OULUN AMMATTIKORKEAKOULU

HAHMOTYYPIN SUUNNITTELU JA TOTEUTUS RPG-VIDEOPELIIN

Ranger/Archer (jousiampuja)

Johannes Huttunen
Opinnäytetyö AMK
Kevät 2025
Tieto- ja viestintätekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Opinnäytetyö AMK
Tieto- ja viestintätekniikan tutkinto-ohjelma

Tekijä(t): Johannes Huttunen
Opinnäytetyön otsikko: Hahmotyypin suunnittelu ja toteutus RPG-videopeliin
Työn valmistumislukukausi ja -vuosi: kevät 2025
Sivumäärä:

Opinnäytetyön aiheena oli luoda hahmotyyppi videopeliin. Työn tavoitteena oli soveltaa yrityslähtöisistä projekteista ja harjoittelusta saatua kokemusta ja kohentaa sitä tekemällä konkreettinen osa peliä. Osana tavoitteena oli myös antaa tilaajalle hyvä pohja, mitä voi käyttää tulevaisuudessa hahmoluokan laajentamiseen.

Työssä käytettiin Unity-pelimoottoria ja 3D-mallien työstöön Blenderiä. Blenderissä suurin osa työstä liittyi jousiaseen 3D-mallin käyttöönottoon. Tarkoitus oli tehdä jouselle luusto, jonka avulla siitä olisi helpompi saada näyttävämpi. Unityssä hoitui esimerkiksi hahmon ja jousiaseen animaatioiden synkronointi ja jousiaseella suoritettavan kyvyn teko.

Työn tuloksena oli hahmoluokan pohja, jolle tehtiin yksi kyky ja on testipelattavassa tilassa. Jatkokehitykselle on paljon mahdollisuuksia. Tähän voisi lisätä lisää jousiaseita, kykyjä ja keskittymispuu. Työ antoi kuitenkin selvityksen, miten asiat voidaan tehdä.

ABSTRACT

Oulu University of Applied Sciences
Information Technology

Author(s): Johannes Huttunen
Title of thesis: Hahmotyypin suunnittelu ja toteutus RPG-videopeliin
Term and year when the thesis was submitted: spring 2025
Number of pages:

This thesis is about producing and planning ranger character class to a RPG-videogame. Objective to me is to increase my knowledge and trying to produce something that can be used as a base to expand.

Big part of the motivation and inspiration to select this kind of project to make my thesis about is my hobby, gaming. Add to that is desire to know more about making games and knowing what goes on below the surface and what the game is processing. I have done game development for around one year now which includes two company driven projects and twenty credits of internship as well.

Game development is a large field, and I am glad to have class design as the subject. It has many parts to it, featuring 3d modelling, coding and planning being the most important ones.

Finished work is a good base for a character class which can be played in a test environment. The class has one usable skill and weapon. In future development there is a lot to do, including more weapons, skills and the focus tree. The work gave a way to do things like modelling the weapons or animating them.

SISÄLLYS

SANASTO

Asset	Elementti, joka on luotu uudelleenkäytettäväksi. Pelimaisemaan tuotuja asseteja sanotaan gameobjecteiksi (peliobjekti). Asset voi olla puu, hahmo tai vaikka äänipätkä.
Blender	3d-mallinnusohjelma
Build	Pelihahmon tietty tyyli rakentaa, joka esittää hahmon käyttämiä aseita tai attribuuttien valinnoista johtuvaa erittelyä. Build voi myös tarkoittaa pelikehityksessä koontiversiota, mikä on pelattava kokonaisuus projektista.
C#	Unity:n pääasiallinen ohjelmointikieli. C# on Microsoftin kehittämä ja ylläpitämä. Käyttää .NET-ohjelmistokehystä. Olio-ohjelmointikieli.
Frame	Tietokonesovelluksissa käytetty termi (suomeksi ”yksi päivitys”). Yleensä käytetään FPS:n (frames per second, kuvaa sekunnissa) yhteydessä, jossa frame on yksi kuva. Monta kuvaa peräkkäin muodostaa liikkuvan kuvan. Pelikehityksessä frame tarkoittaa yhtä päivityshetkeä koodin suorituksessa.
Git	Versionhallintajärjestelmä, mikä mahdollistaa versioiden muuttamista eri oksissa ja niiden yhdistämisen, joka johtaa helpompaan kehittämiseen.
GitHub	Versionhallinta työkalu joka on käyttää Git-järjestelmää
IK	Inverse Kinematics (käänteinen kinematiikka) käytännössä tarkoittaa tapaa ohjata esimerkiksi hahmon kättä, kädessä olevan objektin mukaan. Päinvastoin

	kuin Forward Kinematics:ssa, jossa objekti seuraa käden liikettä.
RPG	Lyhennys roolipelistä (role playing game)
Ranger	Yleinen hahmoluokka roolipeleissä, jousiampuja
Unity	Pelimoottori. Moottorit pelimaailmassa mahdollistavat kaikki pelin toiminnallisuudet. Muun muassa äänen toiston, 3d-grafiikan renderöinnin (esittämisen ruudulla) ja esineiden fysiikkalaskut.
Unity Asset Store	Unityn ylläpitämä nettikauppa, josta saa ostettua tai myytyä assetteja.

1 JOHDANTO

Tämän työn aiheena on luoda Borealuksen kehittämään roolivideopeliin jousiampuja hahmoluokka. Hahmoluokka on valinta, jonka pelaaja tekee päättääkseen minkälainen hänen hahmo on. Monessa pelissä hahmoluokka määrää pystyykö hahmo käyttämään tiettyä aselajia tai tietyn tyyppisiä kykyjä. Nämä ovat monesti myös liukuvia, jolloin hahmoluokalla ei välttämättä ole niin paljon väliä. Miltein aina kuitenkin päätös korostaa tiettyjä attribuutteja, minkä avulla voidaan hahmottaa hahmon vahvuudet. Jousiampuja-luokilla yleensä korostuu näppäryyteen(dexterity) liittyvät attribuutit.

Roolipelit ovat kategoriana hahmon kehittymiseen ja hahmoon eläytymiseen painottuva genre, jotka yleensä sijoittuvat fantasia- tai scifimaailmaan. Tässä tilanteessa kyse on maailmasta, joka sijoittuu fantasia-teemaan. Kyseessä on siis taikaa, miekkoja ja isoja otuksia vihollisena. Roolipelien sisällä on myös liuta eri genrejä (Wieland, 24.3.2024).

Peli on tehty Unity-pelimoottorilla ja se on tarkoitettu julkaistavaksi PC-alustalle Steam-pelikauppa-alustalla. Unity on yksi käytetyimmistä pelimoottoreista tänä päivänä muun muassa Unreal Enginen kanssa. Steam puolestaan on tietokonepelien johtava kauppapaikka.

Työn motivaationa toimi omat harrastukset videopelien äärellä ja halu oppia enemmän siitä, miten pelit toimivat konepellin alla. Tähän perustaa hain vapaasti valittavista opinnoista, missä kävin C#-ohjelmoinnin perusteet-kurssin. C#-kieltä käytetään Unity-pelimoottorin pääohjelmointikielenä. Tämän lisäksi kaksi yrityslähtöistä projektia ja harjoittelun kolmestakymmenestä opintopisteesta kaksikymmentä on myös suoritettu pelisuunnitteluun liittyvässä työssä. Pelisuunnittelussa osana on ohjelmoinnin lisäksi muun muassa suunnittelua, 3d-mallinnusta. Näistä aiheista tulee olemaan tarkemmat osiot.

Tämä opinnäytetyö seuraa prosessia, missä käydään läpi eri asioita jotka tulevat esille hahmoluokkaa tehdessä ja mitä pelikehittämiseen tulee. Peliprojektissa ei ole vielä erikseen jousiaseelle hyökkäysliikkeitä, mutta 3d-malleja ja animaatioita löytyy Unityn Asset Storesta saatuna. Valmiita malleja ja animaatioita voidaan käyttää projektissa, mikä helpottaa työnsarkaa. Näitä voi kuitenkin tehdä uusia käyttäen 3d-mallinnusohjelmaa, kuten Blenderiä. Blenderiä tässä työssä on

käytetty jousiase-assetin luuston tekoon, jonka avulla jousiase liikkuisi luonnollisesti. Sen lisäksi Blenderissä on animoitu jousenaseen liikettä. Aseen ja hahmon animaatioiden yhdistäminen on myös isona osana, sillä jousiaseen animointi hahmon kanssa on hieman monimutkaisempaa, kuin miekan tai taikasauvan kanssa.

Opinnäytetyön toivotaan antavan pohjan jousiasetta käyttävän hahmoluokan kehittämiseen. Tällä mahdollistetaan prosessin helpottamista ja tiettyjen asioiden 3D-mallintamisen ja animoinnin yksinkertaistamista. Lopputulos olisi testikäyttöön valmis tuote, jonka avulla voidaan testata jousiaseiden käytön dynaamiikkaa ja pelituntumaa. Hahmoluokalle olisi hyvä luoda vähintään yksi peruskyky, joka on nimensä mukaan tavallinen jousiaseella ampuminen ilman lisähehkua. Näitä kuitenkin tulee olemaan aika yksinkertainen luoda, ainakin liikkeen runko. Mikä osaltaan mahdollistaa useamman liikkeen luomisen.

Nuolta ja jousta käyttävät hahmot yleensä menevät nimillä ranger tai archer. Kummatkin näistä ovat yleensä jousiasetta käyttäviä, ranger on ehkä laajempi käsitys. Tässä tapauksessa pohjaksi kehitetään jousiasetta käyttäviä liikkeitä. Yleisessa mediassa jousiasetta käyttäviä hahmoja näkee melkein joka sarjassa, elokuvassa, teksteissä ja peleissä. Näistä varmaan tunnetuin on Legolas Taru Sormusten Herrasta-sarjasta. Peleissä jousiaseita nähdään esimerkiksi God of War-sarjassa ja Elder Scrolls V Skyrimissä monen luotto ”build” (hahmon rakenne) on häivejousiampuja.

Kirjoitelmaan sisältyy paljon sanoja ja pääosin 3D- ja pelialan työasioissa käytetään englannin kielisiä termejä. Opinnäytetyön tekstissä käydään läpi osa termeistä englanniksi ja suomennetaan siihen viereen.

2 TYÖKALUJA PELIKEHITYKSEN ÄÄRESSÄ

Pelikehityksessä käytetään lukemattomia työkaluja, sovelluksia, järjestelmiä ja järjestelmien sisällä kaikkia aikaisempia. Tämän työn aikana käytin itse viittä sovellusta. Nämä voidaan jakaa kolmeen kategoriaan, Itse pelimoottori/kehitysalusta, 3D-mallinnusohjelma, versionhallinta- ja viestintäohjelmat. Versionhallinta- ja viestintäohjelmat voisivat olla erillään, mutta tässä tapauksessa ne toimivat limittäin.

2.1 ”Fyysiset työkalut”

Pelit vaativat resursseja niiden toimimiseen ja pelikehityksessä on iso merkitys, minkälaisen balanssin haluaa suorityskykyä, näyttävää grafiikkaa tai vaikkapa suurta määrää simuloitavaa logiikkaa. Tämä myös rajoittaa pelaajia, joiden tietokone voi olla vanhempi tai huonompi tehoinen. Yleensä nettipeleissä myös pelaajamäärä vaatii paljon resursseja tietokoneelta pelistä riippuen. Tätä opinnäytetyötä työstin pöytätietokoneella, jossa on uudehko 10-ytiminen suoritin, 64 gigatavua keskusmuistia ja pari vuotta vanha näytönohjain paremmasta päästä.

