KARELIA-AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutusohjelma

Jani Eronen

ROOLIPELIN KEHITTÄMINEN UNITY 3D:LLÄ

Opinnäytetyö
Huhtikuu 2014

Karelia AMMATTIKORKEAKOULU	**OPINNÄYTETYÖ** **Huhtikuu 2014** Tietojenkäsittelyn koulutusohjelma Karjalankatu 3 80200 JOENSUU p.013 260 600

Tekijä(t)
Jani Eronen

Nimeke
Roolipelin kehittäminen Unity 3D:llä

Toimeksiantaja
-

Tiivistelmä

Tässä opinnäytetyössä tutkitaan roolipelejä, niiden keskeisiä ominaisuuksia ja roolipelin kehittämistä Unity 3D -pelimoottorilla. Opinnäytetyössä vastataan seuraaviin kysymyksiin: mitä ovat tietokoneroolipelit, kuinka ne saivat alkunsa ja mitä ovat niiden keskeisimmät ominaisuudet. Työn tavoitteena on lisäksi kehittää roolipeliprototyyppi Unity 3D-pelimoottorilla. Opinnäytetyön teoriaosuudessa tutkitaan yleisesti roolipelejä, niiden historiaa ja yleisiä ominaispiirteitä. Opinnäytetyön käytännön osuudessa toteutettiin tietokoneroolipeli Unity 3D:llä. Lisäksi vertailtavina olivat eri kehitysympäristöt ja niiden soveltuminen opinnäytetyön toteuttamiseen.

Opinnäytetyön tuloksena syntyi Unity 3D:llä pelattava roolipeliprototyyppi, joka sisältää yleisimpiä roolipeleissä nähtyjä ominaisuuksia. Ominaisuudet määritettiin opinnäytetyön teoriaosuuden tutkimusten pohjalta.

Kieli suomi	Sivuja 51

Asiasanat
tietokoneroolipelit, Unity 3D, pelinkehitys

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS
April 2014
Degree Programme in Business information Technology
Karjalankatu 3

FI 80200 JOENSUU
FINLAND
p.013 260 600

Author(s)
Jani Eronen

Title
Role-Playing Game with Unity 3D

Commissioned by
-

Abstract

This thesis studies computer role-playing games, their most essential features and the development of a role-playing game with Unity 3D game engine. This thesis explores the following research questions: what are computer role-playing games, how were they born and what are the key features. The goal of the thesis is also to develop a role-playing game prototype with Unity 3D game engine. In the theoretical part of the thesis, computer role playing games are studied generally, by examining their history and common features. In the practical part of the thesis, the features of a computer role-playing game were implemented with Unity 3D. Different development environments are also compared, and examined how well they fit in the implementation of the prototype of the thesis.

The end result of this project was a prototype of a role-playing game, that includes features, that are most commonly seen in computer role-playing games. The list of implemented features was defined by the research of the theoretical part of the thesis.

Language
Finnish

Pages 51

Keywords
computer Role-playing games, Unity 3D, game development

Sisältö

Termit ja lyhenteet

.NET Framework	Microsoftin kehittämä ohjelmistokomponenttikirjasto.
Collider	Mahdollistaa eri objektien törmäämisen toisiinsa.
FPS	Lyhenne sanoista First Person Shooter. Ensimmäisen persoonan ammuntapeli, jossa maailma nähdään pelihahmon näkökulmasta kuvattuna.
GUI	Lyhenne sanoista Graphical User Interface. Graafinen käyttöliittymä.
Indie-videopeli	Videopeli, jonka on kehittänyt yksi kehittäjä tai pieni ryhmä kehittäjiä ilman julkaisijan taloudellista tukea.
Mana	Taikaenergiaa, jota tarvitaan taikakykyjen suorittamiseen.
Masterointi	Toimenpide, jonka aikana viimeistellään äänitetty tuote. Toimenpiteen aikana muokataan kokonaisuutta yksittäisten ääniraitojen sijaan.
Miksaaminen	Toimenpide, jonka aikana yksittäiset ääniraidat koostetaan kokonaisuudeksi muokkaamalla raitojen ominaisuuksia erilaisilla miksaustyökaluilla.
NPC	Lyhenne sanoista Non-Player Character. Tekoälyn ohjaama hahmo.
PNG	Lyhenne sanoista Portable Network Graphics. Häviötön bittikarttagrafiikan tallennusformaatti.
Prefab	Uudelleenkäytettävä peliobjekti, joka voidaan luoda lukuisia kertoja Unity-projektin sisällä.
Renderöinti	Kuvan luominen mallista tietokoneohjelman avulla.
Scene	Unityn sisäinen näyttämö, jolle käytettävät objektit ja toiminnallisuus sijoitetaan.
Sprite-grafiikka	2D-kuvista toteutettua tietokonegrafiikkaa, jonka taustavärejä ei piirretä.

Tag	Tunniste, jonka nimen avulla objekteja voidaan tunnistaa Unity-projektin sisällä.
Triggeri	Toiminnon laukaisija. Jos komponentti määritetään triggeriksi, voidaan sen avulla laukaista eri toimintoja.

1 Johdanto

Valitsin opinnäytetyöni aiheeksi roolipelin kehittämisen, sillä vastaavanlaisen projektin toteuttaminen on ollut haaveeni jo pitkän aikaa. Päädyin opiskelemaan ohjelmistotuotantoa tavoitteenani tulla peliohjelmoijaksi, ja motivaationi on vain kasvanut siitä hetkestä kun kirjoitin ensimmäisen Hello World -ohjelmani. Opinnäytetyössäni tutkin roolipelejä ja toteutan roolipeliprototyypin, johon pyrin sisällyttämään yleisimpiä roolipeleissä nähtyjä ominaisuuksia. Opinnäytetyön on tarkoitus vastata kysymyksiin: Mitä ovat tietokoneroolipelit, kuinka ne saivat alkunsa, mitä ovat niiden keskeisimmät ominaisuudet ja kuinka roolipelin ominaisuuksien kehittäminen onnistuu Unity 3D -pelimoottorilla.

Unitysta on viime vuosina tullut yksi suosituimmista pelimoottoreista etenkin indie-videopelien kehittäjien keskuudessa. Myös useat tunnetut kehittäjät ovat huomanneet, kuinka pelinkehittäminen Unity 3D:llä mahdollistaa pelien tehokkaan kehittämisen. Yksi näistä kehittäjistä on Ultima-pelisarjan luoja Richard Garriott (Handrahan 2013). Unity 3D mahdollistaa niin pienen kuin suuren tekijätiimin työskentelyn pelimoottorilla ja helpottaa kääntämistä kaikille suurimmille alustoille oli kyseessä sitten, PC, Mac, Playstation, Xbox tai Android. Unity mahdollistaa myös pelinkääntämisen suoraan selaimeen.

