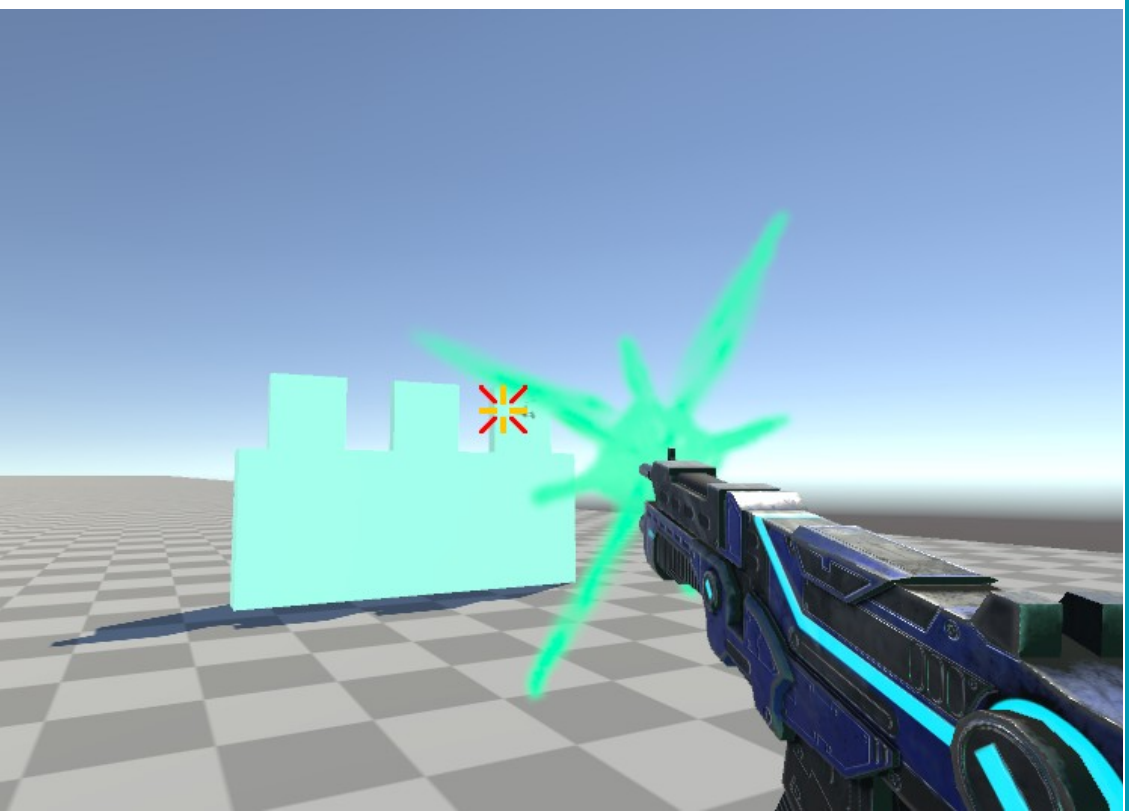
Perttu Ruottinen

Aseen toteutus ensimmäisen persoonan ammuntapeleissä Unity-pelimoottorilla



Tietojenkäsittely

Tradenomi

Kevät 2025

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä(t): Perttu Ruottinen

Työn nimi: Aseen toteutus ensimmäisen persoonan ammuntapeleissä Unity-pelimoottorilla

Tutkintonimike: Tradenomi (KAMK), tietojenkäsittely

Asiasanat: Ensimmäisen persoonan pelit, toimintapelit, peliaseet, pelinkehitys

Tämä tietojenkäsittelyn opinnäytetyö käsitteli aseen toteutusta ensimmäisen persoonan ammuntapelissä. Teksti sisältää eri tavat ohjelmoida aseen toimivuus, aseen ulkonäkö ja sovittelu peliruudulle sekä aseen pelattavuus ja sen vaikutus pelimaailmaan. Tavoitteena oli kuvata, mikä tekee peliaseesta miellyttävän ja responsiivisen ja ymmärtää eri toteutusmetodit ja niiden erot.

Teoriaosuudessa käsiteltiin peliaseen eri toteutusalueet. Ohjelmointipuolella on kaksi eri tapaa implementoida ase: hitscan ja projektiilipohjainen toteutus, joilla on erilainen vaikutus pelituntumaan ja pelin suorituskykyyn. Hyvin toteutettujen äänitehosteiden tärkeydestä kirjoitettiin osio, joka käsittelee äänten vaikutusta pelikokemukseen. Aseen ulkonäköön perehdyttiin graafisella tasolla, mutta myös aseen ideaalinen koko ja sovittelu peliruudulle käsiteltiin. Viimeiseksi pohdittiin aseen pelituntumaa ja responsiivisuutta. Onko sen käyttäminen miellyttävää? Kommunikoiko se pelaajalle kaiken tärkeän informaation kuten osumat ja ampumasuunnat? Näyttääkö sillä olevan vaikutus peliympäristöön?

Toteutusvaiheessa luotiin kaksi omaa peliasetta yksikertaisessa ensimmäisen persoonan peliympäristössä Unity-pelimoottorilla käyttäen ohjeistuksena kaikkea, mitä teoriaosuudessa käsiteltiin. Tuloksena oli sekä hitscan- että projektiilimetodeilla toteutetut yksinkertaiset peliaseet, joita voi ampua ja ladata vapaasti testiympäristössä äänien ja visuaalisten efektien kera.

Aseen toteutuksen ensimmäisen persoonan ammuntapeleissä huomattiin olevan monimutkaisempi prosessi kuin miltä se aluksi vaikuttaa. Toteutuksen aikana oli monia eri toimenpiteitä, joilla on merkittävä vaikutus siihen, miltä aseen pelattavuus tuntuu. Päätökset suunnittelussa, jotka eivät välttämättä edes vaikuta niin suurilta, saattavat täysin määrittää, pitävätkö pelaajat aseesta vai ei.

Abstract

Author(s): Perttu Ruottinen

Title of the Publication: Implementation of a weapon in a first-person shooter with Unity engine

Degree Title: Bachelor of Business Administration

Keywords: First person games, action games, game weapons, game development

This bachelor's thesis examines the implementation of a weapon in a first-person shooter videogame. This thesis contains two different ways of programming the functionality of a game weapon. It also goes through the weapon's appearance and arrangement on the player screen, as well as the gunplay and the weapon's impact on the game world. The goal is to analyze what makes a game weapon satisfying and responsive to use and to understand the different methods of implementation.

The theoretical part explores the two methods of programming a weapon: hitscan and projectile, which both have a difference in game feel and game performance. There's a section about the importance of well executed sound effects, which examines the impact of sound in the game experience. The appearance of a game weapon is studied in terms of graphics but also size and arrangement to the game screen. Lastly gunfeel and responsiveness are discussed. Is the weapon satisfying to use? Does is communicate all the relevant information to the player such as hit indication and shooting direction? Does it seem like it has an effect on the game world?

In the practical part two game weapons are created in a simple first-person environment with the Unity engine, using everything discussed in the theoretical part as guidelines. The result is two functioning game weapons implemented using both hitscan and projectile methods, that can be aimed around, shot and reloaded.

The implementation of a weapon in a first-person shooter is a lot more complex of a process than it seems. There are many steps that have a significant effect on the feel of a game weapon. Design choices that don't even necessarily seem so impactful could be the difference in players liking or disliking the weapon.

Sisällys

Symboliluettelo

Frame: Aika, joka määrittää, kuinka kauan kestää piirtää koko peliruutu näytölle (yleisintä peleissä on 60 framea sekunnissa).

Hitbox: Peliobjektia ympäröivä näkymätön raja, jota käytetään törmäyksen havaitsemiseen peleissä.

Optimointi: Suorituskyvyn parantaminen peleissä. Optimointia voi harjoittaa usealla tavalla, kuten lataamalla vain pelaajalle näkyvät objektit pelimaailmaan kerralla ja poistamalla objekteja, joita ei enää käytetä.

Partikkeliefekti: Efektejä, joita voi käyttää monenlaisiin tarpeisiin, kuten esimerkiksi kipinöiden, tulen tai savun visualisointiin.

Pelimoottori: Ohjelmistokehys, jolla pelejä voidaan rakentaa.

Peliobjekti: Pelin sisällä oleva objekti. Voi sisältää koodia, visuaalisia elementtejä tai olla tyhjä.