Kuten pelin valmista versiota pelatessa, myös peliä tehdessä tarvitaan tietokone, mikä pystyy suorittamaan tekemääsi peliä kunnolla. Esimerkiksi myös koulua varten ostetulla kannettavalla tietokoneella voi myös luoda pelejä, kunhan pysyy simppeleissä grafiikoissa ja pelin ei tarvitse tehdä hirveää määrää laskutoimituksia. Unityn järjestelmävaatimukset ovat sangen pienet. Vaatimuksena toimii käytännössä melkein kaikki toimivat tietokoneet, jossa on moderni käyttöjärjestelmä (Unity, 2025).

Tietokoneen ja sen yleisien oheislaitteiden lisäksi, itsellä oli käytössä myös piirtopöytä Blender käytössä. Se antaa hieman käyttö mukavuutta joihinkin tehtäviin.

2.2 Pelimoottorit

Pelikehityksen päähuomio on pelimoottorissa. Monet isot pelit tehdään pelistudion omalla moottorilla, joka on kustomoitu heidän tarpeisiin. Toinen mahdollisuus on käyttää julkisia moottoreita, kuten Unreal Engineä tai Unityä. Moottoreita on paljon, useita kymmeniä ellei satoja. Moottoreissa on paljon eroa, joista isoimmat erot rajaa, mille järjestelmällä kehityksessä olevaa peliä tulisi pelata. Se on hyvä paikka aloittaa. Unity käyttää ohjelmointikielenä C#, joka on yleisesti helpompi oppia ja käyttää, verrattuna C++ jota Unreal Engine käyttää. Kolmas iso asia on, minkälaista peliä haluaa tehdä. Unreal Engine tarjoaa parempia mahdollisuuksia kolmiuloitteeseen grafiikkaan, kun taas Unity nojaa hieman enemmän kaksiulotteiseen.

Kuva 1, Unityllä luotu, suosittu 2D-peli Hollow Knight (https://www.hollowknight.com/ , 17.06.2025)

Kuvassa 2 on ruutukaappaus Unityllä tehdystä EFT-pelistä (Escape from Tarkov). Pelin 3D-mallit ovat tarkkoja ja pelinäkymää on vaikea erottaa oikean maailman valokuvasta. Kuvassa 3 on ruutukaappaus Unreal Engine-pelimoottorin sisällä renderöitystä pelinäkymästä, jonka valoitus, heijastukset ja sumu tuottavat realistisen näkymän, vaikkakin asetelma onkin futuristinen.

Kuva 2, Escape from Tarkov - Echoes of the Fallen City (Streets of Tarkov teaser #2)-videosta kohdasta 00:01:09)

Kuva 3, Unreal Enginen sivuilta kuvakaapattu esimerkki (https://www.unrealengine.com/en-US 17.06.2025)

Moottoreiden käyttö ei ole kuitenkaan aina ilmaista. On tärkeää tehdä tutkimusta aiheesta, onko moottori kokonaan ilmainen vai ei. Esimerkiksi Unreal Enginestä harrastaja ei joudu maksamaan, ennen kuin julkaisema peli on tuottanut elinaikanaan miljoonan Yhdysvaltain dollarissa (Unreal Engine 2025).

2.3 3D-mallinnusohjelmat

Videopelien pelinäkymä koostuu asseteista. Assetit ovat peliobjekteja, jotka voivat olla seiniä, tuoleja, lamppuja tai jopa musiikkipätkiä. Kolmiuloitteissa peleissä useimmat esineet ovat 3D-malleja. Assetit ovat myös uudelleenkäytettäviä.(Kalinin, 6.10.2023)

Näitä 3D-malleja voi löytää netistä, käytössä olevan pelimoottorin asset-kaupoista tai tehdä itse. Nämä kaikki vaihtoehdot ovat käytettäviä, yksi näistä enemmän työläs, toiset saattavat maksaa rahaa. 3D-mallien luonti on kuitenkin onneksi helppoa, kunhan pysyy homma käsissä, eikä kuroita liian pitkälle. Yksinkertaisen objektin luominen ei vaadi hirveästi harjoittelua ja internet on täynnä ohjevideoita ja kirjallisia tutoriaaleja. Melkein kaikki Blender-aloittelijat tekevät ensimmäiseksi harjoitukseksi donitsin. Tämä tulee Youtubettaja Blender Gurun tutoriaalivideosarjasta, missä luodaan 3D-malli donitsi ja siitä renderöity kuva (Blender Guru, 26.7.2025)

Kuva 4, Tutoriaalin mukaan tehdyt donitsit, oma kuva

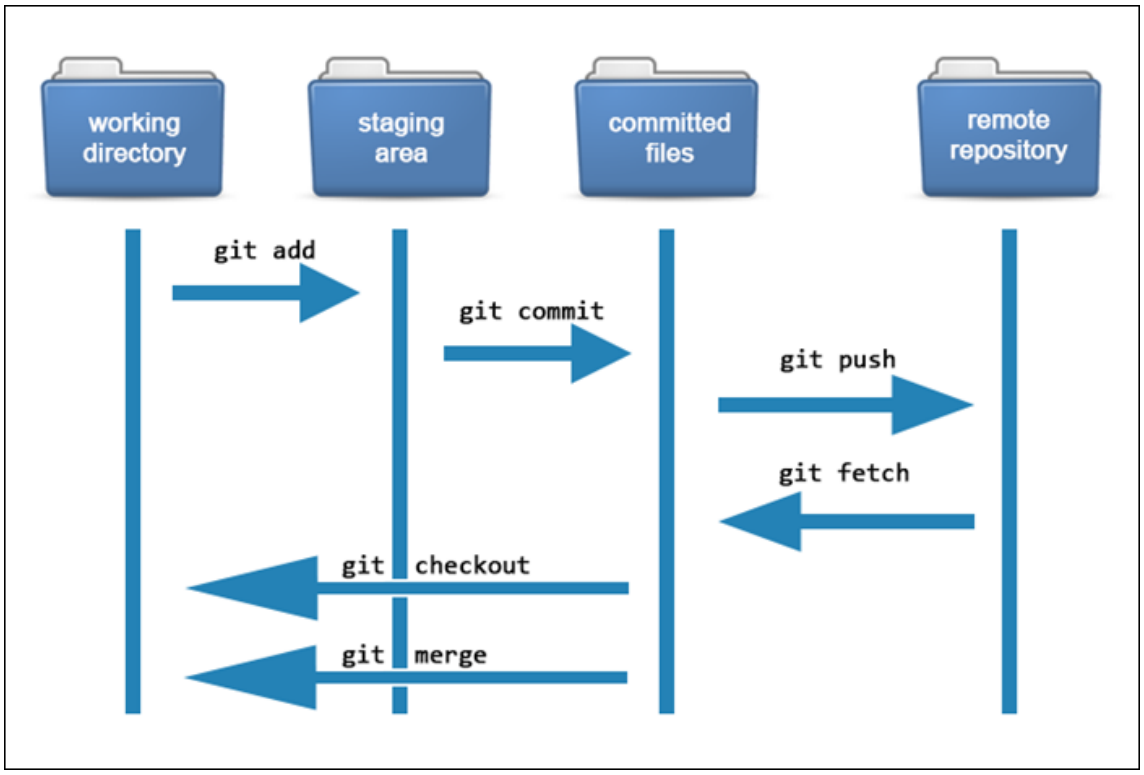
Suosittuja 3D-mallinnus ohjelmia on paljon ja niiden käyttötarkoitus vaihtelee arkkitehtuurista mekaniikkamallinnukseen. Yleisiä ohjelmia ovat esimerkiksi Blender, Autodesk Maya ja Fusion 360.

Opinnäytetyön aikana tein valmiille jousiaseelle luuston ja animoin jousiaseen jännittämisen Blenderissä. Blender oli ennalta tuttu, joten sen käyttö oli jokseenkin jouhevaa.

2.4 Versionhallinta- ja viestintäohjelmat

Versionhallinta- ja viestintäohjelmat ovat hyvin erinlaisia, mutta auttavat projektien kulkuun ainakin yhdellä samalla tavalla. Kummallakin voi viestiä ryhmälle edistymistä.

Versionhallintaohjelmista tunnetuin on GitHub, jota käytetään maailmanlaajuisesti. GitHub käyttää pohjana Git-järjestelmää, mikä on itse versionhallintatyökalu. Git toimii järjestämällä tiedostot eri ”oksiin” (branch), joita voidaan päivittää yksittäin ja yhdistää toiseen tarvittaessa. Tämä tapa työskennellä auttaa saman aikaisen työn eristämistä, mikä estää kollateraali vahingot jos jotain menee projektissa rikki. GitHub mahdollistaa myös kommunikaation, ainakin tikettityylisesti. Tikettityylillä tarkoitan erikseen ilmoitettuja esimerkiksi vika-ilmoituksia, missä on otsikko ja mitä on korjattavana viestinä alla.



Kuva 5, Perus Git työnkulku (Marijan, 2021)

Trello on myös ollut opinnäytetyön alussa käytössä. Sen avulla hahmotettiin aihepiiriä, mistä lähdettiin tekemään tätä. Trello on kanban-tyylinen ideapöytä, missä voi luoda tauluja joiden sisällä on sarakkeita ja kortteja. Se on tarkoitettu tehtävien ja ideoiden lokeroimiseen ja järjestämiseen.

Viestintäohjelmista opinnäytetyön aikana on käytetty Discordia. Se on lähinnä pelaajille suunnattu viestintäpalvelu, jossa voi viestitellä tai soittaa video- tai äänipuhelun. Discordia on käytetty opinnäytetyön aikana työnantoon ja palavereihin.

3 HAHMOLUOKAN KESKITTYMISPUU

Keskittymispuu tässä tapauksessa on valikko, mistä voi valita hahmoluokalle erityisiä kykyjä ja yleisiä attribuutteja. Peleissä yleensä on isompia ja pienempiä solmuja. Isommat sisältävät jotain erikoista, kuten kaikki vahinko mitä hahmo tekee vaihtuu eri elementiksi. Pienemmät taas ovat yleensä jotain, mitä otetaan välimatkalla isompaa solmua kohti. Ne pitävät sisällään esimerkiksi voima/näppätyys/puolustus-tyyppisiä attribuutteja.

Tässä vaiheessa pelin kehitystä Borealuksella ei ole vielä tarkkaa mallia minkälainen puu pitäisi olla, mikä hankaloittaa tätä vaihetta, mutta mielestäni tämä on kuitenkin olennainen osa hahmoluokkaa. Isoa listaa eri kykyjä ja attribuutteja ei ole kannattavaa lähtä miettimään kuitenkaan.

Jousiampujat ovat yleensä näppäröitä ja heikkoja peleissä. Heikolla tarkoitan huonoa puolustusta. Monessa pelissä myös jousiaseihin tarvii näppäryys-attribuuttia tarkkuuteen. Joten keskittymispuun pienemmät solmut olisivat suurimmaksi osaksi näppäryyteen liittyviä tai sitä parantavia. Tietenkin puun laajuudesta riippuen olisi hyvä olla myös erilaisille ”buildeille” (hahmo rakennus) omia niin sanottuja reittejä puussa esimerkiksi voimalle. Isot solmut puun voima-osiossa voisi olla vaikka ampumapituuteen ja seinien läpäisyyn liittyvää.

