Ville Hankipohja

Kenttägeneraattori Zombiefall -peliin

Tradenomi

Tietojenkäsittely

Syksy / 2017

KAJAANIN
AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Tiivistelmä

Tekijä: Hankipohja Ville

Työn nimi: Kenttägeneraattori Zombiefall -peliin

Tutkintonimike: Tradenomi(AMK), tietojenkäsittely

Asiasanat: peliohjelmointi, pelisuunnittelu, mobiilipelit, Android, Unity3D

Opinnäytteen tarkoitus oli luoda kenttägeneraattori mobiilipeliin nimeltä Zombiefall. Generaattorin tuli luoda pelin tarvitsema pelikenttä dynaamisesti pelin aikana. Projektin aikana luotiin kenttägeneraattorin koodi sekä rakennettiin kenttäpalaset, joita generaattori hyödyntää pelikentän luomiseen.

Generaattori ja kenttäpalaset luotiin neljässä vaiheessa. Ensimmäisessä luotiin yksinkertainen generaattori sekä peruspalaset ja toisessa vaiheessa hiottiin generaattoria toimimaan nopeasti mobiilialustoilla. Kolmannessa finalisoitiin kenttäpalaset ja neljännessä lisättiin generaattoriin ominaisuuksia.

Projektin lopputuloksena saatiin toimiva kenttägeneraattori, joka luo pelin aikana yhtenäiseltä vaikuttavan pelialueen, vaikka se rakennetaan useista erillisistä osista. Uusien kenttäpalasten lisääminen on helppoa, koska generaattori hakee käyttöönsä kaikki saatavilla olevat kenttäpalaset automaattisesti pelin alussa.

Abstract

Author: Hankipohja Ville

Title of the Publication: Level Generator for Zombiefall Game

Degree Title: Bachelor's Degree in Information Technology

Keywords: game programming, game design, mobile games, Android, Unity3D

The objective of this thesis was to create a level generator for a mobile game called Zombiefall. The generator was supposed to create the playable game area dynamically during the game. During the project the level generator code was created along with the level parts, which the generator used to create the game area.

The generator and the level parts were created in four phases. A simple generator and basic level parts were created during the first phase. During the second phase, the generator was polished to work on mobile platforms. During the third phase the level parts were finalised and during the fourth phase the generator received more features.

The result of the project was a fully functional game generator, which creates a continuous game area during gameplay, even though it is created from separate parts. Adding new level parts is easy, because the generator retrieves all available level parts automatically at the beginning of the game.

Sisällys

Terminologia

Android: Google LLC -yrityksen valmistama mobiilikäyttöjärjestelmä.

Endless Runner: Pysäyttämättömät pelit määritellään kahdella tavalla: pelaajan ohjaaman hahmon eteenpäin suuntautuvaa vauhtia ei voi pysäyttää ja sillä on jalat. Yleisesti ottaen niitä kuvataan 2D-näkökulmasta sivulta. Ohjaus on yleensä yksinkertaista, keskittyen esimerkiksi esteiden yli hyppäämiseen. (Endless Runner, GiantBomb.com, 2017).

Gamification / pelillistäminen: erilaisten yksitoikkoisten toimintojen muuttaminen pelimäiseen muotoon, jossa aivoja käytetään luovaan, leikittelevään toimintaa. (Juho-Matti Paavola, 2011).

Kenttägeneraattori: järjestelmä, joka luo pelissä käytettävän pelikentän dynaamisesti ennalta määritettyjen sääntöjen mukaan.

iOS: Apple Inc. -yrityksen valmistama mobiilikäyttöjärjestelmä.

Pooling / poolaus: termillä tarkoitetaan objektien uusiokäyttöä niiden tuhoamisen ja uudelleenluomisen sijaan.

Script / skripti: ohjelmointityön sisältävä tekstitiedosto, mikä sisältää vain tavallista, muokkaamatonta tekstiä. Erilliset ohjelmointiin tarkoitetut ohjelmat tarjoavat ohjelmointia avustavia työkaluja, mutta varsinainen ohjelmointityö voidaan tarvittaessa suorittaa käytännössä millä tahansa tekstinkäsittelyohjelmalla.

Zombie: Termillä tarkoitetaan epäkuollutta ihmistä, joka on noussut kuolleista, mutta ei pysty enää toimimaan normaalin ihmisen tavoin. Useimmiten zombie on hidas ja tyhmä olento, jonka tarkoitus on syödä eläviä ihmisiä. (Better Off Dead: The evolution of the Zombie as Post-Human, 2011).

Zombiefall: Zaibatsu Interactive -yrityksen luoma, “endless runner” -genreen kuuluva mobiilipeli.

1 Johdanto

Opinnäytteen aiheena on luoda kenttägeneraattori Zombiefall -peliin. Tarkoituksena on ohjelmoida generaattori, joka dynaamisesti luo pelin tarvitsemat kentät ja optimoida se toimimaan mobiilialustoilla.

Tekstissä opinnäytteen ja kenttägeneraattorin tekijään viitataan termillä ”tekijä”. Zaibatsu Interactive Oy:stä käytetään nimeä Zaibatsu.

1.1 Zaibatsu Interactive Oy

Zaibatsu on vuonna 2014 perustettu jyväskyläläinen peliyritys. Yritys perustettiin kuuden henkilön toimesta, mutta on sittemmin kasvattanut henkilökuntaansa kymmeneen.

Zaibatsun yritysmalli on luoda pelejä, mutta he tarjoavat myös palveluja pelillistämiseen ja ohjelmien valmistukseen. He hyödyntävät pelisuunnittelun vahvoja elementtejä luodakseen kiehtovaa sisältöä (Zaibatsu Interactive – About Us, 2017).

Zaibatsun ensimmäinen peli on nimeltään Elder Goo ja se julkaistiin 2015 yhteistyössä Yleisradio Oy:n kanssa nimellä Elder Goo - Möllit.

1.2 Tehtävänanto

Vuoden 2017 alussa Zaibatsu Interactive valitsi Zombiefall -pelin seuraavaksi julkaistavaksi pelikseen. Tekijä valittiin projektiin kenttäsuunnittelijaksi ja kenttägeneraattorin tekijäksi.

Palaverointi Zaibatsun henkilökunnan kanssa, ja erityisesti projektin vetäjän Wille Hujasen kanssa, antoi hyvän pohjan sille, millaisia toiveita heillä oli kenttägeneraattorin suhteen. Pääasiallisena toiveena oli nopeasti ja luotettavasti toimiva generaattori, koska pelin alustana tulisi olemaan mobiilialustat, tarkemmin Google Android sekä Apple iOS.

Ohjeeksi annettiin tutustua suosittuihin, samaan genreen (”endless runner”) kuuluviin mobiilipeleihin ja selvittää, miten näiden kenttägenerointi on toteutettu.

1.3 Motivaatio opinnäytetyön takana

Tekijä oli harrastanut ohjelmointia jo vuosia ennen kuin haki opiskelemaan Kajaanin ammattikorkeaan. Hän oli myös suunnitellut erilaisia pelejä käytännössä koko ikänsä; nuorempana lauta-, kortti- ja roolipelejä, vanhempana tietokone- ja mobiilipelejä. Kun Zaibatsun edustaja ehdotti kenttägeneraattorin ja siihen liittyvien kenttäpalasten luomista, projekti tarjosi mahdollisuuden yhdistää nämä kaksi intohimoa.

Kenttägeneraattori itsessään oli käytännössä puhdas ohjelmointityö, mutta sen toimintaan tarvittiin kenttäpalasia, joiden avulla generaattori luo varsinaisen pelikentän. Kenttäpalasten luominen tarjosi ohjelmoinnista eroavaa suunnittelutyötä.

Näiden kahden erilaisen osuuden luominen samaan projektiin auttoi pitämään motivaation yllä, koska tekijä ei missään vaiheessa ehtinyt kyllästyä jatkuvaan saman asian tekemiseen.

