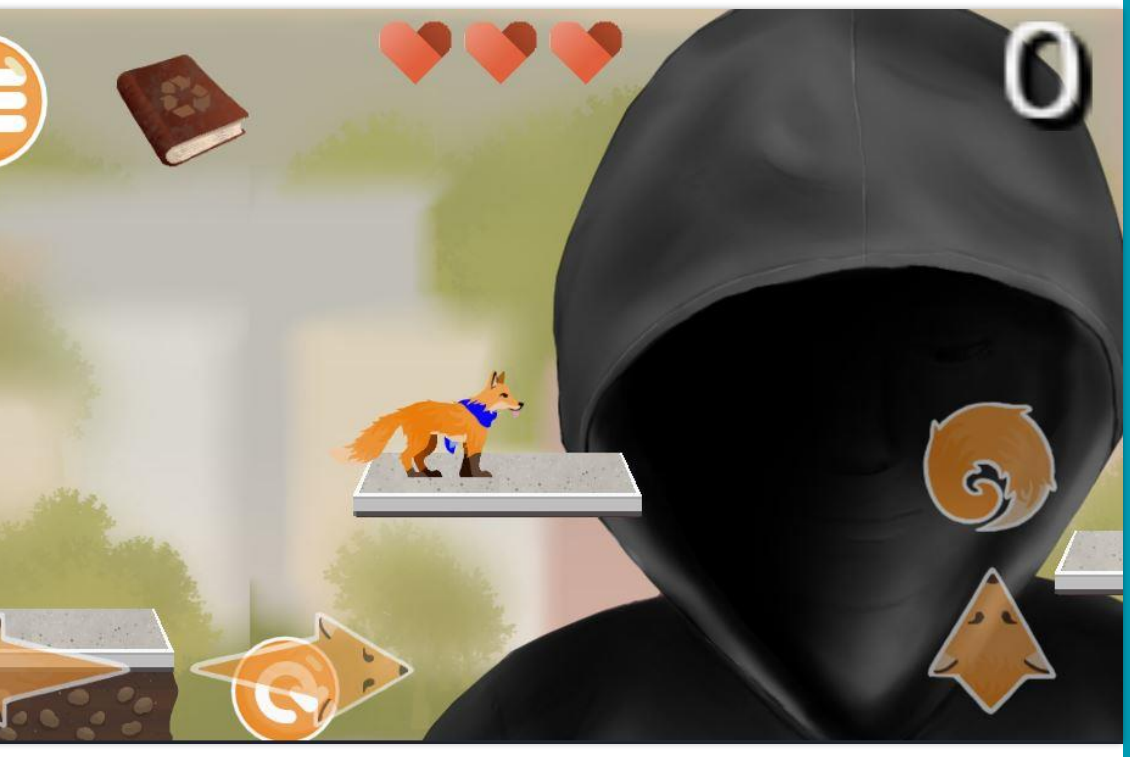
Nevala Laura

# Vastustajan suunnittelu 2D-tasohyppelypeliin



Insinööri (AMK)

Tieto- ja viestintätekniikka

Syksy 2020

KAMK • University of Applied Sciences

**Tiivistelmä**

**Tekijä:** Nevala Laura

**Työn nimi:** Vastustajan suunnittelu 2D-tasohyppelypeliin

**Tutkintonimike:** Insinööri (AMK), tieto- ja viestintätekniikka

**Asiasanat:** 2D, pelisuunnittelu, Godot Engine

Opinnäytetyö on jatkoa peliprojektille, jossa suunniteltiin ja toteutettiin 2D-peliä mobiililaitteilla pelattavaksi. Työn tarkoituksena oli suunnitella peliin sopiva vastustaja. Pelisuunnittelu on laaja ja mielenkiintoinen aihe. Teoriaa pystyy paljon soveltamaan eri peligenreissä, mutta on silti asioita, joita tulee ottaa huomioon omaan peligenreen liittyen. Tässä työssä pääpaino on 2D-tasohyppelypeleissä, vaikkakin käytetään esimerkkejä myös muista genreistä.

Aluksi käytiin läpi pelisuunnittelua yleisesti, keskittyen 2D-tasohyppelypeleihin. Sitten tutustuttiin erilaisten pelien vastustajiin ja niiden ominaisuuksiin. Vastustajat ovat tärkeässä roolissa luomassa onnistunutta pelikokemusta, sekä tärkeitä hahmoja pelien tarinoissa. Pelin päävastustajan voittaminen viimeistelee pelikokemuksen ja monesti jo pelin alusta lähtien pelaajaa valmistellaan taisteluun vastustajaa vastaan.

Vastustaja toteutettiin Godot Enginellä. Godot on ilmainen, kevyt, helppo oppia sekä sopii hyvin tämänkaltaisiin peliprojekteihin. Suunnittelutyö lähti liikkeelle alustamalla peliä, johon vastustaja suunniteltiin, sekä miettimällä minkälainen vastustaja sopisi pelin teemaan. Vastustaja toteutettiin pelin teeman mukaisesti sekä pohdittiin, minkälaisia haasteita työn edetessä kohdattiin ja mitä voisi parantaa.

**Abstract**

**Author:** Nevala Laura

**Title of the Publication:** Boss Design in 2D Platformer Game

**Degree Title:** Bachelor of Engineering, Information and communication technology

**Keywords:** 2D, game design, Godot Engine

This thesis is an extension to a 2D platformer mobile game project. The purpose of this thesis was to design a boss that will match the game. Game design itself is an interesting and wide topic. Theory of game design can be adapted in different genres of games but there are some things that have to be noted in a certain genre. The highlight of this thesis is in 2D platformer games but there are examples of other types of games, as well.

At first, game design in general is covered focusing on 2D platformers. Then, different types of game bosses and their features are introduced. Bosses are in an important role of creating a great gaming experience. They are also important characters in the game's story. Winning the last boss of the game finishes the experience for the player and many games are preparing the player to face the last boss from the very beginning of the game.

The boss was made with Godot Engine. Godot is free to use, lightweight, easy to learn and fits well for projects such as this. When starting to design the boss, basics of the game are introduced and thereafter planning and implementing the boss based on the game. Lastly, difficulties during the project and things that could have been done better are discussed.

Sisällys

**Symboliluettelo**

| | |
|---|---|
| **2D** | Kaksiulotteinen |
| **3D** | Kolmiulotteinen |
| **MMORPG** | Massively Multiplayer Online Role-Playing Game, massiivinen monen pelaajan verkkoroolipeli |
| **NPC** | Non-Playable Character, ei-pelattava hahmo |
| **UI** | User interface, käyttöliittymä |
| **VR** | Virtual Reality, virtuaalitodellisuus |

# 1 Johdanto

Opinnäytetyössä suunnitellaan vastustaja Litter Run mobiilipeliin, jonka pelimoottorina on käytetty Godot Engine 3.2 versiota. Aihe valikoitui kiinnostuksesta tekoälyä ja pelisuunnittelua kohtaan. Vastustaja on suunniteltava niin, että se teemallisesti sopii pelin ideaan ja samalla tarjoaisi uutta mielekästä sisältöä peliin.

Tasohyppelypelit ovat oma videopeligenrensä, joille ominaista on hyppiminen ja kiipeily pelimaailmassa. Osa haasteesta tasohyppelypeleistä tuleekin juuri siitä, että pelaaja osaa ohjata, eli hyppiä oikein epätasaiseksi suunnitelluissa pelikentissä.

Tasohyppelypelejäkin on kuitenkin erilaisia, jotkin vauhdikkaampia kuin toiset, ja joissakin enemmän hyppelyä ja kiipeilyä kuin toisissa. Peleissä on usein erilaisia vastustajia pelaajan haastamiseksi. Monia pelejä yhdistää se, että niissä on yksi tai useampi vastustaja, jotka pelaaja kohtaa, kun on pelannut peliä tarpeeksi pitkälle. Vastustajat ovat aina vaativampia kuin tavalliset viholliset pelikentissä. Vastustajien tarkoitus on testata pelaajan oppimia kykyjä sekä tuottaa onnistumisen tunne tarjoamalla haastetta pelin läpäisemiseksi.

Seuraavissa luvuissa käydään läpi yleisesti pelisuunnittelua keskittyen 2D-tasohyppelypeleihin, kerrotaan esimerkkien avulla, mitä vastustajat videopeleissä ovat ja minkälaisia asioita tulee ottaa huomioon vastustajaa suunnitellessa, sekä kerrotaan oma kokemus vastustajan suunnitteluprosessista.

# 2 Pelisuunnittelu 2D-tasohyppelypeleissä

Tämä luku käsittelee pelisuunnittelun perusteita, keskittyen 2D-tasohyppelypeleihin. Pelisuunnittelu on laaja aihe ja tässä on pyritty kertomaan pääkohtia pelisuunnittelun isoimmista aihekokonaisuuksista, kattaen pelaajahahmon, pelikenttien ja mekaanisen puolen suunnittelua.

## 2.1 Pelin perusmekaniikka

Tasohyppelypeleissä varmasti oleellisinta on se, miten pelaaja liikkuu. Miten nopeasti se juoksee ja miten korkealle pystyy hyppäämään. Näiden lisäksi pelissä usein on vielä lisäominaisuuksia, kuten kiipeily tai ponnahtelu. Tärkeää on myös tietää, pystyykö pelaaja hyppäämään toisen kerran vielä ilmasta ja liukuuko pelaaja eteenpäin vielä laskeutuessaan maahan hypystä. Pelaajan liikkumisen kannalta pelin fysiikat ovat tärkeässä roolissa, koska fysiikka on se, joka määrittää edellä mainitut pelisäännöt. Jotkin tasohyppelypelit käyttävät fysiikkaa haastaakseen pelaajaa enemmän kuin toiset, esimerkiksi pelaaja voi ottaa vahinkoa hypätessään liian kovalla vauhdilla seinään. Tuplahyppy on suosittu mekaniikka useassa tasohyppelypelissä, koska sen avulla voidaan hakea lisäkorkeutta hypylle sitä vaativissa paikoissa tai vaihtaa liikkeen suuntaa ilmassa. Tämä ominaisuus voikin olla ratkaisevassa asemassa siinä, miten hauskaa pelaaminen on. (Gamerforlife 2004-2020.)

Fysiikkaominaisuuksia, joita tasohyppelypeleissä lasketaan, ovat esimerkiksi painovoima, kitka, kiihtyvyys jne. Pelien fysiikoiden ei tarvitse olla täysin realistiset, eikä niiden tarvitse olla samanlaiset kuin muissa peleissä, vaan pelintekijällä on vapaus luoda pelistä omanlainen kokonaisuus. Huonosti toteutetut fysiikat kuitenkin tekevät pelistä helposti kömpelön tuntuisen.

Erilaisten liikkumistapojen lisäksi tasohyppelypeleissä pelaajalla voi olla jonkinlainen kyky puolustautua tai taistella vihollisia vastaan. Tyypillisesti pelaaja pystyy kukistamaan vihollisen hyppäämällä tämän päälle. Pelaajalla saattaa olla kyky myös ampua tai heittää jotain. Muita mekaniikkoja voivat olla esimerkiksi aikaraja, tavaroiden tai pisteiden kerääminen, aseen vaihtaminen, tähtääminen ampuessa tai heitettäessä, pelissä jonkin isomman esineen työntäminen tai raahaaminen ja/tai kokemuspisteiden kerääminen (Kramarzewski & De Nucci 2018, 110).

## 2.2 Tarina

Kaikissa peleissä ei ole kirjoitettua tarinaa. Tarina on kuitenkin loistava tapa kertoa pelaajalle, miksi pelissä tehdään jotain, miksi taistellaan tietynlaisia vihollisia vastaan, miksi ratkotaan tiettyä pulmaa tai etsitään tiettyjä esineitä. Tarinalla saadaan pelaaja uppoutumaan pelin maailmaan. Esimerkiksi roolipeleissä pelaaja tarvitsee tarinan eläytyäkseen peliin. Tasohyppelypeleissä tarina on usein hyvin minimaalinen, koska sellaisia pelejä harvemmin ostetaan tai pelataan tarinan vuoksi. Musiikilla voidaan tehostaa tarinaa luomalla siihen haluttua tunnelmaa. Pelin musiikki voi olla kenttäkohtainen, teemaan sopiva tai peilata hahmon tunteita. Musiikilla voidaan myös viestiä pelaajalle, jos aika on lopussa tai pelaaja on vaarassa. (Fudj 2020.)

Pelien tarinoissa pelaajalla on vastuu tarinan etenemisestä. Juoni menee eteenpäin peliä pelatessa, ja monessa pelissä pelaaja voi itse vaikuttaa tapahtumien kulkuun tekemällä valintoja. Peleihin on jopa tehty erilaisia vaihtoehtoja, miten tarina päättyy perustuen pelaajan tekemiin valintoihin pelin aikana. Tarinaa pelissä kerrotaan muun muassa lyhyillä videoilla, hahmojen välisillä keskusteluilla tai pelimaailmasta löytyvillä esineillä tai teksteillä. Pelin sisäiset videot ja keskustelut tulisi olla ohitettavissa, jos pelaaja ei jaksa katsoa niitä. Hyvä tarina on suunniteltu niin, että pelaajalla pysyy käsitys pelin tarinan kulusta vain pelaamalla, vaikka hän ohittaisi videot ja keskustelut. (Kramarzewski & De Nucci 2018, 188.)

Pelintekoprosessiin kuuluu suunnitelmien muuttuminen projektin edetessä. Välillä huomataan, että jotain pitää muuttaa tai parannella. Joskus käy niin, että kaikkea haluttua ei voida tai ehditä tehdä ja vaarana on, että koko pelin idea kärsii. Hyvänä esimerkkinä sellainen peli, jossa on kerronnallinen juoni, esimerkiksi ritarin pitäisi pelastaa linnanneito, mutta lohikäärmettä ei peliin ehdittykään tehdä. (Vuorela 2007, 69.)

## 2.3 Käyttöliittymäsuunnittelu

Onnistunut pelikokemus rakentuu monesta asiasta, ja ne kaikki on hiottava hyvin yhteen kokonaisuudeksi. Yhdenkin osa-alueen puutteellisuus saattaa vaikuttaa hyvinkin negatiivisesti loppukäyttäjän kokemukseen pelistä. Kaikki osa-alueet siis tuntuvat olevan juuri se yksi tärkeimmistä, kuten käyttöliittymän eli UI:n suunnittelukin. Käyttöliittymän tärkeys saattaa joskus unohtua, koska sitä ei välttämättä tule niin paljoa ajatelleeksi keskittyessään itse pelin suunnitteluun. Käyttöliittymän suunnittelu saattaa myös olla yllättäen yksi vaikeimmista alueista peliä tehdessä.