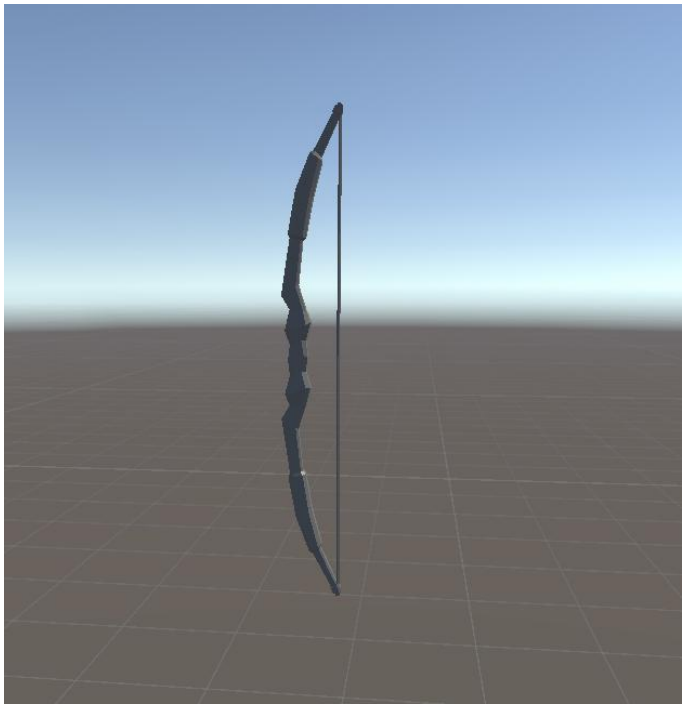
Isoja solmuja voisi olla liittyen myös lähitaisteluun myös. Kuten eri teräaseiden käyttöä avustavia. Isojen solmuen kehittämiseen voi käyttää paljon aikaa ja tehdä siitä erikoisen tai normaalin, vaihtaa jousesta varsijouseen ynnä muuta.

4 HAHMOLUOKAN TOTEUTUS

Toteutus-kappaleessa esitetään työn eri vaiheet ja mitä ongelmia ja suorituksia opinnäytetyön aikana tapahtui. Prosessiin liittyy monia vaiheita ja eri sovelluksien käyttöä, joten ne ovat merkattu selvästi kuvateksteillä ja tekstissä itsessään. Kappale on kirjoitettu kronologisessa järjestetyksessä.

4.1 Jousiase-mallin valmistelu

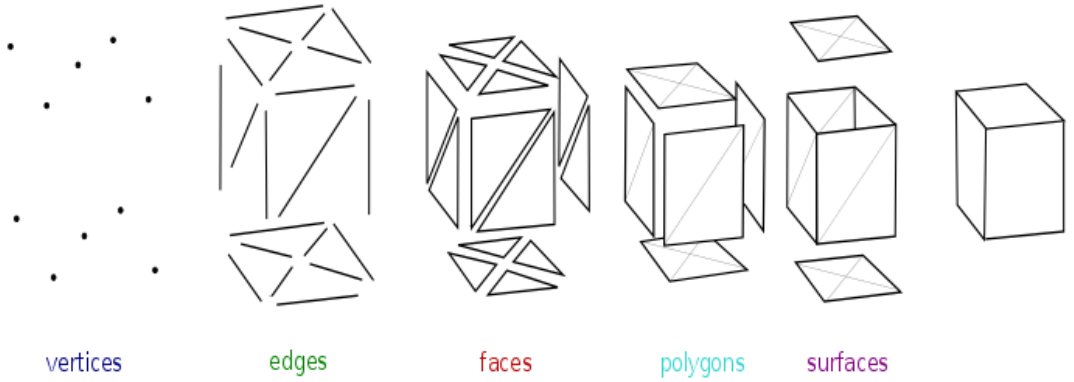
Opinnäytetyö lähti käyntiin katsomalla, voidaanko käyttää valmista 3D-mallia jousiaseesta. Asset-kansioissa oli pari valmista pakettia ja päätettiin aloittaa yksinkertaisella mallilla.



Kuva 6, Kuvakaappaus jousiaseesta Unityssä.

Tällä mallilla ei ole kuitenkaan monimutkaista ”luustoa” tai liikerajoituksia, mikä aiheuttaa deformointia jousta jännittäessä. 3D-malli koostuu monesti eri osasta, joihin kuuluu esimerkiksi mesh (verkkorakenne), rig (luusto) ja weight map (painokartta). Mesh eli verkkorakenne on 3D-mallin runko, joka koostuu

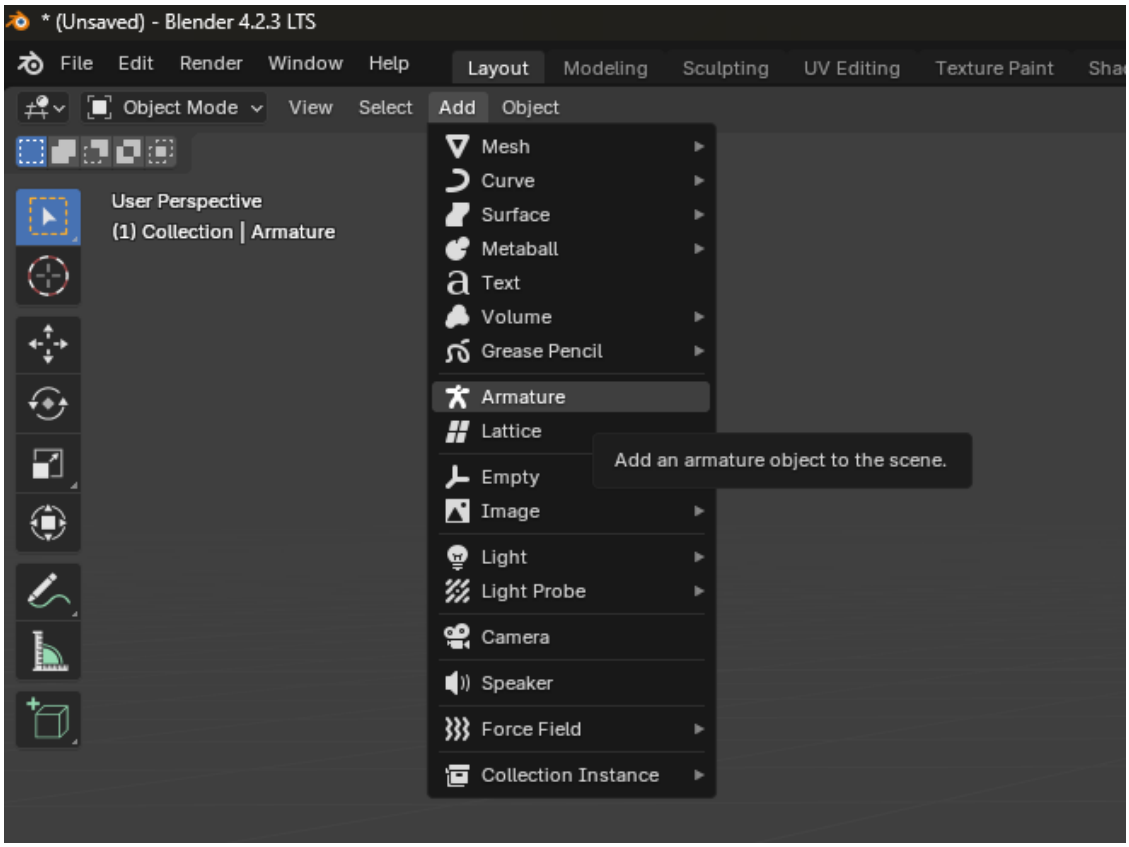
kärkipisteistä (vertex), särmistä (edge), tasoista (face), monikulmioista (polygon) ja pinnoista (surface).



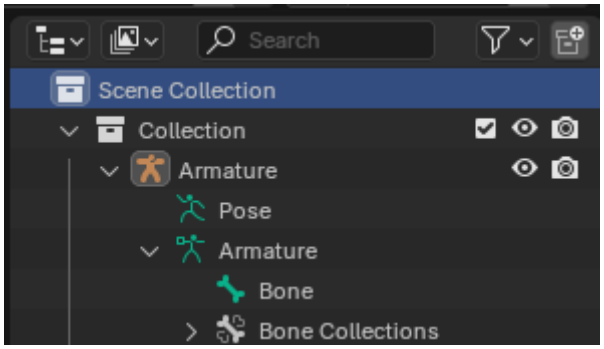
Kuva 7, 3D-mallin koostumus eri osista (https://en.wikipedia.org/wiki/Polygon_mesh)

3D-mallin luusto mahdollistaa verkkorakenteen liikkeen ja animoinnin. Luuston voi luoda monella eri sovelluksella, kun Blenderiä on käytetty, käydään läpi sitä prosessia.

Kuva 9:stä nähdään, miten ensimmäinen askel luuston tekoon sujuu. Painamalla ”add” näppäintä ja sieltä valitsemalla ”armature” luo ”bone” (luu) komponentin Blenderin objektilistaan Collections-ikkunassa (Blender, 2025). Tämä luu on niin sanottu ”root” (juuri), joka alle muut luut asetetaan. Tämä on tärkeää, koska Unityssä juuriluuta liikuttamalla koko objektin liikkuu mukana.