Polygoni: Kolmiulotteisessa grafiikassa renderöintiin käytettävä monikulmion kaltainen primitiivi.

Raycast: Yleensä peleissä käytetty keino, jolla luodaan pisteestä A lähteviä ”säteitä” 3D- tai 2D-ympäristössä osoittamaan määrättyyn suuntaan.

Ray: Yksittäinen raycast ”säde”.

Renderöinti: Peligrafiikan (peliobjektien, geometrian, tekstuurien) näyttäminen peliruudulla.

Rigid Body: Unity-pelimoottorissa oleva fysiikkakomponentti

Teksturointi: Peliobjektien ja geometrian pintojen visualisointi.

Unity: Pelimoottori.

1 Johdanto

Tämän opinnäytetyön aiheena ovat aseet ja niiden toteutus ensimmäisen persoonan ammuntapeleissä. Ensimmäisen persoonan ammuntapelit ovat yksi suosituimmista peligenreistä ja ovat niiden nykyisen määritelmän mukaan olleet olemassa jo 1990-luvulta lähtien. Tämän tyyppiset pelit ovat usein haastavia ja nopeatempoisia. Ne testaavat pelaajien refleksejä sekä käden ja silmän koordinaatiota ja ovat aina olleet minulle mieluisia.

Olen pelannut monenlaisia ensimmäisen persoonan ammuntapelejä jo nuoresta asti, joten kiinnostukseni ja tietämykseni kyseistä peligenreä kohtaan on vahva. Se miten aseet ovat suunniteltu ja toteutettu peleissä kuten Doom, Halo ja Half-Life, on aina kiehtonut minua. Tämä opinnäytetyö perustuu täysin omaan mielenkiintoon ensimmäisen persoonan pelejä kohtaan ja toimeksiantajaa ei ole.

Tämän opinnäytetyön tavoitteena on tutkia, miten ensimmäisen persoonan peliase suunnitellaan ja toteutetaan, miten paljon eri vaiheita näinkin yksinkertaiselta prosessilta vaikuttavaan projektiin tarvitaan ja oppia mahdollisimman paljon uutta tulevaisuutta varten. Käytännön toteutus tehdään Unity-pelimoottorilla, käyttäen C#-ohjelmointikieltä ja peliaseita tehdään yksi sekä hitscan-metodilla että projektiilimetodilla. Tutkin miten hitscan- ja projektiiliaseet implementoidaan Unityllä käyttäen hyvänä useita lähteitä ja tutoriaaleja netistä. Otan selvää, mikä tekee peliaseesta mukavan käyttää ja sovellan parhaani mukaan oppimani toteutusvaiheessa. Aseen 3D-mallin saan Unityn omasta nettikaupasta, mutta ääniefektit ja 2D-grafiikat suunnittelen itse.

2 Ohjelmointi

Ohjelmointi on aseen toteutuksen kaikkein oleellisin osio. Peleissä aseita voidaan toteuttaa kahdella tavalla: hitscan- ja projektiilimenetelmillä, joilla molemmilla on hyvät ja huonot puolensa. Kumpaa tyyliä siis tulisi käyttää, ei ole kysymys, johon kenelläkään olisi suoraa vastausta. Se riippuu täysin siitä, mikä sopii parhaiten peliin.

2.1 Hitscan

Ensimmäisen persoonan ammuntapelien alkuvuosina monet pelit käyttivät hyväkseen raycast-tekniikkaa renderöidäkseen 3D-ympäristöjä näytölle. Raycastin avulla pelimoottori voi määritellä ensimmäisen peliobjektin, mihin yksittäinen ray osuu. Ideana on käyttää tiettyä yksittäistä rayta simuloimaan luotia. Yksinkertaisesti hitscan-aseen implementaatio toimisi seuraavanlaisesti: Selvitetään suunta, johon pelaajan kamera osoittaa ja sitten luodaan yksi ray, joka alkaa pelikameran keskipisteestä haluamallesi etäisyydelle. Viimeiseksi tarkistetaan, osuiko pelaaja kohteeseen. (Jung, 2018.)

Kun pelimoottori on havainnut osuman, voidaan sen kautta määritellä, mitä rayn kanssa törmänneelle kohteelle tapahtuu. Voidaan esimerkiksi laskelmoida haluttu määrä vahinkoa tai tuhota peliobjekti kokonaan. Raycastin käyttäytymistä on myös mahdollista mukauttaa halutulla tavalla. Voidaan esimerkiksi antaa rayn läpäistä useamman kohteen kerrallaan, sen sijaan että vain ensimmäinen objekti rekisteröitäisiin osumaksi. Antamalla raylle myös rajaton pituus saadaan luodille loputon välimatka. Vaihtoehtoisesti voidaan asettaa peliin sopiva rajallinen etäisyys, mihin osumat vaikuttavat. Joitain pintoja peliin ympäristössä on mahdollista jopa ohjelmoida heijastamaan rayta, jolloin voidaan simuloida luotien kimpoamista. (Jung, 2018.)

Raycast on projektiileihin verrattuna helpompi implementoida, nopeasti prosessoitava ja vaatii vain vähän tietokoneen muistia, mutta siinä on myös haittapuolensa. Rayt eivät itse käyttäydy kuten luodit, niiden avulla voidaan vain simuloida luotia. Itsessään rayt käyttäytyvät enemmän kuin laseri. Ne eivät ole fyysisiä peliobjekteja, vaan ikään kuin näkymättömiä viivoja, jotka osuvat määränpäähänsä välittömästi (kuva 1). Niillä ei ole matkustusaikaa, mikä tekee niistä mahdottomia väistää pitkiltäkin matkoilta, jos osuma on jo rekisteröity. Rayt ovat myös täysin suoria, joten fysiikoiden implementoiminen, kuten tuulen ja painovoiman vaikutus, ei ole mahdollista. (Jung, 2018.)

Kuva 1. Hitscan-menetelmällä osuma tapahtuu välittömästi ammuttaessa. Kuvassa aseen suuliekki ja seinällä näkyvät kipinät ilmestyvät täysin samaan aikaan. Kuvattu peli: Doom (1993).

2.2 Projektiili

Projektiili-menetelmällä jokainen ammuttu luoti on oma peliobjektinsa, mikä luodaan pelaajan ampuessa. Näille luodeille voidaan määritellä nopeus ja painovoima.

Projektiileja voi luotien lisäksi käyttää kranaatinheittimien tai ohjusten implementointiin. (NeoFPSDocs, 2019.)

Projektiilipohjainen toteutus on ideaalinen vaihtoehto, jos pelissä tähdätään enemmän realismiin. Jokainen projektiili on oma peliobjektinsa, jolla on määritelty nopeus ja massa, mikä tarkoittaa sitä, että voimme implementoida niille esimerkiksi tuulen ja painovoiman vaikutuksia. Tällä menetelmällä luoti ei myöskään saavuta määränpäätänsä välittömästi, vaan sillä on matka-aika, mikä tekee luodeista helpompia väistää (kuva 2). Nämä edellä mainitut lisäykset kuitenkin vaativat enemmän prosessointia tietokoneelta kuin hitscan-menetelmällä toteutettu systeemi, mutta hyvällä optimoinnilla voidaan välttää hankaluuksia suorituskyvyssä. (Jung, 2018.)

Kuva 2. Aseen projektiilit voidaan nähdä kulkevan ilmassa. Kuvattu peli: Unreal (1998). (ValhaHazred, 2014.)

3 Kommentti aseen äänistä

Ääni on hyvin tärkeä osa aseen toteutusta peleissä. Ei ainoastaan siksi, että pelaaja osaisi hahmottaa, missä päin ampuminen tapahtuu ja reagoida siihen (Alar, 2023.), vaan myös miten ase vastaa sitä käytettäessä. Oikeanlaisten äänitehosteiden implementoimisella erityyppisiin aseisiin on suurempi merkitys aseen tuntuman kuin äkkiseltään voisi kuvitella.