1.4 Käytetyt työkalut

Kenttägeneraattorin ja kenttäpalasten luomiseen käytettiin useita työkaluja, minkä lisäksi projektin aikana käytettiin tiettyjä työkaluja kommunikointiin ja tiedostojen jakamiseen.

1.4.1 Unity

Unity Technologiesin valmistama Unity-moottori oli pääasiallinen työkalu generaattorin tekemisessä. Zombiefall -peli tehtiin käyttäen Unityä, joten generaattori tehtiin luonnollisesti samalla moottorilla.

Unity on saatavilla Windows ja Mac -pohjaisille tietokoneille. Se sisältää valikoiman sekä graafikkoystävällisiä työkaluja immersiivisten kokemusten ja pelimaailmojen luomiseen että vahvoja työkaluja pelilogiikan ja huippuluokan suorituskykyisen pelattavuuden lisäämiseen (Unity Products, 2017).

Pelin tekeminen aloitettiin Unityn versiolla 5.1.1, mutta projektin kuluessa se päivitettiin ensin versioon 5.6.0 ja lopuksi versioon 2017.1. Unity Technologies muutti 5.6 -version jälkeen versiointipolitiikkaansa, mistä johtuu numerointityylin vaihtuminen.

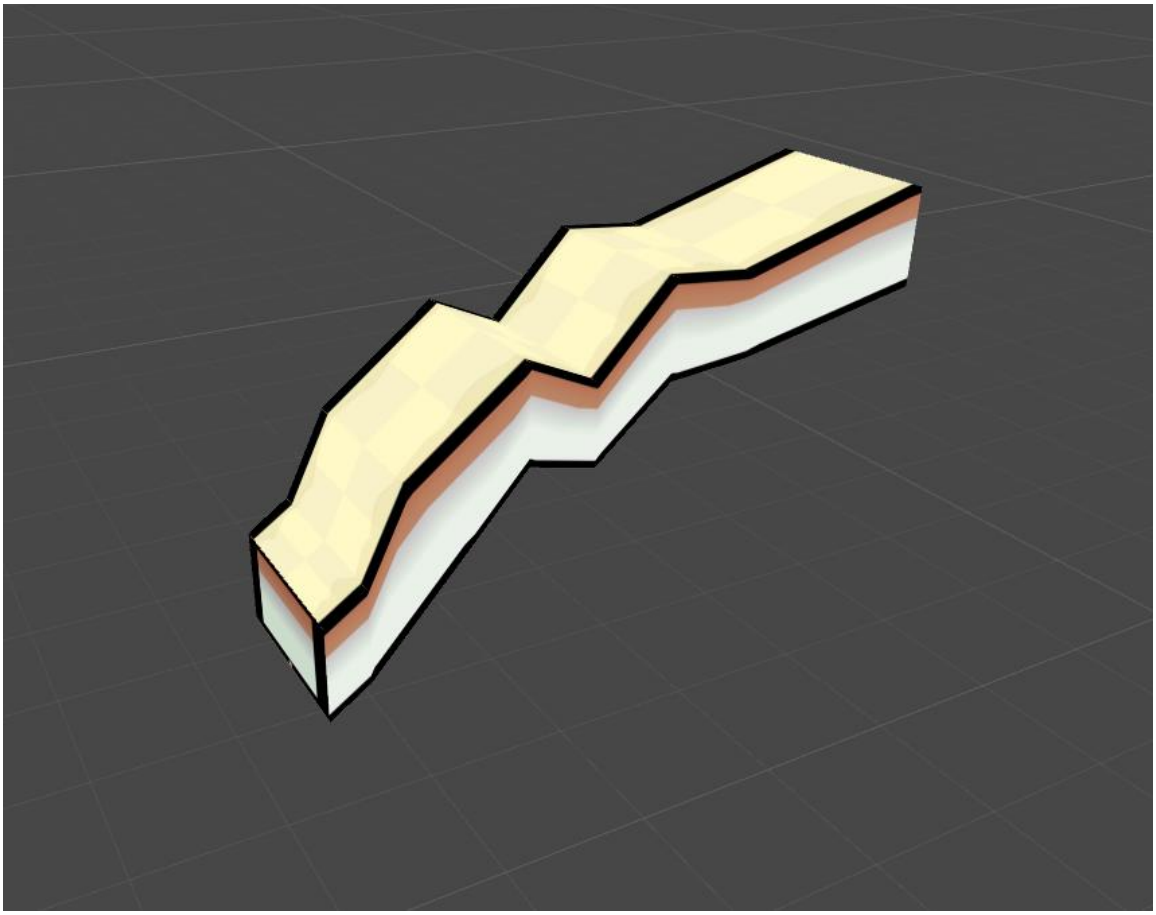
1.4.2 Visual Studio

Ohjelmointityössä käytettiin Microsoftin Visual Studio -ohjelmointiympäristöä. Visual Studio mahdollistaa koodin kirjoittamisen tarkasti ja tehokkaasti menettämättä nykyistä tiedostoyhteyttä (Visual Studio IDE, 2017).

Projekti aloitettiin versiolla 2015, mutta päivitettiin loppuajaksi versioon 2017.

1.4.3 Mesh Deformer

Generaattorin käyttämät kenttäpalaset vaativat tietynlaista vääntelyä sopiakseen pelin visuaaliseen ilmeeseen, joten tehtävään valittiin Unityn Asset Storesta löytynyt Mesh Deformer -työkalu. Kuvassa 1. on Mesh Deformerilla muokattu kuutio.



Kuva 1: Yksikertainen kolmiulotteinen kuutio on muutettu pelin visuaalisen ilmeen vaatimaan muotoon Mesh Deformerin avulla.

1.4.4 Flowdock

Flowdock on selaimessa toimiva keskustelutyökalu, mitä Zaibatsu käyttää kommunikointiin yrityksen tekijöiden kesken.

Yksittäistä ”flowta” voidaan ajatella ryhmähuoneena. Jokainen ”flow” muodostuu tekstikeskustelusta ja ryhmän postilaatikosta. Peukalosääntönä on hyvä idea luoda oma ”flow” jokaiselle ryhmälle. (Help | Flows, 2017).

1.4.5 Trello

Trello on yhteistyötyökalu, jolla voidaan järjestää projektit tauluihin. Trello toimii Flowdockin tapaan selaimessa ja sen avulla voidaan helpottaa projektin tehtävien suunnittelua. Zaibatsu käyttää Trelloa projektin ylläpitoon ja tehtävien jakamiseen tekijöiden kesken.

Yhdellä vilkaisulla Trello kertoo, mitä ollaan tekemässä, kuka tekee mitäkin ja missä kohdin jotain on kesken. (What is Trello? – Trello Help, 2017).

1.4.6 Dropbox

Dropbox on pilvitallennuspalvelu, jonka avulla voidaan myös jakaa tiedostoja käyttäjien kesken.

Dropboxin käyttäminen tietokoneessa on kuin käyttäisi tavallisia kansiota, paitsi Dropbox -kansioon vetämäsi tiedostot synkronoidaan automaattisesti verkkopalveluun sekä muihin tietokoneisiin tai mobiililaitteisiin, jotka on liitetty tiliisi. Sovellus toimii taustalla ja automattisesti synkronoi tiedostot ja pitää ne tallennettuna verkossa. On kuin sama kansio olisi saatavilla kaikilla tietokoneilla ja mobiililaitteilla samanaikaisesti. (What is the Dropbox desktop application?, 2017).

1.4.7 Bitbucket

Bitbucket on Git-pohjainen versionhallintajärjestelmä, jota Zaibatsu käyttää jakaakseen projektitiedostot kehittäjien kesken.