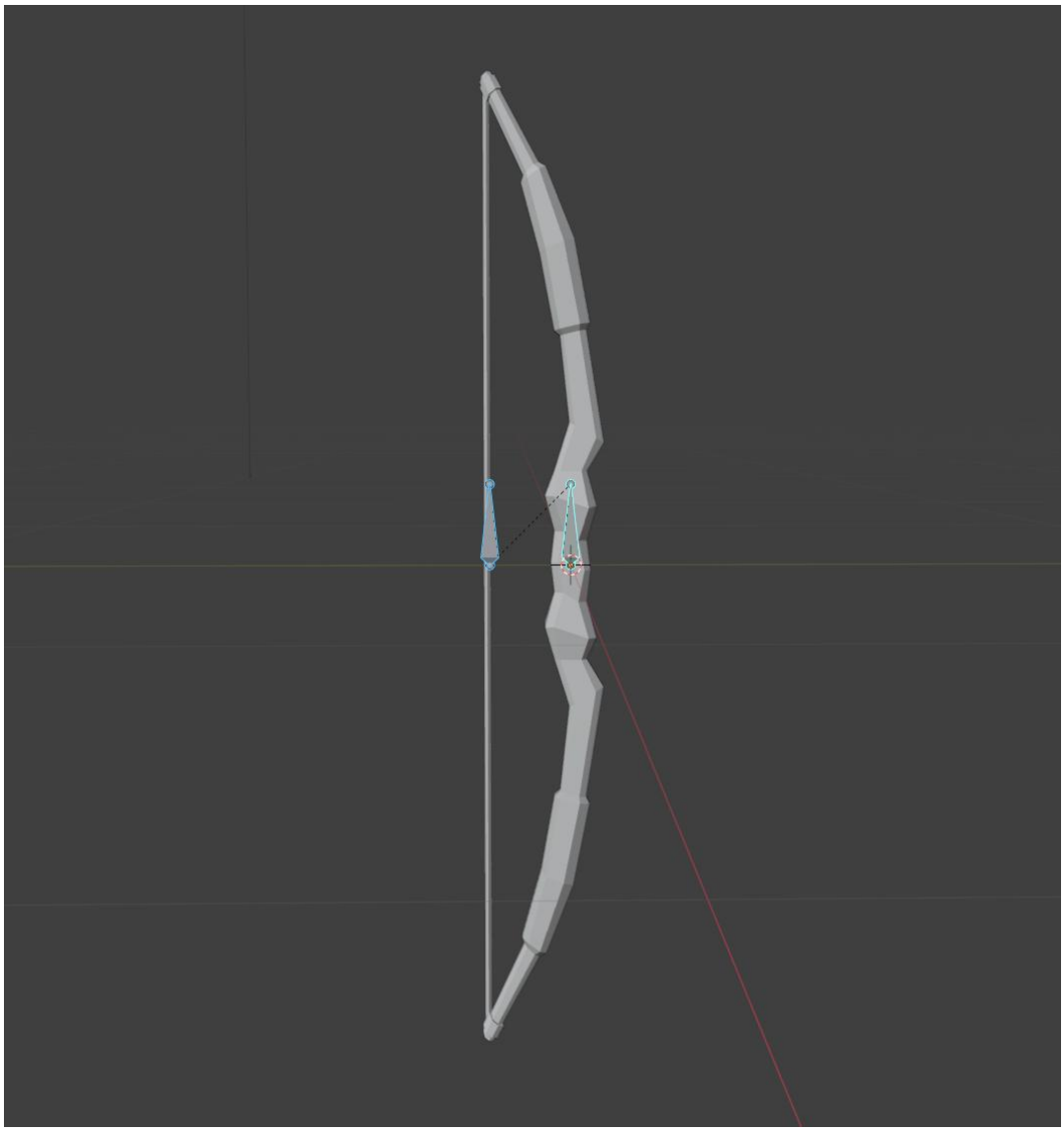
Kuva 8, Blenderissä luuston luonti on helppoa, kuvakaappaus Blenderistä



Kuva 9, Armature komponentin alla on tytär-komponentteja: Pose ja Armature, kuvakaappaus Blenderistä

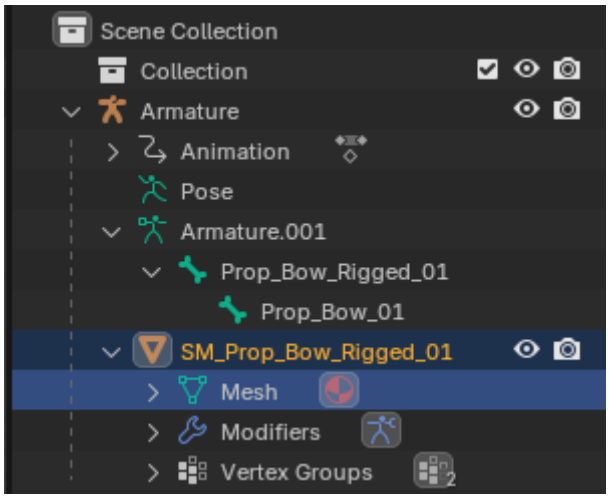
Tämän luodun luun voi kopioida tai extrude-toimintoa (ulos työntö) käyttäen jatkaa toisella luulla. Jos juuriluun kopioi, kopioitu luu menee hierarkiassa saman armaturen (rungon) alle, mutta ei ole silti sen tytär. Kopioidun luun voi silti asettaa parentiksi (vanhemmaksi) valitsemalla kummatkin (vanhemmaksi laitettavan jälkeisempänä) ja valitsemalla oikean painikkeen menusta ”parent” tai painamalla ctrl+p. Nämä aukaisevat uuden menun, missä voi valita asetetaanko uusi tytär luu kiinni vanhempaan vai pitää offset (väli).

Jousiaseen yksinkertaista luustoa tehdessä, haluamme että juuriluu ja tytär on erillään, joten käytämme väli-optiota.

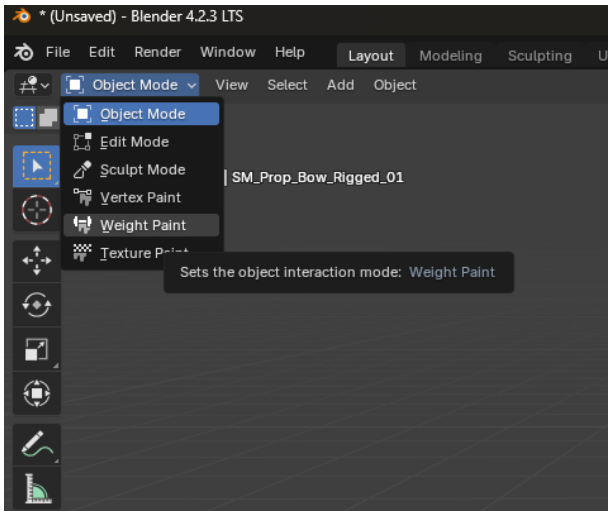


Kuva 10, Sinisellä näkyvät luut, vaaleampi on juuriluu ja musta katkoviiva niiden välissä esittää vanhempi-tytär-suhdetta, kuvakaappaus Blenderistä

Kun luusto on valmis ja verkkorakenne on myös hierarkiassa pää luusto-komponentin alla, voi alkaa miettimään painokarttaa. Painokartta määrää, kuinka paljon tietyllä luulla on vaikutusta verkkorakenteeseen. Tämän onneksi Blender näyttää selvästi. Ensin pitää valita Collections-valikosta verkkorakenne. Sen jälkeen voi vasemmalta ylhäältä valita tila-valikosta Weight Paint-option.

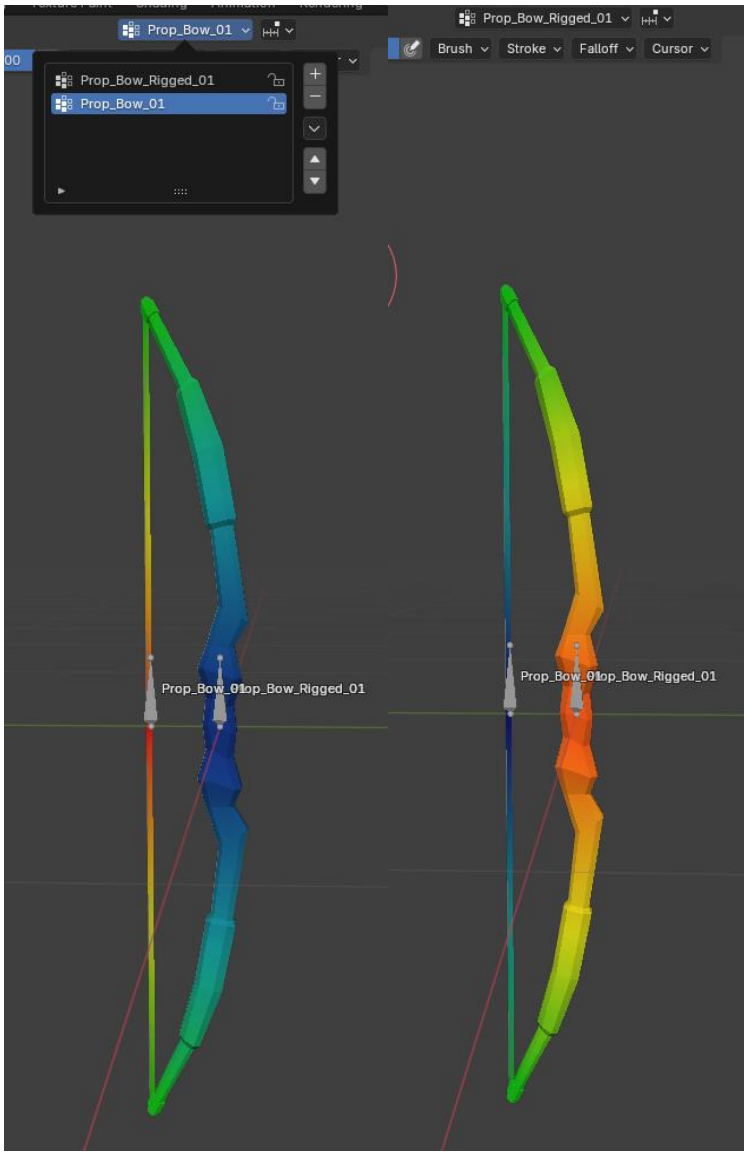


Kuva 11, Mesh on tässä tapauksessa verkkorakenne-objektin nimi, kuvakaappaus Blenderistä



Kuva 12, Tästä näkee verkkorakenteen valitsemisen jälkeisen tila-option, kuvakaappaus Blenderistä

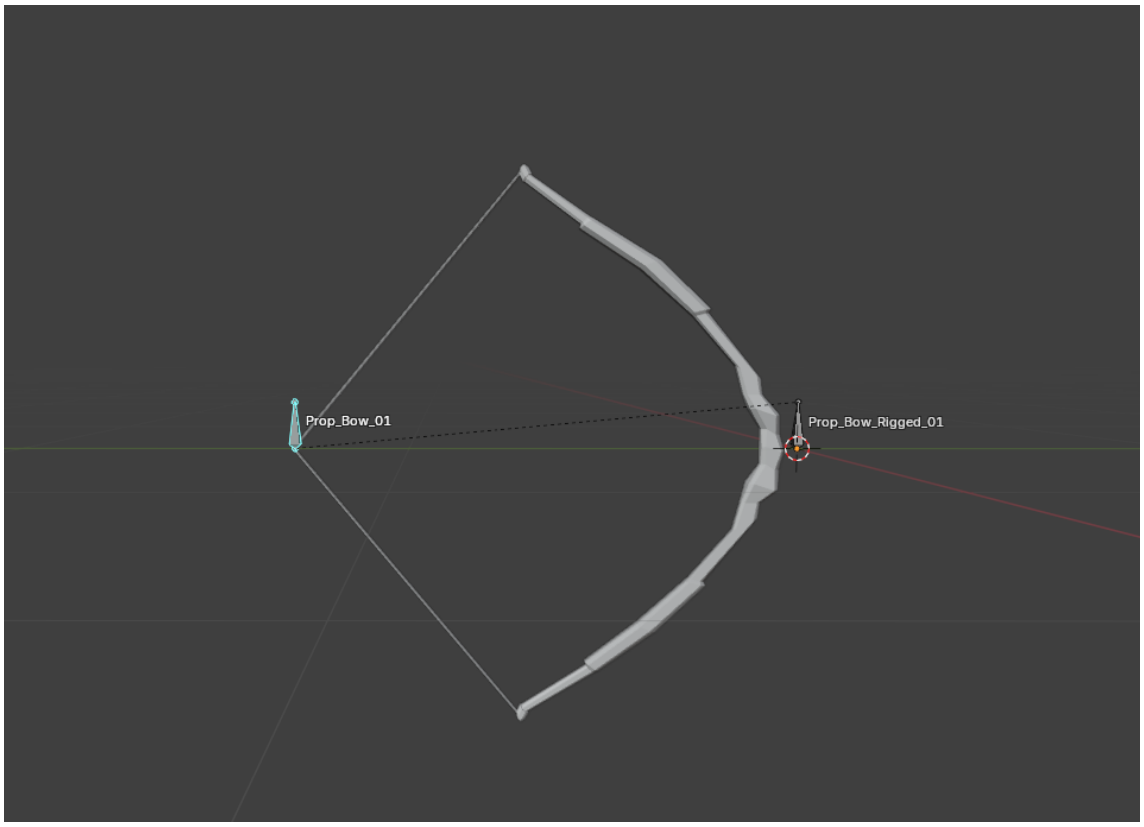
Kuva 13, jousen tytärluu valittuna paino jakautuu enemmän jousen puolelle, kuvakaappaus Blenderistä



Kuva 14, Oikeanpuoleisessa kuvassa juuriluun vaikutus painottuu enemmän jousiaseen varteen, kuvakaappaus Blenderistä

Kuvista näkee, miten paino on jaettu aikalailla puolesta välistä kummallekkin luulle. Tytärluun paino menee aina varteen saakka, siten vaikuttaa myös vartta, kun jousta jännitetään.

