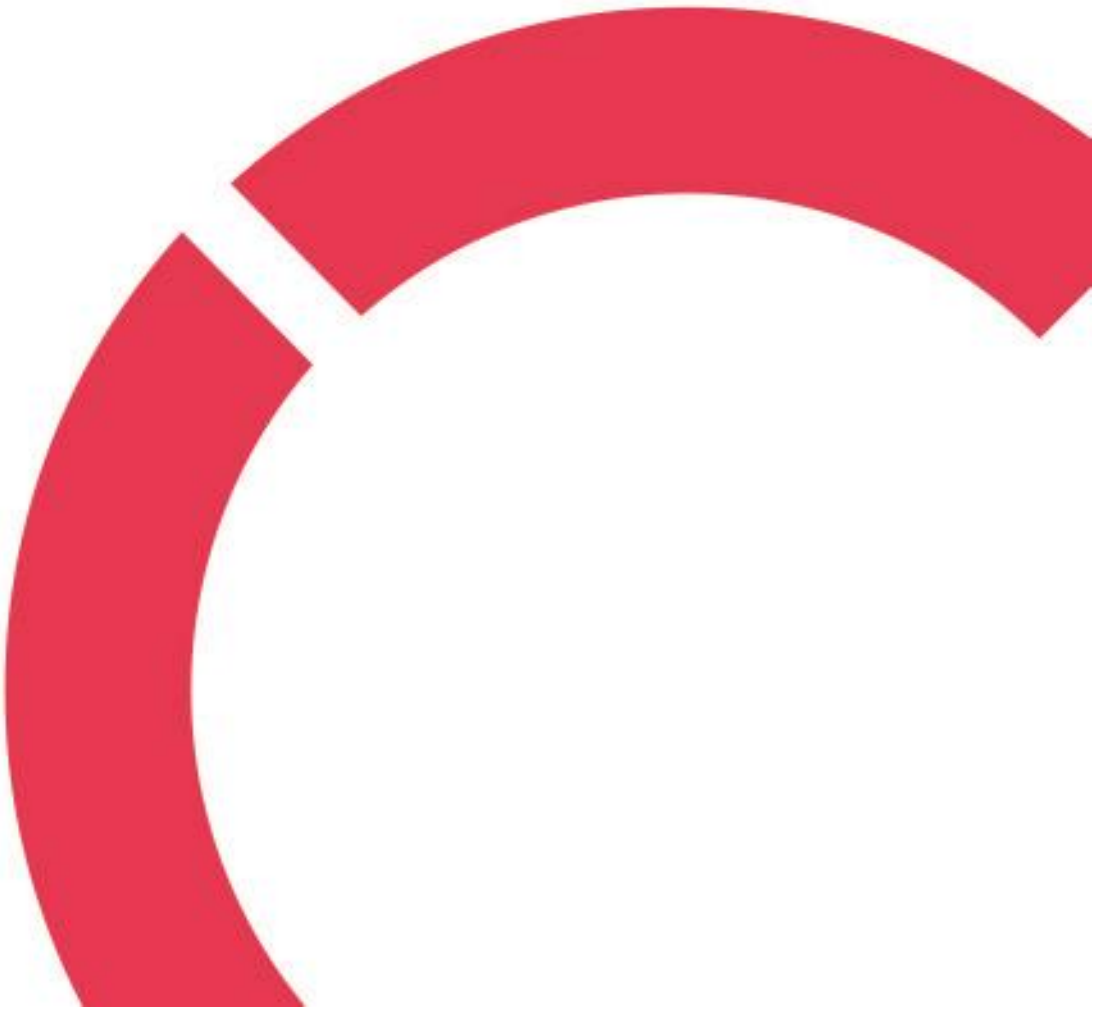
Eemil Keski-Korpi ATIS21K

UNITY ANIMATOR

Pelihahmon animaatiosarjan luominen Unityllä

Opinnäytetyö
CENTRIA-AMMATTIKORKEAKOULU
tieto- ja viestintä tekniikka koulutus
Toukokuu 2025



TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Centria-ammattikorkeakoulu	**Aika** Toukokuu 2025	**Tekijä/tekijät** Eemil Keski-Korpi
Koulutus Tieto- ja viestintätekniikka		☒ AMK ☐ YAMK
Työn nimi UNITY ANIMATOR. Pelihahmon animaatiosarjan luominen Unityllä		
Työn ohjaaja Sari Lipsanen		**Sivumäärä** 19

Opinnäytetyössä tutkittiin, kuinka Unityn Animator-työkalua hyödynnetään pelihahmojen animaatioiden hallinnassa. Työssä keskityttiin yksinkertaisen roolipeliä varten rakennetun hahmon liikesarjan luomiseen hyödyntäen Mixamosta hankittuja valmiita animaatioita. Teoriapohjana käsiteltiin pelimoottoreiden yleisiä ominaisuuksia sekä tarkemmin Unityn ja siihen liittyvien animaatiotyökalujen toimintaa.

Käytännön osuudessa valittiin hahmo ja sille sopivat animaatiot, jotka ladattiin Mixamo-palvelusta ja tuotiin Unityyn. Animaatiot yhdistettiin Animator Controllerin kokonaisuudeksi, jossa kukin määritetään reagoimaan halutulla tavalla. Työssä analysoitiin myös animaatiopuun rakentamisen vaiheita, parametrien hallintaa ja siirtymien vaikutusta pelihahmon toiminnallisuuteen.

Opinnäytetyön lopputuloksena syntyi toimiva ja selkeästi jäsennelty animaatiopuu, joka mahdollistaa sujuvan liikkeen ja reaktiivisen ohjauksen pelihahmolle ilman koodausta. Työ osoitti, että animaatioiden yhdistely ja huolellinen siirtymien hallinta ovat keskeisiä tekijöitä peliohjelmoinnissa.

Asiasanat
Animator Controller, Blend Tree, Mixamo, Unity

ABSTRACT

Centria University of Applied Sciences	**Date** June 2025	**Author** Eemil Keski-Korpi
Degree programme Bachelor of Engineering, Information and Communications Technology		
Name of thesis UNITY ANIMATOR. Creating a character's animation tree with Unity		
Centria supervisor Sari Lipsanen		**Pages** 19

The purpose of this thesis was to explore and learn how Unity's Animator tool can be used to control character animations in game development. The project focused on creating a simple animation tree for a role-playing game character by using pre-made animations from the Mixamo service. The theoretical part included general information of game-engines and a bit more detailed examination of Unity and its animation tools.