Git on ilmainen ja avoimen lähdekoodin versionhallintajärjestelmä, jonka loi alun perin Linus Torvalds vuonna 2005. Toisin kuin keskitetyt versionhallintajärjestelmät Git on hajautettu: jokaisella kehittäjällä on täysi historia omasta paikallisesta koodivarastostaan. Tämä tekee alkuperäisen varaston kopioimisesta hitaampaa, mutta myöhemmät operaatiot ovat huomattavasti nopeampia. (Git basics, 2017).

1.4.8 Sourcetree

Opinnäytteen tekijä käytti Sourcetree -ohjelmaa siirtääkseen tietojaan Bitbucketin ja paikallisen tietokoneen välillä.

Sourcetree yksinkertaistaa sen, kuinka olet vuorovaikutuksessa git-varastojesi kanssa, jotta voit keskittyä ohjelmointiin. Visualisoi ja hallitse varastojasi Sourcetreen yksinkertaisen graafisen käyttöliittymään kautta. (SourceTree, 2017).

2 Zombiefall -peli

Keväällä 2017 Zaibatsu päätti, että heidän seuraava julkaisunsa on Zombiefall -niminen ”endless runner” genreen kuuluva peli.

2.1 Pelin perusmekaniikat

Zombiefallin perusmekaniikka on zombi-hahmon putoaminen päättymättömän ostoskeskuksen läpi. Pelaaja ohjaa zombia koskettamalla älypuhelimen ruudun vasenta tai oikeaa laitaa, jolloin zombi liikkuu haluttuun suuntaan. Pelaajan tarkoituksena on saada zombi etenemään mahdollisimman pitkälle.

Jos zombi osuu ostoskeskuksessa asioiviin ihmisiin, ne muuttuvat zombeiksi, joita pelaaja voi myös ohjata. Pelaajalla on näin ollen pelin aikana usein ohjattavanaan useampi zombi samaan aikaan. Kuvassa 2. on esimerkkitilanne, jonka hetkellä pelaajalla on hallittavanaan viisi zombia. Kaikki zombit reagoivat pelaajan kosketukseen samalla tavalla ja ne liikkuvat aina yhdessä samaan suuntaan. Yksittäisen zombin kuolema ei tässä tapauksessa haittaa, koska peli päättyy vasta, kun kaikki pelaajan ohjaamat zombit ovat kuolleet.

Putouksen aikana pelaaja kerryttää pisteitä sen mukaan montako zombia hänellä on hallussaan. Zombin tartuttamat ihmiset toimivat kertoimena pistelaskimelle, mikä kerryttää pisteitä putoamisen metrimäärän mukaan.

Zombi kuolee osuessaan kentän ylä- ja alalaitaan tai kentällä oleviin vaaroihin, kuten sähkökaappeihin, jolloin peli päättyy. Pelaajalla on käytössään yksi jatkomahdollisuus katsomalla joko mainos tai käyttämällä valuuttaa, jota saa kerättyä kentältä ja ostettua pelin sisäisestä kaupasta.

Kuva 2: Pelikuva tilanteesta, jossa pelaajan hallinnassa on viisi zombia.

2.2 Kenttägeneraattorin osuus pelistä

Kenttägeneraattorin tarkoitus on luoda päättymätön ostoskeskus ennalta rakennetuista kenttäpalasista. Tarkoitus on luoda pelaajalle illuusio siitä, että kenttä jatkuu loputtomiin, eikä pelaaja huomaa kenttäpalasten reunoja.

Generaattori luo siis varsinaisen pelikentän, missä pelaaminen tapahtuu. Sen toimintavarmuus on tärkeässä osassa pelin toimintaa, koska huonosti toimiva generaattori suoraan haittaisi pelaamisen nautintoa.

2.3 Yhteistyö muiden projektissa olevien kanssa

Pääasiallinen yhteistyö tapahtui projektin pääohjelmoijan Tarmo Jussilan kanssa. Yhteistyö sujui erinomaisesti ja Tarmo antoi jatkuvasti hyvää palautetta projektin etenemisestä. Tekijän osuus oli alun perin rajattu vain kenttägeneraattoriin, mutta hän auttoi Tarmoa myös muissa tehtävissä aina tarpeen vaatiessa.

Projektin vetäjänä toimi Zaibatsun Art Director Wille Hujanen, jolta sai kannustavaa kritiikkiä koko projektin ajan. Kommunikaatio hänen kanssaan tapahtui pääasiallisesti Flowdockin ja Trellon välillä, koska hän ei työskennellyt Zaibatsun toimistolla projektin aikana. Kasvokkain tapahtuva kommunikointi rajoittui joka toisella viikolla tapahtuvaan koko tiimin käsittävään palaveriin, mutta se ei varsinaisesti haitannut tekijän toimenkuvaa.

Projektin QA:ta (Quality Assurance; laadunvarmistus) teki Jussi Perttola, jolta myös sai hyvää palautetta sen mukaan toimiko pyydetty toiminto halutulla tavalla.

Projektin graafikkoina toimivat Joni Suhonen ja Timo Koski, joiden kanssa toimittiin vähemmän suorassa yhteistyössä. He toimittivat aina tekijän tarvitseman materiaalin ajallaan ja vaatimusten mukaisesti. Projektin alussa mukana oli graafikko Markus Palomäki, mutta hän jäi pois kevään 2017 aikana ja Timo Koski tuli hänen tilalleen. Markus loi ensimmäiset kenttäobjektit tekijän toiveiden mukaisesti generaattorin alkuperäisen testaamisen aikana.

3 Kenttägeneroinnin teoriaa

Ennen projektin aloitusta, tekijä tutustui Zombiefall -pelin genreen kuuluviin peleihin ja etsi niiden mahdollista taustamateriaalia projektin tueksi. Tällaisen materiaalin löytäminen osoittautui vaikeaksi, joten etsintäparametrejä laajennettiin käsittämään eritasoisia tutoriaaleja antamaan ideaa generaattorin kehitykseen.

3.1 Genren peleihin tutustuminen

Löytääkseen suosittuja pelejä tekijä etsi internetistä listoja, joissa esiteltiin parhaita ”endless runner” -genreen kuuluvia Android -pelejä ja vertasi niiden yhtäläisyyksiä. Näiden listojen varsinainen luotettavuus ei ollut merkityksellistä, koska tekijä yritti vain etsiä suosittuja pelejä. Tarkoitus ei ollut tutkia Android -alustan kaikkein suositumpia pelejä, vaan yksinkertaisesti löytää pelejä, joita pelataan paljon. Siihen tarkoitukseen tällaiset listat olivat omiaan.

Yleisimmät listoilla olevat pelit vaikuttivat olevan Alto’s Adventure, Crossy Road, Jetpack Joyride ja Subway Surfers. Android Play -kauppaa tutkimalla varmistettiin, että näillä peleillä oli runsaasti latauksia ja niitä voitiin käyttää pohjaideana tälle opinnäytteelle.

3.2 Pelien taustamateriaaliin tutustuminen

Edellä mainittujen pelien taustamateriaalin löytyminen osoittautui odotetusti vaikeaksi prosessiksi, joten tekijä päätti ottaa mukaan tutoriaaleja ja muita vastaavia dokumentteja, jotka saattaisivat antaa ideoita kenttägeneraattorin luomiseen tämän kaltaiseen peliin.

3.2.1 Depth in Simplicity: The Making of Jetpack Joyride

Vuoden 2012 Game Developers Conferencessa Luke Muscat piti ”Depth in Simplicity: The Making of Jetpack Joyride” -presentaation, jossa tämä kertoi Jetpack Joyride -pelin tekemisen taustoista. Opinnäytteen kannalta mielenkiintoisinta esityksessä oli kappale

”Procedural Level Generation”, missa Muscat kertoo miksi he päättivät pelin kenttägeneroinnin olevan dynaamisesti luotua eikä ennalta suunnittelijan käsin tekemää. Hän myös käsitteli sitä, miten he päättivät esteiden ilmaantumisen tiettyjen aikavälien mukaan.