Käyttöliittymä viestittää pelaajalle pelin mekaniikan ja myös muiden aistien kautta välittyvän informaation, kyse ei siis ole pelkästään näytöllä näkyvistä nappuloista ja mittareista. Käyttöliittymäsuunnittelu on psykologiaa. Pelaajan halutaan uppoutuvan pelin maailmaan, informaatiota halutaan antaa tarpeeksi ja pelin komentamisen on oltava loogista. Suunnittelussa halutaan pyrkiä tehokkuuteen, mutta myös elämyksellisyyteen, koska pelien pelaaminen on vapaaehtoista toimintaa, jolloin siitä haetaan myös hyvää fiilistä. (Kemppainen 2019, 77.)

Käyttöliittymiä on erityylisiä, ja tärkeintä onkin löytää vaihtoehtojen joukosta sellainen ratkaisu, joka sopii parhaiten omaan peliin. Jotkin pelit on suunniteltu niin, että pelaamisen aikana näytöllä ei ole painikkeita tai pistemääriä näkyvillä, perustuen siihen, että pelaaja uppoutuu peliin helpommin. Kaikkia pelejä ei tietenkään ole mahdollista suunnitella tällä tavalla, jotkin pelit vaativat tietyt painikkeet, ja jotkin vaativat vielä enemmän, riippuen pelin monimutkaisuudesta. (Russel 2011.)

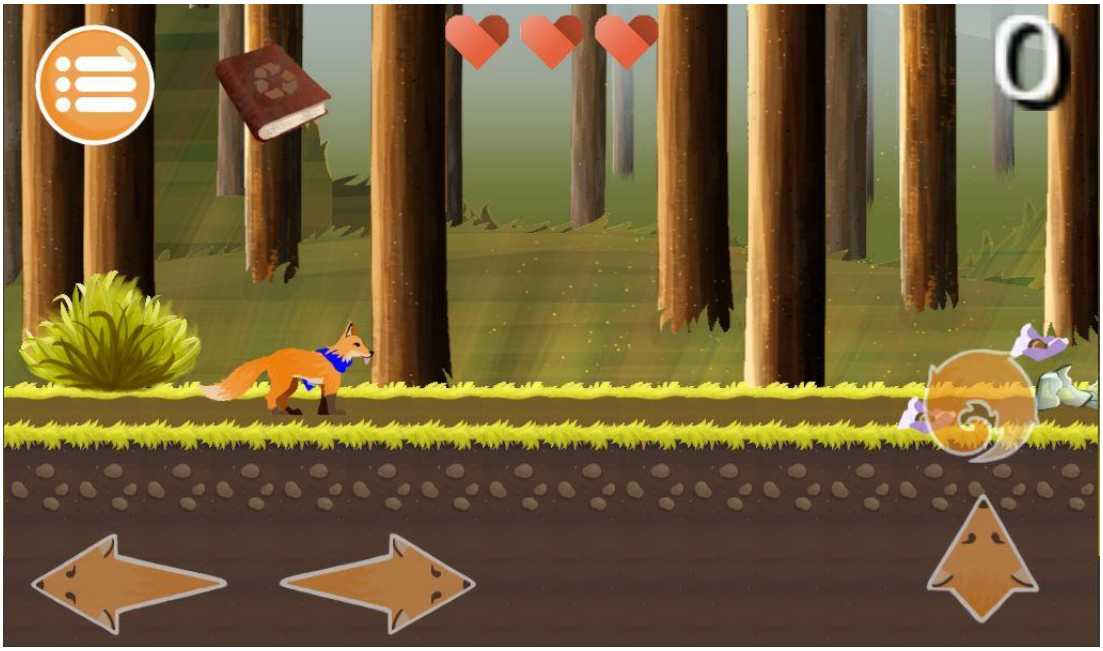
Kuvassa 1 esimerkki pelistä, jossa on jätetty käyttöliittymä tyhjäksi, ilman ohjeita pelaajalle. Joissakin kohdissa pelaaja saa pieniä ohjeita eteenpäin pääsemiseksi.


Kuva 1. Valiant Hearts: The Great War. Esimerkki 2D-pelistä, jonka UI on tyhjä. (Ubisoft Montpellier 2014.)


Kuva 2. Valiant Hearts: The Great War. Joissakin kohtia pelaajalle on asetettu pieniä ohjeita eteenpäin pääsemiseksi. (Ubisoft Montpellier 2014.)

Mobiilipeliä tehdessä on otettava huomioon näytön koko, jolla peliä tullaan pelaamaan. Koska älylaitteissa on suhteellisesti todella pieni näyttö, painikkeiden tulee olla tarpeeksi suuria, jotta niitä on helppo painaa sormella, ja muu informaatio tulee olla helposti luettavissa. Peleissä voi käyttää monella tapaa hyödyksi myös sormella pyyhkäisyä.

Käyttöliittymää suunnitellessa ensin mietitään, mitä ominaisuuksia ja kykyjä pelaajahahmolla on ja mitä tietoa pelaajalle halutaan välittää. Tasohyppelypeleissä sellaisia ovat ainakin liikkuminen eteen ja taakse, hyppy ja ehkä jonkinlainen isku. Sitten mietitään, halutaanko esimerkiksi pelaajan terveyspisteet jollain tavalla näkyviin ruudulle, entä pistemäärä, tai onko joitain muita pelaajalle hyödyllisiä asioita, kuten valikon aukaisu. Kaikkea mieleen tulevaa ei ehkä kannata laittaa näkyviin, koska käyttöliittymä halutaan pitää mahdollisimman yksinkertaisena ja selkeänä. Tulisi pohtia, mitkä ovat pelin pelaamisen kannalta välttämättömimmät, jotka halutaan asetella näkyviin ja mikä olisi loogisin toteutustapa.



Kuva 3. Litter Run. Esimerkki UI:sta, jossa pelaajalle annetaan paljon informaatiota, kuten HP, pisteet ja kontrollit. (Nevala, Saajanne & Vikstedt 2020.)

## 2.4 Kontrollien ja kameramenetelmän suunnittelu

Tasohyppelypelejä ajatellen kaikista tärkeimmät kontrollit ovat pelaajahahmon liikkuminen ja hyppiminen. Niiden on tunnuttava mahdollisimman hyviltä. Liikkeen on oltava sulavaa ja realistisen tuntuista, ja kun pelaaja painaa kontrollia, tapahtuu heti haluttu toiminto ilman viivettä.

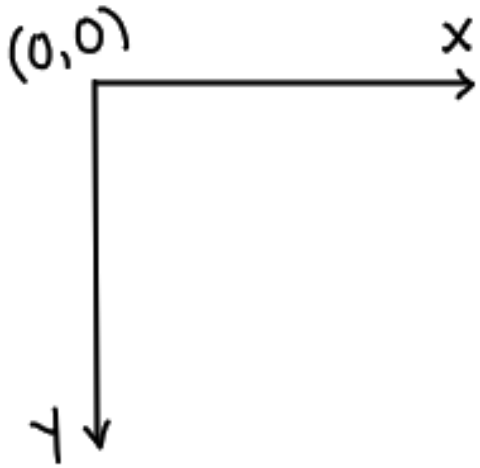
Hahmon liikuttamisen pitäisi olla hauskaa ja mukaansatempaavaa. Sen on myös näytettävä hyvältä, mutta ei kontrollin toiminnallisuuden kustannuksella. (Playtonic Games team 2015.)

Kontrolleja suunnitellessa vastataan kysymykseen, mitä toimintoja halutaan pelaajan osaavan ja miten pelaaja voi vaikuttaa pelin tapahtumiin. Kaikki sellainen pitää päättää etukäteen ja ohjelmoida mukaan, koska tietokone ymmärtää vain ehdottomia sääntöjä. (Vuorela 2007, 59.)

Juoksun nopeuden ja hypyn korkeuden määrittäminen riippuu siitä, minkälaiseen maastoon peli sijoittuu. Juoksuun saadaan realistista tunnetta lisäämällä juoksuvauhdille pieni kiihtyvyys. 2D-pelit ajatellaan kaksiulotteiseen koordinaatistoon, joka tarkoittaa x- ja y-akseleita. Kolmiulotteiseen peliin tulisi mukaan myös z-akseli. 2D-peleissä juoksu tapahtuu vain x-akselin suuntaisesti, joko positiiviseen tai negatiiviseen suuntaan. Kun pelaaja painaa nappia, jolla on tarkoitus liikkua

eteenpäin oikealle, eli positiiviseen suuntaan, pelaajalle lisätään arvoa x-akselin suuntaiselle vektorille. Taaksepäin eli negatiiviseen suuntaan mentäessä vähennetään puolestaan arvoa x-akselin suuntaiselta vektorilta.

Joissakin pelimoottoreissa ja ohjelmointirajapinnoissa y-akselin arvo kasvaa koordinaatistoa alaspäin kuljettaessa. Koordinaatiston origo on tällöin vasemmassa yläkulmassa. Tässä opinnäytetyössä perehdytään Godot-pelimoottoriin ja siinä koordinaatisto toimii tällä tavalla.




Kuva 4. Kaksiulotteinen koordinaatisto.

Pelaajan hyppyä ohjelmoitaessa on tärkeää tietää, mihin suuntaan y:n arvo kasvaa. Silloin kun koordinaatisto toimii näin ja pelaaja haluaa hypätä pelissä, vähennetään arvoa y-akselin suuntaiselta vektorilta. Se, miten korkealle halutaan hypätä, määrittää sen, paljonko arvoa vähennetään. Hypyn ohjelmoinnissa otetaan huomioon myös pelaajahahmon painovoima, jos halutaan, että pelaaja laskeutuu takaisin maan pinnalle.

Tasohyppelypeleissä pelaajalla on yleensä myös määrätty jokin painike lyömiseen, ampumiseen tai muuhun pelimaailman kanssa vuorovaikuttamiseen, esimerkiksi oven avaamiseen.

Peliin liitetään kamera, jotta peli näkyisi ruudulla. Pelimoottoreissa on kameraobjekti sisäänrakennettuna, joka ei tarvitse kuin liittää peliin mukaan. Kameratyyppejä on peleissä erilaisia, esimerkiksi staattinen eli aina paikallaan pysyvä kamera, manuaalinen kamera, jota pelaaja voi itse liikuttaa haluamaansa suuntaan tai pelaajan mukana liikkuva kamera, joka voi myös olla sijoitettu eri kuvakulmiin. Vaihtoehtoja on siis monia. (Kramarzewski & De Nucci 2018, 267-275.) Kameran kuvaamaan alueeseen kannattaa kiinnittää huomiota, koska pelaajan näkökulmasta on turhauttavaa, jos kameran alue on liian pieni, kuvakulma on huono ja pelaaminen vaikeutuu

näistä syistä (Gamerforlife 2004-2020). Tasohyppelypeleissä tyypillistä on kamera, joka seuraa pelaajaa kaksiulotteisesti sivulta kuvattuna. Kamera on siis liitetty pelaajaan, ja se liikkuu pelaajan mukana koko ajan.

## 2.5 Kenttäsuunnittelu

Erityisesti kenttäsuunnittelu on valtava tehtävä pelisuunnittelussa, koska pelin hauskuus ja koukuttavuus on paljon siitä kiinni. Peli käy pian tylsäksi, jos jokainen kenttä toistaa itseään. (Khalifa 2019.) Hauskoissa kentissä on yllätyksellisyyttä, tutkittavia paikkoja, piilotettuja esineitä tai vaikka arvoituksia. Kuitenkin on hyvä muistaa, että kenttä olisi kutakuinkin suoraviivainen, ettei pelaaja eksy sinne. Tunnelman tappaa helposti se, että ei tiedä minne mennä ja mitä tehdä seuraavaksi. Pelaajaa voi myös ohjata oikeaan suuntaan asettamalla maamerkkejä, vinkkejä, tai pisteitä polulle, johon kuuluu mennä. Pisteiden avulla voi myös yrittää houkutella pelaajaa menemään tutkimaan jotain hankalasti mentävää paikkaa. (Playtonic Games team 2015.)

Tasohyppelypeleille ominaista on kentissä olevat esteet ja vaaran paikat. Sellaisia voivat olla esimerkiksi syvät kuilut, joihin pudotessa joutuu aloittamaan alusta. Sitten on erilaisia vahingoittavia ansoja, vieriviä kivenjärkäleitä, tulta, vettä ym. Tasohyppelypeleissä vesi usein on vaarallista, pelaajalla ei ole uimataitoa ja kuolee osuessaan veteen. Pelikenttä itsessään voi olla pelaajan pahin vihollinen. (Gamerforlife 2004-2020.)

Peleissä on yleensä jonkinlainen pistejärjestelmä. Se voi olla aikaan sidonnainen, esimerkiksi mitä nopeammin kentän suorittaa, sitä paremmat pisteet saa. Tai perinteisesti, kenttään on sijoitettu pelaajalle kerättäväksi pisteitä. Monessa pelissä on myös piilotettuja tai harvinaisempia kerättäviä esineitä pelaajalle. Sitten kun pelaaja kerää kaikki pisteet tai löytää kaikki harvinaiset esineet, luvassa saattaa olla lisäpalkintoja. Pelin tulee olla jollain tavalla palkitseva pelaajalle.

Youtube-kanavan Game Dev Underground videossa ”My Level Design Philosophy + Tips For Designing Levels”, henkilö kertoo omasta kokemuksestaan pelisuunnittelijana, minkälaista taktiikkaa käyttää suunnitellessaan kenttiä omaan peliinsä. Hän kertoo huomanneensa, että kentästä tulee parempi, kun hän nimeää kentän etukäteen ja alkaa sitten rakentamaan kenttää sen tietyn nimen ja teeman ympärille. (Game Dev Underground 2017.)