In the practical part of the thesis, a 3D character was rigged in Mixamo, and animations were downloaded and imported into Unity Editor. These animations put into an Animator Controller, where each one is made to respond desired way. The process of building the animation tree, controlling parameters, and defining transitions was analyzed in detail.

The result of the thesis was a functioning and clearly structured animation system that allowed smooth motion for a playable character without additional programming. The findings highlighted the importance of carefully managing animation transitions and parameters when building interactive and believable character behavior in a game environment.

Key words
Animator Controller, Blend Tree, Mixamo, Unity

KÄSITTEIDEN MÄÄRITTELY

Animator Controller

Unityn sisäinen työkalu, jolla hallitaan eri animaatioiden siirtymiä ja yhteyksiä toisiinsa. Se käyttää animaatiopuuta (State Machine) hahmon liikkeitä varten.

Animation Clip

Yksittäinen animaatio, kuten kävely, hyppy tai isku, joka sisältää hahmon liikkeeseen liittyvät key-frame-tiedot. Näitä klippejä käytetään Animator Controllerissa tilojen sisällä.

Blend Tree

Animaatiotekniikka, joka mahdollistaa useiden animaatioiden yhdistämisen sujuvasti. Esimerkiksi juoksu ja kävely voidaan yhdistää yhdeksi jatkumoksi nopeuden perusteella.

Transition

Määritys Unity Animatorissa, joka käskee animaation vaihtumaan toiseen, kun tietty ehto tai parametri täyttyy.

Int, Trigger, Bool, Float

Ovat parametrejä Unity Animatorissa, joilla voidaan aktivoida tietyn siirtymän animaatiosta toiseen esimerkiksi silloin, kun käyttäjä painaa tiettyä nappia.

State (Tila)

Animatorin perusyksikkö, jossa yksi animaatio (animation clip) toistetaan. Esimerkiksi "Idle", "Run" ja "Jump" ovat eri tiloja, joihin hahmo voi siirtyä siirtymien avulla.

Rigging

Prosessi, jossa 3D-hahmoon lisätään luuranko (skeleton), jonka avulla hahmoa voidaan animoida. Mixamo tarjoaa automaattisen riggaustoiminnon, joka mahdollistaa nopean käyttöönoton pelimoottorissa. Mixamossa automaattisessa riggauksessa hahmolle pitää määrittää nivelkohdat: lantio, leuka, ranteet, kyynärpäät ja polvet.

KUVAT

1 JOHDANTO

Digitaalisten pelien kehitys on viime vuosikymmeninä kokenut valtavan harppauksen, ja yhä useampi kehittäjä hyödyntää tehokkaita pelimoottoreita nopeuttaakseen ja helpottaakseen kehitysprosessia. Pelimoottoreiden kehittyessä pelien laatu, visuaalisuus ja monimutkaisuus ovat kasvaneet merkittävästi. Nykyään jopa pienet kehittäjä tiimit voivat luoda ammattimaisia pelejä työkaluilla, jotka olivat aiemmin vain isoimpien studioiden ulottuvilla.

Yksi keskeisistä osa-alueista pelisuunnittelussa on hahmoanimaatioiden hallinta, joka vaikuttaa merkittävästi pelikokemukseen. Sujuvat ja luonnolliset liikkeet lisäävät pelattavuutta ja luovat vahvemman yhteyden pelaajan ja pelimaailman välille. Hyvin toteutetut animaatiot siis parantavat pelin visuaalista ilmettä. Animaatioiden avulla voidaan myös viestiä pelaajalle hahmon tilasta tai ympäristön eri tapahtumista ilman erillistä ilmoitusta.

Tämän opinnäytetyön tarkoituksena on perehtyä Unityn Animator-työkalun käyttöön pelihahmojen animoinnissa ja tutkia, miten yksinkertainen animaatiosarja voidaan rakentaa pelidemon tarpeisiin. Projekti toimii oppimisprosessina, jossa käsitellään muun muassa animaatioiden luonti, niiden hallinta sekä siirtymät tilojen välillä. Unity on valittu kehitysalustaksi sen laajan dokumentaation, käyttäjäyhteisön ja helppokäyttöisyyden vuoksi. Työn lopputuloksena syntyy hahmo, joka osaa suorittaa tyypilliseen roolipeliin liittyviä animaatioita.

2 MIKÄ ON PELIMOOTTORI

Pelimoottori on ohjelmistoalusta, joka mahdollistaa digitaalisten pelien kehittämisen tarjoamalla valmiit työkalut ja järjestelmät pelin keskeisten osa-alueiden toteuttamiseen. Sen avulla pelinkehittäjät voivat keskittyä pelin sisällön suunnitteluun ilman, että heidän tarvitsee rakentaa jokainen tekninen toiminnallisuus, kuten fysiikkalaskenta, renderöinti tai animaationhallinta, alusta asti itse (Lappalainen 2024).

Pelimoottoriin kuuluu tyypillisesti muun muassa grafiikkamoottori, joka huolehtii pelin visuaalisesta esityksestä; fysiikkamoottori, joka simuloi luonnonlakeja kuten painovoimaa ja törmäyksiä; animaatiotyökalut hahmojen liikuttamiseen sekä ääni- ja efektimoottorit, jotka hallinnoivat äänimaisemaa ja visuaalisia tehosteita (Lappalainen 2024). Lisäksi pelimoottoreihin sisältyy usein skriptityökaluja pelilogiikan ohjelmointiin sekä editori, jolla pelin sisältöä voidaan muokata visuaalisesti.

Pelimoottorit ovat tärkeitä myös monialustakehityksen kannalta. Monet nykyaikaiset pelimoottorit tukevat suoraan useita julkaisuympäristöjä kuten PC, mobiili, konsolit ja selainpohjaiset ratkaisut. Tämä mahdollistaa pelin julkaisemisen usealle laitteelle ilman, että koko sovellus täytyy rakentaa uudelleen joka alustalle (RocketBrush Studio 2024).

Oikean moottorin valinta riippuu projektin vaatimuksista, kehittäjätiimin koosta, budjetista sekä kehittäjien ohjelmointitaidosta. Esimerkiksi Unity, Unreal Engine ja Godot tarjoavat erilaisia vahvuuksia, jotka tekevät niistä sopivia erilaisiin käyttötarkoituksiin (RocketBrush Studio 2024; Godot Engine).