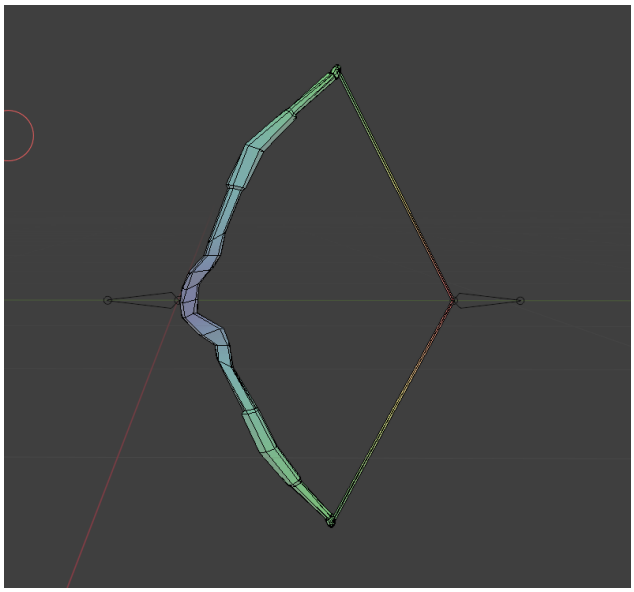
Ilman minkäänlaisia liikerajoitteita tai tarkempaa luustomallia tämä tekniikka kuitenkin kaatuu nopeaa jos seuraa sen toimintaa huolella. Itseasiassa sen huomaaminen ei vaadi kovin paljoa, tässä esimerkki miltä jousen venytys voi näyttää.



Kuva 15, Venytys yksinkertaisella luustolla, kaksi luuta ilman rajoituksia, kuvakaappaus Blenderistä

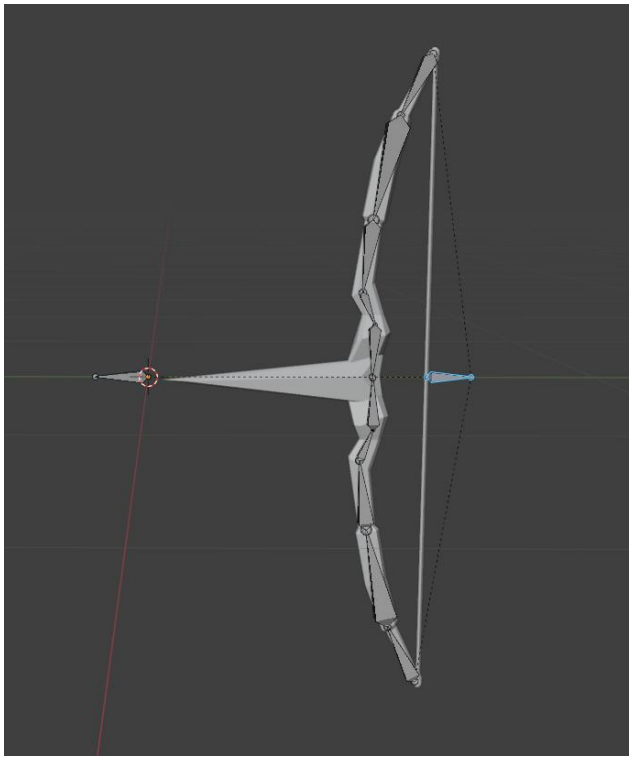
Tälläinen deformaatio ei välttämättä ole ongelmallinen, sillä pelikäytössä 3D-mallia ei välttämättä vedetä niin paljon, että jousen deformaatio olisi selvää. Tätä itse ajattelin kuitenkin välttää tämän käytön ja päädyin yrittämään eri ratkaisuja.

Halusin seuraavaksi yrittää päivittää painokarttaa ja se päättyikin aika huonosti. Oma observaatio oli, että en saanut painokarttaa tarpeeksi sulavaksi. Tällä tarkoitan paino erojen muuttumista.



Kuva 16, Tästä näkee, kuinka tekemäni painokartta on liian kulmikas, kuvakaappaus Blenderistä

Tämän jälkeen, lähdin lisäämään luita jousiaseen varteen, ajatellen että se tekisi siitä enemmän joustavamman. Tämä ei kuitenkaan itseltä onnistunut alkuunkaa, lopputulos oli karu ja ei toiminut kunnolla. Ongelmana oli ettei muu jousi taipunut, vain kahva venyi.

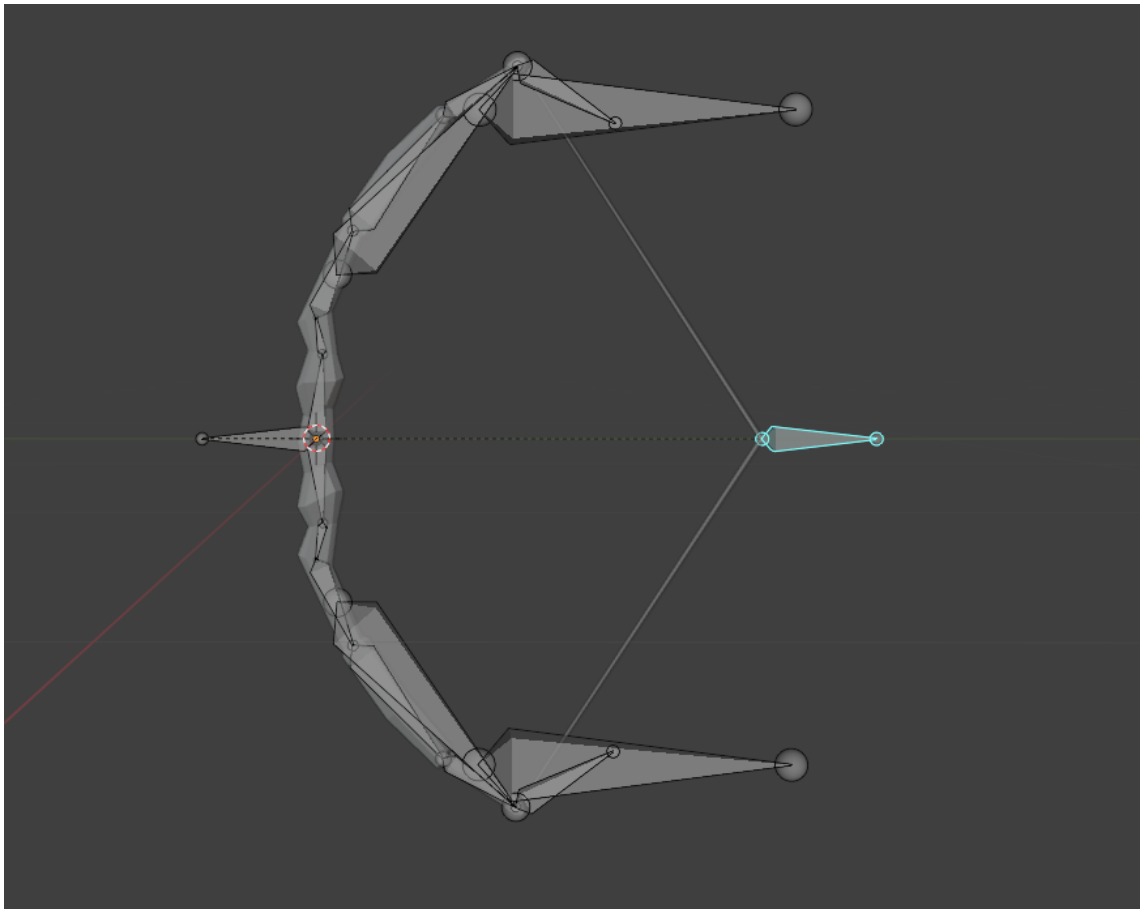


Kuva 17, Kolmas jousiaseen mallinnustesti, kuvakaappaus Blenderistä

Kolmannen yrityksen jälkeen täytyi mennä katsomaan jonkinlainen ohjevideo aiheeseen, sillä kolmas yritys meni vielä huonommin, kuin aikaisempi.

Video minkä katsoin osoittautui erittäin hyväksi ja siinä käytettiin constraintteja (rajoitteita), jotka voivat asettaa rajoittamaan liiketta tietyille akseleille tai kuinka pitkän matkan jokin asia voi liikkua (Erasers45 – Studios, 12.5.2025).

Video oli selvä ja helppo seurata, lopputulos oli monimutkainen. Niin monimutkainen, että tuskin olisin itse saanut tämmöistä tehtyä. Liikerajoitteiden lisäksi, mallissa käytettiin IK-rajoitteita, IK tulee sanoista Inverse Kinematics (käänteinen kinematiikka). Ne toimivat kutakuinkin samanlailla kun liikerajoitteet. Ainoana erona IK-rajoitus rajoittuu toisen luun mukaisesti (Unity, 2025). Lopputuloksessa jousta ei voi vetää yli tietyn rajan ja verkkorakenne ei deformoidu nähtävästi ollenkaan.



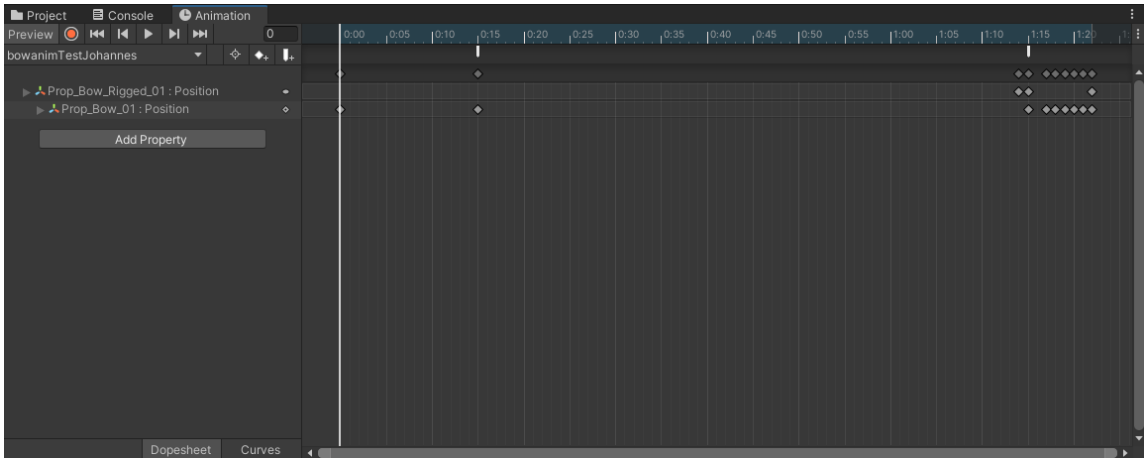
Kuva 18, Ohjeen mukaan tehty jousiase vaikuttaa sekavalta lisäluiden takia. Kuvakaappaus Blenderistä

Jousiaseen malli valmiina, seuraava vaihe olisi tuoda malli Unityyn. Tätä tehdessä kuitenkin selvisi, että liikerajoitteet ja IK-rajoitteet eivät toimi Blenderin ulkopuolella. Tämä tarkoittaa sitä, ettei kuvan 19 mallia voida käyttää.