Kenttägeneraattorin luominen vie projektin alussa enemmän aikaa, mutta projektin loppuvaiheessa proseduraalisesti luodut kentät helpottavat suunnittelijan työtä, koska tämän ei tarvitse enää käyttää kaikkea aikaansa uusien kenttien luomiseen (Luke Muscat, 2012).

Esteiden ilmaantuminen tasaisin väliajoin tekee pelistä helposti tylsän, mutta myös täysin satunnainen ilmaantuminen tuo omat ongelmansa. Välillä pelissä saattaa tulla pitkiä aikoja, jolloin esteitä ei ole laisinkaan, mutta sitten vastaan tulee kolme melkein päällekkäistä estettä. Näin ollen Jetpack Joyrideen luotiin intervalli-systeemi, mikä määrittää lyhyimmän mahdollisen ajan esteiden välille sekä sen miten pitkä aika eri esteiden välillä voi enintään olla. (Luke Muscat, 2012).

3.2.2 Creating an infinite 3D runner game in Unity

Dimitris-Ilias Gkanatsios teki vuonna 2016 tutoriaalin, jossa hän kertoo, miten Unityllä saadaan luotua Temple Runin tai Subway Surfersin kaltainen ”endless runner”. Tutoriaali oli mielenkiintoinen ja tarjosi hyviä ideoita siitä, miten kenttägeneraattorin voisi luoda.

3.3 Johtopäätösten tekeminen

Jetpack Joyride -presentaatio vahvisti tekijän käsitystä, että kenttägeneraattorin luomisella Zombiefalliin oli selkeä tarkoitus, jotta suunnittelijan työ helpottuisi pelin tekemisen loppuvaiheessa. Esteiden ilmaantuminen tietyin aikavälein myös antoi ideaa siihen, miten Zombiefallin kenttäpalasten eri vaikeustasoja voitaisiin käsitellä pelin aikana. Zombiefalliin luotiin systeemi, jolla tarkistettiin, miten monta kenttäpalasta oli käyty läpi ja sen mukaan päätettiin, mikä vaikeustaso oli kulloinkin käytössä.

Gkanatsioksen tutoriaalia ei voinut suoraan hyödyntää, koska pelien toimintapa oli niin erilainen, mutta siitä sai ideoita omaan kenttägeneraattoriin. Tutoriaalissa uudet kenttäpalat ilmaantuvat kentälle, kun pelaaja osuu tiettyyn, pelaajalle näkymättömään, objektiin. Zombiefallissa vastaava toiminto toteutettiin suoraan tarkistamalla pelaajan sijainti kenttäpalasiin nähden.

4 Kenttägeneraattorin toteuttaminen

Kenttägeneraattori toteutettiin pääasiallisesti neljässä eri vaiheessa. Aluksi tehtiin pohjatyö, jolloin luotiin generaattorin perustoimet. Seuraavaksi se hiottiin mobiilialustoille toimivaksi, minkä jälkeen varsinaiset kenttäpalaset finalisoitiin. Lopuksi generaattoriin lisättiin Zaibatsun toivomat ominaisuudet.

4.1 Pohjatyö

Kenttägeneraattorin luominen aloitettiin luomalla kenttäpalasten pohja. Siitä jatkettiin luomalla varsinaisen generaattorin toiminta, millä peli saatiin testattavaksi. Lisäksi luotiin lisäelementtejä peliä varten, kuten seinät ja taustat. Kenttäpalasille myös määriteltiin erilaiset vaikeusasteet ja mahdollistettiin niiden lataaminen käyttöön automaattisesti pelin aloituksen yhteydessä.

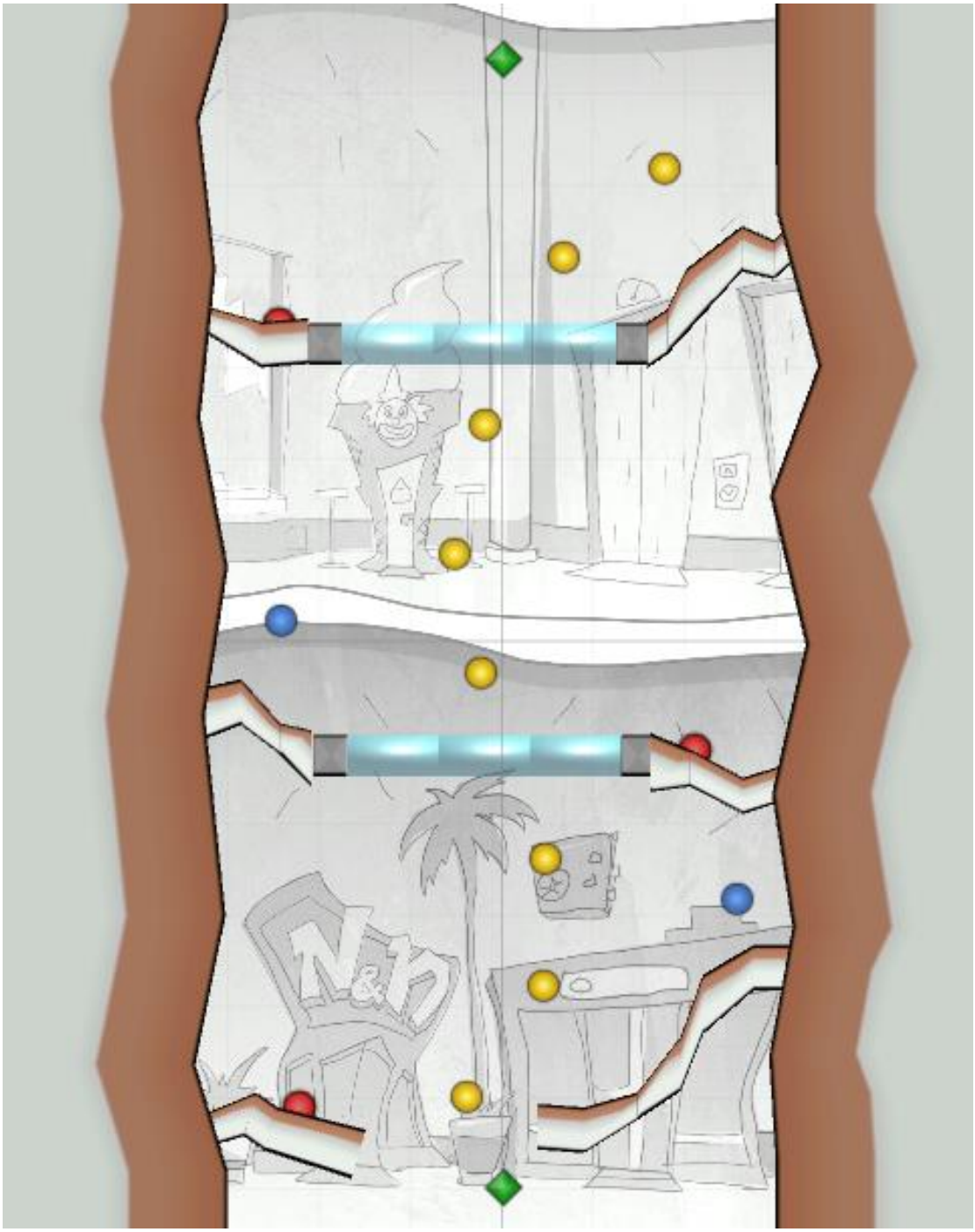
4.1.1 Kenttäpalasten luominen

Kenttägeneraattori -projekti aloitettiin kenttäpalasten pohjan luomisella. Unity -pelimoottorissa voidaan luoda eräänlaisia ”prefab” -objekteja, jotka voidaan rakentaa vapaasti useista täysin erilaisista objekteista. Kenttäpalaset luotiin tällaisina kokoelmaobjekteina, jolloin niihin pystyttiin lisäämään kaikki yksittäiseen kenttäpalaseen kuuluvat tasot ja generaattorin käyttämät merkinnät. Näin ollen varsinaisen pelikentän rakentaminen helpottui, kun peliin voitiin tuoda yksinkertaisesti näitä prefab-objekteja ilman, että jokainen kenttätaso tarvittiin erikseen generoida.