Yhteenvetona voidaan todeta, että pelimoottori on olennainen työkalu modernissa pelinkehityksessä. Se mahdollistaa tehokkaan työnkulun, nopean prototypoinnin ja valmiit integraatiot kriittisiin pelin osa-alueisiin. Ilman pelimoottoria pelin tekninen toteutus olisi huomattavasti hitaampaa, monimutkaisempaa ja kalliimpaa.

2.1 Unity

Unity on yksi maailman käytetyimmistä ja suosituimmista pelimoottoreista, ja sitä käyttää laajasti niin indie-kehittäjät kuin ammattilaistenkin (RocketBrush Studio 2024). Yksi Unityn suurimmista eduista on sen matala aloituskynnys: käyttöliittymä on intuitiivinen ja dokumentaatio kattavaa, mikä tekee ohjelmistosta erittäin aloittelijaystävällisen.

Unityn ohjelmointikielenä käytetään C#:aa, joka on moderni ja helposti opittava kieli verrattuna esimerkiksi Unreal Enginen C++-kieleen (RocketBrush Studio 2024). Lisäksi Unityn Asset Store tarjoaa kehittäjille laajan valikoiman valmiita resursseja, kuten 3D-malleja ja partikkeliefektejä, joiden avulla pelinkehitys voidaan aloittaa nopeasti ilman, että kaikkea tarvitsee rakentaa alusta asti.

Unity tukee monia alustoja, kuten Windows, macOS, Android, iOS, WebGL, PlayStation, Xbox ja Nintendo Switch, mikä tekee siitä erittäin monipuolisen pelimoottorin (RocketBrush Studio 2024). Suuren käyttäjäyhteisön ansiosta Unity tarjoaa myös paljon avoimen lähdekoodin projekteja ja opetusmateriaaleja verkossa.

2.2 Unreal Engine

Unreal Engine on Epic Gamesin kehittämä pelimoottori, joka tunnetaan erityisesti kyvystään tuottaa huippuluokan grafiikkaa ja realistisia visuaalisia efektejä (Tristem 2022). Se on ollut käytössä lukuisissa AAA-pelituotannoissa.

Unreal Engine käyttää pääasiassa C++-ohjelmointikieltä, mutta tarjoaa myös visuaalisen Blueprint-järjestelmän, jonka avulla pelilogiikkaa voi rakentaa ilman syvällistä ohjelmointitaitoa (Tristem 2022). Tämä tekee siitä joustavan työkalun sekä ohjelmoijille että suunnittelijoille.

Moottorin avoimuus, jossa rojaltimaksuja maksetaan vasta, kun pelin tulot ylittävät tietyn rajan, tekee Unreal Enginestä houkuttelevan vaihtoehdon myös pienemmille kehittäjätiimeille (Tristem 2022). Vaikka oppimiskynnys on korkeampi kuin esimerkiksi Unityllä, Unreal tarjoaa palkkioksi laajat ominaisuudet ja huippuluokan suorituskyvyn.

2.3 Godot

Godot on ilmainen ja avoimen lähdekoodin pelimoottori, joka on saavuttanut suuren suosion erityisesti indie-kehittäjien keskuudessa (Godot Engine). Se tarjoaa tehokkaat työkalut niin 2D- kuin 3D-pelien kehittämiseen.

Godotin oma skriptikieli GDScript muistuttaa syntaksiltaan Pythonia, mikä tekee siitä helposti lähestyttävän aloitteleville kehittäjille (Godot Engine). Lisäksi Godot tukee C#-, C++- ja visuaalista koodausta.

Koska Godot on yhteisölähtöinen projekti, se kehittyy nopeasti käyttäjäyhteisön panostuksella, mikä tekee siitä erityisen houkuttelevan vaihtoehdon pienempiin ja kokeellisiin projekteihin (Godot Engine).

Vaikka Godotin grafiikkaominaisuudet eivät vielä täysin vastaa Unityn tai Unreal Enginen tarjoamaa huipputasoa, sen keveys, avoimuus ja joustavuus tekevät siitä erittäin varteenotettavan valinnan erityisesti 2D-pelien kehittämiseen (RocketBrush Studio 2024).

3 MIXAMO

Mixamo on Adoben tarjoama ilmainen verkkopalvelu, jonka avulla käyttäjät voivat ladata valmiita 3D-hahmoja sekä niille valmiita animaatioita. Palvelun avulla voidaan nopeuttaa merkittävästi pelinkehitysprosessia, erityisesti prototyyppivaiheessa, jossa aikaa ja resursseja ei ole tuhlattavaksi omien hahmojen ja animaatioiden luomiseen alusta alkaen. Mixamon suurimpia etuja ovat sen helppokäyttöinen selainpohjainen käyttöliittymä ja automaattinen riggaustoiminto, jonka avulla myös käyttäjän omat hahmomallit voidaan sovittaa valmiisiin animaatioihin.

Mixamo tarjoaa laajan kirjaston erilaisia animaatioita, kuten kävely, juoksu, hyppääminen, taisteluliikkeet ja monia arkisia eleitä. Näitä animaatioita voidaan myös esikatsella suoraan selaimessa ennen lataamista. Animaatiot on suunniteltu yhteensopiviksi Unityn ja Unreal Enginen kaltaisten pelimoottorien kanssa, ja latausvaiheessa käyttäjä voi valita, haluaako ladata animaatiot hahmon kanssa (With Skin) vai pelkkinä liikkeinä (Without Skin). Tämä tekee Mixamosta joustavan työkalun erilaisiin pelinkehityksen tarpeisiin.

Tässä opinnäytetyössä Mixamoa käytettiin hahmon liikesarjojen luomiseen. Työssä valittiin Mixamon kirjastosta muun muassa seuraavat animaatiot: paikallaan seisominen (Idle), juoksu (Run), miekkahyökkäykset (Sword Attack 1, 2 ja 3) sekä kuolemaan liittyvä animaatio (Death). Mixamon käyttö oli erityisen hyödyllistä, koska se mahdollisti animaatioiden nopean integroinnin Unityyn ilman manuaalista keyframe-animaatiota tai kolmannen osapuolen 3D-ohjelmistoja, kuten Blenderiä.