Mietinnän jälkeen päädyin käyttämään valmista mallia, ilman muutoksia. Kuten mainitsin aikaisemmin, mallin deformointi ei välttämättä ole ongelma.

4.2 Jousiaseen animointi

Tarkoitus oli käyttää hahmolla Unityn asset kaupasta saatua animaatiota, joka esittää yksinkertaista ampumista jousiaseella, ilman hienouksia. Jousiaseen animaation tein itse ja testeissä ajoin animaatiot heti testikäynnistyksen alussa tyhjässä maailmassa.



Kuva 19, Unityn animaatio-ikkuna. Kuvakaappaus Unitystä

Jousiaseen animointi kävi helposti Unityssä, käyttämällä isoa scene-näkymääja animaatio-ikkunaa. Vasemmalta voi avata ”Position” kohdan, jonka sisältä voi vaikuttaa valitun luun positioon eri akseleilla. Tässä tapauksessa käytettiin vain yhtä akseliä, minkä mukaan jousi taipuu. 0.15 sekunnin kohdalla veto lähtee käyntiin ja 1.15 sekunnin kohdalla jänne vapautetaan. Näiden välillä jänne menee kohti 1.15 sekunnin kohdalla olevaa positiota lineaarisesti aikaisemmasta positiosta. Jänteen vapautuksen jälkeen kaaviolla näkyvät pisteet ovat jänteen tärinää/heiluntaa.

4.3 Animaatioiden synkronointi

Testeissä ilmeni että, animaatiot olivat ihan hyvän näköisiä työmäärään verrattuna. Animaatiot pyöri oikeaan aikaan, jousen vapautus animaatio tapahtui samaan aikaan kun hahmo päästi irti siitä. Animaatioita tehdessä suunnitelmana oli kuitenkin käyttää Unityn IK-systeemiä, jolla mahdollistettaisiin hieman tarkempi animaatio.

Kuva 20, Animaatiot ilman lisäominaisuuksia. Kuvakaappaus Unitystä

Tähän käyttöön valitsin ”Two Bone IK” tyyppisen IK-rajoitteen, jonka käyttöön tarvitsee kohde-luun ja raajan mikä kiinnittyy siihen. Tässä projektissa käytin oikeaa kättä raajana ja kohdensin sen jousen jänteeseen. Tämä johti raajan kulkemista koko ajan kohteeseen, minkä takia käsi seuraavissa kuvissa menee hieman oudoista paikoista läpi.

Kuva 21, Animaation lopusta kuvakaappaus, jossa käsi menee kehon läpi. Kuvakaappaus Unitystä

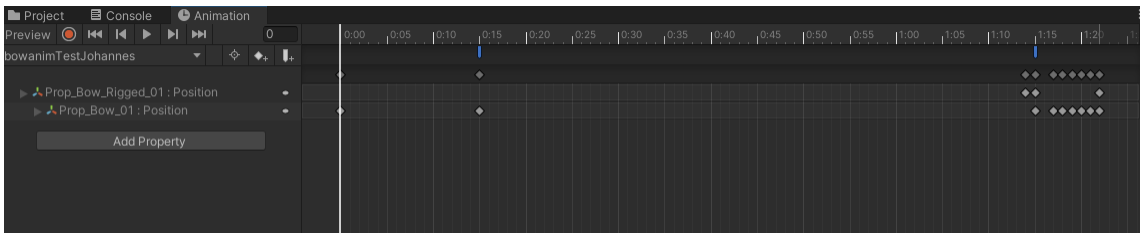
Kuva 22, Ampumisen jälkeen käsi pysyy kohdistetttuna jänteeseen. Kuvakaappaus Unitystä

IK-rajoitteen painon arvoa voi muokata ja yllä olevat kuvissa painon arvo oli 1. Tässä tapauksessa skaala oli nollasta yhteen. Nolla tarkottaisi, ettei IK vaikuta käteen ollenkaan ja tämän ollen yksi, käsi pyrkisi täysin haluttuun kohtaan. Tämän avulla suunnitelma oli muokata paino-arvoa tietyissä kohdissa. Nämä kohdat olivat jänteen veto ja vapauttaminen. Tämän mahdollistamiseksi loin uuden skriptin, jonka kiinnitin jouseen Unityssä. Se mahdollistaa skriptin päällä olon, silloin kun jousi on aktiivinen eli näkyvissä.

```
public class attachIKToBow : MonoBehaviour
{
    2 references
    public GameObject go;
    0 references
    TwoBoneIKConstraint constraint;
    0 references
    public void AddWeight()
    {
        TwoBoneIKConstraint constraint = go.GetComponent<TwoBoneIKConstraint>();
        constraint.weight = 1f;
        Debug.Log("twobone weight is currently at " + constraint.weight);
    }
    0 references
    public void LoseWeight()
    {
        TwoBoneIKConstraint constraint = go.GetComponent<TwoBoneIKConstraint>();
        constraint.weight = 0f;
        Debug.Log("twobone weight is currently at " + constraint.weight);
    }
}
```

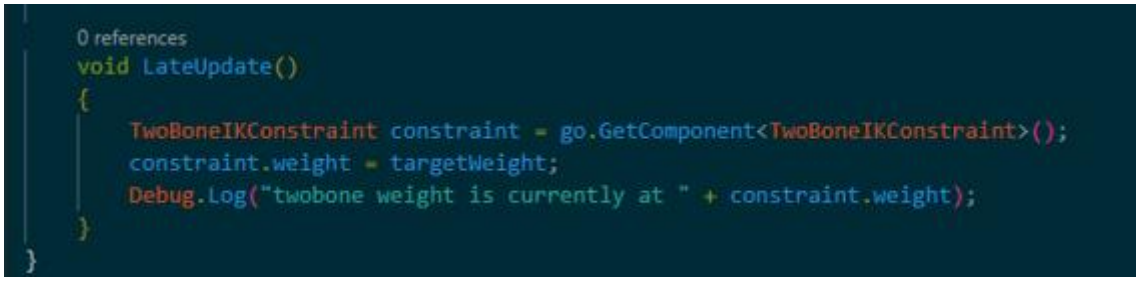
Kuva 23, Ensimmäinen versio koodipätkästä, joka mahdollistaa IK-rajoitteen painon muutokset. Kuvakaappaus omasta koodista

Skriptin sisältö on yksinkertainen ja helppo ymmärtää. Kun halutaan että IK-rajoite aktivoituu, eli käsi ottaa kiinni jänteestä, kutsutaan ”AddWeight” funktiota. Samalla lailla kutsutaan ”LoseWeight” funktiota, kun halutaan irroittaa käsi jänteestä. Kuvasta 24 näkee Unityn animaatio ikkunassa olevan kaksi sinistä kohtaa. Nämä merkkaavat animaatio tapahtumia. Animaatio tapahtumat voidaan asettaa kutsumaan funktiota esimerkiksi (Unity, 2025). Tässä tapauksessa ensimmäinen tapahtuma kutsuu ”AddWeight” funktiota ja toinen ”LoseWeight” funktiota.



Kuva 24, Sinisellä näkyy "animation event". Kuvakaappaus Unitystä

Tämä ratkaisu vaikutti IK-rajoitteen paino-arvoon, mutta ei muuttanut sitä kuitenkaan visuaalisesti pelin sisällä ja paino-arvo palautui takaisin alkuperäiseen arvoon joka pelin frame. Yhden framen aikana peli käy läpi logiikkaa ja se vaikuttaa, kuinka monta framea nähdään sekunnissa (FPS, framea sekunnissa). Tähän ratkaisuksi tuli ”LateUpdate”, mikä on Unityn MonoBehavior-luokkaan sisään rakennettu funktio. Se mahdollistaa koodin läpi käymisen sen jälkeen, kun muut päivitys funktiot ovat suoritettu joka frame (Unity, 2025). Tämän avulla muutos lähetettiin vasta sen jälkeen, kun IK-rajoitteen paino-arvo oli vahvistettu alkuperäiseen lukemaan, täten muuttaen paino-arvon halutuksi.

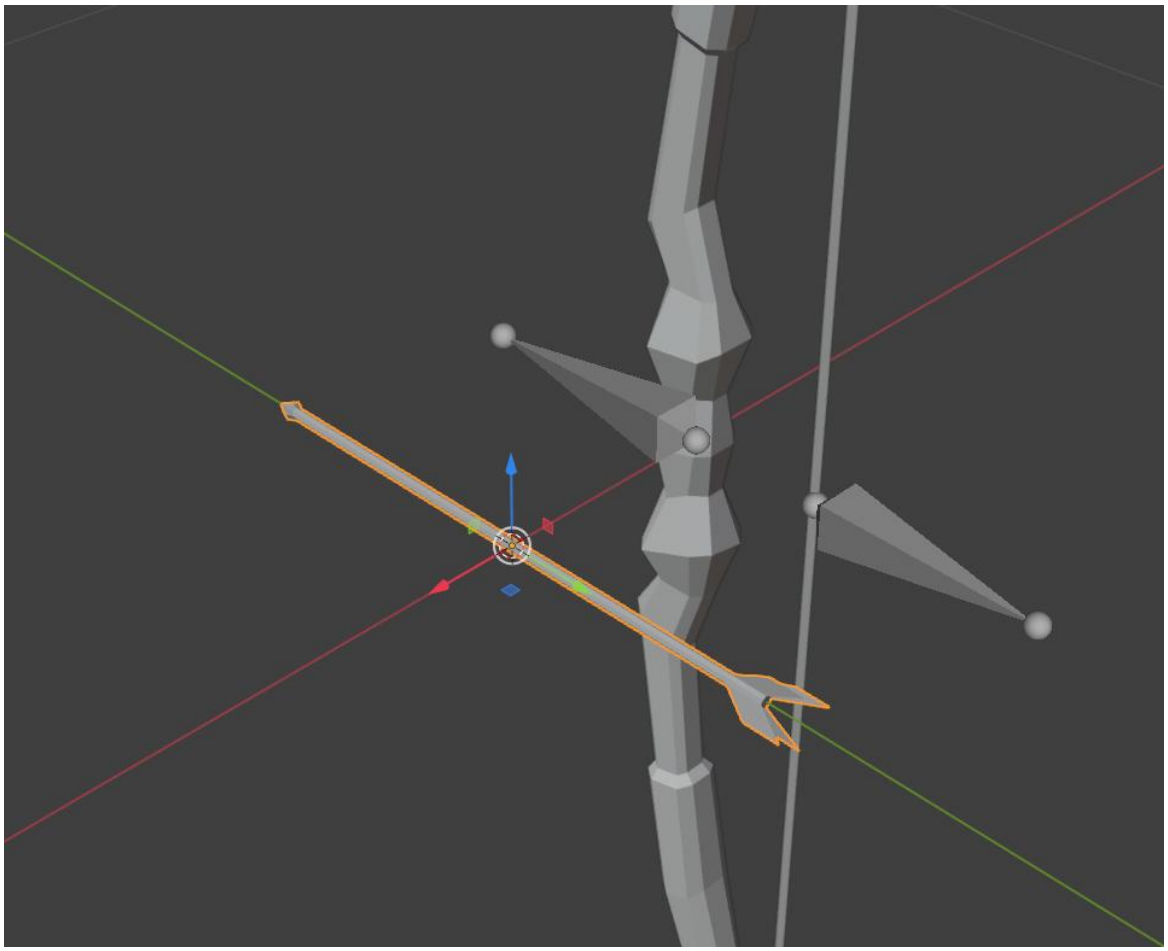


Kuva 25, Päivitys skriptiin, mikä hallitsee IK-rajoitteen paino-arvoa. Kuvakaappaus omasta koodista