Kun pelin idea ja teema on päätetty, voidaan alkaa luonnostelemaan kenttiä. Alkuun pääsemiseksi voi tehdä listan, mitä kaikkea haluaa pelaajalle tapahtuvan tietyssä kentässä. Listaan sisällytetään myös mahdolliset kerättävät esineet tai pisteet. Lista voi olla esimerkiksi tällainen:

- viholliset (minkälaisia ja miten paljon?)
- löydä ensimmäinen tehoste
- opi uusi kyky (esim. tuplahyppy)
- kohtaa ensimmäinen tuhottava seinä
- löydä ensimmäinen lukittu ovi
- löydä avain
- opi vetämään vivusta saadaksesi tasot liikkumaan

Kentän pohjapiirustuksesta tehdään karkea luonnos, johon lisätään listattuja asioita ja esineitä paikoilleen. Luonnosta voi sitten alkaa kehittämään paremmaksi ja yksityiskohtaisemmaksi. Kun luonnoksesta päästään rakentamaan oikea pelattava kenttä tulevaan peliin, on tärkeää testata sitä. Onko se kiva, tylsä, pitääkö jotain siirtää, lisätä tai vähentää. Tärkeä huomio on se, että useampia tai lopullisia kenttiä kannattaa suunnitella vasta, kun pelin perusmekaniikat on tehty ja testattu hyviksi, koska jos esimerkiksi hypyn korkeutta menee jälkeenpäin muuttamaan, joutuu mahdollisesti muuttaa kenttiä uudelleen sopiviksi uusille mekaniikoille. Kun kaikki pelin toiminnan kannalta kaikki tärkeä on luotu, kenttään lisätään pelin teeman mukaiset yksityiskohdat, koristeet, musiikki ja äänitehosteet. (Jonkers 2011.)

## 2.6 Hahmosuunnittelu

Peleissä hahmoja on sekä pelattavia että sellaisia, joita pelaaja ei pysty kontrolloimaan, eli NPC-hahmoja (Non-Playable Characters). Usein jopa pelin kannalta tärkein hahmo ei olekaan itse pelaajahahmo, vaan joku pelaajan pelimaailmassa kohtaama hahmo. NPC:t ovat hahmoja, jotka asuttavat pelimaailman. Pelaaja ei voi kontrolloida niitä, vaan ne ovat ohjelmoitu tekemään pelissä omia asioitaan, tai niillä voi olla jonkinlainen tekoälykin. Jotkut ovat sellaisia, kenen kanssa pelaaja voi kommunikoida, ja jotkut saattavat vain olla taustalla. Monessa pelissä on NPC, jolla on pelin kannalta suuri merkitys. Se voi olla pelaajan apuri, ystävä, rakastaja tai mentori. (Kramar-

zewski & De Nucci 2018, 235-237.) Tällainen hahmo pelissä voi olla ratkaiseva tekijä siinä mielessä, että peli tai sen tarina jää pelaajan mieleen, koska pelaajalla kehittyy myötätuntoa tai tunneside pelin hahmoon, samalla tavalla kuin TV-sarjojen tai elokuvien kohdalla.

Videopeleissä viholliset toimivat tekoälyohjelmointinsa mukaisesti pelaajan etuja vastaan (Vuorela 2007, 60). Vihollishahmot haluavat estää pelaajaa pääsemästä tavoitteeseensa. Vihollisen tehtävä on vaikeuttaa, hidastaa tai jopa tappaa pelaaja. Vihollisen ei ole pakko olla verenhimoinen, se saattaa olla myös sellainen, joka vain haluaa seisoa pelaajan tiellä ja pelaajan on keksittävä keino ohittaa se. (Kramarzewski & De Nucci 2018, 236.) Tasohyppelypeleissä suosittuja vihollisia ovat sellaiset, jotka kuljeksivat omaa reittiään, ovat pelaajan esteenä, pelaajaa ampuvat tai seuraavat otukset. Viholliset tekevät pelistä viihdyttävän, niiden on oltava haastavia mutta hauskoja (Kramarzewski & De Nucci 2018, 259).

Joissakin peleissä pelaaja kontrolloi montaa yksikköä (strategiapelit), mutta useissa peleissä kontrolloidaan pelin päähenkilöä tai sankaria.

Hahmosuunnittelussa on tärkeää muistaa yksinkertaisuus. Hahmon halutaan jäävän helposti ihmisten mieleen. Selkeä ja yksinkertainen hahmo rakentuu selkeästä siluetista, väriteemasta ja pienestä liioittelusta. Siluetilla tarkoitetaan sitä, että jos hahmon muuttaisi kokonaan mustaksi, miltä sen ääriviivat näyttäisivät. Väriteema pidetään yksinkertaisena, valitaan yksi pääväri sekä tehostevärejä mutta muut värit eivät saa kilpailla päävärin kanssa. Liioittelulla tarkoitetaan sitä, että hahmon jotakin ruumiinosaa tuodaan enemmän esille tai lisätään hahmolle jokin asuste, esine tai ase, joka erottuu hahmon siluetista. Ajatus on se, että hahmo olisi helposti tunnistettavissa pelkästään näkemällä hahmon siluetin tai väripaletin. Muodoilla ja väreillä voi tuoda esille myös hahmon luonnetta. Esimerkiksi pyöreät ja pehmeät muodot yhdistetään iloisiin ja ystävällisiin hahmoihin, kun taas terävät ja kulmikkaat muodot ovat ilkeitä tai pahoja. Hahmon mielialalla, asennolla ja vartalon muodolla on myös tärkeä rooli hahmon tarinankerronnassa. (BaM Animation 2020.) Esimerkiksi animaatioelokuvassa Inside Out – Mielen sopukoissa (Docter 2015) värit ovat selkeästi yhdistetty hahmojen luonteisiin.

Kuva 5. Väreillä korostetaan hahmojen luonteita elokuvassa Inside Out – Mielen sopukoissa (Docter 2015).

Vastustaja lukeutuu myös pelin yhdeksi tärkeäksi hahmoksi. Se on eri asia kuin tavallinen vihollinen ja sen suunnitteluun perehdytään tarkemmin seuraavassa luvussa.

# 3 Vastustajahahmon suunnittelu

Tässä kappaleessa kerrotaan mitä vastustajalla tarkoitetaan ja minkälaisiin vastustajiin erilaisissa peleissä törmää. Käydään läpi, minkälaisia asioita yleensä otetaan huomioon vastustajaa suunnitellessa, mikä tekee taistelusta hyvän ja mielenkiintoisen sekä minkälaiset asiat saattavat vaikuttaa pelikokemukseen negatiivisesti.

## 3.1 Vastustajat videopeleissä

Ensimmäinen vastustaja eli pelikielellä ”bossi” (boss engl.) ilmestyi videopeliin vuonna 1975. Peli oli dnd, joka perustui vuonna 1974 julkaistuun Dungeons & Dragons lautaroolipeliin. Dnd:n pelialustana toimi PLATO-tietokone. Kyseessä oli tekstipohjainen roolipeli, jossa oli vain vähän animointia. Vastustaja, Golden Dragon, suunniteltiin peliin, jotta pelissä eteneminen ei olisi liian helppoa. Tällä vastustajalla oli jo samat ominaisuudet, mitä nykypäivän vastustajillakin. Se oli huomattavasti kestävämpi kuin muut pelin viholliset. Golden Dragonilla kasattiin jännitystä pelin edetessä kertomalla sen vartioimasta aarteesta. Kun pelaaja oli kerännyt tarpeeksi kokemusta ja varusteita, hän pystyi haastamaan Golden Dragonin. Tämä lohikäärme toimi pelin viimeisenä haasteena, jonka voitosta sai palkinnoksi kaikkien tavoitteleman aarteen. (Agriogianis 2018, 5; Lee 2015.)

Vastustaja on yleensä pelin haastavin paikka pelaajalle. Monessa pelissä on niin, että pelaaja kohtaa useita välivastustajia tietyissä paikoissa peliä ja pelin viimeisenä haasteena on päävastustaja, joka on kaikista vaikein voittaa. Kun pelaaja on päihittänyt pelin viimeisen vastustajan ja peli on pelattu läpi, tyytyväisyys voitosta riittää palkinnoksi ja pelaaja tuntee olevansa yhden saavutuksen rikkaampi.

Vastustajatkin, kuten tavalliset pienemmätkin viholliset pelissä, tarjoavat haastetta pelaajalle ja yrittävät estää pelaajaa etenemästä pelissä. Vastustaja on tavallista vihollista huomattavasti haastavampi ja pelottavampi. Voittaakseen vastustajan, pelaaja haastetaan käyttämään jotain jo opittua taitoa. Välivastustajat ovat ikään kuin etappeja matkalla päävastustajan luo ja ne yleensä odottavat pelaajaa tiettyjen alueiden tai tasojen lopussa. Esimerkkinä peli Spyro: A Hero’s Tail, jossa pelin kulku ja vastustajat ovat suunniteltu tällä tavalla: pelissä on eri teemaisia alueita ja kenttiä, ja alueen lopussa odottaa teemaan sopiva vastustaja. Viimeisenä pelaaja kohtaa pelin päävastustajan, joka on pelin aikana esiintynyt uhkailemassa Spyroa aina kun tämä on voittanut

edellisen haasteen. (Eurocom 2004.) Vastustajat näyttävät uhkaavilta ja niille on animoitu pelottelevia eleitä näyttääkseen mahtavilta ja voittamattomilta. Vastustajat pyrkivät herättämään pelaajassa tunteita, kuten pelkoa ja jännitystä. Joku tietty vastustaja saattaakin olla juuri se asia, joka pelistä on jäänyt mieleen pitkäksi aikaa. (Kramarzewski & De Nucci 2018, 239.)

### 3.1.1 Tunnettuja pelejä ja vastustajia eri pelityyleissä

Dark Soulsit ovat teemaltaan synkkiä toimintapelejä. Peleissä keskitytään melkein kokonaan vastustajien kanssa taisteluun ja niissä kuljetaan vastustajalta vastustajalle. Dark Souls-peleissä seikkaillaan 3D-maailmassa taistellen erilaisia hirviömäisiä olentoja vastaan. Pelaaja voi löytää itselleen panssareita ja aseita tutkiessaan maailmaa sekä hänellä on myös käytössään taikoja ja parannusrohdoksia. Pelissä tunnelma yksistään on jo pelottava ja vastustajat ovat usein isokokoisia mielikuvituksellisia fantasiaolentoja. Dark Soulsit ovat tunnettuja vaikeustasostaan, joka koettelee pelaajan kärsivällisyyttä. Taisteluita joutuu usein harjoittelemaan kerta toisensa jälkeen ja viime metreillä kuoleminen on pelaajalle hermoja raastavaa. Vastustajat ovat kuitenkin voitettavissa ennen pitkää. Niitä pystyy oppia lukemaan, milloin ne ovat tekemässä kuolettavan iskunsa, jolloin pelaajalla on mahdollisuus väistää tai milloin pelaajalla on hyvä paikka iskeä. (FromSoftware 2011.)

Pelisarjan yksi vaikeimmista vastustajista on Nameless King (Dark Souls 3), Vulture.com artikkelissa ”The 100 Hardest Video-Game Bosses, Ranked By Difficulty” kyseinen vastustaja on päässyt sijalle kolme. Taistelussa on kaksi eri vaihetta (engl. phase), ensimmäisessä vaiheessa taistelun alkaessa Nameless King lentää lohikäärmeen selässä ja yrittää sieltä lyödä pelaajaa jättimäisellä keihäällään samalla kun hänen lohikäärmeensä syöksee tulta pelaajaa kohti. Kun pelaaja luulee voittaneensa, alkaakin seuraava vaihe. Vain lohikäärme onkin poissa pelistä ja pelaajan on vielä kukistettava itse Kuningas vastakkain taistelussa. (Vulture 2020.)

Toinen samalla idealla oleva peli, jossa keskitytään vastustajien päihittämiseen, on Cuphead. Vaikka Dark Souls ja Cuphead ovat täysin eri näköisiä ja erilaisia pelejä, kummassakin vaikeustaso on astetta haastavampi ja niissä liikutaan vastustajalta toiselle. Paljon palkintoja voittanut Cuphead on vanhaan sarjakuvatyyliin piirretty 2D-toimintapeli. Cupheadin yhtenä myyntivalttina toimii sen hieno ulkoasu, käsin piirretyt 1930-luvun tyyliset sarjakuvamaiset grafiikat, vesiväreillä maalatut taustakuvat sekä teemaan ja ajan henkeen sopiva jazz-musiikki.

Cupheadissakin pelaajalta vaaditaan kärsivällisyyttä opetella kunkin vastustajan mekaniikat, koska jokainen vastustaja vaatii myös useamman yrityskerran. Vastustajissa on aina useampi vaihe ja taistelut vaikeutuvat vaihe vaiheelta. Taistelut eivät ole onneksi kovin pitkiä, vaikka pelaaja häviäisi juuri ennen voittoa, on nopeaa aloittaa alusta ja pelata takaisin siihen kohtaan missä aiemmin kuoli. Pelaaja saattaa kokea vahvaa turhautumisen tunnetta mutta voittamisen tuottama hyvänolon tunne tekee siitä kaiken kärsimyksen arvoista. Pelaaja pystyy ostamaan itselleen erilaisia aseita ja kykyjä helpottaakseen taisteluita. (Minotti 2017.) Vastustajat Cupheadissa ovat monen näköisiä ja mielikuvituksellisia. Ilkeitä vihanneksia kasvimaalla, lohikäärme, lampunhenki ym., melkein kaikilla ilkikurinen hymy kasvoillaan. Vastustajille ominaista on myös muodon muuttaminen aina kun seuraava vaihe taistelussa alkaa. Taisteluissa pelaajan on väisteltävä suoraan vastustajalta tulevia iskuja, ammuksia ja kaikkea muuta mitä vastustaja heittelee pelaajaa kohti. Haastavia taisteluista tekee yleensä se, että ruudulla on välillä samanaikaisesti monta erilaista väistettävää asiaa.


Kuva 6. Cala Maria vastustaja pelissä Cuphead (Studio MDHR 2017).

MMORPG eli massiivinen monen pelaajan verkkoroolipeli, World of Warcraft, sisältää paljon erilaisia ja eri vaikeustasoisia vastustajia. Vastustajia pelissä on niin yksin voitettavaksi, kuin 40 pelaajan voimin kaadettavaksi. World of Warcraft eli lyhyemmin WoW, on fantasiamaailmaan sijoittuva roolipeli, siinä on monimuotoinen tarina ja pitkä historia. Pelissä edetään vahvasti tarinan mukaan ja tarinan pahikset ovat vastustajia pelaajille. WoWiin on tänä vuonna tulossa jo kahdek-

sas lisäosa ja jokaisessa lisäosassa, kuten peruspelissäkin, on omat päävastustajansa ja paljon välivastustajia. Tarina on aina jatkunut seuraavassa lisäosassa siitä, kun edellisen lisäosan päävastustaja on voitettu.