Vaikka Mixamo on erinomainen työkalu aloittelijoille ja prototyypeille, sen käyttöön liittyy myös haasteita. Esimerkiksi hahmojen automaattinen riggaus ei aina onnistu täydellisesti, erityisesti jos käytetään ulkoisia 3D-malleja, joiden rakenne ei täysin vastaa Mixamon odotuksia. Tästä syystä on tärkeää tarkastella lopputulosta huolellisesti ennen käyttöä pelimoottorissa.

Kokonaisuudessaan Mixamo tarjoaa helpon ja nopean tavan saada hahmo eloon, ja se sopii erinomaisesti oppimiseen, kokeiluun ja pelidemon varhaisvaiheeseen.

4 UNITY ANIMATOR

Unity Animator on keskeinen osa Unityn animaatiotyökalupakettia, joka mahdollistaa hahmojen ja objektien liikkeiden hallinnan pelin aikana. Animator ei ole pelkästään animaatioiden toistaja, vaan se on järjestelmä, joka yhdistää animaatiot, logiikan ja pelitapahtumat toisiinsa parametrein, siirtymin ja tiloihin perustuen. Käytännössä Animator toimii tilakoneena, jossa hahmo siirtyy tilasta toiseen ehtojen täyttyessä.

Animator Controller on komponentti, joka määrittää animaatioiden kulun. Se sisältää tiloja (states), joissa jokaisessa on yksittäinen animaatio, kuten idle, juoksu tai hyökkäys. Tilojen välillä siirrytään siirtymien (transitions) avulla, ja siirtymiä ohjataan parametreilla kuten "Speed" (liikkumisnopeus), "Bool" (true/false), "Trigger" (laukaiseva tapahtuma) tai "Int" (kokonaisluku).

Animator-järjestelmän keskeisiä osia ovat

- Animator Controller: Rakenne, jonka sisällä animaatiot ja niiden siirtymät hallitaan.
- Animation Clips: Yksittäisiä animaatioita, kuten kävely, juoksu, hyppy tai taisteluliike.
- Transitions: Määrittävät milloin siirrytään animaatiosta toiseen (esimerkiksi kun pelaaja painaa hyppynäppäintä).
- Parameters: Muuttujat, joiden avulla peli seuraa tilaa ja reagoi siihen – esimerkiksi "Speed" kertoo, kuinka nopeasti hahmo liikkuu ja osaa nopeuden perusteella käyttää oikeaa animaatiota.

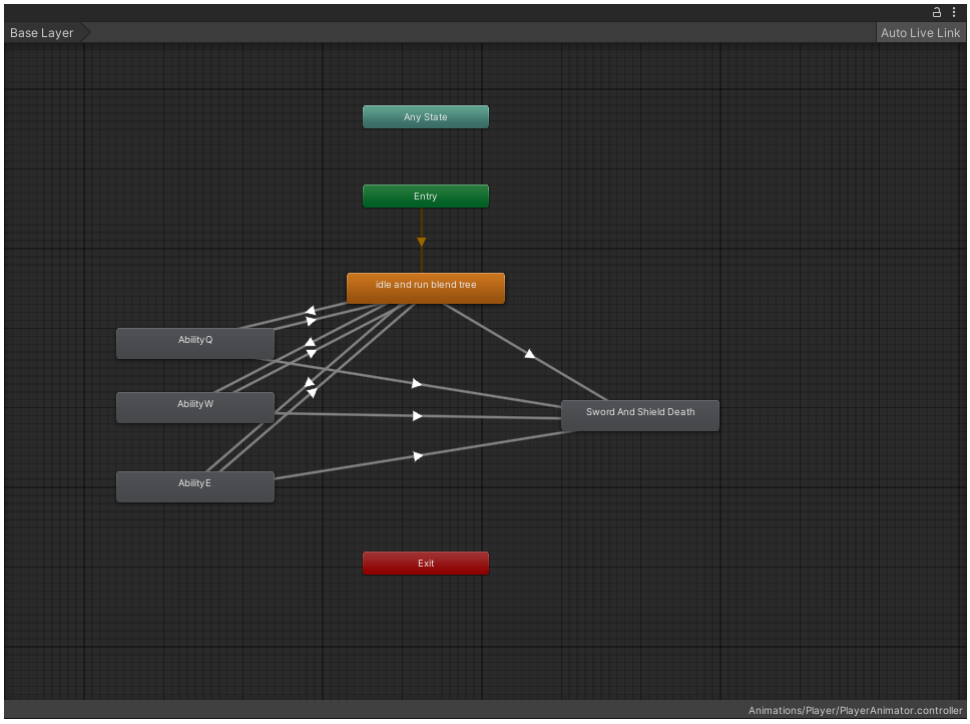
Animatorin käyttö Unityssä mahdollistaa monipuolisen, visuaalisen animaatioiden hallinnan ilman raskasta ohjelmointia. Lisäksi Animator tarjoaa mahdollisuuden yhdistää useita animaatioita sujuviksi jatkumoiksi Blend Treen avulla, mikä parantaa pelin liikkeen realistisuutta ja pelaajan kokemusta. Usein yksi Animator Controller voi sisältää kymmeniä animaatiotiloja ja siirtymiä, joiden logiikkaa optimoidaan tarkasti pelin tarpeiden mukaan.

5 ANIMAATIOPUUN RAKENNE JA PARAMETRIEN HALLINTA

Animaatiopuu muodostaa hierarkkisen rakenteen, jossa kaikki hahmon liikkeet ja niiden siirtymät on järjestetty visuaalisesti. Jokainen tila (State) vastaa yhtä animaatiota. Esimerkiksi idle-tila edustaa paikallaanoloa ja run-tila juoksua. Animaatiopuu alkaa "Entry"-tilasta, josta siirrytään muihin tiloihin riippuen parametreista.
Projektissani toteutin seuraavat tilat:

- Idle and Run Blend Tree: Tämä tila yhdistää paikallaanolon ja juoksun yhteen jatkumoon. Siirtymä tapahtuu "Speed"-parametrin perusteella, jonka arvot vaihtelivat välillä 0–1.
- Ability Q, W ja E: Nämä tilat aktivoivat hahmon kykyihin liittyvät animaatiot, kuten taikaiskut tai hyökkäykset.
- Sword and Shield Death: Tila, jossa hahmo kaatuu maahan ja liike lakkaa täysin.