Kuva 26, Käsi vapautuu jänteestä skriptin päivityksen jälkeen. Kuvakaappaus Unitystä

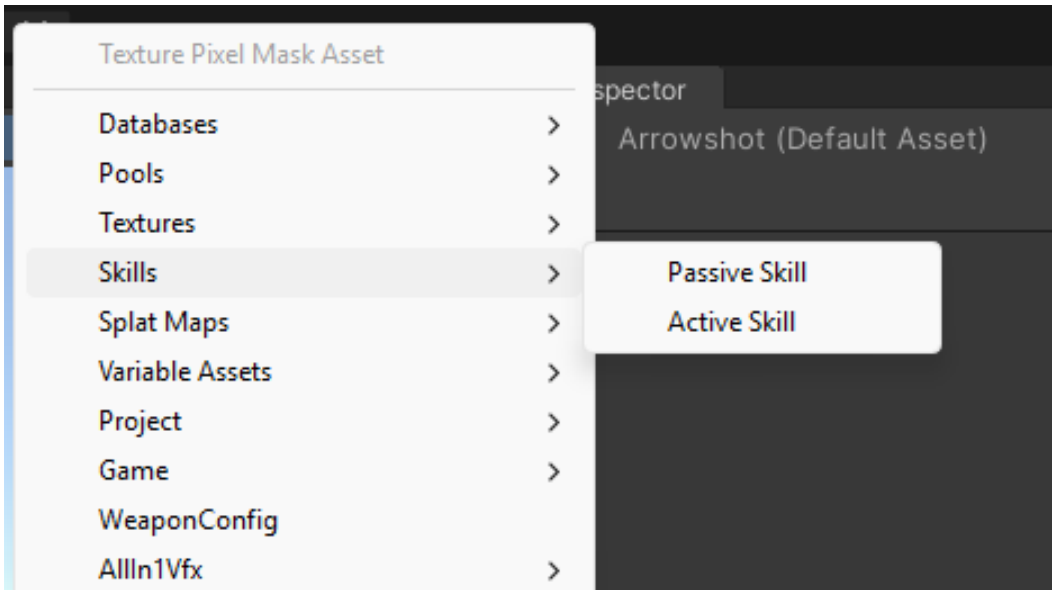
4.4 Kyvyn tekeminen peliin

Borealuksen järjestelmän ansiosta kyvyn kehittäminen oli helppoa. Melkein koko prosessi oli tehty yksinkertaiseksi ja kyvyn voi tehdä kokonaan ilman koodaamista. Yksi manuaalinen asia mikä piti tehdä, oli 3D-malli nuolesta. Tämä onnistui kuitenkin nopeaa. Tekstuurina käytin Unitystä kirkkaan valkoista pintaa.



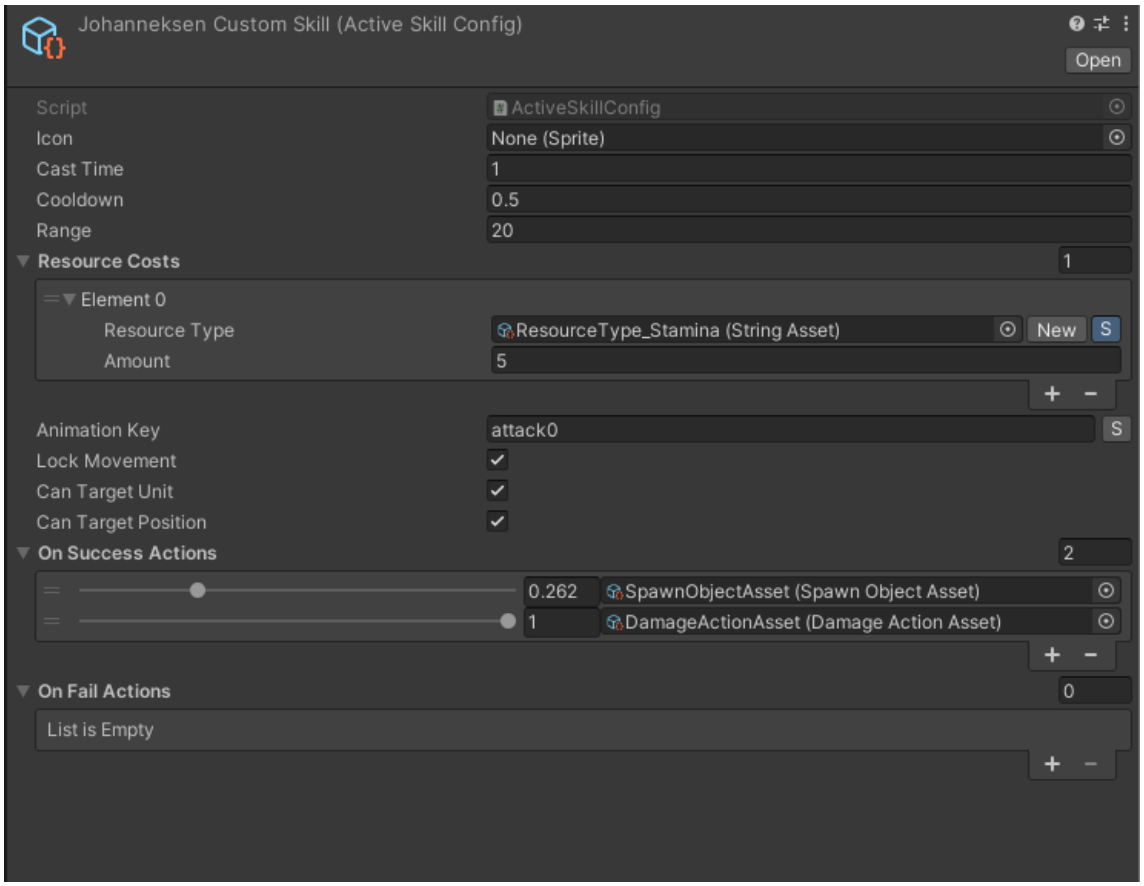
Kuva 27, Valmis yksinkertainen nuoli, jousta käytettiin koko vertailuun. Kuvakaappaus Blenderistä

Kyvyn luonti lähtee käyntiin tässä tapauksessa Unityn puolelta tekemällä assetin käyttäen konteksti menua. Kyky on aktiivinen, joten valitsin ”Active Skill”.



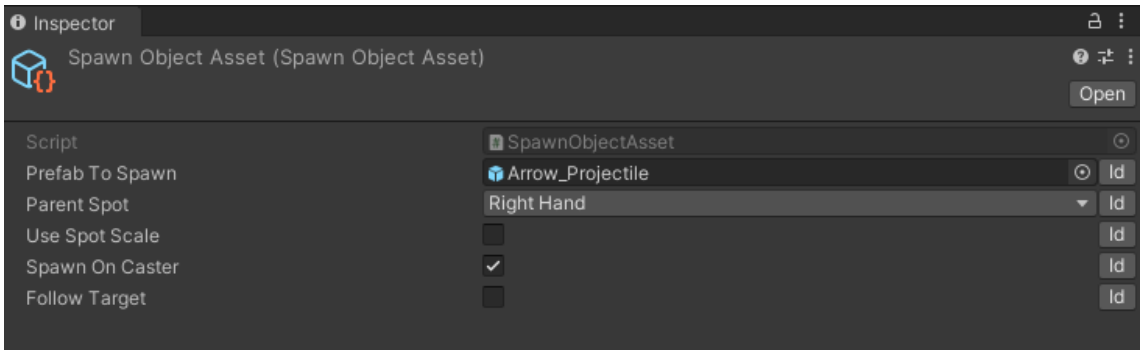
Kuva 28, Konteksti menussa paikka Skillin luontiin. Kuvakaappaus Unitystä

Kuvassa 29 on monta eri vaihtoehtoa, mitkä tekee kyvystä valmiin. Tärkeimmät osiot tästä menusta on ”Animation Key” (animaatioavain) ja ”On Success Actions” (Onnistumisen jälkeiset toiminnot). Animaation avaimen pitää olla sama, kuin hahmolle sen konfiguraatiossa asetetut avain-animaatio parit, jotta järjestelmä osaa valita oikean animaation minkä suorittaa. Onnistumisen jälkeiset toiminnot viittaavat, onko ”cast” eli loihdinta suoriutunut onnistuneesti. Tässä tapauksessa on kyse loihdinnan sijasta jänteen vetäminen. Onnistumisen jälkeiset toiminnot eivät suoriutuisi jos jänteen veto keskeytettäisiin. Tämä sisältää nuolen luonnin ja sen vahingon teon. Nämä ovat kuvassa 29 ”SpawnObjectAsset” ja ”DamageActionAsset”. SpawnObjectAssetille on valittu luku 0.262, joka tarkoittaa toiminnon tapahtuvan kun 26.2% loihdinnasta on onnistunut.