WoWissa muodostetaan ryhmiä muiden pelaajien kanssa ja tavoitteena on hyökätä luolastoihin ja voittaa siellä asuvat vastustajat. Helpoimmat luolastot voitetaan viiden pelaajan ryhmällä ja vaikeimpiin mahtuu jopa 40 pelaajaa samanaikaisesti. Pelaajilta vaaditaan yhteistyökykyä ja vaikeammat vastustajat vaativat hieman taktiikoiden harjoittelua. (Blizzard Entertainment 2004.) Vulture.com vaikeimpien vastustajien listalla sijalla kaksi on yksi WoWin alkuperäispelin 40 pelaajan luolaston vastustaja, C'Thun. C'Thunin lonkerot ja silmistä tuleva säde tekivät selvää jälkeä porukasta mutta se ei yksin riittänyt tälle otukselle, se myös napsi pelaajia ympäriltään kitaansa säännöllisesti. (Vulture 2020.) Monella WoWin vastustajista toistuu samankaltaiset taktiikat, että jotain pitää aina väistää. Joillakin vastustajilla on joitain erikoisempia mekaniikkoja, jotka vaativat etukäteen suunnittelua ja sopimista pelaajien kesken. Lisäosassa Wrath of the Lich King eräs vastustaja, Blood-Queen Lana'thel, puraisi yhtä pelaajaa ja pelaaja muuttui vampyyriksi, jos tämä ei puraissut toista pelaajaa ajoissa. Jos pelaaja muuttui vampyyriksi, hän hyökkäsi tovereitaan vastaan eikä tähän ollut parannuskeinoa.

Pokémon peleissä tarkoitus on kerätä itselleen kaikki maailman Pokémonit. Pokémonin voi yrittää voittaa itselleen taistelemalla sitä vastaan ensin. Halutuin ja vaikein Pokémon versioissa Red, Blue ja Yellow oli Mewtwo. Sen nappaaminen oli koko Pokémon kouluttajan uran kohokohta. Pelaaja saattoi taistella Mewtwota vastaan aiemmin kerätyillä ja koulutetuilla Pokémoneillansa, mutta Mewtwo oli voimakas ja haastava kukistaa, koska se pystyi parantamaan itseään ja iskeä pelaajan Pokémoneihin paljon vahinkoa kerralla. (Vulture 2020.)

Iso kilpikonna nimeltä Bowser haastaa Mariota yhä uudelleen Super Mario pelistä toiseen. Bowser on kaapannut prinsessan ja Mario yrittää pelastaa hänet. Bowser esiintyi ensimmäisen kerran 2D-tasohyppelypelissä Super Mario Bros. vuonna 1985 (Wikipedia 2020). Itse taistelu ei ollut kovin vaikea, Bowser yritti heitellä pieniä kirveen näköisiä esineitä Mariota kohti ja kuoli vain muutamasta iskusta. Haastavampaa oli varmaan matka Bowserin luokse. Kun Bowser oli pois pelistä, prinsessa pelastui. (Nintendo EAD 1985.)

### 3.1.2 Pelityylin vaikutus vastustajan suunnitteluun

Eri genreissä vastustajilla on omanlaisiaan mekaniikkoja, jotka eivät kaikki sovi sekoitettavaksi eri genreihin tai pelityyleihin. Tarkasteltaessa aiemmassa kappaleessa käsiteltyjä pelejä, otetaan esimerkiksi 2D- ja 3D-pelien erot. Huomattavin ero lienee se, että 2D-peleissä ei pysty liikkumaan kuin x- ja y-akseleiden suuntaisesti, joka rajoittaa liikkumista vastustajan ympärillä, kun taas 3D-peleissä pystytään liikkumaan myös syvyysakselin suuntaisesti. 3D-maailmaan on järkevämpää toteuttaa sellaisia monipelejä, joissa pelaaja saa liikkua vapaasti, koska avarassa pelikentässä on luonnollisesti enemmän tilaa liikkumiselle kolmannen ulottuvuuden ansiosta.

Vaikka 2D-peleissä voidaan liikkua vain neljään eri suuntaan, taistelusta pystytään tekemään monimutkaisempi lisäämällä tasoja, joiden päältä pitää hyppiä, tai enemmän väisteltäviä asioita pelaajan esteiksi. 2D-peleissä on onnistuneesti käytetty esimerkiksi vastustajan nopeutta. Yksinkertaisuudessaan vastustaja tekee nopeita liikkeitä, jolloin pelaajalta vaaditaan nopeaa reagointia. Tasohyppelymekaniikkaa nähdään joskus 3D-peleissäkin, joissa se ei kaikkien mielestä ole hauskaa, jos se on huonosti suunniteltu. 3D-pelissä vastustajan kentässä olisi hyvä olla reilusti tilaa, koska liian ahdas kenttä voi olla hankala, jos pitää paljon liikkua ja väistellä vastustajan iskuja. Kun pelikentässä on reilusti tilaa, 3D-pelin vastustajalla on hyvä mahdollisuus käyttää pelaajan eduksi erilaisia asioita, kuten väliaikaisia piilopaikkoja pelaajalle, joihin hän voi väistää iskuja tai mennä parantamaan itseään. 3D-kenttiin on helpompaa sijoittaa taisteluun pelaajalle esimerkiksi vipuja tai muita sellaisia elementtejä, minkä kanssa pelaajan on oltava vuorovaikutuksessa kesken taistelun, koska 2D-kentän tila saattaa olla hyvin rajallinen, kun siellä ei pääse liikkumaan syvyyssuunnassa. Sellaiset eivät toki ole kiellettyjä 2D-peleissäkään, mutta niitä on hyvä käyttää harkiten, jotta kentästä ei tule liian täysi tai sotkuisen näköinen. 3D-vastustajista voidaan tehdä monimutkaisempia, koska pelikenttä on joustava. Tekemällä vastustajasta todella nopealiikkeinen, 2D-vastustajastakin saadaan vaarallinen, vaikka sen mekaniikat olisivat yksinkertaiset. Jos vastustajana on sellainen, joka ei liiku kentässä paikasta toiseen, sillä voikin olla vain yksi heikko kohta, johon pelaajan täytyy kohdistaa iskunsa tehdäkseen vahinkoa vastustajaan. (Gorguinni 2020.)

2D-peleissäkin on hyvin paljon eroja, eikä voida tehdä samanlaista vastustajaa mihin tahansa 2D-peliin. Otetaan tarkasteluun esimerkiksi Cuphead ja Super Mario. Cupheadin vastustaja on liian vaikeaksi suunniteltu Super Marion voitettavaksi. Cupheadin vastustajat vaativat nopeita liikkeitä ja ketteryyttä sekä jatkuvaa tulitusta pelaajalta. Cupheadin vastustajataistelut ovat myös aika pitkiä kestoltaan ja taisteluissa on monta vaihetta. Bowserin tyylinen vastustaja Cupheadissa olisi taas liian tylsä ja tappaisi koko pelin idean tylsyydellään. Pelit ovat myös tehty eri aikakausilla,

joka vaikuttanee vahvasti pelien eriäväisyyksiin. Näiden kahden pelin vertailun tarkoitus on siinä, että vastustajaa suunnitellessa otetaan huomioon, minkälaiseen peliin vastustajaa ollaan suunnittelemassa ja minkälaisia kykyjä pelaajalla on käytössään.

## 3.2 Vastustajataistelun suunnittelu

Vastustaja on sovitettava pelin teemaan siten, että se tuntuu kuuluvan peliin niin mekaniikoiltaan, että tarinaltaan. Jos peliin on tulossa useampi vastustaja, mietitään miten vastustajat sopivat yhteen. Etteivät kaikki vastustajat kuitenkaan tunnu samalta, erilaisilla ja uusilla mekaniikoilla saadaan vaihtelua taisteluihin. Koko pelin, mukaan lukien musiikin ja ympäristön, tulisi tukea etenkin päävastustajaa kasaten intoa ja jännitystä pelaajan edetessä kohti viimeistä haastetta. Matka vastustajan luokse pitäisi olla yhtenäinen ja kun vastustaja ilmestyy, kaiken on tunnuttava oikealta ja uskottavalta. (de Wit 2015.)

Vastustajan halutaan olevan palkitseva, todellinen testi pelaajan taidoille sekä jännittävä ja erilainen kuin muut pelin viholliset. Hyvä vastustaja on testi pelaajalle, sillä hän pääsee näyttämään pelin varrella oppimansa kyvyt ja taidot. Jo pääsy vastustajan luo on palkitsevaa pelaajalle. Ja onhan vastustajan taisteleminen yleensä vaihtelua muuhun seikkailutoimintaan pelissä. Tarinan puolesta päävastustajan luo pääseminen on pelin tavoite. Välivastustajat rytmittävät peliä sekä rakentavat jo innostusta päävastustajan luo pääsemiseksi. Peli yleensä vaikeutuu ja intensiivisyys nousee jokaisen välivastustajan jälkeen. Jokainen voitto välivastustajalla on yksi askel eteenpäin päävastustajaa ja pelin läpäisyä. Pelaaja taistelee tiensä kukistamaan sen, joka on kylvänyt kaikkea pahaa ympäristöönsä ja kenen kätyreitä vastaan on joutunut taistelemaan koko pelin varrella. Vastustajan voittamisen ilo ei usein ole ainoa palkinto, jonka pelaaja saa, mutta se on kuitenkin erittäin tärkeä. Voitosta voi poikia esimerkiksi jotain uutta peliin, kuten uuden alueen aukeaminen, joku uusi esine, kyky tai taikavoima. (Stout 2020.)

Pelaajan tulee erottaa selkeästi, mitkä ovat tavallisia vihollisia ja milloin kohdataan vastustaja, koska vastustajan tulee olla erottuva muista pelin vihollisista sekä ulkonäöltään, että käyttäytymiseltään. Usein vastustajan huoneeseen astuessa pelissä alkaa joku lyhyt videopätkä, jossa vastustaja uhittelee pelaajalle. Vastustajalle on animoitu jonkinlaisia uhkaavia eleitä tai dialogi ennen kuin taistelu alkaa. Vastustajat ovat yleensä huomattavasti isokokoisempia, joka tekee niistä uhkaavan näköisiä. Aina ei välttämättä olekaan tarpeen pelotella pelaajaa heti alussa, vaan vastus-

tajan voikin naamioida näyttämään tavalliselta ja helpolta, mutta sitten tapahtuu käänne juonessa, jolloin vastustaja muuttuu hurjaksi. Vastustajalla haetaan kontrastia muuhun peliin, esimerkiksi Dark Souls -pelissä päävastustaja, Gwyn Lord of Cinder, onkin tavallisen ihmisen näköinen eikä edes merkittävän suuri, kun välivastustajat ovat isokokoisia ja erilaisia demoneita. Tällä ihmismäisellä päävastustajalla on kuitenkin erittäin suuri tulimiekka käytössään, joka luo pelottavuutta vastustajan olemukseen. Vastustajat ovat haastavampia kuin muut viholliset, päävastustaja kaikista haastavin, mutta taistelu saattaa tuntua pitkäveteiseltä, jos vastustaja kestää liian paljon vahinkoa ja sen tappaminen ottaa paljon aikaa tai vastustaja pystyy parantamaan itseään liikaa tai liian usein. Vaikeustasoa voi miettiä muustakin kuin vastustajan kestävyyden näkökulmasta. Taistelun voittamisesta saadaan vielä palkitsevampi, kun se haastaa pelaajaa käyttämään ongelmanratkaisukykyjään, eikä pelkästään puhdasta voimaa. (Snoman Gaming 2016.)

Vaikka taistelun vaikeustaso ja pituus ovat tärkeitä asioita, niin tärkeämpää taistelun viihdearvolle on se, että onko taistelu mukaansatempaavaa. Jos keskittyy vain siihen, onko taistelu liian helppo tai vaikea, liian lyhyt vai pitkä, saattaa helposti unohtaa olennaisen. (Design Doc 2019.) Aiemmin on lueteltu jo paljon rakennuspalikoita hyvään vastustajataisteluun, mutta syvennytään vielä muutamaan asiaan tarkemmin. Vastustajan ja taistelun mekaniikkoja suunnitellessa otetaan huomioon mitä pelaaja jo osaa. Aiemmissa tasoissa on ehkä vähitellen opittu uusia kykyjä tai taikoja. Koska taistelussa halutaan haastaa pelaajan jo oppimat taidot, on tärkeää suunnitella elementtejä, joissa pelaajan on hyödynnettävä niitä. Näin taistelu myös peilaa aiempaa kokemusta pelistä, siinä missä graafinenkin teema.

Taistelu on hauskaa silloin, kun vastustaja on tarpeeksi haastava mutta reilu. Vaikka taistelu olisi vaikea, se ei saa tuntua mahdottomalta. Reiluuden tunnetta saa luotua ainakin niin, että vastustajan käyttäytyminen noudattaa tiettyä kaavaa ja sen eleitä pystyy oppia lukemaan. Kun vastustaja on esimerkiksi tekemässä iskua, pelaaja pystyy lukemaan vastustajan eleestä mitä on tapahtumassa, jolloin pelaaja saa mahdollisuuden väistää. Taistelua voidaan jakaa myös eri vaiheisiin, jolloin se voi vaikeutua vaihe vaiheelta. Taistelun vaihe yleensä vaihtuu, kun vastustajaan on saatu tehtyä tietty määrä vahinkoa. (Snoman Gaming 2016.) Vastustajalla voi olla myös tietty ele, jolla se näyttää olevansa erityisen haavoittuvainen ja pelaajalle olisi hyvä aika iskeä takaisin. Näin voidaan myös miettiä asiaa vastustajan kestävyyden kannalta; halutaanko että pelaaja pystyy vahingoittamaan vastustajaa vain vähän kerrallaan vai sallitaanko pelaajalle vakavampia iskuja, jolloin vastustajalta saisi kerralla paljon elämää pois. Kun vastustaja vihdoinkin kaatuu, pelaajalle pitäisi jäädä euforinen olo, aivan kuin olisi ratkaissut haastavan aivopähkinän. (de Wit 2015.)