Parametreilla ohjataan, milloin ja miten hahmo siirtyy eri tilojen välillä. Esimerkiksi "Trigger"-parametri voidaan aktivoida, kun pelaaja painaa tiettyä näppäintä, jolloin animaatiopuu siirtyy hyökkäystilaan. "Speed" puolestaan muuttuu jatkuvasti riippuen hahmon liikkeestä, ja sitä käytetään Blend Tree -rakenteessa yhdistämään eri liikkeet saumattomaksi kokonaisuudeksi.



KUVA 1. RPG Animaatiopuu (Keski-Korpi Näyttökuva 2025)

Animatorin visuaalinen editori mahdollistaa näiden tilojen ja siirtymien muokkaamisen graafisesti. Kuvassa 1 näkyvä rakenne näyttää esimerkkinä pelihahmon Animator-puun, jossa siirtymät hahmon eri toimintojen välillä on määritelty tarkan logiikan mukaan.

Animatorin tehokas käyttö vaatii huolellista suunnittelua ja testailua, erityisesti kun hahmolla on paljon erilaisia toimintoja. Oikein rakennettu animaatiopuu ja selkeät siirtymät takaavat sujuvan pelikokemuksen ja reaktiivisen ohjauksen, mikä on tärkeää erityisesti toimintapohjaisissa peleissä.

6 TOTEUTUS

Hahmojen ja niiden animaatioiden integrointi pelimoottoriin, kuten Unityyn, on olennainen osa pelinkehityksen työprosessia. Työn käytännön osuudessa keskeisenä tavoitteena oli selvittää, kuinka hahmo ja siihen liittyvät animaatiot voidaan tehokkaasti yhdistää ja testata Unityn käyttöympäristössä. Tämä vaihe sisälsi valintojen tekemistä, teknisten asetusten määrittelyä sekä kokeiluja ja havaintoja, jotka esitellään seuraavissa alaluvuissa.

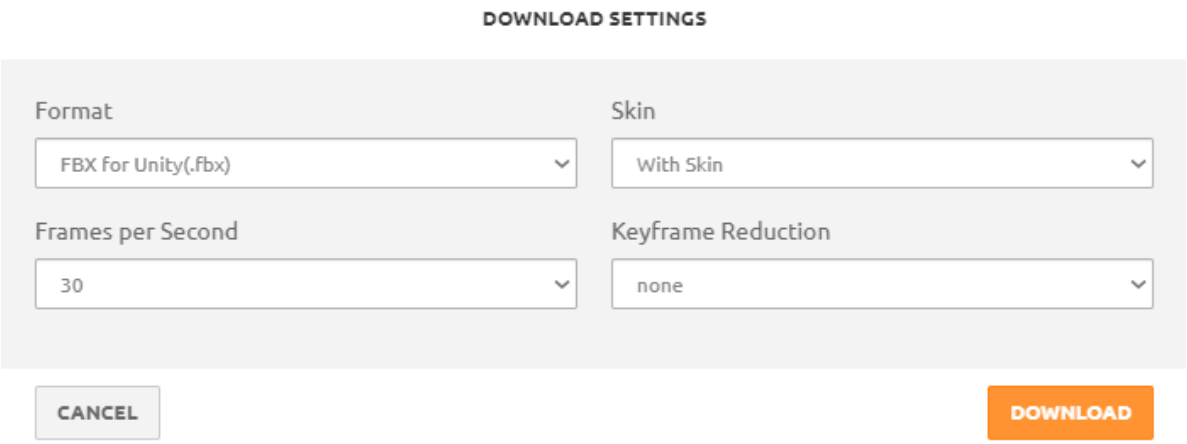
6.1 Hahmo ja Mixamo

Ensimmäiseksi projektin kannalta oli tärkeää valita sopiva hahmo, joka tukee työn tavoitteita ja toimii teknisessä ympäristössä. Työhön valittiin oma hahmo, joka tuotiin Mixamo-palveluun. Mixamo on selainpohjainen työkalu, jolla voidaan rigata hahmoja automaattisesti sekä ladata niille valmiita animaatioita.

Koska työssä käytettiin ulkopuolista hahmomallia, on tärkeää määrittää luuranko (rig) oikein Mixamossa. Jos luurankoa ei sijoita tarkasti hahmon rakenteeseen sopivaksi, animaatiot voivat vääristyä tai liike ei näy oikein Unityssa. Tämä voi aiheuttaa virheitä, kuten liikkeiden epärealistisuutta tai hahmon venymistä.
Mixamosta valittiin hahmolle seuraavat animaatiot:

- Idle (paikallaanolo)
- Run (juoksu)
- Sword Attack 1–3 (miekkahyökkäykset)
- Death (kuolema).

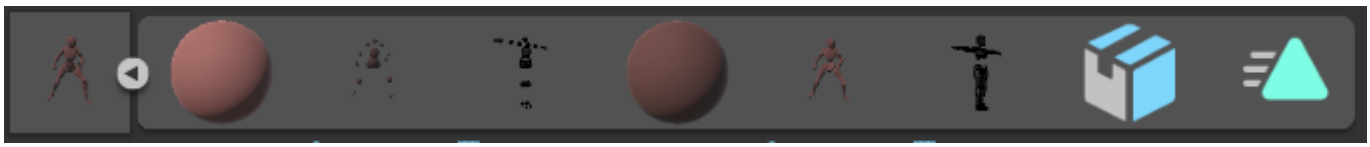
Mixamon etuna on joustavuus: se mahdollistaa sekä valmiiden hahmojen että omien mallien käytön. Latausvaiheessa oli tärkeää valita asetus With Skin, jotta hahmot tulevat animaatioiden mukana. Kuvan 2 mukaiset latausasetukset on valittu tähän työhön.