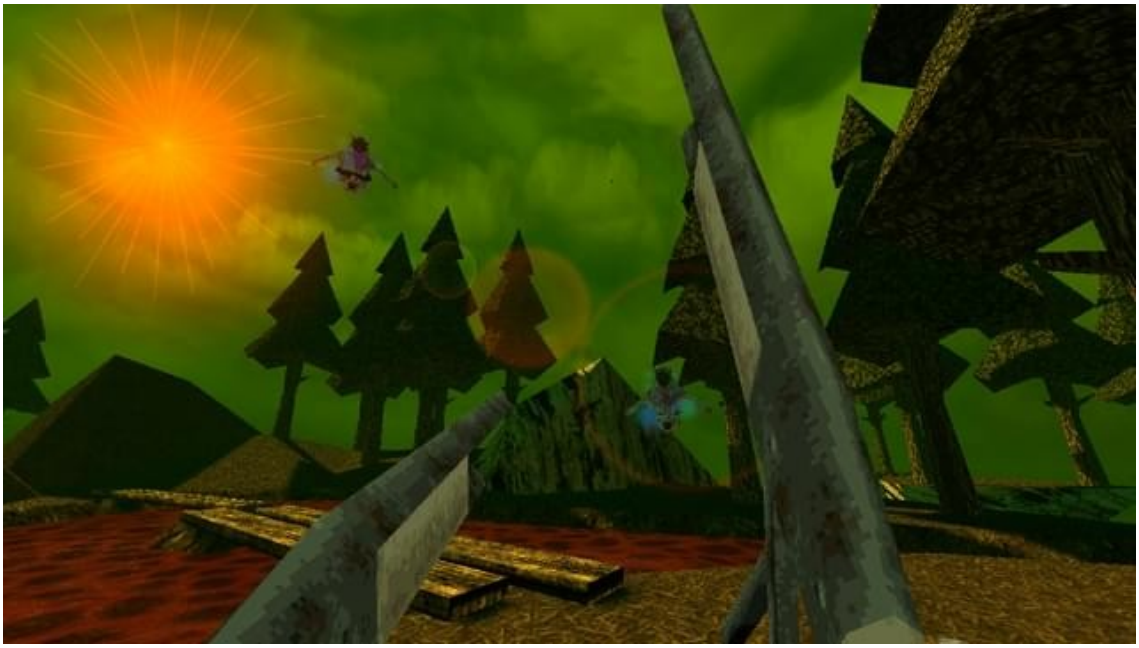
Hyvät äänet kommunikoivat pelaajalle välittömästi aseiden tehokkuudesta ja vaikutuksesta pelimaailmaan. Jos aseen ääni ei ole laadukas eikä vastaa aseen ulkonäköä ja vaikutusta peliympäristössä, pelikokemus saattaa kärsiä. Esimerkiksi pienemmiltä käsiaseilta yleensä odotetaan terävää paukahdusta, kun taas raskaamman tulivoiman aseelta, kuten vaikka haulikolta voisi odottaa voimakasta pamahdusta. (Holstrom, 2021.) Tämä opinnäytetyö ei sovellu sen syvemmin äänisuunnitteluun, mutta toteutusvaiheessa aseille annetaan kuitenkin sopivat ääniefektit.

4 Ulkonäkö

Se miltä peliaseet näyttävät, on tietysti tärkeä osa-alue toteutuksessa. Estetiikka on se, mitä pelaajat näkevät ensin poimiessaan aseen ensimmäistä kertaa. Hyvin suunnitellun peliaseen ulkonäkö on tunnistettava ja antaa pelaajille välittömästi idean siitä, minkä tyyppinen ase on kyseessä, oli se sitten kivääri, haulikko tai mikä tahansa muu.

4.1 Grafiikat

Mitä aseen ulkonäköön tulee ensimmäisen persoonan ammuntapeleissä, grafiikan ’laatu’ on se vähiten tärkeä tekijä. Aseessa voi olla satoja tai tuhansia polygoneja, uusimman sukupolven teksturointia ja heijastavia metallipintoja, mutta ne eivät ole välttämättömiä erinomaisen aseen toteutukseen. Sitä tärkeämmät seikat ovat enemmän käytäntöön perustuvia kuin kosmetiikkaan ja realismiin. Kunhan aseiden ulkonäkö on mielenkiintoinen sekä hyvin suunniteltu ja toteutettu, grafiikoiden tasolla ei ole niin suurta merkitystä. (Qingxiang, 2024.) Minimalistiset tai retrografiikat ovat tyylisuuntaus itsessään (kuva 3). (Couture, 2016.)



Kuva 3. Ase voi olla visuaalisesti yksinkertainen. Kuvattu peli, Dusk (2018) pyrkii grafiikoiltaan imitoimaan 1990-luvun loppupuolen ensimmäisen persoonan ammuntapelejä.

4.2 Sovittelu

Tärkeämpi tekijä aseen ulkomuodossa on sen ideaalinen koko ja asettelu pelaajan näkökentässä oikein. Näytön määritteleminen eri alueisiin ja niiden noudattaminen sovitellessa on suositeltavaa (kuva 4). Kriittisin alue on näytön keskiosa, koska pelaajan katsekontakti on keskitetty siihen suurimman osan ajasta. Jos ase sisältää esimerkiksi latausanimaatioita, sovitetaan ne ruudun sivulle ja pidetään huoli, että ne peittävät mahdollisimman vähän näytön tärkeimmistä alueista (kuva 5). Voidaan jopa käyttää latausanimaatioita korostamaan näytön keskipistettä ja keskittää pelaajan huomio siihen. (Helsby, 2015.)

Kuva 4. Keskellä olevaa punaista aluetta tulisi välttää peittämästä lähes kaikissa olosuhteissa. Kuvattu peli: Destiny (2014). (Helsby, 2015.)

Kuva 5. Kuvassa esitetty latausanimaatio tukee pelaajan näkökentän kriittistä aluetta ja keskittää pelaajan katseen sinne. Kuvattu peli: Destiny (2014). (Helsby, 2015.)

On havaittu, että pelaajan silmä yleensä kiertelee tähtäimen ympärillä ja harvoin poistaa katsekontaktin tältä alueelta (Kuva 6). On luonnollisempaa kääntää kamera pelin sisäisesti osoittamaan haluttua kohdetta ja katsoa sitä tähtäimestä käsin kuin kääntää omaa katsekontaktia ympäri näyttöä. (Helsby, 2015.)

Kuva 6. On havaittu, että pelaajien silmä yleensä kiertelee tähtäimen ympäri. Kuvattu peli: Destiny (2014). (Helsby, 2015.)

5 Pelattavuus

Aseiden pelattavuus ja vaikutus pelimaailmaan ovat keskeisiä tekijöitä aseen toteutuksessa ensimmäisen persoonan ammuntapeleissä. On monia keinoja parantaa pelikokemusta ja kommunikoida pelaajalle aseen toimivuudesta itse pelaajahahmosta käsin tai peliympäristöä käyttäen.

On useita visuaalisia indikaattoreita, joilla voidaan kommunikoida aseen vaikutusta pelimaailmassa. Esimerkiksi ampuessa hyvä tapa saada ase tuntumaan vahvalta on tärisyttää hieman pelaajan kameraa, aina kun asetta käytetään. Näin voidaan simuloida rekyyliä. Kameraliikkeiden lisäksi vielä hyvät animaatiot antavat aseille hyvän pelituntuman.

Pelaajan osuessa vihollisiin tai vaikka seiniin tulee jättää jälkiä, jotta aseella tuntuisi olevan vaikutus ympäristöön. (Scott-Jones, 2020.) Laukausten aikana voidaan näyttää valojuovia luotien lentäessä, jotta nähdään, mistä ja mihin päin luodit kulkevat. Vihollisia ampuessa tähtäimen ympärille voisi välähtää merkki osumasta (kuva 7). Näillä indikaattoreilla on suuri merkitys aseiden pelattavuuteen.

Kuva 7. Tähtäimen ympärillä esiintyy valkoisilla viivoilla merkattu tunniste osumasta. Kuvattu peli Call of Duty: Modern Warfare 2 (2009). (Lahti, 2012.)

6 Alkuvalmistelut

Toteutus alkoi lyhyellä suunnitteluvaiheella ja alkuvalmisteluilla. Suunnitellessa päätin, minkälaiset toteutetut aseet tulevat olemaan. Päätin, että projektiiliaseesta tulee tavallinen luoteja käyttävä rynnäkkökivääri, jossa on puoliautomaattinen tuli ja sarjatuli, joiden välillä voi vaihtaa. Hitscan-aseesta päätin tehdä futuristisen laser-aseen. Alkuvalmisteluun kuului visuaalisten efektien valmistaminen, aseen 3D-mallin hankkiminen ja valmiin testikentän muodostaminen.