Suunnitellessa hyvää taistelua, on myös asioita, joita halutaan välttää. Joitakin seikkoja mainittiin jo aikaisemmin, mutta ei vielä kaikkia. Yleisesti asiat, jotka tuntuvat pelaajan mielestä ärsyttäviltä ja turhilta. Pitkäveteinen, samoja asioita toisteleva dialogi, jota ei voi edes ohittaa, kun pelaaja odottaa jo pääsevänsä haasteen kimppuun. Vielä pahempaa on se, että pelaajan kuollessa ja yrittäessä uudelleen, täytyy katsoa sama videonpätkä tai dialogi kuin aloittaessa ja jota ei edelleenkään saa edes ohittaa. Kun pelaaja on koko pelin ajan odottanut kohtaamista vastustajan kanssa, joka on kylvänyt tuhoa ja uhkaillut pelaajaa vähän väliä, niin kohdatessaan tämän roiston, vastustaja juokseekin pakoon pelkurimaisesti ja heittelee kentälle omia pieniä kätyreitään taistelemaan puolestaan. Huonosti suunnitellut mekaniikat; pelaajalle opetetaan vasta vastustajataistelussa jokin uusi taito. Vältettävää on myös kaikki, mitkä tekevät taistelusta epäreilun tuntuisen, kuten se, että vastustajan aikeita ei pysty ennustaa, jos ne eivät menettele mitään kaavaa, vaan toistuvat sattumanvaraisesti. Vastustajalta tulevia iskuja, joita ei ole mitään mahdollisuutta väistää tai ne toistuvat liian usein tai ruudun ulkopuolelta tulevat iskut. Vastustaja muuttaa pelin sääntöjä yllättäen, eikä noudata pelin mekaniikkoja. Vastustaja saa elämää takaisin jostain syystä, vaikka pelaaja tekee parhaansa tehdäkseen mahdollisimman paljon vahinkoa. On myös tärkeää antaa palautetta pelaajalle niin, että pelaaja näkee selvästi, onnistuiko hän vahingoittamaan vastustajaa iskullaan vai ei. Senkin voisi laskea huonoksi suunnitteluksi, jos vastustaja auttaa pelaajaa liikaa. Esimerkiksi niin, että vastustajan heikkouksia esitellään liikaa vieden voittamisesta kaiken haasteen tunnun. (de Wit 2015; Houghton 2013.)

Pieni yhteenveto asioista, joiden tärkeys korostui asiaa tutkiessani:

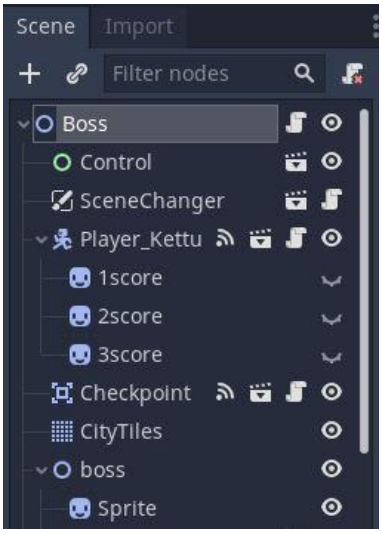
- ✓ vastustajasta on tultava tunne, että se kuuluu peliin
- ✓ palkinto voitosta
- ✓ vastustaja eroaa selkeästi tavallisista vihollisista ja päävastustaja eroaa välivastustajista
- ✓ vastustaja on pelottava ja uhkaava
- ✓ vastustaja testaa pelaajan aiemmin opittuja taitoja
- ✓ taistelun rakenteen suunnittelu: vaiheet, kesto
- ✓ vastustaja on reilu: ei tunnu mahdottomalta päihittää, ennustettavuus käyttäytymisessä

# 4 Godot Engine

Godot Engine on täysin ilmainen, avoimella lähdekoodilla oleva pelimoottori. Godot toimii eri käyttöjärjestelmillä, kuten Windows, Linux, macOS, FreeBSD ja OpenBSD sekä 32-, että 64-bittisenä. Godot on myös varsin kevyt ohjelma. Godotilla onnistuu 2D, 3D, VR ja web-pohjaisten (HTML5) pelien kehittäminen, niin konsoleille kuin älylaittelekin. Godotilla tehdyistä peleistä ei tarvitse maksaa kehittäjille edes rojalteja. Godot Engine on voittoa tavoittelematon, vapaaehtoisten kehittämä pelimoottori. (Linietsky & Manzur 2007-2020.)

## 4.1 Yleistä

Solmut (node engl.) ovat pelien rakennuspalikoita Godotissa. Solmuja on eri toiminnoilla eri käyttötilanteita varten. Solmun ominaisuuksia pystyy muokata, ja sitä voi laajentaa lisäämällä funktioita. Solmulla voi olla ns. lapsisolmuja. Solmurakennelmaa, joka koostuu vanhempi- ja lapsisolmuista, kutsutaan solmupuuksi (nodetree engl.) (Linietsky & Manzur 2014-2020a).




Kuva 7. Näkymä rakentuu juurisolmusta ja lapsisolmuista (Nevala ym., 2020).

Jotta peliä voidaan aloittaa rakentamaan solmuilla, luodaan näkymä (scene engl.). Näkymällä on juurisolmu (root node engl.), jolle aletaan lisäämään lapsisolmuja. Peli voi koostua monesta näkymästä. Projektille määritetään päänäkymä, joka avautuu aina, kun peli käynnistetään. (Linietsky & Manzur 2014-2020a.)

Kun projektissa on useampi näkymä, yhdestä näkymästä voidaan luoda ilmentymä (instance engl.) eli tuodaan haluttu näkymä toiseen näkymään. (Linietsky & Manzur 2014-2020b). Esimerkiksi solmu ”Player_Kettu” (Kuva 7) on oma näkymänsä, joka on tuotu, eli luotu ilmentymä, pelissä jokaiseen kenttänäkymään.

## 4.2 Ohjelmointi

Godotiin on sisäänrakennettuna GDScript ja VisualScript. Godot tukee myös C++ ja C# -ohjelmointikieliä, mutta niitä varten tulisi käyttää erillistä ohjelmointiympäristöä.

GDScript on Godotin pääkieli. GDScript on yksinkertainen, Pythonin kaltainen ja helppo osata, etenkin jos on kokemusta Pythonista tai Luasta. GDSCriptissä on valmiit koodit muun muassa solmuja ja signaaleja varten, joka tekee työskentelystä tehokasta. Vektorityypitkin ovat sisäänrakennettu, jolloin sillä voi hyvin käyttää myös lineaarista algebraa ohjelmoinnissa.


Kuva 8. Esimerkki GDScript -tiedostosta Godotissa.

VisualScript on nimestään päätellen visuaalista ohjelmointia, jossa ”koodilaatikoita” yhdistellään toisiinsa. Sitä on helpompi käyttää, jos ei omaa aiempaa kokemusta ohjelmoinnista. Se ei kuitenkaan syrjäytä perinteistä ohjelmointia, koska se ei skaalaudu yhtä hyvin ja koodin muokkaaminen ei käy yhtä helposti vain muutaman merkin lisäämällä. VisualScript on yleensä kätevä vain silloin, kun ohjelmointitaidoton haluaa kokeilla pelin kehittämistä. Godotissa skriptitiedosto liitetään solmuun, jota halutaan ohjelmoida. Skriptillä lisätään solmuun toiminnallisuus ja vuorovaikutus muiden projektin solmujen kanssa. Skripti perii sen solmun funktiot, johon se on liitetty. (Linietsky & Manzur 2014-2020c.)

## 4.3 Tekoäly Godot Enginessä

Godotissa on valmiita solmuja, joilla voi luoda tekoälyominaisuuksia NPC:lle. 2D-peliä tehdessä solmujen nimessä on käytetty ”2D” päätettä, esimerkiksi ”Path2D” tai ”Area2D”. Koska työn aiheena on 2D-pelin kehittäminen, käytetään esimerkkeinä 2D -solmuja. Godotilla on mahdollisuus ohjelmoida NPC:t käyttäytymään eri tavoilla eri solmuja käyttämällä.

Path2D-solmua käyttämällä voidaan ohjelmoida NPC, vihollinen tai vaikka liikkuva taso pelissä seuraamaan ennalta määrättyä kulkureittiä. Solmuun määritetään pisteet, jotka merkitsevät mistä mihin reitti kulkee. Solmulle lisätään lapseksi PathFollow2D, joka osaa seurata Path2D:n merkittyä reittiä. (KidsCanCode 2019.)

Area2D-solmulla pystytään lähettämään signaali, jos tietty objekti ylittää määritetyn alueen reunan. Tämä solmu on kätevä silloin, jos halutaan määrittää tapahtuma, joka sattuu vain pelaajan astuessa tietylle alueelle. Esimerkkinä pelaajaa jahtaava vihollinen: jos ei haluta, että vihollinen pystyy seuraamaan pelaajaa kentän toiselta puolelta asti. Viholliselle voidaan lisätä Area2D-solmu ja sijoittaa solmun alue niin, josta halutaan, että vihollinen näkee pelaajan. Kun pelaaja astuu alueen rajan yli, Area2D saa signaalin, että ylitys on tapahtunut. Silloin vihollinen ymmärtää lähteä seuraamaan pelaajaa vasta, kun pelaaja on tämän näköetäisyydellä. (KidsCanCode 2019.)

RayCast2D on myös kätevä 2D-tasohyppelypeleissä sellaisten vihollisten kulkureittien hallitsemiseen, jotka liikkuvat kentässä edestakaisin. Jos viholliset saavat muuten liikkua vapaasti edestakaisin, mutta halutaan että ne vaihtavat suuntaa kielekkeen reunalla. RayCast2D toimii niin, että se piirtää viivan haluttuun suuntaan ja se viiva palauttaa RayCastille informaatiota mitä on tapahtumassa. Eli jos ei haluta, että NPC tippuu alas kielekkeeltä, RayCast2D voidaan asettaa suuntaamaan lattiaan, jolloin se ilmoittaa, kun lattia on häviämässä alta ja on aika vaihtaa kulkusuunta. (UmaiPixel 2018.)

Navigation2D-solmua voidaan käyttää esimerkiksi AStar-tyyppiseen reitin etsintään kaksiulotteisen alueen sisällä. Sen toiminta perustuu siihen, että sillä voidaan laskea lyhyin reitti kohteeseen. (Linietsky & Manzur 2014-2020d.)

Yksi vaihtoehto on myös tilakoneiden ohjelmointi tekoälykäyttäytymistä varten. Menetelmiä ovat esimerkiksi: hahmolla voi olla lapsisolmu jokaiselle tilalle, enum-tietotyyppiä voidaan käyttää yhdessä GDScriptin match-lausekkeen kanssa tai tilojen omat skriptit voidaan vaihtaa dynaamisesti solmusta ajon aikana. (Linietsky & Manzur 2014-2020e.)

## 4.4 Miksi Godot Engine?

Kuten esittelykappaleessakin jo todettiin, Godot on todella kevyt ohjelma, tiedostokoko on vain 60MB ja se ei vaadi asennusta. Lisäksi se toimii monella käyttöjärjestelmällä. Kynnys aloittaa Godotilla on matala sen vuoksi, että se on ilmainen ja pelin tekijä omistaa Godotilla tehdyn pelinsä kokonaan itse, joten pelistä ei tarvitse maksaa rojalteja tai mitään muutakaan. Godotiin on luotu oma skriptikielensä, jossa on vaikutuksia Pythonista ja Luasta. GDScript on räätälöity yhdistämään näiden kahden kielen hyvät ominaisuudet yhdeksi kieleksi ja GDScriptille on oma sisäänrakennettu ohjelmointiympäristö. GDScript on monien mielestä ollut helppo oppia, mutta jos ei kuitenkaan halua tarttua uuteen kieleen, Godotissa on mahdollista käyttää myös muita ohjelmointikieliä. Godotissa on erilainen näkymäsysteemi kuin esimerkiksi Unityssä ja Unreal Enginessä, niissä pelikentät rakentuvat näkymän ympärille, mutta Godotissa näkymä on solmurakennelma. Godotissa näkymät voivat periä tietoa toisiltaan, kunhan niillä on samantyyppinen juurisolmu. Tämä mahdollistaa laajasti monipuolisen työskentelyn. (Buckley 2019.)

# 5 Projektityö

Litter Run on 2D-tasohyppelypeli, joka on kehitetty kokonaan Godot Enginellä mobiililaitteille, joissa on Android-käyttöliittymä. Peliprojekti lähti käyntiin jo loppuvuodesta 2019, mutta kesällä 2020 sitä työstettiin paljon eteenpäin pienellä kolmen hengen ryhmällä. Seuraavassa luvussa kerrotaan taustatietoja pelistä sekä aletaan suunnittelemaan vastustajaa kyseiseen peliin.

## 5.1 Kuvaus pelistä

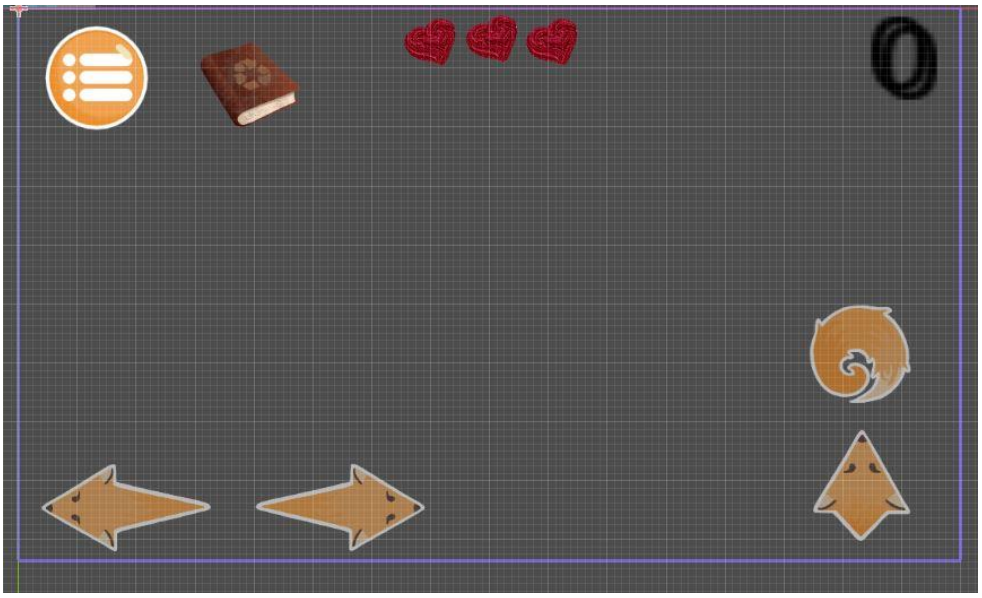
Litter Run kertoo ympäristömme roskaisuudesta. Suomen luonnosta tutut metsän eläimet ovat hätää kärsimässä, koska heidän elinympäristönsä on niin siivottomassa kunnossa. Päähahmo, jota pelaaja kontrolloi, ottaa tehtäväkseen siivota metsää. Pelin edetessä pelaaja kohtaa myös kaupunkiympäristön ja näkee miten siivottomasti ihmiset käyttäytyvät. Pelaaja kerää erilaisia roskia ja oppii sekä yleisten, että hieman harvinaisempien, mutta hyvin haitallisten roskien ympäristövaikutuksista. Pelaajan löydettyä erikoisemman roskan, se tallentuu roskakirjaan, josta voi lukea kyseisen roskan tietoja.