Tärkeimmät asiat kenttäpalasen sisällä olivat ”Linker” ja ”Target” -objektit. Target oli yksinkertaisesti tyhjä objekti, jolla ei ollut muuta tarkoitusta kuin toimia sijoituskohteena kenttägeneraattorille ja se sijoitettiin kenttäpalasen äärimmäiseen alalaitaan. Linker puolestaan sisälsi kaikki kenttäpalasen sisällä olevat objektit ja se laitettiin kentän ylälaitaan.

Kentän generoinnin alussa ensimmäinen kenttäpalanen sijoitetaan tiettyyn, ennalta määritettyyn kohtaan kentässä. Seuraavien kenttäpalasten Linker-objektit kohdistetaan aina edellisen kenttäpalasen Target-objektiin, jolloin saadaan luotua saumaton siirtymä kahden kenttäpalasen välillä.

Kenttäpalasten luominen helpottui näiden kahden objektin myötä, koska Linker ja Target -objektit määrittivät luonnolliset rajat kentälle. Kentän sisältö yksinkertaisesti sijoitettaisiin näiden kahden objektin väliselle alueelle. Kuvassa 3. näkyy kenttäpalasten sisältämät merkit, joiden tilalle generoidaan muita objekteja, kuten kolikoita tai ihmisiä.



Kuva 3: Kenttäpalanen merkkeineen, joiden tilalle generoidaan tietyt objektit.

4.1.2 Varsinainen generaattori

Varsinaista generointia varten luotiin kaksi skriptiä: ”LevelGenerator” ja ”LevelUtility”. Syy näiden erillään pitämiseen oli välttää yhden tiedoston paisumisen liian suureksi.

LevelGeneratorin tarkoitus oli tehdä varsinainen työ eli itse kentän luominen. Se loisi objektit kentälle ja määrittäisi niiden tarkan paikan.

LevelUtility puolestaan sisältäisi generaattorin käyttämät asetukset ja erinäisiä tarvittavia työkaluja. Kaikki, mikä ei varsinaisesti liittyisi kenttäpalasten tai objektien luomiseen, laitettaisiin LevelUtilityyn pois häiritsemästä varsinaista generointia. LevelUtilitystä tehtiin myös ”instance”, jolloin sitä pystyisi kutsumaan mistä tahansa muusta skriptistä. Tällä mahdollistettiin LevelUtilityn sisältämien työkalujen käyttäminen koko pelin läpi.

4.1.3 Lisäelementit

Peliä varten luotiin myös taustaelementit ja seinät. Taustaelementit loivat tunnelmaa, mutta eivät varsinaisesti vaikuttaneet peliin. Seinät puolestaan olivat tärkeä pelillinen elementti.

Seinät loivat luonnollisen putken, mitä pitkin zombi putosi. Alkuperäiset seinät olivat yksinkertaisesti suorat seinämät kentän laidoilla, joilla vain estettiin zombia putoamasta kentän sivuilta ulos. Myöhemmässä vaiheessa kenttiin tulisi vähän muotoa, joten niiden tarvittiin toimivan kenttäpalasten kanssa tarkasti yhteen.

Seinät ja taustaelementit asemoitiin siten, että niiden Linker -objekti sijoitettiin alalaitaan, jolloin ne saatettiin sijoittaa vastaavan kenttäpalasen Target -objektin kohdalle. Tällöin ne täyttivät kenttäpalasen ympäristön halutulla tavalla. Kuvassa 4. näkyvät seinät ja tausta ilman kentällä olevia muita objekteja.

Toinen vaihtoehto olisi ollut laittaa Linker ylälaitaan kenttäpalasten tapaan, mutta tällöin olisi pitänyt sijoittaa seinän/taustan Linker -objekti kenttäpalasen Linker -objektiin, mikä rikkoisi generaattorin sisäisen logiikan.

Kuva 4: Seinät ja tausta.

4.1.4 Vaikeusasteet

Kenttäpalasille määritettiin viisi eri vaikeusastetta numeroilla 1-5. Ensimmäinen on helpoin ja viides vaikeusaste on kaikkein vaikein. Ensimmäisen vaikeusasteen tasoissa ei ole laisinkaan vaaroja ja niiden tarkoitus on toimia välipalasina muiden vaikeusasteiden kenttäpalasten välillä. Näin ollen vältetään liittämästä peräkkäin kaksi vaikeaa kenttää, jolloin pelaaja saa aina pienen hengähdystauon niiden välillä.

Kenttäpalasen valmistuttua sen vaikeus arvioitiin ja siirrettiin sopivaan vaikeusasteeseen. Testauksen yhteydessä kenttäpalanen saatettiin siirtää eri vaikeusasteeseen käyttäjäpalautteen mukaan.

4.1.5 Resources -kansio

Kenttäpalasten luomisen yhteydessä havaittiin nopeasti, että kun kenttäpalasten määrä alkaisi kasvamaan, niiden käsittely muuttuisi huomattavasti vaikeammaksi. Jos niitä varten luotaisiin yksinkertaiset listat, kenttien siirtäminen vaikeusasteesta toiselle olisi vaikeaa ja uusien kenttäpalasten luomisessa tulisi aina muistaa lisätä ne sopivaan listaan.

Tämä ratkaistiin käyttämällä Unityn Resources -kansiota, koska sen sisältä voitiin dynaamisesti ladata objekteja muuttujiin. Kenttäpalaset jaoteltiin Resources -kansion alle vaikeusasteiden mukaisesti, joista ne ladattiin haluttuun listaan pelin alussa. Näin ollen uuden kenttäpalasen luomisessa riitti, kun se luotiin oikeaan kansioon ja peli huolehti lopusta. Myös kenttäpalasten vaihtaminen vaikeusasteesta toiseen oli helppoa, koska kenttäpalanen tuli vain siirtää toiseen kansioon.

4.2 Hiominen mobiilialustoille

Pääasiallinen valmistelu mobiilialustoja varten tehtiin luomalla poolaus-järjestelmä. Järjestelmän tarkoitus oli luoda, poistaa käytöstä sekä tuoda uudelleen käyttöön objekteja, joita kenttägeneraattori tarvitsee.

Kun kenttägeneraattorin kenttään laittama objekti poistui kameran näköpiiristä, se poistettiin käytöstä. Tämä käytännössä tarkoitti, että se yksinkertaisesti piilotettiin, mutta jätettiin alkuperäiseen sijaintiinsa, kunnes sitä jälleen tarvittiin.