6.1 Aseiden ja pelaajahahmon rakenteet

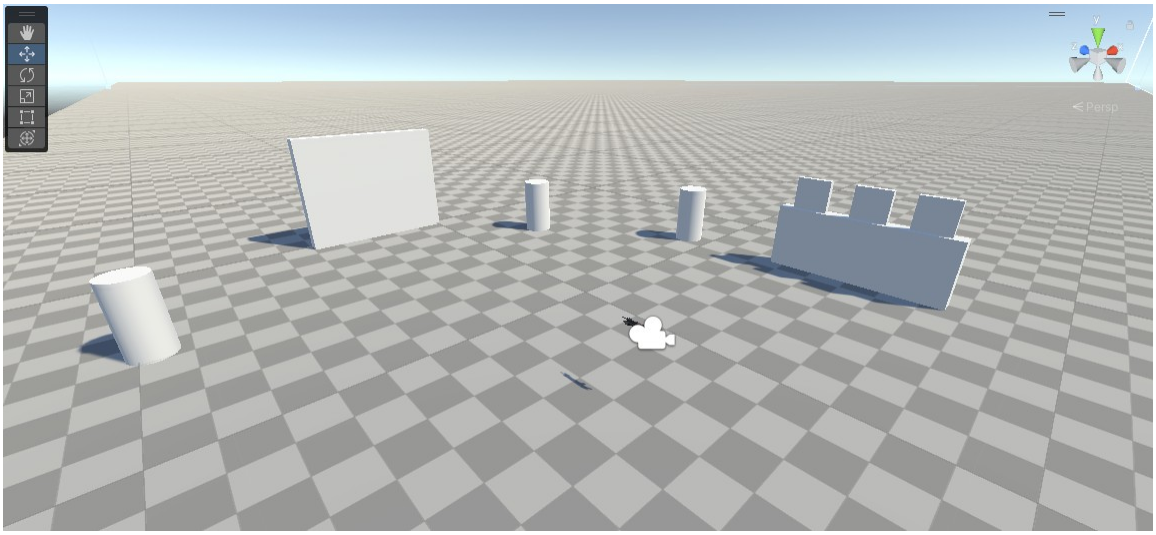
Aseiden ja pelaajahahmon rakenteet ovat varsin yksinkertaiset. On pelaajahahmo, jossa on pelikamera. Pelaajahahmoon asetetaan toteutetut aseet sopivalle korkeudelle ja etäisyydelle kamerasta, jotta ne näyttäisivät niin kuin pelaajahahmo pitäisi niitä käsissään pelaajan silmien näkökulmasta. Asettelusta myöhemmin lisää. Tarvitaan myös erillinen kamera renderöimään vain aseen. Tämän tehdään vian korjaamisen vuoksi, jotta ase ei mene peliympäristöstä läpi, jos pelaaja on vaikka seinää vasten (kuva 8). Pääkamerassa renderöidään kaikki paitsi ase. Luodaan erillinen kamera, joka renderöi aseen ikään kuin eri tilassa ja asettaa sen takaisin peliruudulle.

Kuva 8. Vasemmalla pelkällä pääkameralla renderöity peliruutu näyttää aseen menevän seinistä läpi. Oikealla pääkamera renderöi kaiken paitsi aseen. Aseelle on oma kamera, joka renderöi aseen erillisessä tilassa.

6.2 Visuaalit ja testikenttä

Tein tarvittavat visuaaliset elementit aseihin, jotka ovat oleellinen osa pelattavuutta ja tärkeitä hyvin toteutetuissa peliaseissa. Valmistin tähtäimen, ampumavälähdykset, indikaattorin osumaa varten ja luodinreikien grafiikat täyttääkseni nämä kriteerit. Aseiden 3D-mallit on ilmaiseksi ladattu Unityn omasta nettikaupasta.

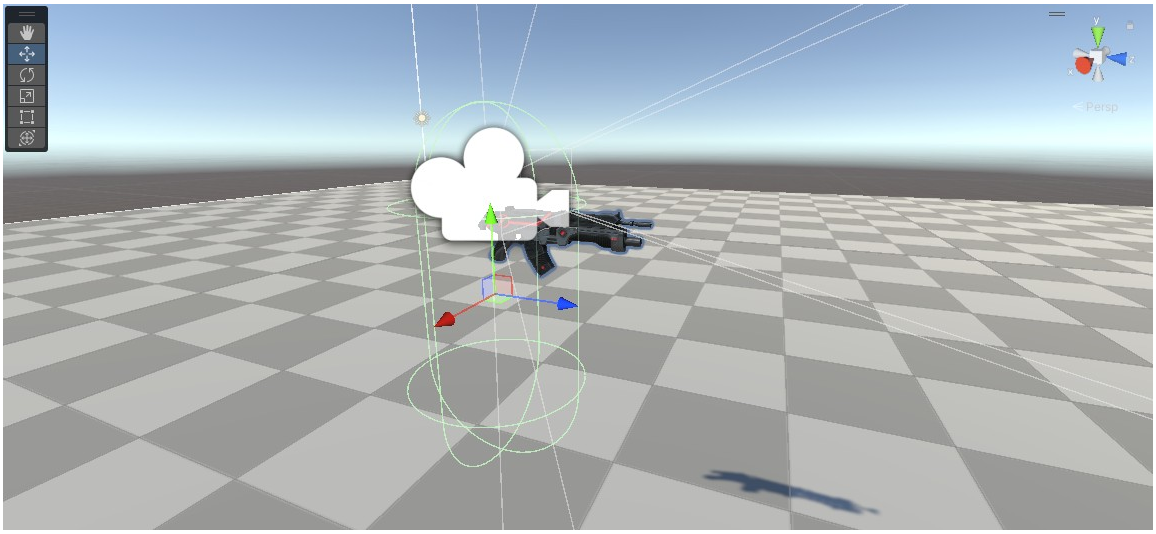
Viimeiseksi loin testikentän, jossa voidaan kokeilla aseiden toimivuutta (kuva 9). Tyhjään pelimaailmaan asetin tasaisen maaston, seiniä sekä kohteita. Tein vielä pelaajahahmon ja ohjelmoin liikkumisen ja ohjaukset (kuva 10).



Kuva 9. Testikenttä, tehty Unityn tasoeditorilla.

7 Toteutus – projektiili

Kahdesta metodista päätin ensin toteuttaa projektiiliaseen. Laitetaan aseen piipun päähän kaksi tyhjää peliobjektia. Toisesta lähetetään luodit ja toisesta näytetään ampumavälähdykset. Nämä peliobjektit ovat tyhjiä siksi, koska tarvitaan niistä vain niiden sijainnit pelimaailmassa, luoteja ja ampumavälähdyksiä varten.



Kuva 10. Liikuteltava pelaaja hahmo ja ase.