Litter Runissa on perinteiset tasohyppelymekaniikat. Pelaaja liikkuu kaksiulotteisessa pelimaailmassa eteen- tai taaksepäin x-akselin suuntaisesti sekä osaa hypätä y-akselin suunnassa ylöspäin. Pelaajalla on käytössään myös tuplahyppy, jolloin pelaajan on mahdollista hypätä normaalia korkeammalle. Vihollisten ja esteiden ylittämiseen pelaaja osaa ”lyödä”, joka on animoitu ketulle hännänheilautukseksi ja tehdä myös eräänlaisen syöksyiskun hypyn kautta maahan.

Kuva 9. Ylärivin kuvasarja on ilmasta syöksyn animaatiolle, ja alempi hännänheilautukselle (Nevala ym. 2020).

Peliin on sijoitettu esteitä, joista osa tekee vahinkoa pelaajaan, jos pelaaja osuu niihin edetessään. Osa esteistä on sellaisia, jotka pelaaja pystyy rikkomaan ja näin ollen ne on helpompi ylittää. Pelissä on myös vihollisia vähän erilaisilla mekaniikoilla; ensimmäiset vain kävelevät edestakaisin, toiset hyppivät ja kolmannet pudottelevat vahinkoa tekeviä pieniä roskia pelaajan tielle. Pelaaja ottaa vahinkoa osuessaan viholliseen. Pelaajalla on kolme terveyspistettä ja yksi osuma tekee vahinkoa yhden pisteen verran. Jos pelaajalta loppuu terveyspisteet, hän herää ja pääsee jatkamaan peliä eteenpäin uudelleen joko kentän alusta tai välitallennuspisteeltä.

Litter Run on tehty pelattavaksi mobiililaitteilla. Käyttöliittymä on suunniteltu niin, että peliä pelataan laite vaakatasoon käännettynä. Pelaajahahmon ohjaukseen on neljä painiketta: eteen ja taakse liikkumiseen, lyömiseen (hännänheilautus tai syöksy) sekä hyppäämiseen. Käyttöliittymään on sijoitettu myös painikkeet pelin keskeyttämiselle/valikkoon pääsemiselle ja roskakirjan aukaisemiseen.



Kuva 10. Käyttöliittymä (Nevala ym. 2020).

Pelaajan kontrollit skriptien puolella ovat jaettu kahteen eri skriptitiedostoon. Nämä toimivat niin, että muuttujat ovat ohjelmoitu globaaleiksi muuttujiksi tiedostoon, josta voidaan hakea muuttujia mihin tahansa muuhun skriptitiedostoon projektin sisällä. Esimerkiksi nopeus x-suunnassa on määritelty globaaliksi. Pelaajahahmoon on liitetty oma skriptinsä, johon on ohjelmoitu se, mitä tapahtuu, kun painetaan eteen tai taakse -painikkeita. Tähän pelaajahahmon skriptiin on haettu se globaali muuttuja, joka määrittää nopeuden arvon. Silloin hahmo liikkuu määritellyn nopeuden arvon mukaisesti x-akselilla.

Hyppynappia painamalla pelaaja tekee hypyn näköisen liikkeen y-akselin suuntaisesti ylöspäin. Lyöntinappiin on ohjelmoitu kaksi eri vaihtoehtoa. Siinä katsotaan ensin se, mistä tilanteesta pelaaja painaa nappia. Jos pelaaja on maassa tai hypännyt vain yhden kerran, pelaajahahmo tekee hännänheilautuksen. Jos taas pelaaja on juuri tehnyt tuplahypyn, lyöntinapista hahmo syöksyy alas maahan.

Kamera on liitetty pelaajahahmoon siten, että hahmo on aina kuvaruudun keskellä ja kamera seuraa pelaajan liikkeitä. Ainoa poikkeus milloin pelaaja ei ole keskitetty, on silloin kun pelaaja on lähellä pelikentän reunaa, esimerkiksi ihan kentän alussa. Kameralle on asetettu raja niin, ettei pelaaja näe kentän reunojen ylitse ja siksi kamera pysähtyy reunalle, mutta pelaaja voi silti liikkua myös reunaan kiinni.

Kenttäsuunnittelussa on edetty pelin tarinan pohjalta. Tarina alkaa metsästä, joten pelin ensimmäinen kenttä sijoittuu myös metsään. Metsäkenttien jälkeen pelaaja etenee kaupunkialueelle. Kentät etenevät suoraviivaisesti ja matkan varrella pelaaja kohtaa vihaisia roskapusseja, jotka yrittävät vahingoittaa pelaajaa. Pisteiden roolissa toimii yksittäiset roskat, jotka kuvaavat yleisiä roskia, joita ympäristömme on pullollaan. Kenttiin on myös piilotettu erikoisempia pisteitä, jotka tallentuvat kirjaan kerättäessä. Kirjasta voi sitten lukea kunkin roskan haitallisuudesta. Käyttöliittymässä on yläreunassa kirjan kuva, jota painamalla kirja aukeaa ja sitä voi lukea.

Kun Litter Run -peliä lähdettiin suunnittelemaan ja kehittämään, pelaajahahmoksi haluttiin valita jokin kaikkien tuntema eläin, jota esiintyy myös Suomen luonnossa. Kettuhahmot ovat suosittuja, ja vetoavat helposti ihmisten tunteisiin. Kettuja on paljon erilaisissa peleissä ja animaatioelokuvissa. Ketut ovat usein suloisia ja kuvataan sankareinakin, hyvänä esimerkkinä Disneyn Robin Hood -animaatioelokuva. Sankarillinen hahmo sopii myös Litter Runiin, kun hän lähtee vaaralliselle seikkailulle pelastaakseen kotimetsänsä. Kettu on piirroshahmona helppo tunnistaa oranssivalkoisen turkkinsa ansiosta.

Usein pelihahmo 2D-pelissä on kaksijalkainen. Litter Runissa pelihahmosta on tehty nelijalkainen, joka onkin luonut tiettyjä haasteita pelin kehitykselle, koska hahmo on pitkän mallinen. Esimerkiksi ylä- tai alamäessä juoksua on hankalampi saada realistisen näköiseksi.

Kuva 11. Toinen esimerkki siitä, miksi nelijalkainen hahmo ei ole yhtä hyvä valinta kuin kaksijalkainen, koska etujalat näyttävät seisovan ilmassa, kun takajalat ovat vielä tason päällä. (Nevala ym. 2020).

Pelaajahahmo rakentuu monesta eri solmusta omassa näkymässään. Juurisolmuna toimii KinematicBody2D (nimetty ”Player_Kettu”), jonka alle luodaan kuvat, animaatiot ja törmäyksentarkistukset (CollisionShape2D) lapsisolmuina.

Suurin haaste on ollut pelaajan törmäyksien säätö pelin eri objektien kanssa, koska suorakulmion muotoinen CollisionShape2D ei toimi joka tilanteessa. Se toimii kuitenkin paremmin kuin ConvexPolygonShape2D, koska silloin jos pelaajan törmäyksen tarkistus on määritelty monimutkaisen muotoiseksi, se saattaa jäädä helpommin jumiin omituisiin paikkoihin, jolloin pelikokemus on huono. Suorakulmion muotoinen ei jää mihinkään jumiin, se suurimmaksi osaksi näyttää vain hassulta joskus pelaajan seisoessa puoliksi tason päällä, niin että joko etu- tai takaruumiista puolet näyttää olevan ilmassa. Pelihahmon liikkuminen on erittäin tärkeä osa hyvää pelikokemusta, koska kun se on tehty hyvin, pelaaja ei kiinnitä siihen huomiota. Jos se on taas tehty huonosti tai se on jollain tavalla puutteellista tai kömpelöä, sen huomaa varmasti.

## 5.2 Vastustajan suunnittelu

Hyvin toteutetut vastustajataistelut ovat yleensä koko pelin kohokohtia. Pelin tarina johtaa vastustajan luo ja vastustaja jatkaa tarinaa. Vastustajataistelut ovat mieleenpainuvia, koska niissä on ollut jotakin hienoa tai ne eivät ole päästäneet pelaajaa liian helpolla. Vastustaja lopulta kaadettuaan palkitsee pelaajan vähintäänkin onnistumisen ilolla.

Vastustajan ja taistelun suunnittelu ei ole aivan yksinkertaista. On paljon kysymyksiä, joihin pitää vastata. Kun vastustaja on saatu kuviteltua mielestään hyväksi ja siitä on ehkä tehty jo ensimmäinen versio, testausvaiheessa se saattaakin tuntua huonolta ja moni asia saatetaan joutua korjaamaan. Varsinkin jos yksin on tehnyt kaiken työn ja ensimmäistä kertaa saa ulkopuolista palautetta.

Ensimmäiset tärkeät kysymykset vastustajan suunnittelulle voisivat olla siihen peliin liittyviä kysymyksiä, johon vastustajaa suunnitellaan. Minkälainen tarina pelissä on? Kuka on tämä pahin vastustaja ja miksi? Mikä on vastustajan tavoite? Mitä vastustajalla halutaan viestiä pelaajalle? Mikä on pelin kohderyhmä, vaikeustasoa ajatellen?

Ajatus Litter Runissa oli korostaa ihmisen välinpitämättömyyttä luontoa kohtaan ja näyttää ongelmaa eläinten näkökulmasta, koska eläimet kärsivät elinympäristönsä roskaamisesta eniten. Ensimmäinen idea oli, että vastustaja olisi iso roska, roskapussi tai roska-astia, mutta se tuntui ristiriitaiselta, koska roskat tai roska-astiat eivät ole roskaamisen takana, vaan ihminen. Ihminenhän oli kaiken pahan takana, joten vastustajaksi hahmottui ihminen.

Taistelua suunnitellessa ihmisen vahingoittaminen ei tuntunut kuitenkaan oikealta ratkaisulta, vaan ihmistä vastaan olisi taisteltava jollain muulla tavalla. Roska-astian käyttämisestä taistelussa jäi myös vielä ajatus. Suunnittelu tuntui hieman haastavalta, koska ajatuksesta tehdä perinteinen taistelu, jossa vastustajaa itseään vahingoitettaisiin, luovuttiin. Toisaalta siinä oli se hyvä puoli, että näin taistelusta ei tulisi niin tavallinen, vaan olisi erilainen kuin yleensä.

Taistelun keskeiseksi ideaksi muotoutui roskien kerääminen roskikseen samalla kun ihminen yrittää estää pelaajan siivousoperaatiota. Vaikka ihminen tässä onkin päävastustaja, niin silti häntä vastaan ei suoraan taistella, vaan hän on pelikentän taustalla heittämässä roskia sekä yrittäen vaikeuttaa pelaajan toimintaa. Ihminen heittelee kentälle pisteinä toimivia roskia, jotka pelaaja voi kerätä roskikseen. Kun roskiksen saa kerättyä täyteen, peli on voitettu. Ihminen heittelee myös ilkeitä roskapusseja, jotka yrittävät vahingoittaa pelaajaa. Välillä ihminen yrittää myös tyhjentää roskista, johon pisteitä on kerätty ja eteneminen saattaa ottaa takapakkia, jos pelaaja ei estä ihmisen yritystä ajoissa. Roskis ei voi tyhjentyä edelliseen vaiheeseen takaisin, joten vastustaja ei tee tyhjennysliikettä, jos roskiksen täyttöaste on vaiheiden minimirajalla. Taistelussa on kolme eri vaihetta, jotka vaihtuvat roskiksen täyttöasteen mukaan. Jokainen vaihe on edellistä hieman haastavampi. Kolme terveyspistettä koko taistelun ajaksi saattaa joidenkin pelaajien mielestä tuntua liian vähältä, joten vaiheen kaksi ja kolme alkaessa kentälle putoaa yksi lisäelämä pelaajalle kerättäväksi.

Vaihe 1: pisteroskia ja vihollisia ilmestyy pelikentälle. Roskiksen täyttöaste 0 %-33 %.

Vaihe 2: samat kuin vaiheessa yksi, lisänä tippuvia myrkkyroskia ja ihminen yrittää ensimmäisen kerran tyhjentää roskista. Roskiksen täyttöaste 34 %-66 %.

Vaihe 3: samat kuin edellisessä vaiheessa, mutta ihminen yrittää tyhjentää roskista useammin. Roskiksen täyttöaste 67 %-100 %.

Pelaaja pystyy kerätä kyytiinsä kolme roskaa kerrallaan. Pelaajan päällä näkyy niin monen roskan kuvat, kun pelaajalla on sillä hetkellä mukana. Aina kun pelaaja menee roskiksen luo, roskat siirtyvät roskikseen. Yllä olevassa kuvassa oikeassa reunassa näkyvä roskis on se, johon roskat kerätään. Siinä on kiinnitettynä mittari, joka kertoo roskiksen täyttöasteen. Godotissa on mittarille oma solmunsa, TextureProgress. Sille voi valita monesta eri vaihtoehdosta mittarin täyttymistavan, ja siihen liitetään halutut tekstuurit sekä tyhjälle mittarille, että täydelle mittarille. Mittarille määritetään myös minimi- ja maksimiarvot.

