J. & Mäkinen R.A.E. (2003). *Introduction to Shape Optimization: Theory, Approximation and Computation,* SIAM.

Komura, T., Lueng, H. & Kuffner, J. (2004) Animating Reactive Motions for Biped Locomotion. *VRST '04: Proceedings of the ACM symposium on Virtual reality software and technology.* 32-40. https://dl.acm.org/doi/10.1145/1077534.1077542

Harold S. & Dimitris, M. (2001) Automating gait generation. *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques.* 261-270. https://doi.org/10.1145/383259.383288

Multon, F., France, L. Cani-Gascuel, M. & Debunne, G. (1999) Computer Animation of Human Walking: a survey. *The Journal of Visualization and Computer Animation* 10*,* 39-54.

Koritnik, T. Bajd, T. & Munih M., Ljubljana. (2010). A Simple Kinematic Model of a Human Body for Virtual Environments. *Advances in Robot Kinematics*, 401-408. https://doi.org/10.1007/978-90-481-9262-5_43

Ruuskanen, A. (2018). *Inverse Kinematics in Game Character Animation* (Opinnäytetö, Kajaanin ammattikorkeakoulu). Haettu osoitteesta http://urn.fi/URN:NBN:fi:amk-2018120419993

Rantala, T. (2013). *Animation Of A High-Definition 2D Fighting Game Character* (opinnäytetyö, Kajaanin ammattikorkeakoulu). Haettu osoitteesta http://urn.fi/URN:NBN:fi:amk-201305148471

Shih-kai, C. & Hahn, J. (1999) Animation of Human Walking in Virtual Environments. *Proceedings Computer Animation*, 4-15. https://doi.org/10.1109/CA.1999.781194

Bruderlin, A. & Calvert, T. (1989). Goal-Directed, Dynamic Animation of Human Walking. *Computer Graphics* 23, 233-241. https://doi.org/10.1145/74334.74357

Lasseter, J. (1987) Principles of Traditional Animation Applied To 3D Computer Animation. *Computer Graphics* 21, 35-44. https://doi.org/10.1145/37402.37407

Burtnyk, N. & Wein, M. (1976) Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation. *Communications of the ACM* 19. https://doi.org/10.1145/360349.360357

Thomas, F. and Johnston, O. (1981). *Disney Animation-- The Illusion of Life*, Abbeville Press.

Wink, S. (2009). *Game Feel: A Game Designer's Guide to Virtual Sensation*, Morgan Kaufmann Publishers.

Saarva, A. (2015). *Animaation 12 perusperiaatetta – työkaluja nukenkäsittelijälle* (Opinnäytetyö, Turun ammattikorkeakoulu). Haettu osoitteesta http://urn.fi/URN:NBN:fi:amk-2015100815242

Cappellini, G., Ivanenko, Y., Poppele, R. & Lacquaniti, F. (2006). Motor Patterns in Human Walking and Running. *Journal of Neurophysiology* 95, 3426-37. doi:10.1152/jn.00081.2006

Psarras, A., Mertyri, D & Tsaklis, P. (2016). Biomechanical Analysis Of Ankle During The Stance Phase Of Gait On Various Surfaces: A Literature Review. *Human Movement* 17, 140-147. doi:10.1515/humo-2016-0026

Meyns, P., Bruijn, S. & Duysens, J. (2013). The how and why of arm swing during human walking. *Gait & Posture* 38, 555-562. https://doi.org/10.1016/j.gaitpost.2013.02.006

Zelik, K. (2012). Mechanical Work as an Indirect Measure of Subjective Costs Influencing Human Movement. *PLoS ONE* 7. https://doi.org/10.1371/journal.pone.0031143

Satran, M. (30.3.2017) Why Transformation Order Is Significant [Dokumentaatio]. Haettu osoitteesta https://docs.microsoft.com/en-us/dotnet/framework/winforms/advanced/why-transformation-order-is-significant

Joensuu, J. (2016). *3D-ALAN SANASTO – 3D-grafiikan termit suomeksi* (Opinnäytetyö, Kajaanin ammattikorkeakoulu). Haettu osoitteesta http://urn.fi/URN:NBN:fi:amk-2016060612045

Hinton-Jones, A. (4.12.2018) Getting Started with 2D Inverse Kinematics [Blogikirjoitus]. Haettu osoitteesta https://blogs.unity3d.com/2018/12/04/getting-started-with-2d-inverse-kinematics/

Hilton-Jones, A. (11.9.2018) Getting Started with Unity's 2D Animation Package [Blogikirjoitus]. Haettu osoitteesta https://blogs.unity3d.com/2018/11/09/getting-started-with-unitys-2d-animation-package/