Opinnäytetyön toisessa luvussa käsitellään tietokoneroolipelejä yleisesti; kuinka ne syntyivät, millaisia olivat ensimmäiset tietokoneroolipelit, millaisiksi ne ovat kehittyneet ja mitkä ovat yleisimpiä roolipelien ominaisuuksia. Näitä ominaisuuksia käsitellään sekä yleisesti että vertailemalla eri pelejä keskenään. Kolmannessa luvussa esitellään eri kehitysympäristövaihtoehtoja ja määritetään ominaisuudet, joiden pohjalta kehitysympäristön valinta suoritettiin. Neljännessä luvussa käyn läpi muita kehitysprosessissa käyttämiäni työkaluja ja kerron, mihin kutakin työkalua kehitystyössä käytettiin. Viidennessä luvussa käyn läpi opinnäytetyön käytännön osuuden eri vaiheita, kuinka loin pohjan peliprototyypilleni ja kuinka suunnittelin ja toteutin opinnäytetyön teoriaosuudessa tutkimani roolipeliominaisuudet. Kuudennessa luvussa pohdin yleisesti, kuinka opinnäyte-

työn toteutus sujui prosessina, millaisia teoreettisia ja käytännön tuloksia sain opinnäytetyössäni aikaiseksi ja millaisia jatkokehitysmahdollisuuksia opinnäytetyölläni voisi mahdollisesti olla.

2 Tietokoneroolipelit

Tietokoneroolipeli on pelilajityyppi, jossa ohjataan joko yhtä hahmoa tai kokonaista ryhmää. Pelimaailma on yleensä laaja, ja pelaaja viettää aikaansa siellä taistellen ja ympäristöä tutkien. Roolipelin päätavoitteena on tehtävien suorittaminen ja hahmon ominaisuuksien kehittäminen (Manninen 2007, 20). Roolipeleissä pelaajilla on lukuisia rooleja, joista valita. Pelaajat voivat esimerkiksi ottaa taistelijan tai parantajan roolin, ja nämä roolivalinnat vaikuttavat moniin pelaajan peruskykyihin (Fullerton 2008, 51).

Tietokoneroolipelimääritelmän alle voi laskea esimerkiksi pelit Dragon Age Origins, Mass Effect, Deus Ex: Human Revolution, Witcher 2 ja Final fantasy 7. Hahmonkehityksen ja taistelun lisäksi tarina on iso osa pelikokemusta, jonka nämä pelit tarjoavat. Pelit sisältävätkin runsaasti dialogia, jonka avulla juonikuvioita kuljetetaan eteenpäin. Osassa peleistä on jopa mahdollisuus vaikuttaa hahmonsa dialogivalintoihin ja näin muuttaa pelikokemustaan pelimaailman reagoidessa pelaajaan eri tavalla.

2.1 Tietokoneroolipelien historiaa

Tässä luvussa käsittelen teoksia, jotka ovat omalta osaltaan eniten vaikuttaneet tietokoneroolipelien syntymiseen. Lisäksi käyn läpi muutamia ensimmäisiä tietokoneroolipelejä, jotka esittelivät myöhemmin genren standardeiksi muodostuneita ominaisuuksia.

Ensimmäiset tietokoneroolipelit syntyivät pöytäroolipelien, kuten Dungeons & Dragonsin, J.R.R. Tolkienin kirjoitusten, urheilusimulaatiopelien ja Will Crowt-

her's Colossal Cave Adventure -seikkailupelin inspiroimina (Barton 2008, 13). Vuonna 1971 ilmestyi miniatyyripohjainen pöytäroolipeli nimeltä Chainmail. Peli sijoittui keskiaikaan ja esitteli useita myöhemmin roolipelien standardeiksi muodostuneita käytäntöjä. Chainmailin fantasialisäosa muun muassa lisäsi peliin monia Tolkienin töistä tuttuja olioita sekä velhoja, jotka pystyivät loitsimaan tulipalloja ja salamoita. Chainmail johti Dungeons and Dragonsin syntyyn, jonka ensimmäisen version säännöt jopa ehdottivat, että pelaajilta löytyisi Chainmail. (Barton 2008, 17.)

Vuonna 1974 julkaistu Dungeons & Dragons tarjosi monia innovaatioita, kuten sen että pelaaja ohjasi armeijan sijasta yhtä hahmoa. Aluksi pelaajat valitsivat hahmoluokan, kuten soturin tai velhon, ja sitten pelin edetessä kartuttivat kokemuspisteitä, löysivät aarteita ja muita esineitä ja taistelivat aina vain voimakkaampia vihollisia vastaan. Kun kokemuspisteitä oli saatu tarpeeksi, pelaaja ansaitsi uuden tason, jonka avulla hänen hahmonsa sai lisää taitoja, kykyjä ja osumapisteitä. Hahmon osumapisteiden määrä kertoo kuinka paljon vahinkoa hahmo voi kestää kuolematta. Osumapisteiden määrän mennessä nollaan tai sen alapuolelle hahmo joko kuolee tai menee tajuttomaksi. Suuremman tason saavuttaneilla hahmoilla on enemmän osumapisteitä ja he voivat selviytyä kovemmista taisteluista. (Barton 2008, 19−20.)

Vuonna 1977 julkaistu science fiction -pöytäroolipeli Traveller tarjosi taitopohjaisen hahmonkehitysjärjestelmän. Suurin ero aiempiin peleihin oli se, että pelin hahmonluontijärjestelmä sisälsi mahdollisuuden päättää hahmon koulutuksesta ja taustasta. Tämä mahdollisti sen, että pelaajat pystyivät aloittamaan pelin hahmoilla, joilla oli enemmän koulutusta ja parempia taitoja, mutta joutuivat kohtaamaan iän mukana tuomia rangaistuksia. Uusien tasojen ansaitsemisen sijaan hahmon tuli seurata valittua urapolkua ja nousta asemassaan ja saada uusia arvonimiä ja poliittista valtaa. (Barton 2008, 22.)

Yksi ensimmäisistä tietokoneroolipeleistä oli Gary Whisenhuntin ja Ray Woodin luoma dnd. Pelissä oli jo monia roolipelien peruselementtejä, kuten kyky luoda hahmo ja muokata hänen ominaisuuksiaan. (Barton 2007.) Dnd sisälsi kaupan

ja kokemuspisteisiin perustuvan hahmonkehitysjärjestelmän ja hirviöt, jotka olivat sitä vaikeampia, mitä syvemmälle tyrmään pelaaja laskeutui. Peli sisälsi myös juonen. Tarinassa mentiin tyrmään, tapettiin lohikäärme ja otettiin haltuun tämän taikapallo(orb). Myöhemmin samainen tehtävä esiintyi yhtä uudelleen muissa tietokoneroolipeleissä. Peli kehitettiin 1970-luvun puolivälissä ja sitä jatkokehitettiin aina vuoteen 1985. Se vaikutti moniin tuleviin tietokoneroolipelien kehittäjiin. (Barton 2008, 32.)