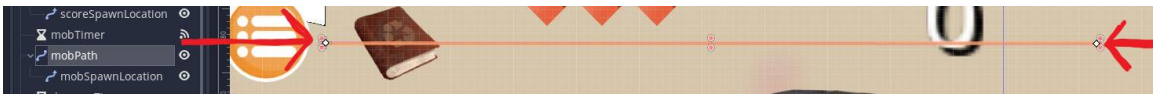
Pelin muut grafiikat on piirtänyt peliprojektimme graafikko, mutta piirsin itse vastustajahahmon. Sillä ei ole vielä erikoisia animaatioita, koska niiden piirtäminen olisi vienyt paljon aikaa, joten piirsin vain yhden kuvan ja animoin pelkästään vastustajan katsetta. Normaalitilassa vastustajan katse on alaspäin. Kun vastustaja on iskemässä roskikseen, sen katse suuntautuu roskikseen ja se varoittaa pelaajaa vilkkumalla punaisena. Jos vastustaja onnistuu iskemään roskikseen, roskis on myös animoitu heilumaan merkiksi tyhjentymisen onnistumisesta.

Jos pelaaja ottaa liian monta kertaa vahinkoa ja kuolee kesken taistelun, taistelu alkaa alusta. Monessa pelissä pelaaja saa armoa tässä kohtaa, eikä joudukaan aloittamaan koko taistelua alusta, vaan esimerkiksi sen vaiheen alusta, johon hän kuoli. Tässä kohtaa kannattaa ottaa huomioon taistelun vaikeusaste, pituus tai se, miten vaikean haluaa vastustajan päihittämisestä tehdä. Tämä taistelu ei ole kovin vaikea eikä pitkä, joten alusta aloittaminen ei ole ajallisesti suuri menetys.

## 5.3 Vastustajan tekoäly

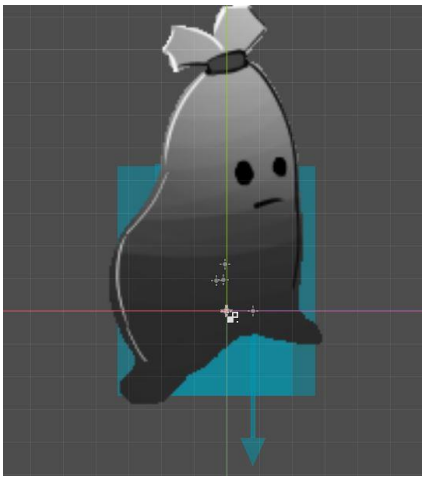
Pisteiden ja vihollisten ilmestymiseen pelikentälle on käytetty Timer-solmua, joka toimii ajastimena. Jotta pisteet ja viholliset saadaan ilmestymään aina eri kohtiin pelikenttää, on hyödynnetty Path2D ja PathFollow2D-solmuja. Path2D-solmulle on määritetty pisteet, joiden välissä reitti kulkee. Timer eli ajastin, lähettää signaalin halutun ajan välein PathFollow2D-solmulle, joka laskee

satunnaisen paikan Path2D:n varrelta. Silloin uusi vihollinen lisätään add_child()-funktiolla ja asetetaan satunnaisesti laskettu sijainti uudelle viholliselle. (Linietsky & Manzur 2014-2019.)




Kuva 12. Oranssi viiva merkitsee Path2D reittiä. Punaiset nuolet osoittavat Path2D alku- ja loppupisteen reitillä.

Viholliset ovat edestakaisin kävelevä roskapusseja, jotka vahingoittavat pelaajaa törmätessään siihen. Pelaaja pystyy hävittämään vihollisen hyppäämällä tämän päälle tai iskemällä tätä hännänheilautuksella. Vihollinen on oma näkymänsä, josta on luotu ilmentymä vastustajan näkymään. Vihollisilla on käytössä RayCast2D-solmu, jotta ne eivät putoaisi alas tasojen reunoilta. RayCast2D näkyy alla olevassa kuvassa alaspäin osoittavana nuolena, joka havaitsee törmäyksen alla olevan tason eli lattian kanssa. Kun tason reuna tulee vastaan eikä nuoli havaitse enää törmäystä lattian kanssa, vihollinen vaihtaa kulkusuuntaa. Jos vastaan tulee seinä, Godotissa on tähän tarkoitukseen käypä funktio valmiina, is_on_wall(), joka ottaa arvokseen joko tosi (true) tai epätosi (false). Kun tämän arvo on tosi, eli kun vihollinen osuu seinään, sen voi ohjelmoida vaihtamaan kulkusuuntaa.



Kuva 13. Edestakaisin kävelevä roskapussivihollinen. Alaspäin osoittava nuoli tulee RayCast2D-solmusta, joka tunnistaa, jos vihollinen on kävelemässä tason reunan yli, jolloin vihollinen vaihtaa kulkusuuntaa. (Nevala ym. 2020.)

Taistelun vaiheiden vaihtuminen on ohjelmoitu mahdollisimman yksinkertaisesti. Ohjelma tarkistaa roskiksen täyttöasteen. Jos täyttöaste on pienempi kuin 33 %, ohjelma lukee match-lausekkeelta vaiheen yksi koodia. Jos roskiksessa on yli 66 %, ohjelma siirtyy vaiheen kaksi koodiin jne. Kun roskis on täynnä, peli loppuu ja pelin näkymä vaihtuu.

```
65  if scorebar.value <= 33:
66      state = 1
67
68  elif scorebar.value > 33 and scorebar.value <= 66:
69      state = 2
70
71  elif scorebar.value > 66 and scorebar.value != 100:
72      state = 3
73
74  else: #scorebar.value == 100:
75      state = 4
76
77
78  match state:
79      1:
80          print("phase 1")
81      2:
82          print("phase2")
83      3:
84          print("phase3")
85      4:
86          print("voitto")
87          SceneChanger.change_scene(SaveSystem.mainmenu)
```

Kuva 14. Vaiheiden vaihtumisen logiikka. Match-lausekkeessa oleviin ehtoihin tulee vaihekohtaiset koodit.

Vaiheessa kaksi vastustaja pudottelee pisteroskien lisäksi myrkyllisiä roskia, jotka pelaajaan osuessaan tekee vahinkoa yhden terveyspisteen verran. Se on toteutettu niin, että myrkkyroskilla on oma ajastin, joka käynnistyy, kun taistelu siirtyy vaiheeseen kaksi. Vaiheen kaksi alkaessa vastustaja alkaa myös yrittämään roskiksen tyhjentämistä pelaajan kiusaksi. Tyhjentämiselle on myös oma ajastin, joka käynnistyy vaiheen kaksi alussa. Ajastin roskiksen tyhjentämiselle on 15 sekuntia, kun ajastimen käynnistymisestä on kulunut tuo aika, vastustaja varoittaa pelaajaa vilkkumalla ja katsomalla roskiksen suuntaan. Tässä kohtaa pelaajalla on mahdollisuus estää vastustajaa iskemästä. Jos pelaaja epäonnistuu, vastustaja vähentää roskiksesta pienen määrän pisteitä. Vaiheessa kolme roskiksen tyhjentämiselle oleva ajastin alkaa pyöriä nopeammin, jolloin vastustaja yrittää iskeä roskikseen useammin. Tässä vaiheessa ajastin asetetaan antamaan pelaajalle merkki vastustajan iskusta kahdeksan sekunnin välein. Yhtenä ideana oli myös lisätä vaikeampia vihollisia vaiheeseen kolme. Silloin olisi saattanut olla järkevämpää olla nopeuttamatta vastustajan iskujen väliä. Jos uusi vihollinen olisi lisätty pelikentälle, riskinä olisi ollut se, että kentällä on liikaa kaikkea. Yksi toinenkin syy miksi ne jäivät pois, oli se, että tässä haluttiin keskittyä enemmän itse vastustajaan, eikä vain heitellä muita vihollisia pelaajan eteen.

Pelikentässä on ”vipu” pelaajalle, jonka pelaaja aktivoi käyttämällä hännänheilautus-iskuaan vipuun. Vivussa on Area2D-solmu, joka tunnistaa pelaajalta tulevan iskun. Kun pelaajan isku osuu vivun Area2D:hen, se pysäyttää ja uudelleen käynnistää vastustajan iskun ajastimen alusta sekä pysäyttää vastustajan varoitusanimaation.

## 5.4 Testaus

Pelin testaaminen on hyvin oleellinen osa pelin kehitystä. Peliä tai pelin osia on hyvä testailla paljon pelin koko kehityskaaren ajan. Pelitestaajia on hyvä ottaa pelinkehitystiimin ulkopuoleltakin, jotta saa palautetta sellaisiltakin henkilöiltä, joille peli ei ole aikaisemmin tuttu. Litter Runia kehittäessä huomasimme, että pelistä sai arvokasta palautetta myös sillä tavalla, että katsoimme, kun testihenkilö pelasi peliä. Koska sillä tavalla näimme itse, miten pelaaja toimii pelin eri tilanteissa. Esimerkiksi, jos joku asia vaikutti olevan toistuvasti monelle pelaajalle epäselvää, niin osasimme korjata sen paremmaksi.

Pelissä ei pitäisi tulla pelaajalle sellaista oloa, ettei pelaaja tiedä ollenkaan, miten edetä. Testaamisessa kannattaa kirjoittaa ylös myös kaikki yleiset mielipiteet ja havainnot pelistä, niin positiiviset, kuin negatiivisetkin. Lisäksi myös, minkälaisia tuntemuksia pelin pelaaminen herättää. Pelintekijää kiinnostaa, millaisia ovat hyvät hetket pelissä ja mitkä kohdat mahdollisesti aiheuttavat turhautumista tai ärsyyntymistä. Saatua palautetta kannattaa analysoida ottaen huomioon pelitestaajien tausta. Palautteeseen saattaa vaikuttaa merkittävästi testaajien pelitottumukset tai -mieltymykset, ja he ovat vasta aloittelijoita testattavassa pelissä, joten peli saattaa tietysti olla vaikeampi aloittelijalle. (Vuorela 2007, 70-71.)

Vastustajaa testatessa päähuomio on siinä, minkälaisella tahdilla kaikki tapahtuu. Ilmestyykö vihollisia liikaa tai liian vähän, entä pisteitä tai lisäelämiä. Sitäkin on pitänyt miettiä testatessa, miten monta prosenttia roskista saa täyteen yhdestä pisteestä. Sitten on vielä se, että miten paljon taistelussa voi ottaa takapakkia tyhjentämällä roskista ilman, että se tuntuu turhauttavalta pelaajan mielestä. Tällaisten hienosäätöjen lisäksi voidaan kartoittaa mahdolliset suunnitteluvirheet.

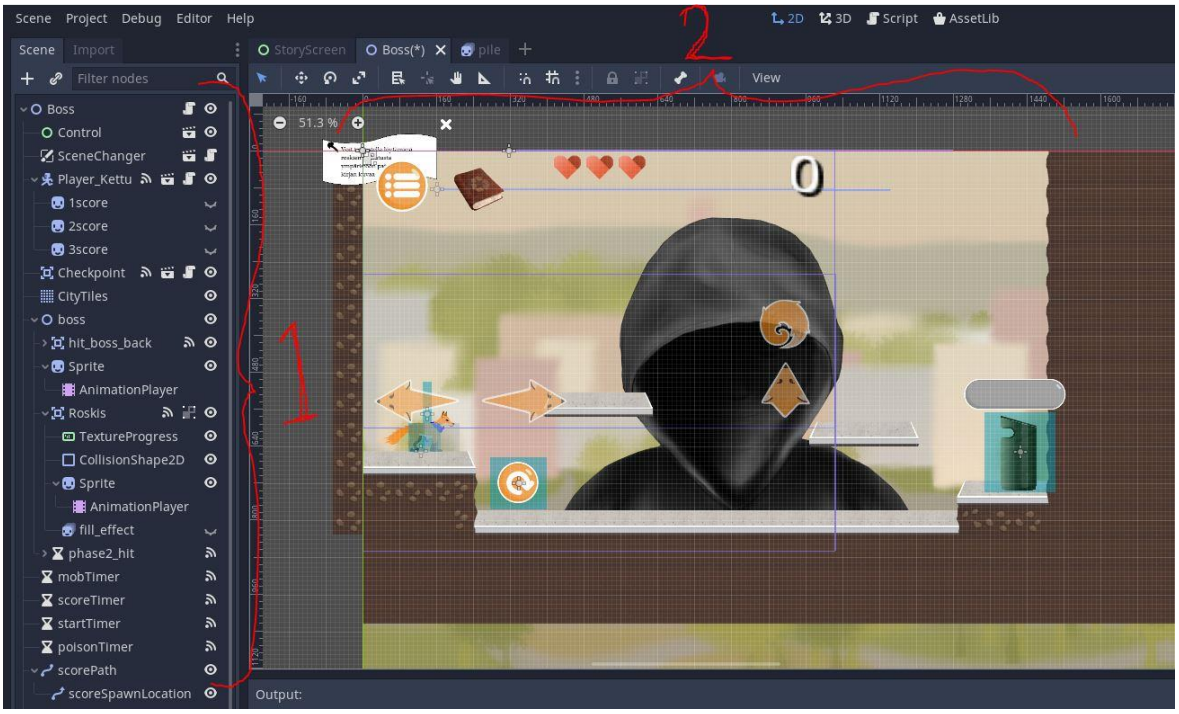
Jotkin asiat saattavat jäädä ajattelematta loppuun asti ja sitten ne huomataan vasta kun testataan lopputulosta. Tässä tapauksessa kävi niin, että jälkeenpäin vasta ajattelin, että vastustaja olisi kannattanut animoida vilkkumaan jollain toisella värillä, kuin punaisella. Koska pelaaja vilkkuu pu-

naisena, kun hän ottaa vahinkoa jostakin, joten tässä yhdistetään punaisena vilkkuminen vahingoittamiseen. Vastustajan ei ole vilkkumisella tarkoitus näyttää ottavansa vahinkoa, vaan varoittaa omasta iskustaan.

Kuva 15. Vastustajan vilkkuminen punaisena ennen roskikseen iskemistä (Nevala ym. 2020).

# 6 Lopputulos

Vastustaja rakentuu monesta solmusta, kuten kuvista ja animaatioista. Vastustajan näkymään on myös luotu ilmentymiä solmuista, jotka sijaitsevat muissa näkymissä, esimerkiksi pelaajan solmu. Vastustajan kenttä on aika yksinkertainen. Alue ei ole kovin iso, se on vain vähän isompi kuin kameran kattava alue. Kentässä (Kuva 16) on pelaajalle kaksi erillistä tasoa, joiden päältä hän voi hypätä. Alhaalla vasemmalla on vipu, jonka aktivoimalla pelaaja voi estää vastustajaa iskemästä roskikseen. Oikeaan reunaan on sijoitettu roskis, johon pelaaja kerää pisteroskat.