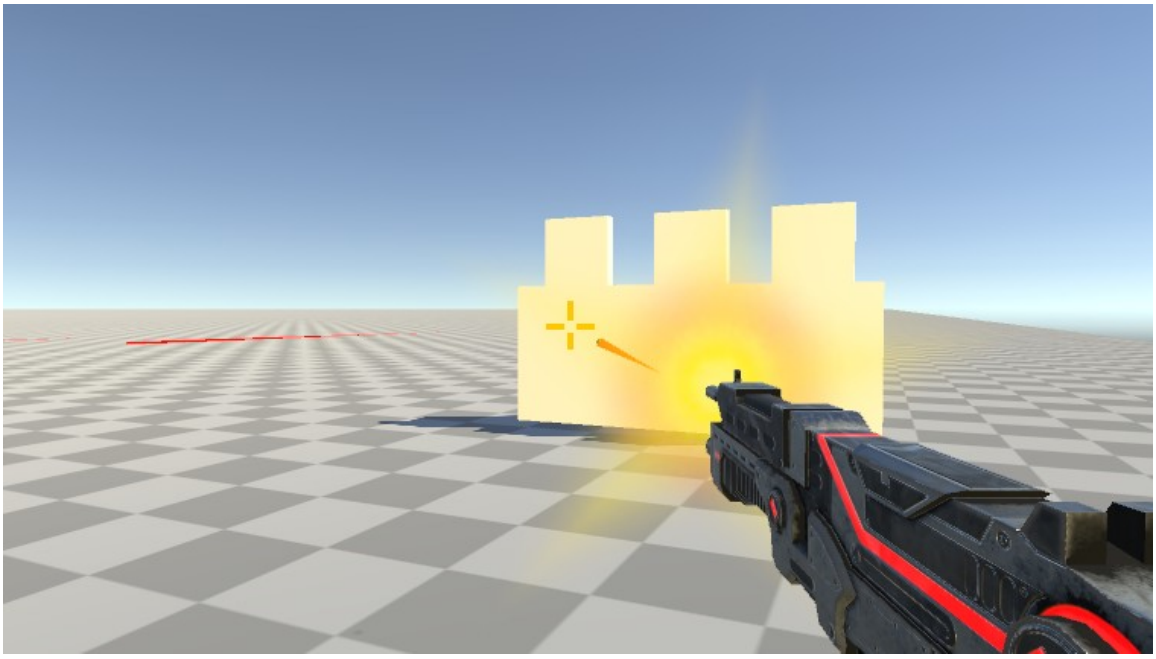
7.1 Toimivuus

Ensimmäisenä käydään läpi oleellisimmat muuttujat. Sovitaan luodeille nopeus ja elinaika. Elinaika siksi, koska jokainen luoti on oma objektinsa pelissä ja jos niitä ei tuhota aikanaan, pian pelimaailmassa on satoja luoteja ja tästä aiheutuisi tarpeetonta ylikuormitusta koneelle. Halutaan jokaisen laukauksen jälkeen tietää, kuinka kauan kestää, että ase on seuraavan kerran valmis ampumaan. Näin voidaan hallita tulinopeutta ja varmistaa, että se on sopiva. Annetaan luodeille hieman levitystä, jotta jokainen niistä ei osu robottimaisesti tismalleen samaan kohtaan. Asetetaan latausaika, eli kauanko latausanimaatio kestää, lippaaseen ammusmäärä ja tarkistetaan aina ennen ampumista, onko latausanimaatio käynnissä. Pelaajan ei pitäisi kyetä ampumaan asetta ladatessa.

Kun peli suorittaa ampumisen funktion, vähennetään ensin tämänhetkistä ammusmäärää yhdellä. Asetetaan luodille suunta ja luodaan se pelimaailmaan aseen piipun päähän. Asetetaan luodille lentovoima edellä määritetyn nopeuden mukaan ja aloitetaan ajastin luodin eliniälle. Viimeiseksi on viive, jolla tarkistetaan, milloin ase valmis ampumaan uudelleen. Kun ase ladataan, asetetaan ammusmäärä takaisin maksimimäärään.

7.2 Luoti

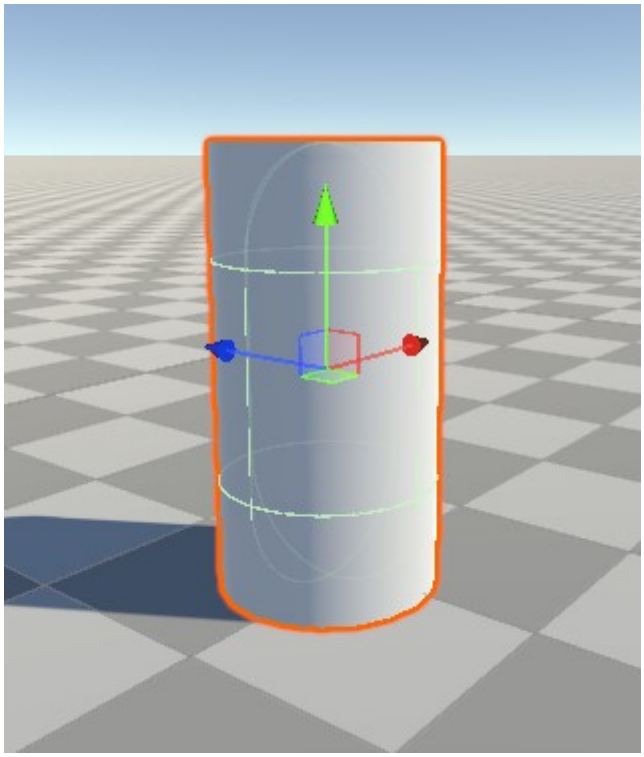
Annetaan luodille massa Unityn valmiilla fysiikkakomponentilla (Rigid Body). Luodaan myös valojuova, joka kulkee luodin perässä, jotta voidaan nähdä se paremmin pelimaailmassa (kuva 11). Asetetaan luodille, kuinka paljon numeraalista vahinkoa se tekee ja tarkistetaan joka framella, osuuko luoti kohteeseen. Voidaan tarkistaa osuma antamalla kohteena toimivalle peliobjekteille ”Enemy” ID:n. Jos saadaan osuma, vähennetään kohteelta elämäpisteitä ja tuhotaan luodin peliobjekti. Jos luoti ei osu kohteeseen, tuhotaan luoti aiemmin asetetun luodin eliniän mukaan.



Kuva 11. Valojuovat, auttavat pelaajia näkemään luodit paremmin. Unityn valmiilla fysiikoilla saadaan luodit kimpoamaan seiniltä.

7.3 Kohteet

Kohteet tässä projektissa ovat simppeleitä valkoisia sylintereitä (kuva 12), joille on asetettu hitbox tai ”collider” peliobjektien törmäyksen rekisteröimiseksi ja edellä mainittu ID tai ”tag”, jotta saadaan tieto osumasta. Viholliselle asetetaan elämäpisteet ja funktio nimeltä TakeDamage, joka vähentää elämäpisteitä. Samalla funktiolla tarkistetaan myös aina, jos kohteen elämäpisteet ovat nolla, jolloin tuhotaan peliobjekti. Tämä funktio suoritetaan joka kerta, kun osuma on havaittu.



Kuva 12. Kohde Unity-editorissa.

8 Toteutus – hitscan

Alkuasettelu on melkein täysin samanlainen kuin projektiiliaseessa. Ainoa ero on, että piipun päähän asetetaan tällä kertaa vain yksi tyhjä peliobjekti, edelleen ampumavälähdyksiä varten, mutta toinen korvataan partikkeliefektillä, jota käytetään visualisoimaan luodit valojuovilla.

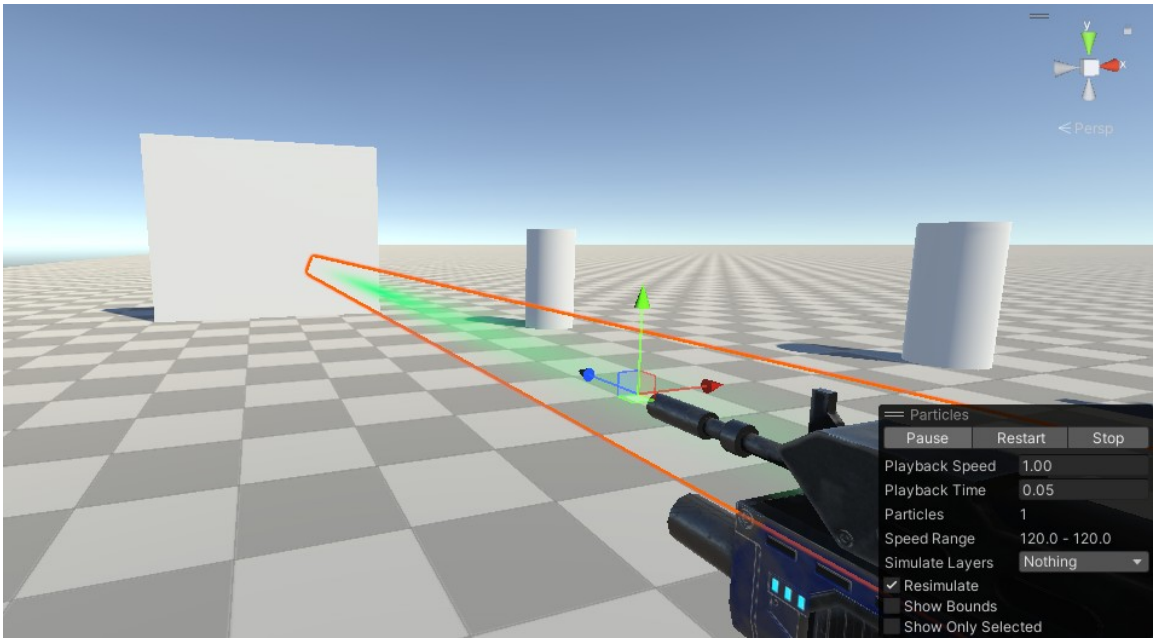
8.1 Toimivuus

Jälleen käydään läpi ensin muuttujat. Halutaan tietää, kuinka paljon vahinkoa jokainen osuma tulee tekemään ja mikä niiden etäisyys tulee olemaan, eli kuinka pitkälle osumat rekisteröidään. Samalla tavalla, kuin projektiiliaseessa asetetaan latausaika, ammusmäärä ja aina ennen ampumista tarkistetaan, onko latausanimaatio käynnissä.