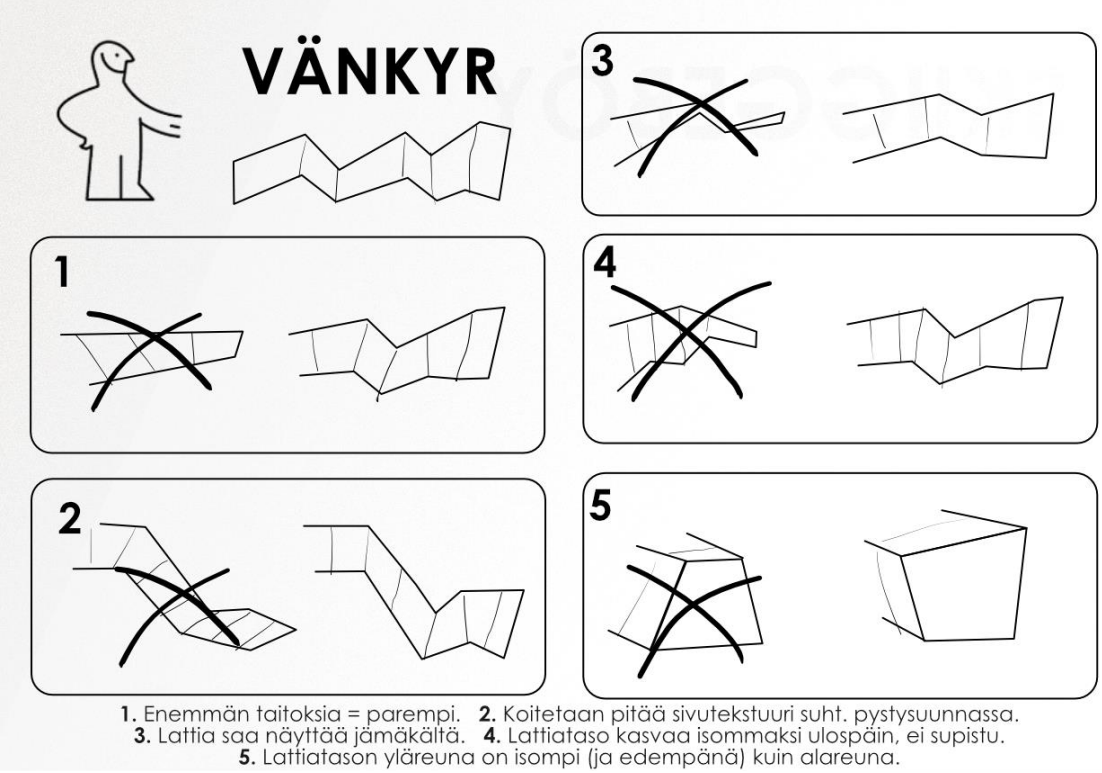
Kun kenttägeneraattori tarvitsi esimerkiksi uutta kenttäpalasta, se pyysi tätä poolaus-järjestelmältä. Jos objektia ei oltu vielä käytetty kertaakaan tai sitä ei ollut sillä hetkellä saatavilla (koska se oli yhä kameran näköpiirissä), järjestelmä loi siitä uuden kappaleen, jolloin kenttägeneraattori saattoi hyödyntää sitä. Jos esine oli jo luotu, käytetty ja poistettu käytöstä, järjestelmä palautti sen aktiiviseksi ja antoi kenttägeneraattorin käytettäväksi, joka siirsi sen oikeaan kohtaan.

Poolausjärjestelmä luotiin LevelUtilityyn, koska se ei suoraan liittynyt varsinaiseen kenttägenerointiin vaan toimi ainoastaan sen tukena.

4.3 Kenttäpalasten finalisointi

Kenttäpalaset finalisoitiin määrittämällä niille koko, minkä sisällä toimittiin. Varsinaiset ja pääasiassa lopulliset kenttäpalaset rakennettiin tässä vaiheessa projektia. Kenttäpalasia viilattiin myöhemmin muun muassa tekstuurien osalta ja säädettiin yksittäisten tasojen tarkempaa muotoa, mutta kenttäpalasia ei enää varsinaisesti enää muutettu isommalla skaalalla. Kuva 5. on pelin graafikon tekemä humoristinen ohjeistus siitä, minkä näköisiä kenttätasoista tulisi tehdä.

Lisäksi kenttäpalasten kääntäminen mahdollistettiin ja niille luotiin erillinen testiobjekti toimintavarmuuden takaamiseksi.



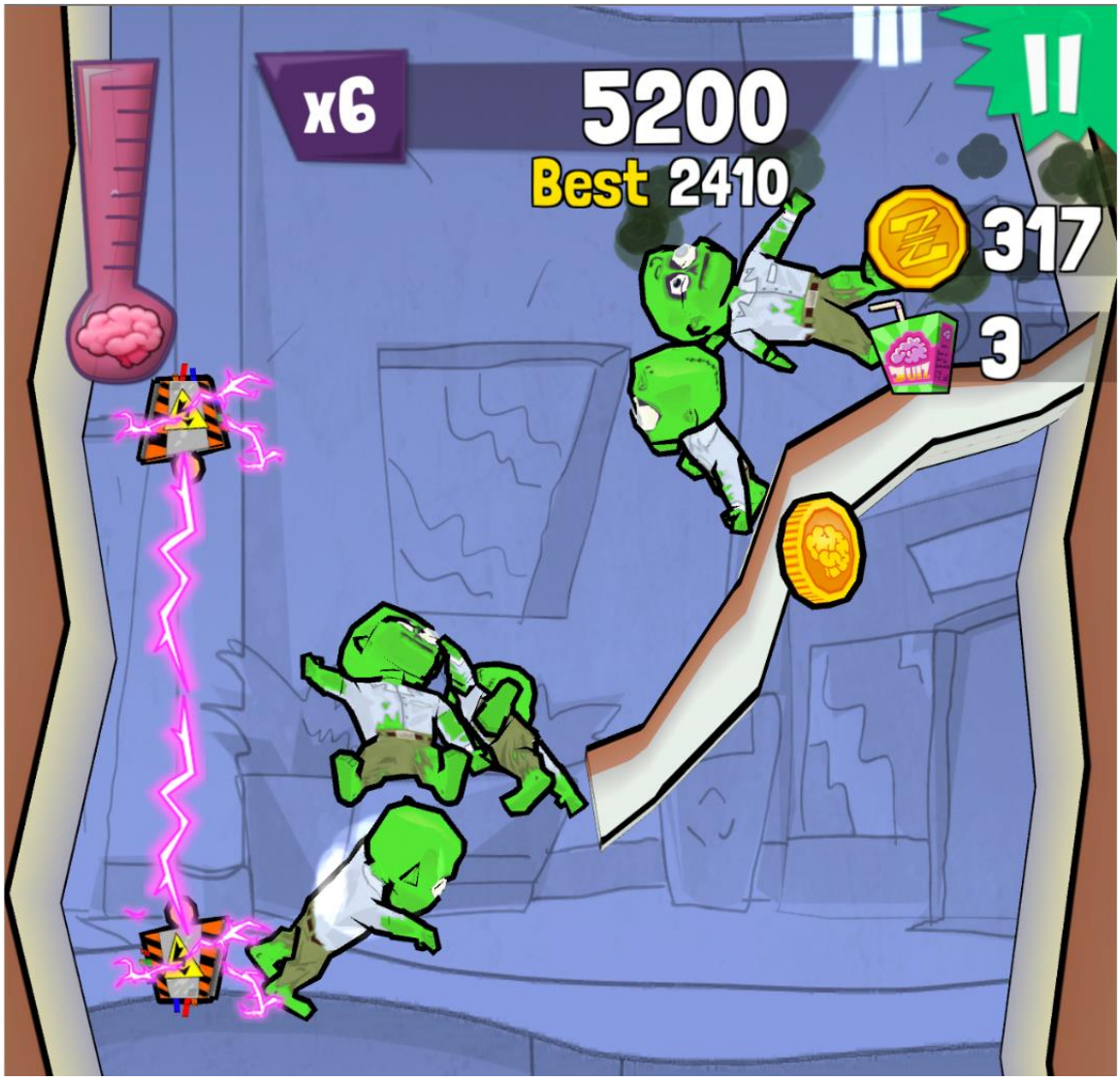
Kuva 5: Graafikko Joni Suhosen tekemä ohjeistus kenttätasojen muotojen määrittämisestä. Se mukailee Ikean huonekalujen kasaukseen annettujen ohjeiden tyyliä.