Rogue on vuonna 1980 julkaistu, Michael Toyn ja Glenn Wichmanin kehittämä tietokoneroolipeli, joka poikkesi ominaisuuksiltaan monista muista tietokoneroolipeleistä. Peli muun muassa käytti numeroihin, kirjaimiin ja symboleihin pohjautuvaa grafiikkaa ja järjestelmää, joka generoi sattumanvaraisesti tyrmät aina kun pelaaja aloitti uuden pelin. Näin pelaajalla oli aina uusia alueita tutkittavana. Tämän lisäksi peli ei sisältänyt kauppaa, josta pelaaja olisi kyennyt ostamaan varusteensa, vaan kaikki aseet, haarniskat ja muut esineet tuli löytää joko tyrmien lattioilta tai ruumiiden luota. Rogue inspiroi satoja muita pelejä, joita usein kutsutaan genrenimellä "roguelike". Näitä ovat muun muassa Hack(1982), Moria(1983), Larn(1986), Angband(1990) ja Ancient Domains of Mystery vuodelta 1994. (Barton 2008, 36 –37.)

Akalabeth: World of Doom on Richard Garriottin kehittämä tietokoneroolipeli, joka julkaistiin vuonna 1980. Peli oli monella tavalla edellä aikaansa niin teknisesti kuin ominaisuuksiltaan. Peli sisälsi rautalankapohjaisen (wire-frame) ensimmäisen persoonan kuvakulman, joka vaihtui ylhäältä päin kuvatuksi pelaajan ollessa maanpinnalla. Tämä innovaatio nähtiin myöhemmin lukuisissa tietokoneroolipeleissä. Pelaajan tehtävänä pelissä oli laskeutua tyrmiin, lahdata vihollisia ja palata sitten takaisin pinnalle hankkimaan uusia varusteita ja tehtäviä. Suorittamalla tehtäviä pelaaja pystyi kohottamaan ominaisuuksiaan ja etenemään arvonimissä. (Barton 2007.)

2.2 Tietokoneroolipelien elementtejä

Tietokoneroolipelien yleisimpiä ominaisuuksia ovat hahmon luominen ja kehittäminen, taisteleminen, resurssienhallinta ja keskusteleminen. Lisäksi tietokoneroolipelit sisältävät usein suuren maailman, jonka sisällä kaikki toiminta tapahtuu. Tässä luvussa tarkastelen tietokoneroolipelien ominaisuuksia ja niiden toteutusratkaisuja vertailemalla erilaisia pelejä keskenään.

2.2.1 Hahmon luominen

Monissa roolipeleissä pelaaja voi luoda hahmon mieleisekseen. Hahmolle voi nimen ja ulkonäön lisäksi antaa eri taitoja hyödyntävän hahmoluokan. Joissakin peleissä hahmon luonnissa tehdyillä valinnoilla voi myös vaikuttaa pelin tarinaan. Yksi esimerkki tästä on Biowaren kehittämä Dragon Age Origins (kuva 1), jossa pelaaja voi alussa valita hahmon nimen ja ulkonäön lisäksi hahmolleen taustatarinan (Pyykkönen 2009). Pelaaja voi aloittaa tarinansa esimerkiksi syrjittynä kaupunkihaltiana tai kuninkaallisena kääpiönä. Jokaisessa taustatarinassa lähtöasetelman lisäksi aloitusympäristö on omansa, mikä tekee jokaisesta alusta aloitetusta pelikokemuksesta erilaisen.

Kuva 1. Kuvakaappaus Dragon Age Origins-pelin hahmonluonnista(Bioware Edmonton 2009).

Toiset pelit taas, kuten esimerkiksi Witcher-sarja, perustuvat siihen että pelaaja pelaa ennalta määrätyllä hahmolla läpi pelin, eikä voi näin vaikuttaa pelaajansa taustatarinaan, nimeen tai hahmoluokkaan. Pelaaja voi kuitenkin tehdä valintoja hahmonsa käyttämien varusteiden ja tiettyjen dialogivalintojen suhteen.

Witcher-pelisarja perustuu puolalaisen Andrzej Sapkowskin Witcher-kirjasarjaan. Pelisarjassa, kuten kirjasarjassa, keskiössä on witcher eli suomeksi noituri, nimeltään Geralt Rivialainen. Hän, kuten muutkin noiturit, on eräänlaisia hirviöiden tappamiseen erikoistunut palkkamurhaaja. (Kärkkäinen 2014.)

2.2.2 Hahmon kehittäminen

Pelihahmon kehittäminen on yksi yleisimmistä roolipelin elementeistä (taulukko 1). Osassa peleistä hahmonkehitys on yksi keskeisimmistä pelielementeistä, kun taas osa on virtaviivaistanut hahmon kehittämistä ja keskittynyt pelin muihin osa-alueisiin. Perinteisesti roolipeleissä kehitetään useaa pelaajan hahmon tai hahmojen ominaisuuksia, jotka yhdessä muodostavat hahmon toimintakyvyn. Yleisimpiä kehitettäviä ominaisuuksia ovat hahmon voima, kestävyys ja energia, jotka vaikuttavat hahmon selviytymiseen taistelussa. Lisäksi hahmoilla on omat erikoiskykynsä, jotka ovat myös kehitettävissä ajan mittaan vahvemmiksi. Tämän lisäksi osa peleistä tarjoaa myös mahdollisuuden kehittää muita hahmon ominaisuuksia riippuen pelimaailman tarjoamista aktiviteeteista.

Taulukko1. Hahmon luominen ja kehittäminen eri roolipeleissä.

Peli	Hahmon luominen	Hahmon kehittäminen
Dragon Age Origins	X	X
Diablo 3	X	X
Witcher 2		X
Mass Effect	X	X
Deus Ex Human Revolution	X	X
Final Fantasy 7		X
The Elder Scrolls: Skyrim	X	X
Dust: An Elysian Tail		X

Fallout: New Vegas on Obsidian Entertainmentin kehittämä, SPECIAL-roolipelisääntöjä hyödyntävä roolipeli. SPECIAL on lyhenne pelihahmon tärkeimmistä ominaisuuksista, jotka ovat Strenght (voima), Perception (tarkkaavaisuus), Endurance (kestävyys), Charisma (karismaattisuus), Intelligence (älykkyys), Agility (notkeus) ja Luck (onni) (Honkala 2010, 29). Pelaaja saa pelin alussa rajoitetun määrän pisteitä, jotka hän voi käyttää haluamallaan tavalla hahmonsa ominaisuuksien määrittämiseen.