Ampuessa vähennetään taas ammusmäärää yhdellä, mutta tällä kertaa ei luoda luotiobjektia. Käytetään Unityn valmista raycast-funktiota, joka katsoo, mihin pelikameran keskipiste osoittaa ampumahetkellä. Tarkistetaan osuma jälleen käyttämällä kohteisiin asetettua ”Enemy” ID:tä. Jos osuma on havaittu, suoritetaan taas kohde peliobjektissa sijaitsevan TakeDamage-funktion. Lataus on toteutettu samalla tavalla kuin projektiiliaseella.

8.2 Laserin visualisointi

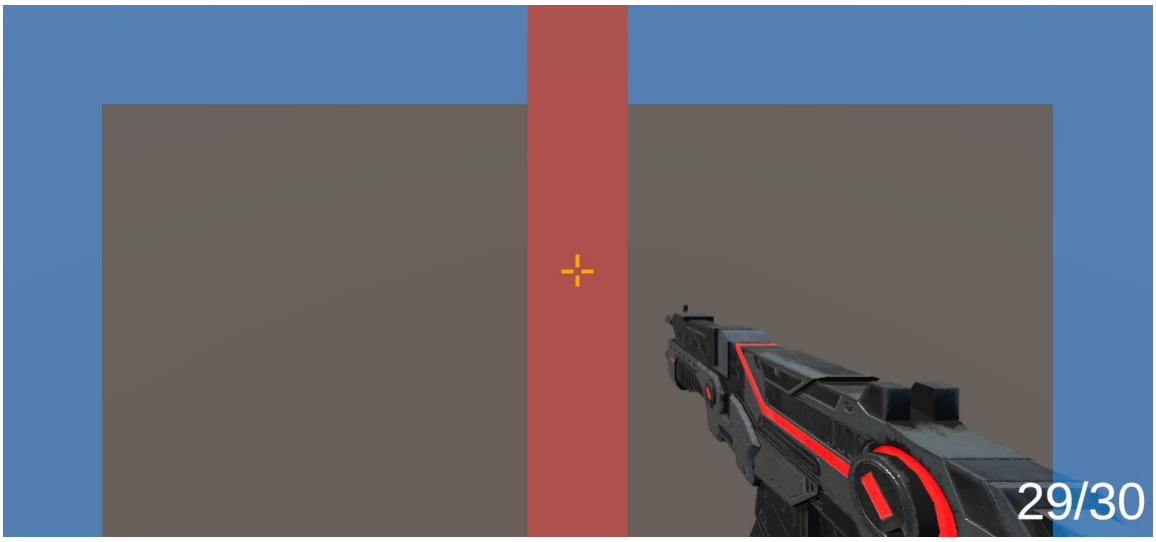
Koska hitscan-aseet eivät luo fyysisiä peliobjekteja luodeille, ei saada samanlaista visuaalista informaatiota ammuksen lentosuunnasta. Voidaan kuitenkin visualisoida ammukset Unityn partikkeliefekteillä. Luodaan valojuova, joka lähtee aseen piipusta ampumasuuntaan (kuva 13). Todellisesti hitscan-aseella osutaan kohteeseen samaan aikaan, kun valojuova ilmestyy peliruudulle. Siksi asetetaan valojuovan kulkunopeus korkealle, mutta sopivaksi, jotta pelaaja ehtii kuitenkin nähdä sen.



Kuva 13. Hitscan-aseen valojuova Unityn editorissa.

9 Toteutus – asettelu ja visuaalit

Noudattamalla teoriaosuudessa käytyjä sääntöjä asetetaan pelaajan ase ruudulle siten, että ruudun keskikohta jää tyhjäksi, eikä ase estä pelaajaa näkemästä, mitä hänen edessään on. Alla olevassa kuvassa (kuva 14) punaisella alueella merkattu ”Combat Corridor” on peliruudun tärkein alue ja sitä ei sovi peittää. Siniset reunat ovat myös syytä olla peittämättä mahdollisimman vähän siltä varalta, että pelaaja havaitsee sivusilmällä pelin sisäisiä uhkia, jolloin niihin voi reagoida.

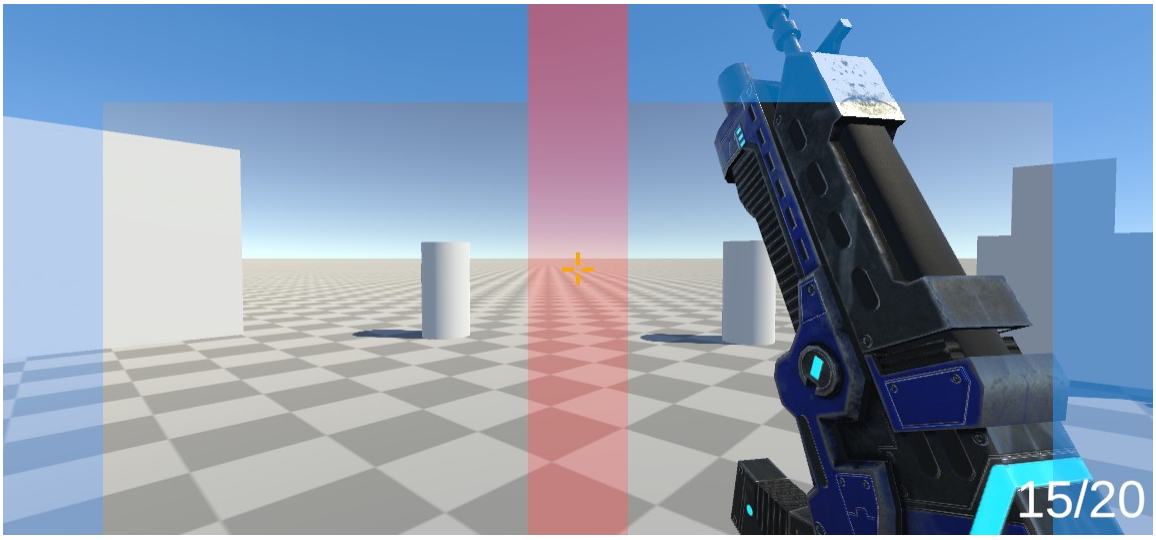


Kuva 14. Oikeaoppisesti peliruudulle asetettu pelaaja-ase.

9.1 Animaatio

Molemmille aseelle on kolme animaatiota. Ensimmäisenä on aseen oletusanimaatio, joka pyörii loputtomasti siihen asti, että joku muu animaatio aloitetaan. Sitten on tietysti ampuma-animaatio, joka visualisoi aseen rekyyliä. Tämä tekee aseesta uskottavamman ja antaa pelaajalle idean sen vahvuudesta. Ampuma-animaatio soitetaan joka kerta, kun ampumisfunktio suoritetaan, minkä jälkeen palaudutaan oletusanimaatioon, ellei pelaaja ammu uudestaan tai lataa asetta.