4.3.1 Kenttäpalasten kääntäminen

Kenttägeneraattoriin lisättiin toimintametodi, jolla kenttäpalasten lukumäärä saatiin käytännössä tuplattua. Tämä toteutettiin kääntämällä satunnaisesti kenttäpalasen Linker -

objekti 180 astetta ympäri, jolloin se tuotiin kentälle peilikuvana. Kenttäpalasten tasot toimivat suoraan tähän tarkoitukseen, koska niillä oli olemassa sekä etu että takapuoli. Eräs tapa optimoida kolmiulotteisia objekteja on poistaa niistä puolet, joita pelaaja ei koskaan näe (kuten laatikon alaosa, joka on lattiaa vasten). Kenttäpalasten tasoista ei ollut näitä poistettu, koska niiden mahdollinen kääntäminen oli otettu huomioon alusta lähtien.

Joidenkin kenttäpalasten sisällä olevia objekteja piti erikseen kääntää kenttäpalasen kääntämiseen jälkeen, koska niillä ei ollut olemassa taustapuolia (esimerkiksi sähkö, mikä esiintyy kuvassa 6, oli esitetty yksinkertaisella tasolla, jossa oli vain yksi puoli). Generaattoriin lisättiin näin ollen toimintoja, joissa käännettiin tarvittavat objektit ympäri, jos kenttäobjekti tuotiin peliin peilikuvana.



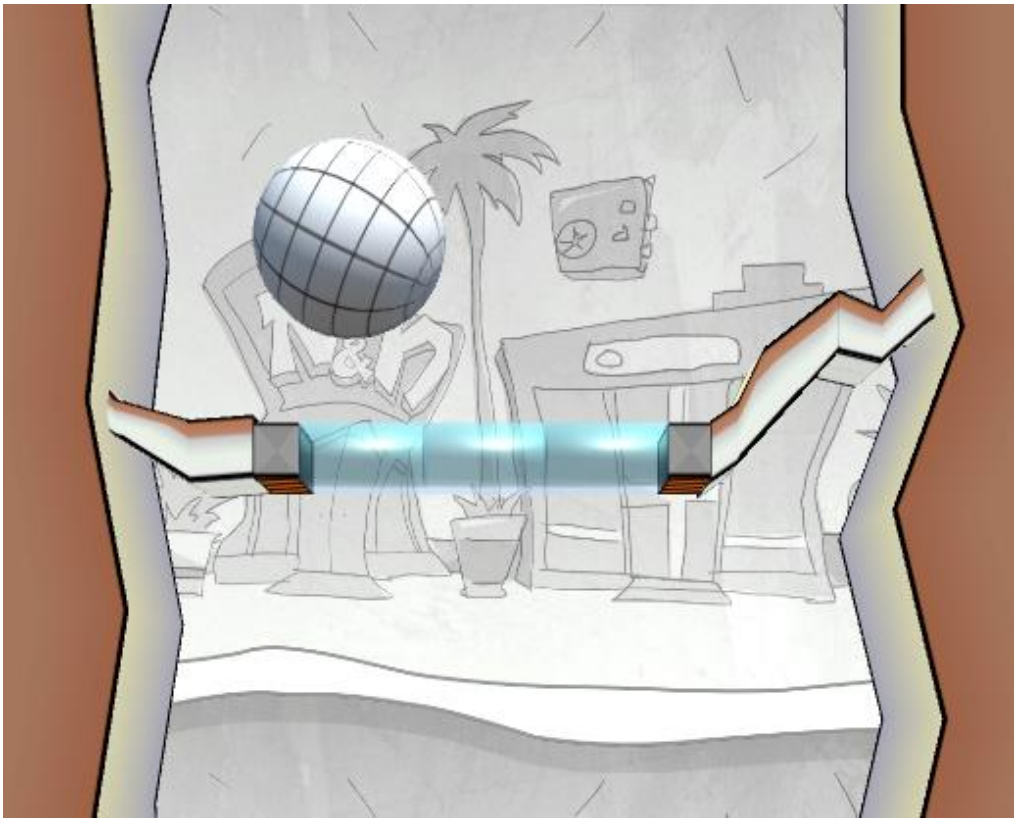
Kuva 6: Kentällä oleva sähkökaappi.

4.3.2 Testiobjekti

Kenttäpalasten testaamista varten luotiin erityinen testiobjekti. Kyseessä oli yksinkertainen ruudukolla teksturoitu pyöreä pallo (kuten näkyy kuvassa 7), jota voitiin ohjata näppäimistöllä. Siihen vaikuttivat pelin fysiikat, joten se putosi alaspäin samalla tavalla kuin pelaajahahmo. Pitämällä välilyöntiä pohjassa, peli poisti testiobjektin fysiikat käytöstä, jolloin sitä voitiin vapaasti leijuttaa näppäimistöllä mihin suuntaan tahansa.

Pallon koko oli määritetty pelaajahahmon enimmäiskorkeuden mukaan, jotta testatessa voitiin varmistaa pelaajan mahtuvan kaikista raoista. Testiobjektin kokoa voitiin kuitenkin vapaasti pienentää tai kasvattaa testaamisen helpottamiseksi. Koska pelaajahahmo ei ollut pallon muotoinen, pienemmällä testiobjektin koolla voitiin testata, miten hahmo mahdollisesti pääsisi pienemmistä aukoista. Isomman koon avulla taas voitiin varmistaa, että pelaajahahmo varmasti mahtuisi myös kaikkein vaarallisimmista paikoista.

Varsinaista pelihahmoa ei voinut käyttää, koska siitä olisi pitänyt tehdä räsynukke-versio, jonka ohjaaminen olisi ollut vaikeaa. Testiobjektin käyttö oli luotettavampaa, helpompaa ja se ajoi asiansa kenttien testaamiseen.



Kuva 7: Testiobjekti.

4.4 Ominaisuuksien lisääminen

4.4.1 Vaikeusasteiden säännöt

Kenttäpalasten lataamiselle eri vaikeusasteiden mukaan tehtiin yksinkertaiset säännöt. Alussa ladataan vain 2-3 vaikeusasteen kenttäpalasia (alkuperäiseen tapaan 1 vaikeusasteen kenttäpalaset ladataan aina joka toisena palasena). Kun pelaaja on mennyt tietyn verran kenttäpalasia, siirrytään lataamaan 2-4 vaikeusasteen palasia. Sitten siirrytään 3-4 vaikeusasteen palasiin, siitä 3-5 kenttäpalasiin, ja lopulta 4-5 palasiin. Varsinaista määrää kenttäpalasia, jonka pelaajan pitää ohittaa, säädettiin myöhemmin tarkemmin.

4.4.2 Pelihahmon lataaminen

Pelihahmon lataamiselle tehtiin aputoiminto LevelUtility -skriptiin. Kun pelaaja kuolee ja jatkaa peliä jatkotoiminnolla, pelihahmo tuodaan uudelleen peliin. Tällöin pitää tarkastaa alue, mihin pelihahmo luotaisiin, jotta se ei synny tason sisälle. Alueelta etsittiin tyhjä kohta ja pelihahmo luotiin sen keskelle.

4.4.3 Taustaelementtien muokkaaminen

Taustaelementeille luotiin vaihtojärjestelmä, jolla ne saatiin vaihtumaan satunnaisesti erilaisiksi. Niille luotiin myös värisysteemi, jolla niitä voitiin värittää suoraan pelin aikana, millä saatiin lisää vaihtelua kenttien ulkonäköön. Värisysteemillä voitiin joko värittää kaikki samanvärisiksi tai vaihtoehtoisesti vaihdella niitä kahden värin kesken.

4.4.4 Tutoriaalikentät

Generaattoriin lisättiin myös tutoriaalikentät, jotka luodaan pelin alkuun, jos pelin asetuksissa on käytössä tutoriaali-asetus. Tämä asetus on automaattisesti voimassa, kun pelin aloittaa ensimmäisen kerran.

Tutoriaalikentät eivät sisällä vaaroja muuten kuin varoituksena pelaajalle. Niissä myös opetetaan pelin pelaaminen ja asioiden kerääminen kentältä.