Pisteytys säilyy suurimmalta osin muuttumattomana läpi pelin ja pelin aikana kokemustasoista ansaitaan vain taito- ja etupisteitä. Näitä taitopisteitä ovat esimerkiksi hiipiminen, puhetaidot, korjaus ja lukkotiirikointi. Tarpeeksi suuri määrä tietynlaisia taitopisteitä mahdollistaa erilaiset toimintatavat pelimaailman eri tilanteissa. Esimerkiksi tarpeeksi korkeilla puhetaitopisteillä pelaaja kykenee jopa puhumaan pois itsensä tilanteesta, jossa muuten tulisi turvautua väkivaltaan. Hyvä lukkojen tiirikointitaito puolestaan avaa mahdollisuuden päästä paikkoihin, joihin alhaisimmilla pisteillä ei pääsisi mitenkään. Uudet edut taas tuovat monenlaista hyötyä, kuten esimerkiksi mahdollisuuden kantaa enemmän tavaraa tai paremman aseella tähtäyskyvyn. Pisteitä voi hallinnoida käyttämällä pelihahmon ranteessa kulkevaa Pip-Boy-laitetta (kuva 2).

Kuva 2. Kuvakaappauksessa Fallout: New Vegas: SPECIAL- ja taitopisteet(Obsidian Entertainment 2010).

Yksi hahmonkehityksen pelin keskiöön nostavista peleistä on Diablo. Diablo on Blizzard Entertainmentin kehittämä suurta suosiota osakseen saanut roolipeli (Clayman 2010). Diablossa pelaaja valitsee ohjattavakseen yhden kolmesta eri kyvyillä varustetusta hahmosta ja lähtee ulkomaailmaan taistelemaan hirviöitä vastaan. Matkalla pelaaja löytää jatkuvasti uusia esineitä, aseita ja haarniskoita, joilla parantaa hahmonsa kykyjä. (Ward 1997.)

Roolipeligenren toista laitaa edustaa Mass Effect. Mass Effect on Biowaren kehittämä roolipelisarja (Bioware 2013), jossa pääpaino on siirretty mikromanageroinnista tarinankerrontaan ja hahmon yksittäisten ominaisuuksien lisäksi pelaaja kehittää koettavan tarinan olosuhteita. Hahmolleen voi valita pelin alussa luokan, joka vaikuttaa olennaisesti pelaajan kykyyn taistella, mutta itse hahmonkehittäminen ei vaikuta olennaisesti lopputulokseen.

Pääpaino on siirretty hahmon persoonan kehittämiseen ja eri tilanteissa tehtyihin valintoihin. Peli sisältää moraalijärjestelmän, jossa pelaajan valinnat lasketaan joko esikuva- tai luopiopisteinä moraalimittariin. Tämä hahmonkehitys näkyy niin, että pelaaja voi halutessaan tehdä valinnan tietyissä tilanteissa sen mukaan, mitä pisteitä hänellä on eniten moraalimittarissaan. Tämä tarkoittaa myös sitä, että jos valinnan edellyttämiä moraalipisteitä on liian vähän, ei pelaaja voi tehdä kyseistä valintaa. Tehtävien suorittamisen lomassa osaksi pelikokemusta muodostuvat ihmissuhteet pelaajan mukana kulkevien NPC:iden kanssa.

2.2.3 Taistelu

Taistelut ovat aina olleet iso elementti tietokoneroolipeleissä. Osa peleistä toteuttaa taistelut käyttämällä vuoropohjaista taistelujärjestelmää ja osa reaaliaikaista taistelujärjestelmää. Lähes kaikissa näissä peleissä yhteistä on kuitenkin se, että taistelusta saadaan hahmonkehityksen edellyttämiä kokemuspisteitä, pelin sisäistä valuuttaa, esineitä ja rahaa. Taistelussa pärjäämiseen vaaditaan niin hahmonkehitystä, varusteiden päivittämistä kuin strategista ajatteluakin.

Oikein tehdyt hahmon kyky- tai luokkavalinnat voivat kokonaan ratkaista taisteluissa menestymisen.

Monissa japanilaisissa roolipeleissä, kuten Final fantasy 7:ssä, taistelut ovat vuoropohjaisia. Taistelun alkaessa pelaajan ohjaama hahmo tai ryhmä hahmoja on sijoitettu peliruudun toiseen laitaan ja kohdattava vihollinen toiseen laitaan (kuva 3). Pelaajan hahmot ja viholliset hyökkäävät vuorotellen toistensa kimppuun kunnes toinen osapuoli on lyöty. (Kasavin 1997.) Pelaajan hyökkäykset, taiat, elvyttäminen ja muut toiminnot tapahtuvat valikoista valitsemalla kunkin pelihahmon oman vuoron aikana.

Pelaaja näkee taisteluiden aikana oman energiansa ja taikomiseen tarvittavan manan määrän. Energiaa pelaaja tarvitsee pysyäkseen elossa ja manaa pystyäkseen käyttämään taikoja. Lisäksi taisteluvalikossa on erikoiskyky- ja aikamittari. Erikoiskykymittari kertoo, kuinka paljon pelaajan on kärsittävä vahinkoa, jotta hän voi käyttää erikoiskykyään taistelussa, aikamittari taas kertoo, kauanko pelaajan pitää odottaa, jotta hän voi toteuttaa seuraavan hyökkäyksensä. Jos pelaaja voittaa taistelun, siirtyy hänen ohjaamansa hahmo taistelutilasta taistelun yhteenvetoikkunaan, jossa hänelle kerrotaan, paljonko taistelusta ansaittiin kokemusta, rahaa ja esineitä. Tämän jälkeen hän siirtyy takaisin pelimaailmassa kulkemistilaan. Taistelut alkavat satunnaisesti pelaajan kulkiessa läpi maailman, eikä vihollisia voi millään tavalla etukäteen vältellä. Vastaavanlainen vuoropohjainen taistelujärjestelmä on ollut käytössä lukuisissa eri tietokoneroolipeleissä.

Kuva 3. Kuvakaappaus pelin Final Fantasy 7 taistelusta (Square 1997).

Deus Ex Human Revolution on roolipeli, jonka taistelujärjestelmä muistuttaa FPS-pelin taistelujärjestelmää (kuva 4). Pelimaailma nähdään pelaajan ohjaaman Adam Jensenin silmin ja taistelu hoidetaan joko asein tai tyrmäämällä viholliset hiipimällä selustaan. Taistelu on pelissä myös monesti vapaaehtoista, joten pelaaja voi välttää konfliktitilanteet hiipimällä vihollisten ohi.

Kuva 4. Kuvakaappaus pelin Deus Ex Human Revolution taistelusta (Eidos Montreal 2011).

Dust: An Elysian Tail -pelissä taistelujärjestelmä on sekoitus tasohyppelyä ja nopeita refleksejä vaativaa toimintaa. Taistelu tapahtuu kaksiulotteisessa maailmassa pelaajan suunnistaessa eteenpäin miekka ja erityiskyvyt aseinaan(kuva 5). Toisin kuin Final Fantasyssa, taistelu ei ole vuoropohjaista vaan reaaliaikaista ja pelaajan selviytyminen on etenkin korkeammilla vaikeustasoilla kiinni oikein ajoitetusta väistöliikkeestä tai torjunnasta.

Kuva 5. Kuvakaappaus pelin Dust: An Elysian Tail taistelusta(Humble Hearts 2012).