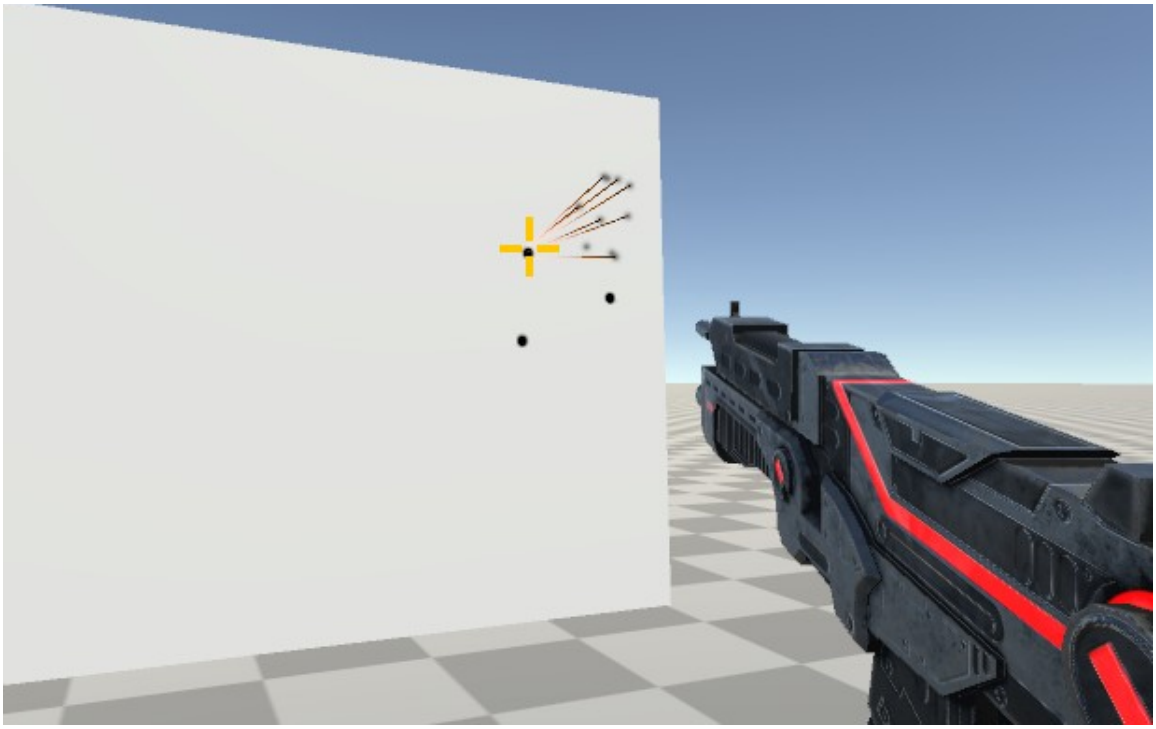
Viimeiseksi on latausanimaatio, jonka kanssa pitää olla kaikista varovaisin. Suunnitellaan latausanimaation siten, että peliruudulle kriittisiä alueita ei peitetä, vaan pitäydytään mahdollisimman paljon turva-alueille (kuva 15). Koska aikaisemmin asetettiin varmistin, joka estää pelaajaa ampumasta latauksen aikana, latausanimaatiota ei voi estää kesken kaiken. Latauksen jälkeen palaudutaan taas oletusanimaatioon. Riippuen pelistä aseelle voi myös luoda oletus animaation lisäksi kävelyanimaation sekä erillisen animaation tähtäämiseen. Kävelyanimaatiolla voidaan antaa pelaajalle idea aseen painosta. Tähtäysanimaatiota luotaessa on myös hyvä muistaa olla peittämättä mahdollisimman vähän kriittisiä alueita.



Kuva 15. Aseen latausanimaatio ei käytä peliruudulla kriittisiä alueita.

9.2 Efektit

Kun osuma on havaittu, voidaan ottaa tieto kohdasta, johon luoti tai raycast osui ja asettaa siihen visuaalinen efekti. Osuma voidaan tarkistaa jälleen ID:n avulla. Asetetaan testialueella sijaitsevalle seinälle myös oma ID, jolloin saadaan rekisteröityä osumat seiniin. Näytetään partikkeliefekti ja luodaan myös luodinreikä osuttuun kohtaan (kuva 16). Jotta pelin performanssi pysyisi taas koko ajan hyvänä, tuhotaan aina luodinreiät tietyn ajan kuluttua.



Kuva 16. Seinille ilmestyvä efekti visualisoi osuman.

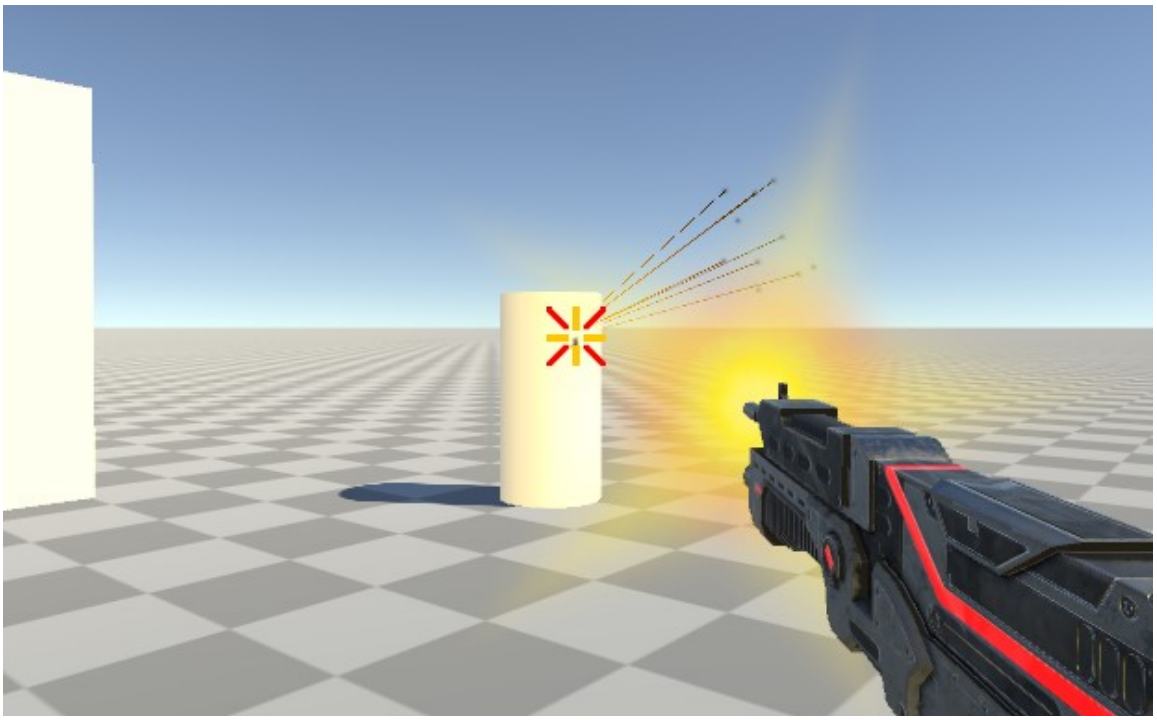
Ampumavälähdykset ovat tässä tapauksessa vain simppeli peliobjekti, jossa on .PNG-kuvatiedosto ja valoefekti. Ampuessa asetetaan välähdysobjekti edellä mainittuun aseen piipun päähän sijoitettuun tyhjään peliobjektiin. Annetaan ampumavälähdykselle hyvin lyhyt elinikä (tässä tapauksessa 0,05 sekuntia), minkä jälkeen se tuhotaan. Joka kerta, kun välähdys näytetään, asetetaan sille myös satunnainen rotaatio, jotta se näyttäisi luonnollisemmalta korkealla tulinopeudella. Näin luodaan välähdyksen näköinen efekti joka kerta, kun pelaaja käyttää asetta.

9.3 Indikaattorit

Keskellä peliruutua on aseen tähtäin, jolla tiedetään, mihin luodit tulevat tarkalleen ottaen osumaan. Samaan kohtaan peliruutua tehdään myös osumaa indikoiva visuaalinen efekti, joka on merkattu punasilla viivoilla (kuva 17). Oletuksena asetetaan

indikaattori näkymättömäksi. Aina kun peli rekisteröi osuman kohteeseen, jälleen käyttäen aiemmin asetettua ID:tä käyttäen, vaihdetaan indikaattori aktiiviseksi ruudulla.

Asetetaan indikaattorille aika, jolloin se on aktiivisena (0,05 sekuntia), jonka jälkeen se laitetaan taas pois näkyvistä. Indikaattorin näkyvyydelle annetaan lyhyt aika siltä varalta, että jos pelaaja saa lyhyessä ajassa monta osumaa, hän ehtii selkeästi nähdä, että jokainen niistä osui.