KUVA 2. Mixamo animaation latausasetukset (Keski-Korpi Näyttökuva 2025)

6.2 Animaatioiden siirtäminen Unityyn

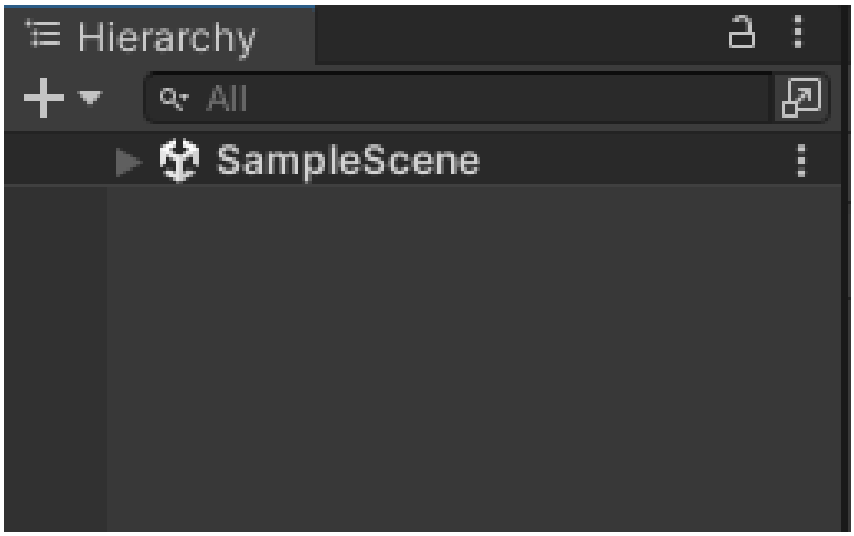
Kun hahmo ja siihen liittyvät animaatiot oli ladattu, tiedostot tuotiin Unityyn raahaamalla ne suoraan projektikansioon. Unity tunnistaa FBX- ja muut yleiset 3D-formaatit ja sijoittaa ne selkeästi projektinäkymään, josta niitä voi helposti käsitellä. Kuvan 3 mukaiset sisällöt tulisi saada, jotta hahmolla on kaikki tarvittavat sisällöt projektia varten.



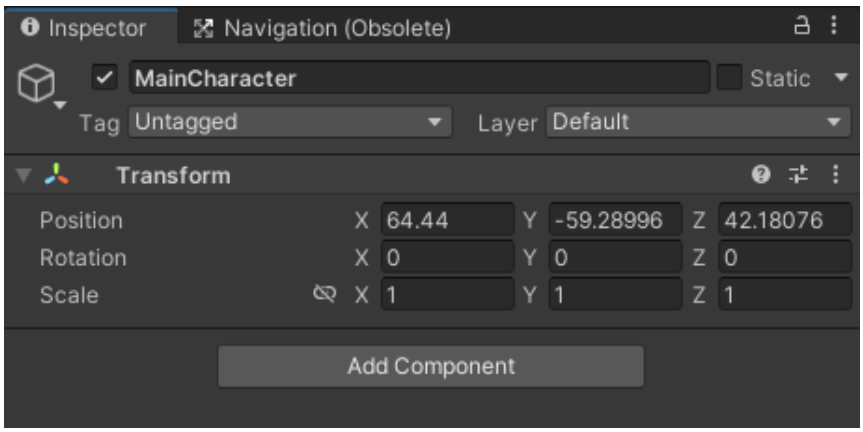
KUVA 3. Hahmon visuaaliset osat ja animaatiot (Keski-Korpi Näyttökuva 2025)

Seuraavaksi hahmo siirrettiin Scene-näkymään, missä sen sijaintia ja ominaisuuksia voidaan tarkastella. Hahmon lisäämisen jälkeen Unityn Hierarchy-paneeliin (Kuva 4) ilmestyy uusi objekti, joka edustaa hahmoa. Tämän objektin valitsemalla avautuvat oikeassa reunassa Inspector-paneelin (Kuva 5) kautta sen komponentit, kuten Transform ja mahdollisesti valmiiksi liitetty Animator-komponentti.

Mikäli Animator-komponentti puuttui, se lisättiin käsin Add Component -painikkeella. Tämä komponentti on tärkeä, sillä sen avulla yhdistetään hahmoon kaikki ohjauslogiikka ja siirtymät eri animaatiotilojen välillä.



KUVA 4. Unity editorin Hierarchyosio, johon hahmon siirtämällä saa sen näkymään pelissä (Keski-Korpi Näyttökuva 2025)



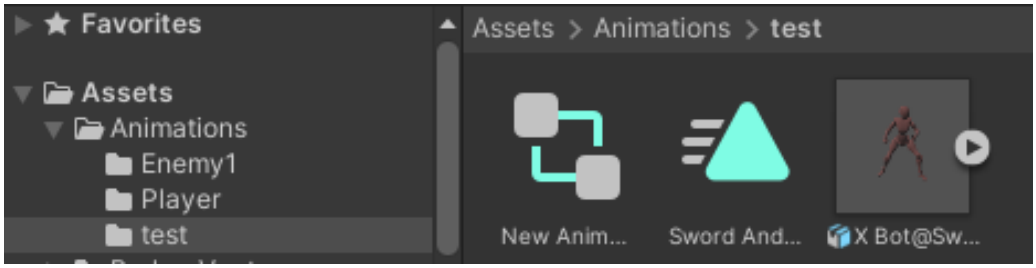
KUVA 5. Unity editorin Inspector-näkymä, jossa näkyy valitun objektin sijainti ja voi lisätä uusia komponentteja (Keski-Korpi Näyttökuva 2025)

6.3 Animator Controller-testaus

Seuraavaksi luotiin uusi Animator Controller -tiedosto projektikansioon, joka on kuvassa 6. Tämä tiedosto toimi hallintalaitteena animaatioille ja liitettiin hahmon Animator-komponenttiin. Animator-komponentin löytää valitsemalla hahmon scene-osiosta. Uuden komponentin voi ladata inspector-ikkunasta. Animator-komponentin lisättyä ja avattua inspector-ikkunassa löytyy kohta, johon voi raahata haluamansa animator controllerin.

Controlleriin lisättiin ensimmäisessä testissä yksi animaatio, jonka automaattinen yhdistyminen Entry-tilaan osoitti, että integraatio oli onnistunut. Animator Controllerin liittäminen hahmoon tapahtuu yksinkertaisesti vetämällä controller-objekti Animator-komponentin kenttään Inspector-näkymässä. Tämä toimii drag-and-drop-tyylisesti.

Controllerissa näkyvät kaikki määritetyt animaatiotilat sekä niiden väliset siirtymät. Ensimmäinen testaus tehtiin lisäämällä Idle-animaatio ja tarkastelemalla sen aktivoitumista Play-tilassa. Näin varmistettiin, että Unity tunnistaa hahmon ja osaa toistaa siihen liitetyn liikkeen.