2.2.4 Resurssit ja niiden hallinta

Monet roolipelit antavat pelaajan kerätä ja hallinnoida pelimaailmasta löytyviä peliobjekteja. Näitä ovat esimerkiksi erilaiset kerättävät resurssit kuten raha, aseet, haarniskat, taikajuomat ja muut esineet. Näillä resursseilla voidaan muun muassa parantaa pelattavan hahmon kykyjä, elvyttää taisteluissa menetettyä elinvoimaa tai muokata hahmon ulkonäköä.

Kerätyt resurssit voi myös monesti myydä eteenpäin ja ostaa saaduilla pelin sisäisillä rahoilla toisia resursseja. Kerätyt resurssit menevät eräänlaiseen tavaraluetteloon (inventory), jonka saa monessa pelissä näkyviin erillisenä ikkuna-

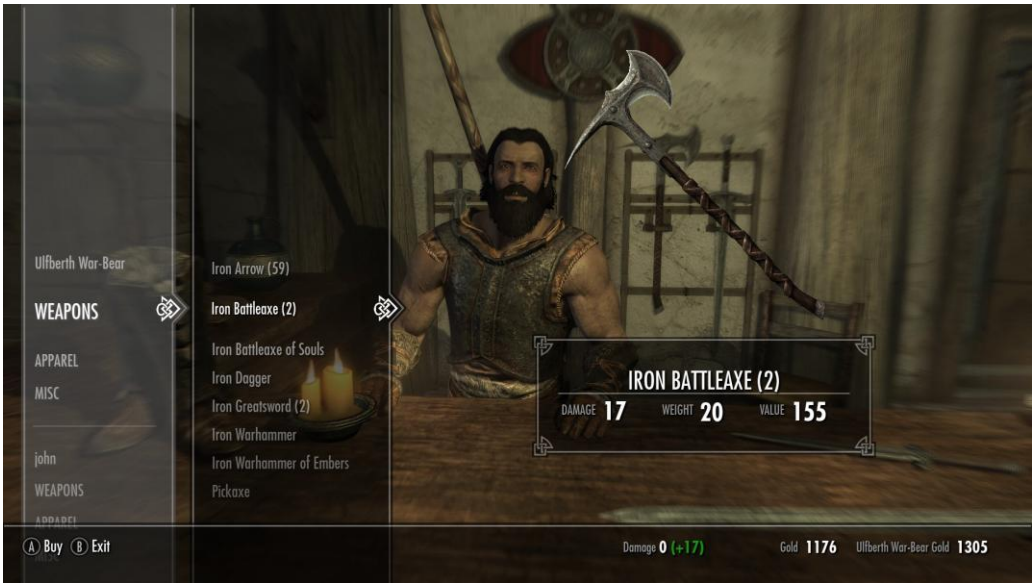
na, josta käsin resursseja voi pelistä riippuen muun muassa ottaa käyttöön, myydä tai pudottaa(kuva 6).

Kuva 6. Kuvakaappaus pelin Darksiders 2 tavaraluettelosta(Vigil Games 2012).

Tietyt roolipelit, kuten esimerkiksi Diablo 3, nostavat resurssienhallinnan keskeiseen asemaan osana pelattavuutta. Hahmon kykyä selviytyä pelimaailmassa kehitetään jatkuvasti keräämällä uusia aseita ja varusteita, ja hahmon kantamaa varustusta vaihdetaan ja muokataan useita kertoja pelin edetessä. Jokaisella pelaajan keräämällä esineellä on omat erikoisominaisuutensa, jotka omalla tavallaan muuttavat pelaajan kykyjä tämän pitäessä niitä osana hahmonsa päällä olevia varustuksia. Esimerkiksi ohessa olevassa kuvakaappauksessa(kuva 7) hahmolla on tavaraluettelossa määritetty aseeksi The Executioner -niminen kirves, joka pystyy tuottamaan vahinkoa 145−174 vahinkopisteen verran ja sisältää ensi- ja toissijaisia ominaisuuksia, jotka omalta osaltaan muuttavat pelaajan pelikokemusta muun muassa lisäämällä liikkumisnopeutta ja hirviöiden kukistamisesta saatavien kokemuspisteiden määrää. Mikäli pelaaja päättää vaihtaa asetta, muuttuu pelattavan hahmon kyky toimia pelimaailmassa.

Kuva 7. Kuvakaappaus pelin Diablo 3 tavaraluettelosta(Blizzard Entertainment 2012).

Perinteisen resurssienhallinnan lisäksi moni roolipeli tarjoaa mahdollisuuden myydä ja ostaa resursseja eri puolilla pelimaailmaa sijaitsevista kaupoista. Peleissä on monesti pelin sisäinen valuutta, jota saa muun muassa taistelemalla, suorittamalla tehtäviä tai myymällä pelissä keräämiään esineitä. Kaupankäynti tapahtuu perinteisesti tavaraluetteloikkunoiden välityksellä, joista pelaaja valitsee kaupankäynnissä käytettävät tuotteet (kuva 8).



Kuva 8. Kuvakaappaus Kaupankäynnistä pelissä The Elder Scrolls V: Skyrim(Bethesda Game Studios 2011).

2.2.5 Keskustelujärjestelmä

Moni tietokoneroolipeli sisältää runsaasti dialogia. Pelaaja kykenee keskustelemaan muiden pelimaailman hahmojen kanssa; kuljettamaan pelin tarinaa eteenpäin, saamaan heiltä tehtäviä ja kehittämään oman pelihahmonsa suhdetta muihin hahmoihin. Vanhemmat ja pienemmän budjetin pelit pohjautuvat monesti pelkkään tekstipohjaiseen keskustelujärjestelmään, kun taas isomman budjetin roolipeleissä pelin hahmot ovat osittain tai kokonaan ääninäyteltyjä.

Pelissä Mass Effect 3 keskustelu hoidetaan käyttämällä dialogipyörää, josta pelaaja valitsee kuhinkin tilanteeseen omasta mielestään sopivan vaihtoehdon. Pyörä ei näytä pelaajalle koko sanottavaa lausetta, vaan lähinnä vastauksen ydinajatuksen muutamalla sanalla(kuva 9). Dialogivalinnoilla on niin pienempiä kuin suurempia seurauksia pelin kuluessa tilanteesta riippuen. Tietyissä olosuhteissa pelaaja saatetaan laittaa muun muassa tilanteeseen, jossa hän joutuu puoltamaan tilanteen jotakin osapuolta ja tämän tietyn osapuolen tukeminen koituu toisen osapuolen kohtaloksi.

Kuva 9. Kuvakaappaus keskustelusta pelissä Mass Effect 3(Bioware 2012).

2.2.6 Pelimaailma

Pelimaailma on aina ollut iso osa tietokoneroolipelejä. Toiset roolipeleistä ovat keskittyneet tarjoamaan lineaarisempia kokemuksia kun taas toiset tarjoavat suuria pelialueita, joissa pelaaja voi vapaasti kulkea, kehittää hahmoaan ja tutkia pelimaailmaa.