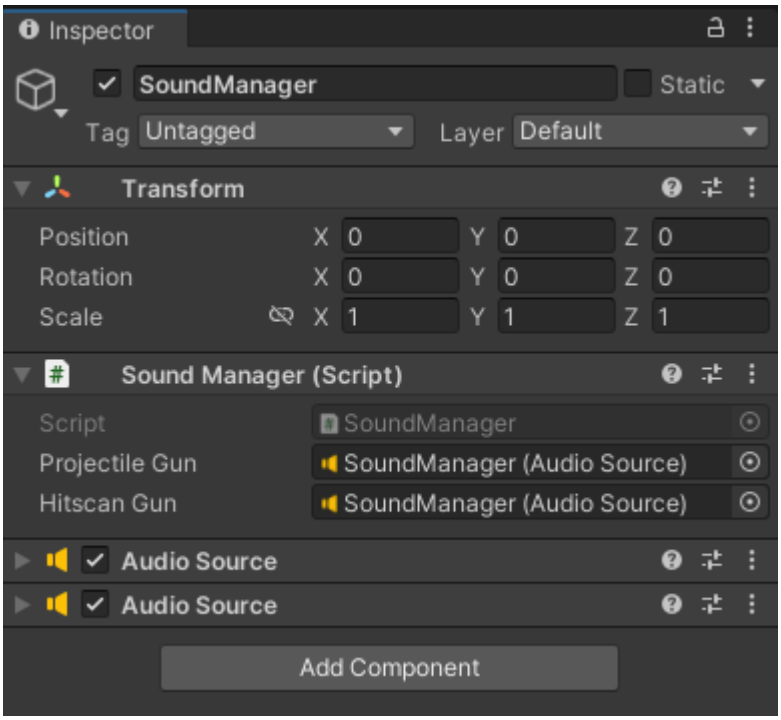


Kuva 17. Tähtäimen ympärille ilmestyvä punaisilla viivoilla visualisoitu indikaattori, kohteeseen osuessa.

Halutaan myös antaa pelaajalle informaatio siitä, kuinka paljon ammuksia hänellä on milläkin hetkellä lippaassa. Asetetaan peliruudun oikeaan alakulmaan indikaattori, joka näyttä numeraalisesti lippaan maksimimäärän ja tämänhetkisen ammusmäärän. Luodaan ammusmanageri, joka ottaa informaation ammusmääristä peliaseesta ja päivittää ne peliruudulla olevaan indikaattoriin joka kerta, kun pelaaja ampuu tai lataa aseen. Näin saadaan reaaliaikaisesti tieto ammustilanteesta.

10 Toteutus – äänet

Viimeiseksi käydään läpi äänten toteutus. Loin molemmille aseille ääniraidat ampumiseen ja lataamiseen sekä liipaisimen äänet siltä varalta, että pelaaja yrittää ampua, kun lipas on tyhjä. Luodaan äänimanageri, johon asetetaan audiolähteet molemmille aseille (kuva 18). Audiolähteet toimivat ikään kuin kanavina, joista voidaan soittaa ääniraitoja. Asetetaan ääniraidat peliaseisiin ja joka kerta, kun tarvitaan tietyn ääniefektin kuuluvan joko ampuessa tai ladatessa, kutsutaan sen aseen audiolähdettä ja soitetaan sitä kautta haluttu ääniraita.



Kuva 18. Äänimanageri Unity-editorissa.

11 Pohdinta

Lopputuloksena on kaksi täysin funktionaalista asetta toteutettu sekä hitscan että projektiilimetodeilla animaatioilla ja äänitehosteilla varustettuina. Aseita voi ampua ja ladata vapaasti yksinkertaisessa peliympäristössä, jossa on seiniä ja kohteita. Apuna oli kaikki teoriaosuutta varten tehty tutkinta, YouTube-tutoriaalit ja aiempi kouluprojektien aikana saatu kokemus. Käytännön osuuden tulos, oli loppujen lopuksi onnistunut hyvin, ja kaikki tärkeimmät ominaisuudet saatiin implementoitua.

Lisäanimaatioita olisi voinut tehdä aseen tähtäämiseen ja peliympäristössä liikkumiseen, mutta en kokenut niiden olevan välttämättömiä tätä opinnäytetyötä varten. Varsinkin aseen erillinen tähtääminen on yksilöllinen ominaisuus, eikä sitä kaikissa peleissä tarvita. Alun perin halusin kokeilla implementoida nämä aseet käyttäen myös Unreal- ja ehkä mahdollisesti Godot-pelimoottoreita ja nähdä erot niiden toteutuksissa, mutta koin, että se olisi tehnyt projektista liian suuren ja itseään toistavan.

Opin paljon tämän opinnäytetyön aikana. Unityn animaatio-ohjaimet tulivat tutummaksi, äänitehosteiden implementointi Unityn äänimanagerin avulla oli aiemmin tuntematon menetelmä ja projektiiliaseen toteutus tuli myös täysin uutena. Näytön jakaminen eri alueisiin peliaseen sijoittamista varten on hyvä periaate, mitä tulen varmasti käyttämään tulevaisuudessakin.

Aseiden kehittäminen ensimmäisen persoonan ammuntapeleissä vaikuttaa ideana todella yksinkertaiselta ja itsestään selvältä, mutta todellisuudessa prosessi on monivaiheinen ja se vaatii pelikehittäjiltä paljon enemmän kuin osaisi aluksi kuvitella. Ensimmäisen persoonan ammuntapeleissä aseet ovat ne todelliset päähenkilöt ja niiden toteutus usein määrittää pelin positiivisen tai negatiivisen vastaanoton.

Tässä opinnäytetyössä toteutetut aseet ovat suhteellisen yksinkertaisia, mutta niitä voi kehittää pidemmälle lisäämällä ominaisuuksia, kuten aiemmin mainittu tähtäysanimaatio. Erilaiset ampumamoodit, kuten sarjatuli ja puoliautomaatti tai kiinnitykset, kuten kranaatinheittimet, tekevät aseesta myös dynaamisemman. Näiden ominaisuuksien lisäys riippuu kuitenkin täysin siitä, minkälainen peli on kyseessä.

Lähteet

Alar, S. (2023) The Art of Sound: A Deep Dive into Sound Design in FPS and RPG Games
Saatavilla 11.2.2025
https://www.linkedin.com/pulse/art-sound-deep-dive-design-fps-rpg-games-semih-alar-ujumf

Couture, J. (2016) Why are so many devs employing a retro low-poly mid-1990s aesthetic?
Saatavilla 11.2.2025
https://www.gamedeveloper.com/art/why-are-so-many-devs-employing-a-retro-low-poly-mid-1990s-aesthetic-

Helsby, D. (2015) The art of Destiny's firs-person animation
Saatavilla 11.2.2025
https://gdcvault.com/play/1022297/The-Art-of-First-Person

Holstrom, J. (2021) Resident Evil and the Art of Good Sound Design
Saatavilla 11.2.2025
https://www.dreadxp.com/editorial/resident-evil-and-the-art-of-good-sound-design/

Jung, T. (2018) How do bullets work in video games?
Saatavilla 11.2.2025
https://medium.com/@3stan/how-do-bullets-work-in-video-games-d153f1e496a8

NeoFPSDocs (2019) Hitscan vs Projectiles
Saatavilla 11.2.2025
https://docs.neofps.com/manual/weapons-firearms-hitscan-projectiles.html

Qingxiang, H. (2024) Simple Article: Do Realistic Graphics Matter in Games?
Saatavilla 11.2.2025
https://redharegames.wordpress.com/2024/07/15/simple-article-do-realistic-graphics-matter-in-games/

Scott-Jones, R. (2020) Doom Eternal shows FPS games how to make guns feel like they hurt
Saatavilla 11.2.2025
https://www.pcgamesn.com/doom-eternal/weapons

Kuvalähteet

Helsby, D. (2015) The art of Destiny's firs-person animation [Kuvakaappaus]
Saatavilla 11.2.2025
https://gdcvault.com/play/1022298/The-Art-of-First-Person

Lahti, E. (2012) The origin of Call of Dutys most-heard sound [Kuvakaappaus]
Saatavilla 11.2.2025
https://www.pcgamer.com/the-origin-of-call-of-dutys-most-heard-sound/

ValhaHazred (2014) [Kuvakaappaus]
Saatavilla 11.2.2025
https://forums.warframe.com/topic/162227-so-happy-that-the-flak-cadrakgoon-exists/