KUVA 6. Animator controller ikoni (kolme neliötä) sekä animaation ikoni (kolmio) (Keski-Korpi Näyttökuva 2025)

6.4 Koonti

Hahmon animaatioiden hallinta pelikehityksessä ei perustu vain yksittäisten animaatioiden käyttöön, vaan keskeistä on ymmärtää animaatioiden välinen vuorovaikutus ja niiden siirtymien taustalla olevat syyt. Animaatioiden liittämisessä toisiinsa tarvitaan tarkkaa suunnittelua, jotta pelin hahmon liike olisi mahdollisimman sujuvaa ja loogista. Tämä osio käsittelee animaatioiden hallintaa yksinkertaisessa roolipelihahmon liikesarjassa sekä sitä, miten siirtymät ja blend tree on toteutettu työssä.

Työssä toteutettiin animaatiot seuraavalla tavalla (Kuva 1), mikä kuvastaa niiden välistä hierarkiaa ja siirtymiä:

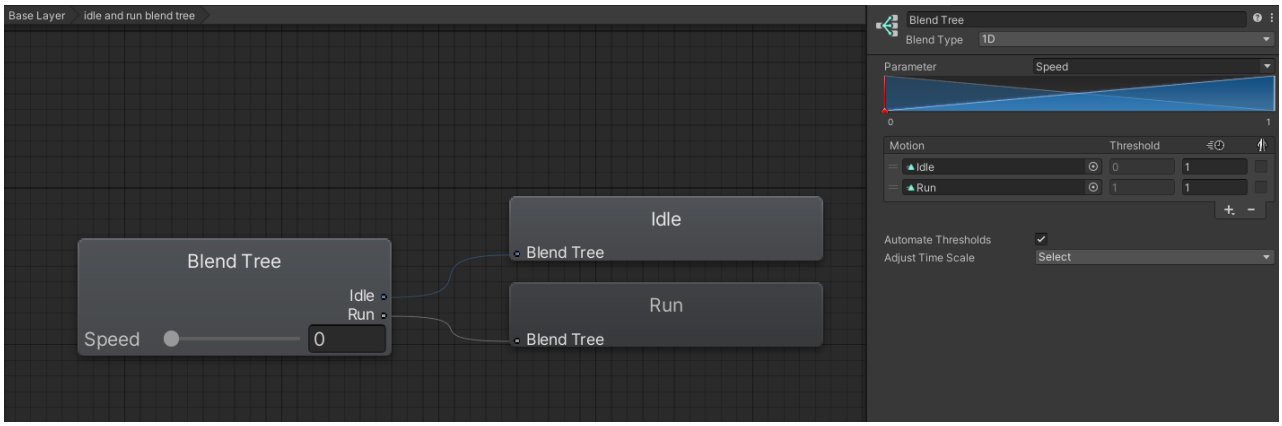
- Entry → IdleAndRun blend tree
- IdleAndRun Blend Tree → AbilityQ, AbilityW, AbilityE ja Death
- AbilityQ → Blend tree ja Death
- AbilityW → Blend tree ja Death
- AbilityE → Blend tree ja Death.

Tämä rakenne tarjoaa yksinkertaisen mutta toimivan liikesarjan roolipelihahmolle, jossa animaatiojärjestelmässä hahmo voi siirtyä perustilasta (esim. paikallaanolo tai juokseminen) erikoistoimintoihin (kyvyt) tai esimerkiksi kuolemaan. Keskeistä oli huomioida, että siirtymät eli transitions eivät muodostu automaattisesti Unityssä, vaan ne täytyy määritellä manuaalisesti projektin tavoitteiden mukaisesti.

6.4.1 Animaatioiden sulauttaminen

Blend tree (Kuva 7) on keskeinen työkalu, joka mahdollistaa hahmon liikkeen sujuvan siirtymisen äärimmäisen erilaisten animaatioiden välillä esimerkiksi paikallaanolosta juoksuun hahmon nopeuden perusteella. Tässä työssä toteutettu blend tree yhdisti hahmon idle-tilan ja run-tilan seuraavalla logiikalla:
Määritettiin blend treen Unityn Animator-näkymässä, jossa luotiin yhteys kahden animaation välille. Blend Treen sisällä pitää valita Add Motion, sillä täällä ei voi suoraan tuoda animaatioita.

Hahmon nopeus toimi parametrina, jonka avulla animaatio siirtyi dynaamisesti paikallaanolosta (0-nopeus) juoksemiseen (1-maksiminopeus). Blend tree päivitti animaatiota reaaliajassa hahmon liikkumisnopeuden muuttuessa, mikä varmisti hahmon liikkeen realistisuuden ja reagoivuuden pelaajan toimintaan.



KUVA 7. Blend Tree (Keski-Korpi Näyttökuva 2025)

Tämä yksinkertainen mutta tehokas ratkaisu vähensi tarvetta monimutkaisille koodiratkaisuille ja varmisti sujuvan pelikokemuksen. Jotta hahmo liikkuisi pelitilassa, se vaatisi tämän projektin ulkopuolista ohjelmointia. Tässä vaiheessa voi testata animaatio vaihtoa liikuttamalla nopeus-säädintä ja nähdä muutos animaatiossa. Nopeus-säädintä on esitetty kuvassa 7.