Rotenberg, S. (2016), Inverse Kinematics (part 2) [Opetusmateriaali]. Haettu osoitteesta https://cseweb.ucsd.edu/classes/sp16/cse169-a/slides/CSE169_09.pdf

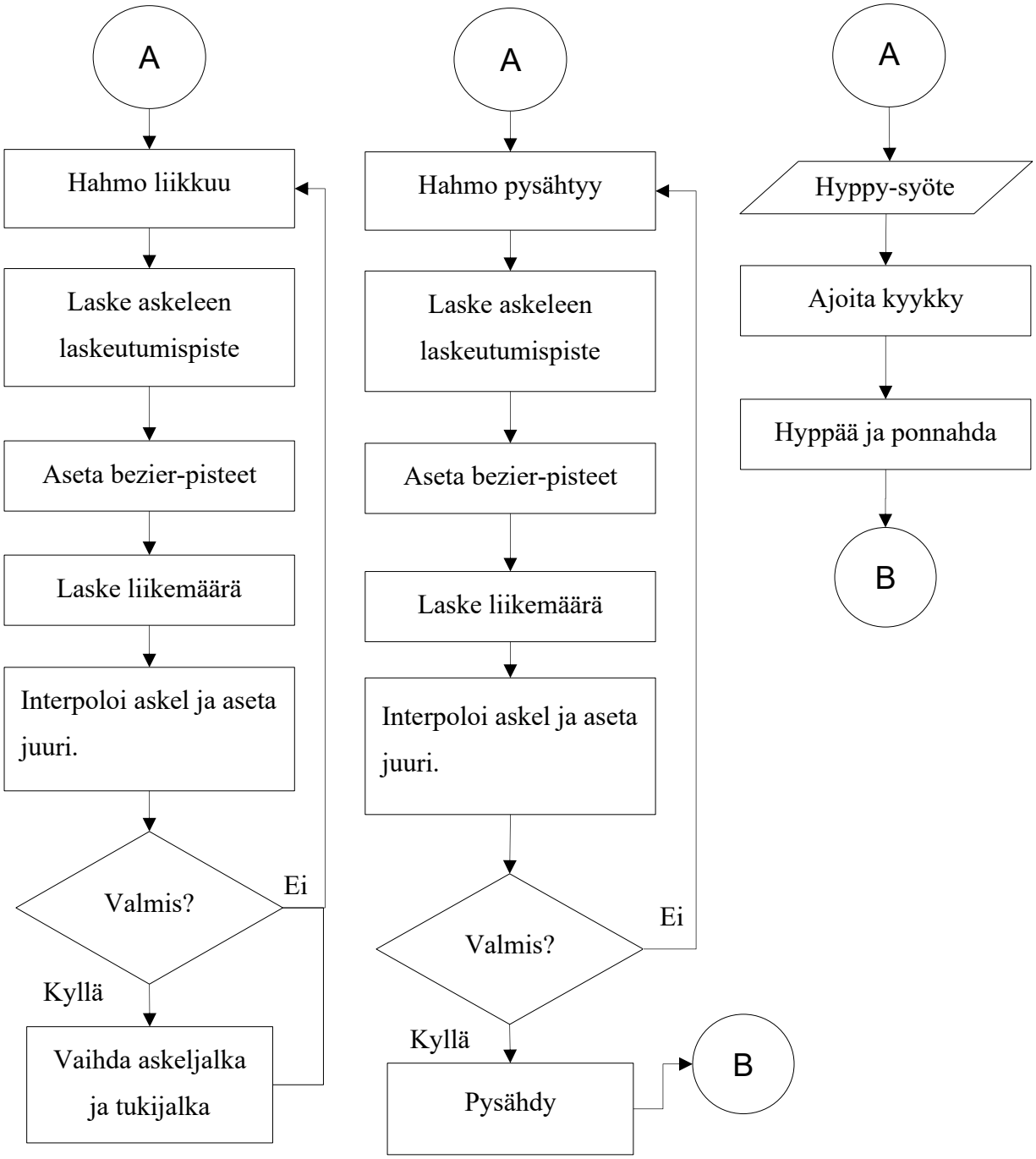
Unity. (2019). Using Components. Haettu osoitteesta https://docs.unity3d.com/Manual/UsingComponents.html

Unity. (2018). 2D Inverse Kinematics (IK). Haettu osoitteesta https://docs.unity3d.com/Packages/com.unity.2d.ik@1.3/manual/index.html

Liitteet

Liite 1

Askeleiden vuokaavio



Liite 2

Komponenttikaavio