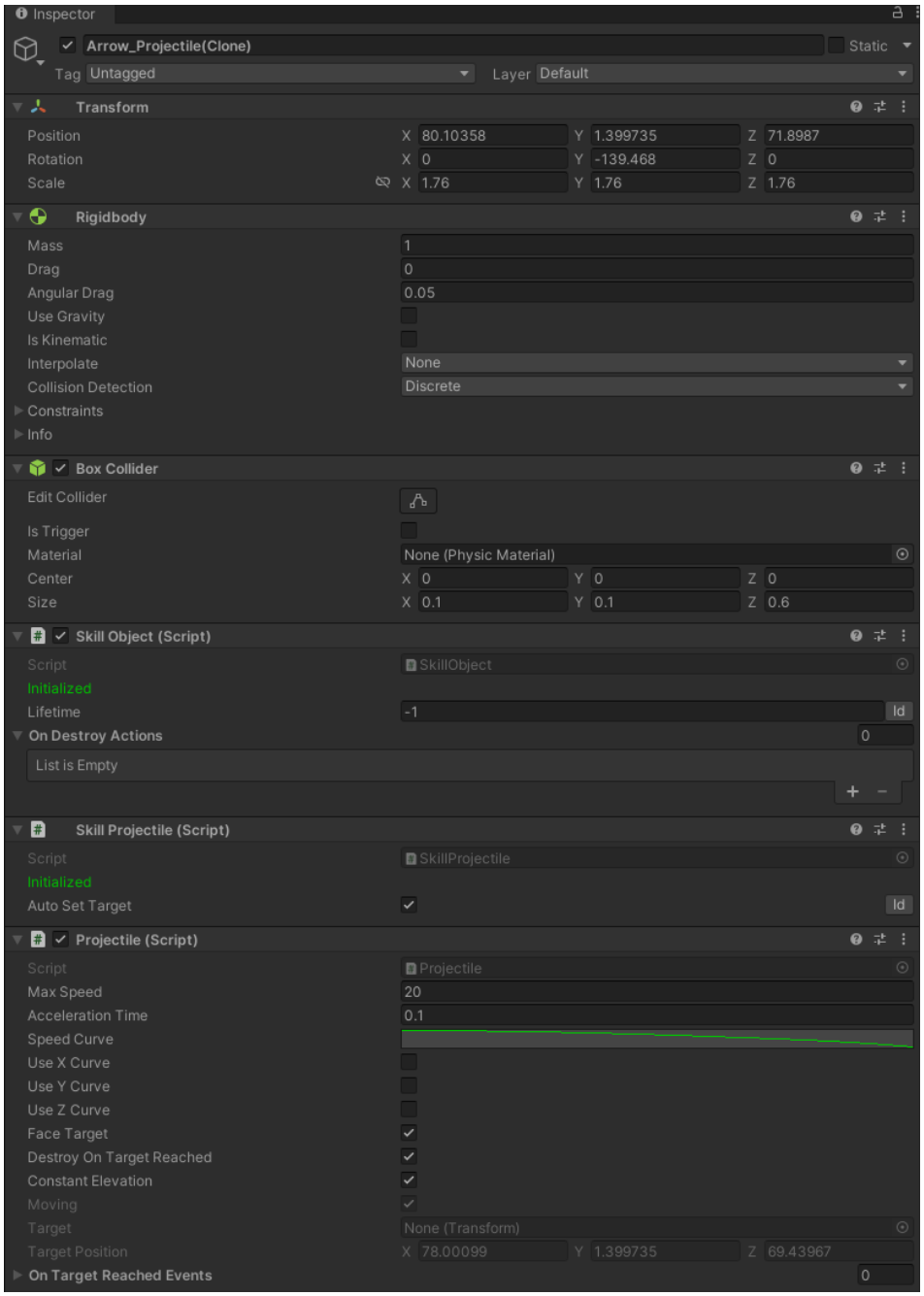
Kuva 29, kyvyn muokkaus menu. Kuvakaappaus Unitystä

Tässä tapauksessa kyky luo nuolen ja se lähtee oikean käden kohdalta. Myös ”Spawn On Caster” on tärkeä, ilman sitä nuoli ilmestyisi tähdättyyn paikkaan. Nuolen luonti onnistuu Spawn Object Assetin avulla, mikä luo nuolen ja pitää tiedossa esimerkiksi mistä nuoli lähtee.



Kuva 30, Objektin luonti-assetin menu. Kuvakaappaus Unitystä

Arrow_Projectile, mikä on kuvassa 30 kohdassa ”Prefab to Spawn” on nuolen 3D-mallista luotu objekti, millä on kollaasi erinlaisia skriptejä kiinni, mahdollistaen fysiikan ja integraation pelin logiikkaan.



Kuva 31, Nuoli-objektin inspector sivu. Kuvakaappaus Unitystä

Nuolen komponenteista hieman. Rigidbody (jäykkä runko) asettaa objektin liikkeen Unityn fysiikkamoottorin huostaan. Box Collider (Laatikko törmäytin) on objektin niin sanottu Hitbox (törmäyslaatikko), joka vastaa törmäämisestä ilmoittamisen muulle logiikalle. Sitä voi käyttää esimerkiksi kyvyn vahingon

laskemisessa ja esittämisessä hyödyksi. Projectile (ammus) komponentista voi muuttaa nuolen kulkuvauhtia ja sen nopeuden laskemisen/lisäämisen lisäksi muita attribuutteja, kuten tuhotaanko nuolen peliobjekti kun se osuu johonkin. Se auttaa lisäämään pelin suorituskykyä: Jos sitä ei tuhottaisi, peli joutuisi laskemaan sen toiminnan vaikka se olisi osunut maahan ja olisi toimeton.

Animaatio ja hahmon kyvyt pakattiin assettiin, mikä liitettiin hahmon skriptiin Player Behavior (pelaajan käyttäytyminen). Kun käytin kyvyn animaatio avaimena ”attack0” osasi järjestelmä käyttää aikaisemman konfiguraation avulla oikeaa animaatiota.

Kuva 32, Testissä animaation alku. Kuvakaappaus Unitystä

Kuva 33, Tilannekuva kykyä käytettäessä. Kuvakaappaus Unitystä

5 YHTEENVETO

Opinnäytetyön tavoitteena oli tehdä hahmoluokka rpg-peliin ja siihen päästiin mutkien kautta. Hahmoluokka sisältää monta eri osaa, joista suurin osa saatiin valmiiksi hyvin. Hahmoluokan kehittäminen on monimutkainen prosessi, joka kattaa muitakin osia pelien kehityksestä. Esimerkkinä tässä työssä tehtiin 3D-mallinnusta ja koodaamista.

Työ lähti käyntiin jousiaseen valmistelulla, mikä sisältää luuston luonnin ja animoinnin. Borealuksella oli erinlaisia valmiita jousiase malleja Unityn asset storesta, näistä valittiin malliksi yksinkertainen jousi tarkoituksella. Tämän valmistelussa tuli ongelmia ja työtä jota ei suoraan voinut käyttää. Lopputulos animoidusta jousesta kuitenkin on tyydyttävä. Jousen animaatio piti sovittaa aikaisempia päätöksien takia hahmon animaatioiden kanssa, joka päättyi myös hyvin. Kyvyn luonti oli helppo Borealuksen järjestelmän ansiosta. Siinä ei ollut hirveästi ongelmia.

Hahmoluokan keskittymispuu on olennainen osa pelejä. Sen avulla hahmosta luodaan omalaatuinen. Tässä vaiheessa pelin kehittämistä ei kuitenkaan voi kehittää valmista kokonaisuutta, joten tämä osio jäi pohjustukseksi tulevaa varten.

Työn aikana myös tutustuin isomman projektin piirteisiin ja toimimaan järkevästi ison tiedostomäärän kanssa. Versionhallinta on aina tärkeämpää mitä isompi projekti on ja tässä käytettiin GitHubia, mikä on ennaltaan tuttu, mutta lisäkäyttö ei haittaa ollenkaan.

Melkein kaikki mitä tein tämän projektin aikana oli jokseenkin uutta. Kun kyse ei ole vain eri kielien funktioista tai loopeista, on vaikea asettua tekemään jotain projektissa, mistä ei ole etukäteen tietoa. Viittaan tässä siihen, kun on iso projekti, jossa tehdään lukemattomia eri systeemejä pelimoottorin omien systeemien lisäksi. Näiden käyttö vaatii ymmärtämistä ja sain aina Borealukselta apua näissä tilanteissa. Jokainen eri systeemi on tehty jotain varten ja oli mukava ymmärtää mitä ne tarkoittivat lukemalla eri skriptejä.

Työ antoi pohjan, millä Borealus voi jatkaa kehittämistyötä. Valmis hahmoluokka on nykypeleissä hieman mysteer. Jos kyse on moninpelistä, todennäköisesti jotain hahmoluokassa tai aseiden suorituskykyä tullaan muuttamaan jossain vaiheessa. Yksinpeleissä balanssi ei ole välttämättä niin tärkeä, kunhan se ei tunnu huonolta pelata. Itselle työstä jäi paljon käteen ja tulen varmasti hyödyntämään oppimaani.

Oman pelin kehittäminen on iso taakka ja voi olla vaikea hahmottaa mitä se sisältää. Miltein koko ikäni olen pelannut pelejä jollain tavalla. Aloitin Playstation 1:sellä ja nykyään tietokoneella. Pelin kehittäminen on aina ollut mielessä ja en ollut pitäny sitä mahdollisena ennen kuin alotin koodaamaan. Tämän opinnäytetyön avulla voin nähdä itseni kehittämässä peliä tulevaisuudessa.

LÄHTEET

Blender Guru, 26.7.2024. Beginner Blender Tutorial – Full Course. Katsottavissa: https://www.youtube.com/watch?v=4haAdmHqGOw. Haettu: 12.2.2025

Blender, 2025. Introduction, Your First Armature. https://docs.unity3d.com/6000.1/Documentation/Manual/script-Animation-WindowEvent.html. Haettu 11.2.2025.

Erasers45 – Studios, 12.5.2025. How to RIG a BOW for Animation in Blender 4.4 [Tutorial]. Youtubesta, alkuperäinen video oli vanha versio tästä, melkein sama prosessi. Katsottavissa: https://www.youtube.com/watch?v=4QltyD6ha9s. Alkuperäisen videon hakupäivä 28.2.2025.

Escape from Tarkov, 2025. Verkkosivun etusivulta olevasta videosta otettu kuvakaappaus, luettavissa: https://www.escapefromtarkov.com/. Haettu: 13.6.2025.

Wieland, R. 24.3.2024. What Is A Role-Playing Game? RPG Types Explained. Forbes. Luettavissa: https://www.forbes.com/sites/technology/article/what-are-rpg-games/. Luettu: 27.6.2025.

Hollow Knight, 2025. Verkkosivun etusivu, luettavissa/nähtävissä: https://www.hollowknight.com/. Haettu: 13.4.2025.

Kalinin, E. 6.10.2023. What Are Assets in Game Design? RetroStyle Games. https://retrostylegames.com/blog/what-are-assets-in-game-design/. Luettu: 23.5.2025

Marijan, B. 16.9.2021. How Does Git Work? phoenixNAP. Luettavissa: https://phoenixnap.com/kb/how-git-works. Luettu: 12.5.2025.

Unity, 2025. Add an Animation Event. https://docs.unity3d.com/6000.1/Documentation/Manual/script-AnimationWindowEvent.html. Haettu 15.6.2025.

Unity, 2025. Inverse Kinematics. https://docs.unity3d.com/6000.1/Documentation/Manual/script-AnimationWindowEvent.html. Haettu: 28.2.2025.

Unity, 2025. MonoBehaviour.LateUpdate. https://docs.unity3d.com/6000.0/Documentation/ScriptReference/MonoBehaviour.LateUpdate.html. Haettu 23.3.2025

Unity, 2025. System requirements for Unity 6.1. https://docs.unity3d.com/6000.1/Documentation/Manual/system-requirements.html. Haettu 26.3.2025

Unreal Engine, 2025 FAQ (Usein kysytyt kysymykset), How much do I have to pay for Unreal Engine. Luettavissa: https://www.unrealengine.com/en-US/faq

Unreal Engine 2025. Verkkosivun etusivulta otettu kuvakaappaus. Luettavissa: https://www.unrealengine.com/en-US. Haettu: 13.6.2025.

Wikipedia, Polygon Mesh, kuva 3D-mallin osista. https://en.wikipedia.org/wiki/Polygon_mesh. Haettu: 25.4.2025.