Esimerkiksi tietokoneroolipeli nimeltä The Elder Scrolls V: Skyrim tarjoaa todella suuren maailman, jossa pelaaja voi vapaasti matkustaa valtavien alueiden välillä(Kuva 10) tutkien ympäristöä, taistellen lohikäärmeitä vastaan tai keräten esimerkiksi ruokaa tai kirjoja. Pelimaailmaan on lisätty lukematon määrä erilaisia sivutehtäviä ja kyliä täynnä NPC:itä, joiden kanssa jutella tai käydä kauppaa. Pelaaja voi matkustaa mihin tahansa ilmansuuntaan ilman, että pelimaailmassa tulisivat näkymättömät seinät vastaan. Jos esimerkiksi horisontissa näkyy vuori, voi sen päälle todella kiivetä.

Kuva 10. Kuvakaappaus pelistä The Elder Scrolls V: Skyrim (Bethesda Game Studios 2011).

Tietokoneroolipelit hyödyntävät vielä nykyisinkin tyrmän kaltaisia ympäristöjä luodessaan pelimaailmaa. Näihin ympäristöihin sijoitetaan erilaisia tehtäviä, taistelua etsiviä vihollisia ja aarteita pelaajan löydettäväksi. Siinä missä pelin päämaailmassa liikkuminen on monissa peleissä vapaampaa, ovat tyrmät liik-

kumista ajatellen paljon lineaarisempia paikkoja, joissa eteneminen saattaa olla kiinni esimerkiksi juuri tietyn salaoven löytämisestä.

The Elder Scrolls V:Skyrim esimerkiksi sisältää paljon erilaisia luolastoja, jotka on sijoiteltu ympäri pelimaailmaa. Niiden sisältä pelaajalla on mahdollisuus löytää monia uudenlaisia varusteita, aseita ja aarteita, joiden hankkiminen on edellytys hahmon selviytymiskyvyn parantamiselle.

2.2.7 Kartta

Tietokoneroolipelit sisältävät usein suuren maailman, jossa navigoidakseen pelaaja tarvitsee apuvälineitä, kuten kartan. Monet roolipelit sisältävätkin kartankäyttötoiminnon, joka perinteisistä kartoista poiketen myös näyttää pelaajan sen hetkisen sijainnin kartan piirtämällä alueella sekä pelaamisen kannalta oleellisia mielenkiinnon kohteita. Näitä voivat olla esimerkiksi tehtävien sijainnit, rakennukset ja pelimaailman eri hahmot. Monet pelit antavat pelaajalle myös mahdollisuuden merkitä karttaan paikan, johon tämä haluaa matkustaa. Tämä ominaisuus on tietyllä tavalla eräänlainen pelinsisäinen navigaattori, joka parantaa pelimaailman sijaintien paikannettavuutta.

Osa peleistä mahdollistaa niin sanotun pikamatkustamisen eri sijaintien välillä käyttämällä karttaa määränpään määrittämiseen. Näin pelaaja pääsee nopeammin paikasta toiseen ilman, että hänen tarvitsisi erikseen kulkea läpi pelimaailman. Koko ruudun peittävän kartan lisäksi monessa pelissä on myös pienoiskartta(minimap), joka näyttää pelaajan ja tätä lähellä olevia kiinnostuksen kohteita ilman erillistä kartan aktivointia (kuva 11).

Kuva 11. Kuvakaappaus pelistä Witcher 2: Assassins of Kings (CD Projekt RED 2011).

3 Kehitysympäristön valinta

Ennen kehitysympäristön valintaa mielessäni oli useita tarkoitukseen sopivia vaihtoehtoja. Olin opiskelujeni aikana käyttänyt useita eri ohjelmointikieliä, työkaluja ja pelimoottoreita, joten valinta ei alusta alkaen ollut täysin selvä. Tutkin eri vaihtoehtoja ja listasin kaipaamiani ominaisuuksia. Kehitysympäristön tulisi olla hyvin dokumentoitu, sisältää 2D-pelinkehitystä tukevat työkalut ja tukea olio-ohjelmointikieltä. Lisäksi olisi hyvä, jos kehitysympäristö kykenisi kääntämään pelin verkkoselaimelle. Tämä muun muassa mahdollistaisi pelin helpon jaettavuuden eri ihmisten kesken ilman, että peliä tarvitsisi erikseen asentaa koneelle.

3.1 Adobe Flash

Adobe Flash Professional CS on Adoben kehittämä ohjelma (Paananen 2011, 7). Flash Professional -ohjelmalla voi kehittää web-sovelluksia, erilaisia animaatioita ja pelejä. Aikomukseni oli alusta lähtien tehdä peli, jossa käyttäisin Sprite-pohjaisia hahmoja, joten Flash vaikutti aluksi aiempien kokemuksieni perusteel-

la hyvältä vaihtoehdolta. Hyvän olio-ohjelmointikielen lisäksi Flash CS -ohjelman työkaluilla pystyy hyvin toteuttamaan niin grafiikan, animaation kuin ohjelmoinninkin.

3.2 XNA

XNA on Microsoftin kehittämä pelien luomiseen suunniteltu ohjelmointiympäristö (Microsoft 2013). XNA:lla on luotu monia kaksiulotteisia indie-pelejä ja tiesin tapauksesta, jossa yksi ihminen on kyennyt tekemään laadukkaan pelin XNA:n avulla (Sarkar 2012). Olin käyttänyt XNA:ta aiemmin opinnoissani ja kehitysympäristöstä saamani kokemukset olivat hyviä, joten opinnäytetyön tekeminen sillä tuntui hyvältä ratkaisulta. Suunnitelmiin tuli kuitenkin nopeasti muutos kun luin uutisen, että XNA:n kehitys on lopetettu (Rose 2013). Tämä sai minut etsimään uutta vaihtoehtoa työkaluista, joiden hallitsemisesta minulle voisi myös tulevaisuuden pelinkehityksessä olla hyötyä.

3.3 Unreal Development Kit

Unreal development Kit eli UDK on Unreal Engine 3 -pelimoottorin ilmaisversio (Epic 2013). Olin aiemmin opintojeni aikana tutustunut kyseiseen pelimoottorin ja tiesin sen tarjoamista mahdollisuuksista jonkin verran. UDK:lla on luotu useita indie-pelejä, joten työkaluna se olisi varteenotettava vaihtoehto. UDK käyttää ohjelmointiin UnrealScript-ohjelmointikieltä (Epic 2012).

3.4 Unity 3D

Unity 3D on pelimoottori, jolla pelinkehittäjä voi vaivattomasti luoda pelejä useille eri alustoille. Unityn sisäiset työkalut mahdollistavat niin pelin sisäisen testauksen, objektien muokkauksen kuin pelin suorituskyvyn mittaamisen (Unity 2013a). Unityn tukemat ohjelmointikielet ovat Javascript, C#: ja Boo. Unitysta on tarjolla sekä maksullinen että ilmainen versio. Maksullinen versio sisältää ilmaista versiota kattavamman määrän pelikehityksessä käytettäviä työkaluja.