6.4.2 Transition

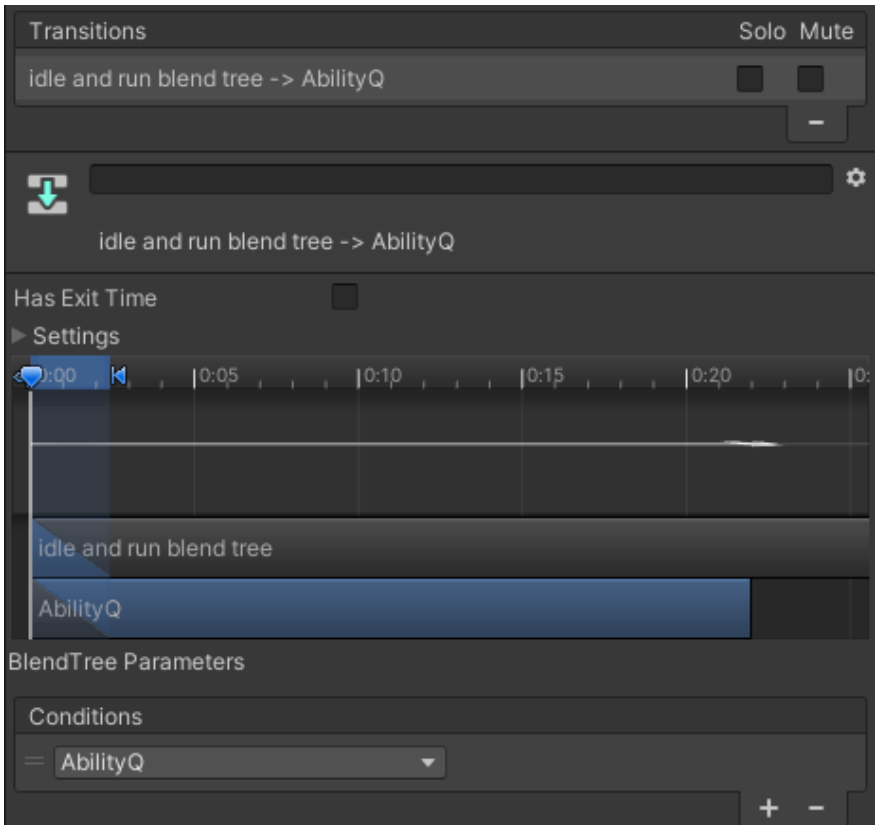
Siirtymien luominen on kriittinen osa hahmoanimaatioiden hallintaa. Siirtymät määrittävät, milloin ja millä ehdoilla hahmo vaihtaa animaatiotilaa toiseen, kuten hyökkäyksestä takaisin perusliikkeeseen. Siirtymien asettamisessa on tärkeää huomioida oikeat aikaviiveet ja ehdot, jotta liikesarjat eivät tunnu pelaajalle kömpelöiltä tai epäsynkronisilta.

Työssä siirtymät toteutettiin seuraavilla askelilla:

- Ensimmäinen vaihe
  Unityn Animator-näkymässä siirtymä luotiin valitsemalla kaksi tilaa ja yhdistämällä ne siirtymäviivalla. Kuva 8 näyttää näkymän, joka avautuu, kun valitsee siirtymäviivan editorissa.

- Toinen vaihe
  Siirtymän yhteydessä määriteltiin ehto, kuten napin painallus (Trigger), joka laukaisee siirtymän. Tässäkin kohtaa voi testata toimivuutta asettamalla kyseisen triggerin päälle ja pois nähdäkseen kuinka hahmon animaatiot vaihtuvat.

- Kolmas vaihe
  Kuvassa 8 näkyy näkymä siirtymä BlendTree → AbilityQ. Kohta Has Exit Time valitsin on pois päältä tässä työssä, sillä tavoite on saada animaatio vaihtumaan heti eikä vasta kun edellinen on loppunut. Animaation palautuessa tavoite on, että lyöntianimaationi loppuu kokonaan ennen kuin se menee takaisin BlendTreehen. Kuvassa näkyy myös aikaikkuna missä ovat molemmat animaatiot päällekkäin. Sininen kohta kertoo kauan siirtymä kestää siihen asti, että tuleva animatio pyörii ”puhtaana” ilman sekoitettua animaatiota. Tämä on hieman samantapainen kuin blend tree, mutta tässä tapauksessa siirtymä vain tapahtuu automaattisena samanlailla aina lineaarisesti.



KUVA 8. Animaatioiden siirtymisen asetukset ja ehdot (Keski-Korpi Näyttökuva 2025)

7 POHDINTA

Tämän opinnäytetyön tavoitteena oli perehtyä Unityn Animator-työkalun käyttöön pelihahmojen animaatioiden hallinnassa sekä toteuttaa käytännön projekti hyödyntäen Mixamo-palvelun valmiita animaatioita. Työn aikana syvennyttiin sekä teoreettiseen että käytännölliseen ymmärrykseen siitä, miten hahmojen liikesarjat voidaan rakentaa pelimoottorissa.

Projektin aikana korostui erityisesti animaatiopuun ja siirtymien tarkka suunnittelu. Pienilläkin asetuksilla, kuten parametrien hallinnalla ja siirtymäehdoilla, oli suuri vaikutus hahmon liikkeiden sujuvuuteen ja pelattavuuden laatuun.

Mixamon käyttö nopeutti animaatioiden hankintaa ja integrointia merkittävästi, mutta toi esiin myös joitain rajoitteita, kuten automaattisen riggauksen aiheuttamat mahdolliset haasteet. Automaattisen riggauksen ongelmakohtina oli, että hahmon animaatiot saattoivat vääristyä. Tämä vaati huolellista nivelkohtien asettelua Mixamossa.

Työn haasteellisimpia osa-alueita oli hahmon animaatiopuun rakenteen hallinta siten, että siirtymät tapahtuivat luontevasti. Visuaalisen Animator-työkalun käyttö oli kuitenkin intuitiivista, ja sen hallinta parani projektin edetessä.

Kokonaisuutena projekti tarjosi arvokasta kokemusta animaatioiden hallinnasta pelikehitysympäristössä sekä käytännönläheisen näkökulman pelihahmojen toiminnallisuuden rakentamiseen. Tulevaisuudessa työn pohjalta voisi kehittää laajempia ja monimutkaisempia animaatiorakenteita sekä tutkia syvemmin animaatioiden yhdistämistä koodin avulla pelilogiikkaan.

LÄHTEET

Mixamo. Saatavissa: https://www.mixamo.com. Viitattu 18.4.2025.

Godot Engine. Godot Engine - Official Site. Saatavissa: https://godotengine.org. Viitattu 18.4.2025.

Lappalainen, N. 2024. Markkinoiden johtavat pelimoottorit. Jyväskylän yliopisto. Saatavissa: https://jyx.jyu.fi/jyx/Record/jyx_123456789_95632?sequence=1&isAllowed=y. Viitattu 18.4.2025.

RocketBrush Studio. 2024. Godot Engine vs Unity: Which one suits you best in 2025. Saatavissa: https://rocketbrush.com/blog/godot-vs-unity. Viitattu 18.4.2025.

Tristem, B. Udemy Blog. 2022. Unity vs Unreal: Which Game Engine is Best for You? Saatavissa: https://blog.udemy.com/unity-vs-unreal-which-game-engine-is-best-for-you. Viitattu 18.4.2025.

iHeartGameDev. YouTube. 2021. Unity's Animation System. Soittolista. Saatavissa: https://www.youtube.com/watch?v=-FhvQDqmgmU&list=PLwyUzJb_FNeTQwyGujWRLqnfKpV-cj-eO. Viitattu 18.4.2025.