Kuva 16. Vastustajan näkymä Godotissa. Numero yksi osoittaa solmupuun, josta vastustaja rakentuu. Numero kaksi osoittaa vastustajan pelikentän ja siinä on nähtävillä myös pelaajan käyttöliittymä. (Nevala ym. 2020.)

Pelien kehityksessä tapahtuu virheitä myös isoille kehitystiimeille tai peliyrityksille. Joskus todetaan, että pelin säännöt vaativat muutoksia ja joskus virheet jäävät huomaamatta siihen asti, kunnes peliä pelaavat jo jopa sadat tuhannet pelaajat. Virheet saattavat jäädä huomaamatta kiireellisen aikataulun vuoksi. Joskus voidaan joutua tekemään kokonaisia työvaiheitakin uudestaan. Lopputuloksessa kannattaa miettiä muutakin kuin virheitä, esimerkiksi sitä, missä on onnistuttu paremmin kuin odotettiin. (Vuorela 2007, 87.)

Itse näen paljon virheitä tämän työn lopputuloksessa. Kun ajatus vastustajan suunnittelusta tuli, visio lopputuloksesta oli hienompi. Ajatellessa syytä sille, miksi siitä ei sitten tullut niin hyvä kuin alun perin kuviteltu, niin tulee päällimmäisenä mieleen, että pelin teemaan, tarinaan ja mekaniikkoihin sopiminen oli haastavaa, koska halusin säilyttää tietyn ideologian vastustajassa. Haasteena oli myös se, että suunnittelin vastustajan yksin suhteellisen lyhyessä ajassa, suunnittelutyöhön olisi kaivattu muitakin mielipiteitä ja ideoita.

Vastustaja vaikuttaa hieman tylsältä. Se kaipaisi jotakin jännittävämpää. Vastustajalla käytetty tekoälyosuuskin jäi hyvin yksinkertaiseksi. Vastustaja olisi kaivannut enemmän suoraa interaktiivisuutta pelaajan kanssa. Kappaleessa 3.2 Vastustajataistelun suunnittelu käsittelin asioita, mitkä eivät välttämättä ole parhaita ratkaisuja vastustajataisteluissa. Yhtenä oli se, että saattaa olla ärsyttävää, jos vastustaja ei itse taistele pelaajaa vastaan, vaan lähettää kätyreitään taistelemaan. Siksi en halunnut, että tämän työn vastustaja pelkästään heittelisi ilkeitä roskapusseja pelaajan tielle, vaan että sillä olisi muutakin. Nämä viholliset, joita vastustajataistelussa käytettiin, ovat kuitenkin passiivisia, eivätkä jahtaa pelaajaa yrittäen vahingoittaa tätä, joten ne toimivat vain pelaajan hidasteena. Yhtenä vaarana oli myös se, että jos vastustaja pystyy parantamaan itseään liikaa, jolloin pelaaja turhautuu taisteluun. Tässä on siis samankaltainen idea siinä, että vastustaja yrittää haitata pelaajan etenemistä taistelussa. Pelaajalla on kuitenkin mahdollisuus estää se. Pelikentän rakennettakin voisi miettiä pidemmälle, että tarvitseeko jotakin siirtää eri kohtiin. Varsinkin vipu, josta pelaaja estää vastustajaa iskemästä roskikseen, ei tunnu sopivan kenttään, joten se vaatii jatkokehittelyä. Vastustajan voitosta tarvitsisi vielä jonkun merkittävän palkinnon. Peliin ei ole suunniteltu loppuun asti palkintosysteemiä (muuta kuin pisteet), mutta palkintona voisi olla esimerkiksi uuden pelialueen aukeaminen.

Projekti on ollut kaiken kaikkiaan opettavainen. Vaiheiden ohjelmointi sujui loppujen lopuksi hyvin. Lähdin aluksi suunnittelemaan sitä paljon monimutkaisemmalla tavalla, joka ei toiminut. Olen tyytyväinen vastustajan ideaan, mutta kuten yllä mainitsin, se vaatisi vielä jotakin. Peliä testatessa tuli mieleen, että viholliset eivät katoa itsestään pelikentältä, jos pelaaja ei itse lyö niitä. Niillä on mahdollisuus kerääntyä kentälle isoksi joukoksi. Tämä ei kuitenkaan tuntunut sittenkään niin huonolta asialta, koska se vähän lisäsi haastetta. Pelaajan siis kannattaa putsata vihollisia pois samalla kun kerää pisteitä. Litter Run on vielä keskeneräinen peli, joten tämä vastustaja oli ikään kuin välivastustaja ja päävastustaja samaan aikaan, koska pelissä ei ole vielä enempää sisältöä, mutta siihen saattaa tulla. Jatkokehittäessä peliä eteenpäin, tämä jää ensimmäiseksi välivastustajaksi. Jotta vastustaja olisi täysin valmis, se tarvitsisi vielä kunnolliset animaatiot ja taustamusiikin sekä äänitehosteet. Koko pelissä meillä ei vielä ollut musiikkeja eikä ääniä.

# 7 Yhteenveto ja pohdinta

2D-pelien saralla tasohyppelypelit ovat hyvin perinteinen genre. Vaikka kaikissa peleissä pätevät monet samat säännöt pelinkehityksen kannalta, on silti monia seikkoja, jotka on hyvä tiedostaa suunnitellessaan 2D-tasohyppelypeliä. Peleihin voi tehdä yhtä mukaansatempaavat tarinat tai samaistuttavat hahmot, mutta pelien mekaniikat riippuvat paljon siitä, onko peli 2D vai 3D ja mille alustalle peli on tehty pelattavaksi.

Vastustajat ovat merkittävässä roolissa luomassa peleistä mieleenpainuvaa kokemusta. Vastustajan suunnittelussa on teoriassa monta asiaa, miten saadaan luotua hyvä tai huono vastustaja. Loppujen lopuksi ratkaisee se, että miten hyvin vastustaja sopii pelin teemaan ja tarinaan sekä millä tavalla se palkitsee pelaajan. Vastustaja yleensä kruunaa pelin, joten sen täytyy olla pelaajalle hyvä kokemus.

Projektityössä itse suunnitteluvaihe tuntui haastavimmalta, koska vastustaja piti saada sopimaan muun pelin kanssa yhteen ja sen piti olla tarpeeksi yksinkertainen pelin mekaniikkoihin ja aikarajaan nähden. Litter Runissa siivotaan ihmisten sotkemaa ympäristöä. Lopuksi pelaaja kohtaa ihmisen, joka halveksuu pelaajan siivousyritystä ja haluaa mieluummin heittää roskansa maahan. Kun pelaaja saa siivottua hänen ja ystäviensä kotiympäristön, muut metsän eläimet tulevat onnellisiksi ja pitävät pelaajaa sankarina.

Peliprojekti on toteutettu kokonaisuudessaan Godot Enginellä, joten Godot oli jo tuttu työkalu koko pelinkehityksen ajalta. Valitsimme Godotin projektiamme varten, koska Godot on kevyt ja ilmainen ohjelma, sekä siihen löytyy hyvät dokumentaatiot Godotin nettisivuilta. Godotia oli mukava käyttää ja se sopi erinomaisesti tämänkaltaiseen peliprojektiin.

Lähdeluettelo

Agriogianis, T. (2018). *The Roles, Mechanics, and Evolution of Boss Battles in Video Games.* Undergraduate Honors College Theses 2016-. 45.

BaM Animation. [*GOOD vs BAD Character Design: Tips and Tricks!* (5.3.2020)]. (video) Saatavilla 28.10.2020. https://www.youtube.com/watch?v=8wm9ti-gzLM&t=1097s

Blizzard Entertainment. (2004). *World of Warcraft.* Blizzard Entertainment.

Buckley, I. (25. 4 2019). *10 Reasons to Use Godot Engine for Developing Your Next Game*. Saatavilla 2.11.2020. https://www.makeuseof.com/tag/reasons-godot-engine-game-development/

Design Doc. [*Boss Battle Design Vol. 2 – Designing Engaging Boss Fights in Games* (3.10.2019)]. (video). Saatavilla 11.10.2020. https://www.youtube.com/watch?v=o0CKHLumTG0

de Wit, R. (2015). *Rob de Wit Game Design*. Saatavilla 11.10.2020. http://www.rob-wit.com/boss-battle-research-concept/

Docter, P. (Ohjaaja). (2015). *Inside Out - Mielen sopukoissa* [Elokuva].

Eurocom. (2004). *Spyro: A Hero's Tail*. Vivendi Universal Games.

FromSoftware. (2011). *Dark Souls.* Namco Bandai Games.

Fudj. [*Storytelling in Platformers* (7.1.2020)]. (video). Saatavilla 29.9.2020. https://www.youtube.com/watch?v=L17mtUWn0vA&t=758s

Game Dev Underground. [*My Level Design Philosophy + Tips For Designing Levels* (25.11.2017)*].* (video). Saatavilla 30.9.2020. https://www.youtube.com/watch?v=HyLL0W4mHnc

Gamerforlife. (2004-2020). *Platforming games 101: running, jumping & more*. Saatavilla 30.9.2020. http://www.racketboy.com/retro/platforming-games-101-all-you-need-to-know

Gorguinni. [*The Originality Between 2D and 3D Bosses* (20.10.2020)]. (video). Saatavilla 4.11.2020. https://www.youtube.com/watch?v=mk2UzKGbXo0

Houghton, D. (16. 10 2013). *8 tired boss fight tropes that need to die*. Saatavilla 16.10.2020. https://www.gamesradar.com/8-boss-fight-tropes-need-die/

Jonkers, D. (4. 7 2011). *How to design levels for a platformer*. Saatavilla 30.9.2020. http://devmag.org.za/2011/07/04/how-to-design-levels-for-a-platformer/

Kemppainen, J. (2019). *Pelisuunnittelun peruskirja.* Aviador Kustannus.

Khalifa, A. (10. 6 2019). *Level Design Patterns in 2D Games*. Saatavilla 30.9.2020. https://www.gamasutra.com/blogs/AhmedKhalifa/20190610/344344/Level_Design_Patterns_in_2D_Games.php

KidsCanCode. (2019). *AI/Behavior*. Saatavilla 20.10.2020. https://kidscancode.org/godot_recipes/ai/

Kramarzewski, A.;& De Nucci, E. (2018). Teoksessa A. Kramarzewski;& E. De Nucci, *Practical Game Design.* Birmingham: Packt Publishing.

Lee, T. (28. 9 2015). *An annonated history of video game boss battles*. Saatavilla 11.10.2020. https://www.polygon.com/features/2015/9/28/9333685/annotated-history-boss-battles

Linietsky, J.;& Manzur, A. (2007-2020). *Features.* Saatavilla 18.10.2020. https://godotengine.org/features

Linietsky, J.;& Manzur, A. (2014-2020b). *Instancing*. Saatavilla 18.10.2020. https://docs.godotengine.org/en/latest/getting_started/step_by_step/instancing.html

Linietsky, J.;& Manzur, A. (2014-2020d). *Navigation2D*. Saatavilla 19.10.2020. https://docs.godotengine.org/fi/stable/classes/class_navigation2d.html

Linietsky, J.;& Manzur, A. (2014-2020a). *Scenes and nodes*. Saatavilla 18.10.2020. https://docs.godotengine.org/en/latest/getting_started/step_by_step/scenes_and_nodes.html

Linietsky, J.;& Manzur, A. (2014-2020c). *Scripting*. Saatavilla 18.10.2020. https://docs.godotengine.org/en/stable/getting_started/step_by_step/scripting.html

Linietsky, J.;& Manzur, A. (2014-2020e). *State design pattern*. Saatavilla 19.10.2020. https://docs.godotengine.org/en/stable/tutorials/misc/state_design_pattern.html

Linietsky, J.;& Manzur, A. (2014-2019). *Your First Game*. Saatavilla 19.10.2020. https://docs.godotengine.org/en/3.1/getting_started/step_by_step/your_first_game.html

Minotti, M. (7. 10 2017). *Cuphead review — a uniquely beautiful and worthwhile challenge*. Saatavilla 10.10.2020. https://venturebeat.com/2017/10/07/cuphead-review-a-uniquely-beautiful-and-worthwhile-challenge/

Nevala, L.;Saajanne, M.;& Vikstedt, S. (2020). *Litter Run*.

Nintendo EAD. (1985). *Super Mario Bros.* Nintendo.

Playtonic Games team. (13. 5 2015). *6 musts for a perfect platformer, from the Yooka-Laylee team*. Saatavilla 30.9.2020. https://www.gamasutra.com/view/news/243310/6_musts_for_a_perfect_platforme

Russel, D. (2. 2 2011). *Video game user interface design: Diegesis theory*. Saatavilla 2.10.2020. http://devmag.org.za/2011/02/02/video-game-user-interface-design-diegesis-theory/

Snoman Gaming. [Good Game Design – Bosses (26.11.2016)]. (video). Saatavilla 17.10.2020. https://www.youtube.com/watch?v=YmwLPF11eos&t=180s

Stout, M. (2020). *Boss Battle Design and Structure*. Saatavilla 17.10.2020. https://www.gamasutra.com/view/feature/134503/boss_battle_design_and_structure.php?print=1

Studio MDHR. (2017). *Cuphead*. Studio MDHR.

Ubisoft Montpellier. (24. 6 2014). *Valiant Hearts: The Great War*. Ubisoft.

UmaiPixel. [*Godot 3 – Platformer Tutorial – Part 8 – We Need Enemies* (21.9.2018)]. (video). Saatavilla 19.10.2020. https://www.youtube.com/watch?v=4HLUu_CB1Kw

Wikipedia. (13. 10 2020). *Bowser (character)*. Saatavilla 11.10.2020. https://en.wikipedia.org/wiki/Bowser_(character)

Vulture. (2020). *The 100 Hardest Video-Game Bosses, Ranked By Difficulty*. Saatavilla 16.10.2020. https://www.vulture.com/article/100-hardest-video-game-bosses-ranked.html#comments

Vuorela, V. (2007). *Pelintekijän käsikirja.* Helsinki: BTJ Finland Oy.