Taulukko 2. Kehitysympäristöt ja niiden tuki tarvituille ominaisuuksille.

Kehitysympäristö	2D-työkalut	Kattava dokumentaatio	Olio-ohjelmointi	Selaintuki
Adobe Flash CS	X	X	X	X
XNA		X	X	
UDK			X	
Unity 3D	X	X	X	X

Valitsin projektin kehitysympäristöksi lopulta Unityn. Vertaillessani eri kehitysympäristöjä ja niiden soveltumista tarpeisiini (Taulukko 2), ymmärsin nopeasti Unityn sopivan parhaiten projektin toteuttamiseen. Tarvittavien ominaisuuksien lisäksi valintaani vaikuttivat aiemmat opintoni ja työharjoittelu, jossa työskentelin kyseisen pelimoottorin parissa. Työharjoitteluaikana pääsin syventämään ymmärrystäni Unityn keskeisimmistä työkaluista ja siitä, kuinka peleihin pystytään toteuttamaan toimivaa ohjelmointikoodia.

4 Työkalut

Unity 3D -pelimoottorin lisäksi projekti tarvitsi myös muita työkaluja, joiden avulla peliprototyyppi saataisiin toteutettua. Pelimoottorin lisäksi projektin työkaluina käytettiin Unityn mukana tulevaa Monodevelop-ohjelmointieditoria, kuvan- ja äänenkäsittelyohjelmia sekä projektinhallintaa edistäviä työkaluja.

4.1 Monodevelop

Monodevelop on ohjelmistokehitystyökalu, joka on ensisijaisesti suunniteltu C#- ja muille .NET-ohjelmointikielille. Ohjelma tukee useaa eri käyttöjärjestelmää ja sen avulla pystyy esimerkiksi kääntämään Visual Studiolla luotuja .NET-sovelluksia Linux- ja Mac OSX -käyttöjärjestelmillä (Monodevelop 2013). Monodevelop tulee ilmaiseksi Unity 3D:n mukana ja sen avulla ohjelmoija kykenee ohjelmoimaan kaiken pelin tarvitseman ohjelmointilogiikan.

Päädyin valitsemaan Monodevelopin, sillä sen lisäksi että ohjelma on ilmainen, sillä pystyy kääntämään myös Unityn ilmaisversiolla luodun projektin. Yrittäessäni kääntämistä Visual Studio 2013 -ohjelmalla huomasin nopeasti, ettei se avannut käsiteltäväksi muita kuin vain valitsemani tiedoston.

4.2 Photoshop, Flash CS ja Wacom-piirtoalusta

Adobe Photoshop on Adoben kehittämä kuvankäsittelyohjelma (Paananen 2010, 5). Photoshop mahdollisti projektissa tarvittavien tekstuurien nopean luomisen, sillä minulla oli jo ennestään usean vuoden kokemus kyseisestä ohjelmasta. Photoshop on erittäin käytetty ohjelma kuvankäsittelyssä ympäri maailman, ja se tarjoaa erittäin laadukkaat työkalut työskentelyyn. Sen avulla onnistuin vaivattomasti rajaamaan, värittämään ja muokkaamaan projektini grafiikoita haluamaani suuntaan. Toisena ohjelmavaihtoehtona oli GIMP, mutta päädyin Photoshopiin säästääkseni aikaa. Näin pystyin paremmin keskittymään ohjelman suunnitteluun ja tekniseen toteuttamiseen.

Vaikka päädyinkin käyttämään Unitya projektin pääkehitysalustana, niin halusin silti toteuttaa osan pelin graafisesti ulkoasusta käyttämällä Adobe Flash CS-ohjelman työkaluja. Adobe Flash CS on erinomainen ohjelma grafiikan luontiin ja olen jo aiemmin toteuttanut sillä ulkoasun erilaisiin projekteihin. Ohjelmalla voi muun muassa helposti muokata jokaista piirtämäänsä viivaa ja saada näin laadukkaita kuvituksia.

Apuna grafiikoiden luomisessa käytin lisäksi omaa Wacomin tietokoneeseen kytkettävää piirtopöytää. Sen avulla kykenin piirtämään asiat suoraan Flash CS-ohjelmaan, ilman että minun tarvitsi erikseen piirtää asioita paperille ja skannata tietokoneelle. Tämä virtaviivaisti prosessia ja säästi aikaa huomattavasti. Lisäksi lopputuloksesta tuli paljon parempi kuin mitä tulos olisi ollut, jos olisin käyttänyt piirtämiseen tietokoneen hiirtä.

4.3 Cubase, Fl-studio ja sfxr

Cubase on Steinbergin kehittämä audiotyöasema (Laaksonen 2006, 380). Ohjelmalla käyttäjä pystyy äänittämään, editoimaan, miksaamaan ja masteroimaan ääntä. Opinnäytetyössä Cubasea käytettiin pelissä esiintyvien äänitehosteiden äänittämiseen, editointiin ja miksaamiseen. Valitsin Cubasen työkalukseni, sillä minulla on sen käytöstä usean vuoden kokemus niin ammatillisen koulutuksen kuin musiikkiharrastuksen puolesta. Cubase oli yksi niistä ohjelmista, joita käytin eniten opiskellessani audiovisuaalista viestintää ammattiopistossa.

Fl studio on Image-linen kehittämä digitaalinen musiikin teko-ohjelma, jolla pystyy säveltämään, äänittämään, editoimaan ja miksaamaan musiikkia (Image-line 2013). Opinnäytetyössä Fl studiota käytettiin pelissä esiintyvien äänitehosteiden toteuttamiseen. Fl Studio on toinen ohjelma, josta minulla on paljon aiempaa kokemusta muun muassa erillisten konemusiikkitaustojen säveltämisestä.

Sfxr on Tomas Petterssonin kehittämä työkalu äänitehosteiden tekemiseen. Työkalu luotiin alun perin Ludum Dare -kilpailun yhteydessä pelinkehittäjille, jotka haluavat saada pelinsä valmiiksi kilpailun määrittämän 48 tunnin aikarajan sisällä. Sfxr on avoimen lähdekoodin ohjelma, jota kuka tahansa voi vapaasti käyttää (Petterson 2012). Ohjelmaa käytettiin opinnäytetyössä osana pelin äänitehosteiden toteuttamisprosessia.

4.4 Dropbox

Käytin Dropboxia käytännön projektini versionhallintaan, varmuuskopiointiin ja jakamiseen. Dropboxin avulla kykenin helposti jakamaan viimeisimmän pelattavan version projektini musiikin säveltäjälle. Minun ei tarvinnut käytännössä kuin jakaa html-tiedosto kerran hänelle ja päivittää muutokset projektini Web Player -versioon. Lisäksi Dropbox mahdollisti vaivattoman projektin läpikäymisen myös mobiililaitteilla. Kykenin pääsemään käsiksi lähdekoodiin niin kännykällä kuin tabletillanikin ja arvioimaan tulosta ilman, että minun tarvitsi olla yhteydessä pöytätietokoneeseeni.