4.4.5 Brain Party

Pelissä on erityinen Brain Party -moodi, johon pelaaja pääsee tartutettuaan tarpeeksi monta ihmistä. Sen aikana pelihahmo putoaa vapaasti alas ilman vaaraa ja kerää kolikoita.

Brain Partyn aikana generoidaan eri kenttäpalasia kuin tavallisessa kentässä ja tämä mahdollistettiin luomalla toinen generointijärjestelmä alkuperäisen vierelle. Alkuperäinen generointi käsittelee kaikkea tavallisiin kenttäpalasiin liittyvää, kuten sähkökaappien kääntämistä. Brain Partyssa hahmot putoavat nopeasti alaspäin, jolloin pelitehoa ei kannattanut käyttää turhien asioiden tarkistamiseen.

Kun Brain Party lähtee käyntiin, alkuperäinen generaattori pysäytetään ja käynnistetään Brain Partyn oma generaattori. Sama tehdään toisinpäin, kun Brain Party päättyy.

5 Kenttägeneraattorin lopputulos

Kenttägeneraattori saatiin aikataulussa hyväksyttävään malliin. Peliä testattiin perinpohjaisesti eikä generaattorin toiminta rikkoutunut sen valmistuttua.

5.1 Toiminta

Kenttägeneraattorin toiminta on luotettavaa. Se tuo kenttäpalaset kentälle nopeasti ilman katkoksia ja vaihtaa dynaamisesti normaalien sekä Brain Party -kenttien välillä.

Kenttägeneraattori toimii itsenäisesti taustalla eikä isotkaan muutokset pelissä vaikuta sen toimintaan. Sen käyttäminen on helppoa, koska uudet kenttäpalaset tarvitsee vain lisätä sopivaan kansioon Unity -projektin sisällä ja järjestelmä ottaa ne automaattisesti käyttöön. Samoin ihmishahmojen, taustojen ja seinien lisääminen tapahtuu automaattisesti, jos niitä lisätään oikeisiin kansioihin.

Mahdollisia virheitä voi käytännössä tapahtua vain kehittäjän virheenä, jos asioita laitetaan vääriin kansioihin. Nämä virheet on kuitenkin helppo havaita pelin testaamisen yhteydessä, koska asioita generoituu paikkoihin, joissa niiden ei pitäisi olla.

5.2 Vastaanotto

Zaibatsun henkilöstö testasi peliä intensiivisesti sekä sisäisesti että erillisillä, ulkoisilla pelaajilla. Peli sai hyvää palautetta eikä generaattorin toiminta vaikuttanut negatiivisesti pelikokemukseen missään vaiheessa sen valmistuttua.

Generaattorin katsotaan näin ollen onnistuneen, koska pelaajat eivät varsinaisesti havainneet sen toimintaa, mikä oli hyvän kenttägeneraattorin merkki.

Kenttäpalaset itsessään olivat myös onnistuneet suunnittelunsa osalta, koska pelitestaajat eivät pystyneet erottamaan eri kenttäpalasten rajoja toisistaan, vaikka he erikseen yrittivät havaita niitä. Varsinainen pelikenttä vaikutti näin ollen yhtenäiseltä kentältä, mikä oli toivottu lopputulos.

5.3 Tulevaisuus

Kenttägeneraattori on viimeistelty kokonaisuus, minkä toimintaan ei varsinaisesti tarvitse puuttua, ellei siihen haluta lisätä uusia ominaisuuksia. Generaattori on jaettu kahteen selkeästi kommentoituun skriptiin, joihin osaavan ohjelmoijan on helppo tulevaisuudessa lisätä haluttuja ominaisuuksia.

Pelin sisällön kasvattaminen tapahtuu helposti lisäämällä uusia kenttäpaloja, ihmishahmoja, taustoja tai seiniä ilman, että generaattorin toimintaa tarvitsee muuttaa.

Kenttägeneraattoria pystyy myös halutessa käyttämään vastaavanlaisiin peleihin ilman, että se vaatii suurempia muutoksia. Pelikentän suunnan vaihtaminen pystyakselista vaakasuuntaiseen ei vaadi kuin hyvin pieniä muutoksia generaattorin koodissa. Näin sitä pystyisi käyttämään myös perinteisempään ”endless runner” -genreen kuuluvaan peliin, jossa pelihahmo liikkuu vaakatasossa eteenpäin.

Lähteet

Andriyanov, Vadim. Mesh Deformer 2.04u. Haettu 6.4.2017 osoitteesta: https://assetstore.unity.com/ packages/tools/modeling/mesh-deformer-41155

Atlassian Corporation Plc. Git basics. Haettu 1.9.2017 osoitteesta: https://www.atlassian.com/git

Atlassian Corporation Plc. SourceTree 2.3.10. Haettu 18.5.2017 osoitteesta: https://www.sourcetreeapp.com

CA Technologies. Help | Flows. Haettu 28.8.2017 osoitteesta: https://www.flowdock.com/help/flows

Christie, Deborah ja Lauro, Sarah Juliet (2011). Better Off Dead: The evolution of the Zombie as Post-Human (p. 169). Fordham University Press. Haettu 1.7.2017 osoitteesta: https://books.google.fi/books?id=0oZIlm84F2oC&pg=PA57&redir_esc=y#v=onepage&q&f=false

Dotson, Carter. Top 10 Endless Runners for Android (18.3.2017). Haettu 15.6.2017 osoitteesta: https://www.lifewire.com/top-endless-runners-for-android-3957673

Dropbox, Inc. What is the Dropbox desktop application? Haettu 1.9.2017 osoitteesta: https://www.dropbox.com/help/desktop-web/desktop-application-overview

Filipowicz, Luke. Best Endless/Auto Runner Games for iPhone and iPad (16.2.2017). Haettu 15.6.2017 osoitteesta: https://www.imore.com/best-endless-and-auto-runner-games-iphone-and-ipad

Fog Creek Software. What is Trello? – Trello Help. Haettu 28.8.2017 osoitteesta: http://help.trello.com/article/708-what-is-trello

Geig, Mike. Unity - Object Pooling. Haettu 10.6.2017 osoitteesta: https://unity3d.com/learn/tutorials/topics/scripting/object-pooling

GiantBomb.com. Endless Runner (11.07.2017). Haettu 1.9.2017 osoitteesta: https://www.giantbomb.com/endless-runner/3015-7179/

Gkanatsios, Dimitris-Ilias (7.3.2016). Creating an infinite 3D runner game in Unity. Haettu 20.7.2017 osoitteesta: https://dgkanatsios.com/2016/03/07/creating-an-infinite-3d-runner-game-in-unity-like-temple-run-subway-surfers-part-1

Hindy, Joe. 10 best endless runner games for Android (1.3.2017). Haettu 15.6.2017 osoitteesta: https://www.androidauthority.com/best-endless-runner-games-android-690566/

Microsoft Corporation. Microsoft Visual Studio 2017 15.3.5. Haettu 7.6.2017 osoitteesta: https://www.visualstudio.com/vs/

Muscat, Luke. Depth in Simplicity: The Making of Jetpack Joyride (5-9.3.2012). Haettu 1.9.2017 osoitteesta: http://www.gdcvault.com/play/1015527/Depth-in-Simplicity-The-Making

Paavola, Juho-Matti. Elämä pelissä, peli elämässä (Aviisi 04/2011). Haettu 1.9.2017 osoitteesta: http://arkisto.aviisi.fi/artikkeli/?num=04/2011&id=8c12f9d

Unity Technologies. Unity 2017.1.1f1. Haettu 4.9.2017 osoitteesta: https://unity3d.com

Unity Technologies. Unity Products. Haettu 1.9.2017 osoitteesta: https://unity3d.com/unity

Zaibatsu Interactive. Zaibatsu Interactive - About Us. Haettu 1.6. 2017 osoitteesta: http://zaibatsu.fi/about