4.5 Google Drive

Google Drive on Googlen tarjoama ilmainen palvelu, jolla voi luoda, muokata ja jakaa dokumentteja, piirroksia ja muita tiedostoja. Lisäksi palvelu myöntää käyttäjälleen tallennustilaa, johon kaikki käyttäjän luoma sisältö tallennetaan automaattisesti (Google 2013).

Käytin opinnäytetyössäni Google Drivea dokumenttien luomiseen ja hallinnointiin. Palvelun avulla kykenin käsittelemään dokumenttejani millä tahansa tietokoneella tai tabletilla ilman, että minun olisi pitänyt huolehtia tekemistäni muutoksista eri versioiden välillä.

4.6 Trello

Trello on ilmainen selainpohjainen projektinhallintasovellus (Fog Creek Software 2013). Olen käyttänyt Trelloa aiemmin useassa eri projektissa ja huomannut sen auttavan niin dokumenttien organisoinnissa kuin tehtävien ja aikataulujen toteuttamisessa. Trello helpottaa asioiden muistamista ja sillä pystyy tarvittaessa hallinnoimaan kokonaisen projektiryhmän toimintaa jakamalla muille projektin jäsenille tehtäviä.

5 Roolipeliprototyypin toteuttaminen

Tavoitteeni oli toteuttaa roolipeliprototyyppi, joka sisältäisi yleisimpiä roolipeleissä esiintyviä elementtejä. Tutkin useita roolipelejä ja päätin näiden pohjalta, mitkä elementit olisi kaikkein tärkeintä saada mukaan opinnäytetyöni käytännön osuuteen. Lopullinen peliprototyyppi sisältää pelimaailman, kaupan, hahmonkehityksen sekä taistelu-, keskustelu- ja resurssienhallintajärjestelmän.

Pelissä pelaaja kykenee tutkimaan ympäröivää pelimaailmaa, olemaan vuorovaikutuksessa pelimaailman muiden hahmojen kanssa ja taistelemaan vihollisia vastaan. Pelaaja pystyy myös keräämään, ostamaan ja myymään pelimaail-

masta löytyviä esineitä. Voittamalla vihollisia pelaajan ohjaama hahmo saa kokemuspisteitä, joiden avulla hahmoa voi roolipeleille ominaiseen tapaan kehittää vahvemmaksi.

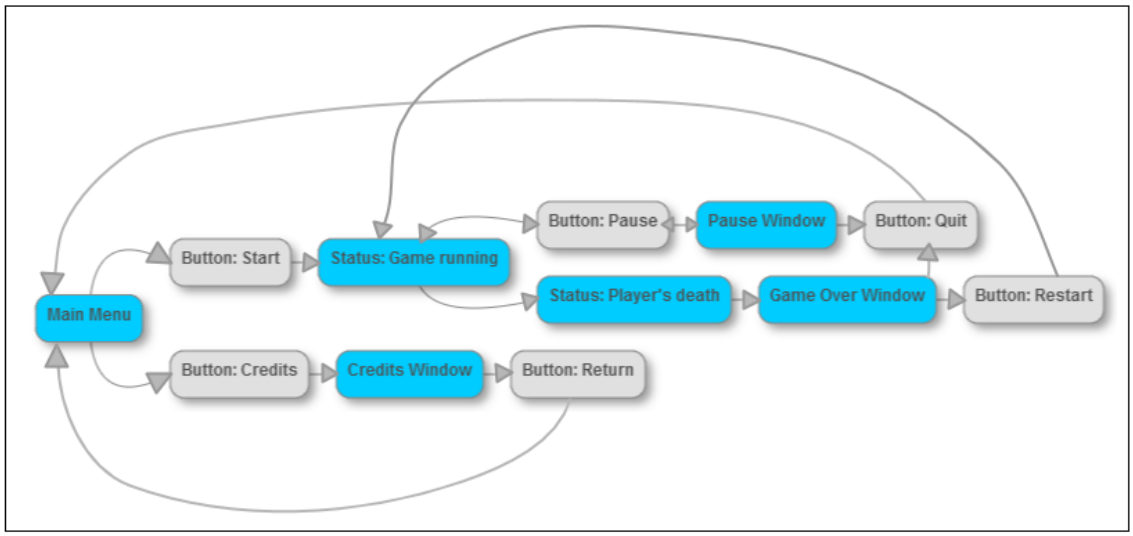
Äänitehosteet toteutettiin mikrofonin, nauhurin ja eri ääniin erikoistuneiden tietokoneohjelmien avulla. Näistä ohjelmista kerron tarkemmin luvussa 4, jossa käyn läpi kaikki pelimoottorin lisäksi projektissa käytetyt työkalut. Äänitehosteista editoitiin kaikki ylimääräinen tyhjä ääniraita pois niin, että tehosteet toistuisivat Unityssa pelin vaatimalla tavalla. Tämä on erityisen tärkeää, sillä Unityn toistaessa ääntä äänen pitää toistua juuri oikealla hetkellä. Tämä ei välttämättä olisi mahdollista, mikäli ääniraidassa olisi ylimääräistä pituutta tiedoston alussa tai lopussa, sillä se loisi todennäköisesti viivettä, joka saisi äänet toistumaan eri tahdissa animaation ja ohjelmoitujen toimintojen kanssa. Äänitehosteiden miksausvaiheessa ääniin lisättiin muun muassa keinotekoisesti luotua jälkikaikua (reverb) ja muita tehosteita. Näin äänet saatiin paremmin sopimaan osaksi pelin äänimaailmaa. Myös Unityssa on mahdollisuus lisätä ääniin tehosteita, mutta ne sisältyvät vain Unityn maksulliseen versioon (Unity 2011), eivätkä näin olleet käytettävissä opinnäytetyössäni. Sain peliin myös siihen sopivaa musiikkia pyytämällä apua säveltäjäystävältäni Tero Kangasmaalta. Hän toteutti mielellään useamman kappaleen prototyypin taustaksi. Näistä kolme pääsi osaksi peliä; yksi alkuvalikkoon, toinen pelin aloituspaikkaan ja sokkeloon ja kolmas pelin kauppapaikkaan.

5.1 Valikoiden luominen

Aloitin pelin luomisen suunnittelemalla rakenteen valikoille, joiden avulla peliä käytetään (kuva 12). Tämän jälkeen aloin suunnitella valikoiden ulkoasua. Halusin toteuttaa ulkoasun itse, joten loin sopivan fontin Wacom-piirtoalustalla. Halusin pitää ulkoasun yksinkertaisena, mutta toimivana ja keskittyä toiminnallisuuden luomiseen. Loin päävalikolle Unityssa oman scenensä, johon aloin luomaan tarvittavia objekteja. Tein Photoshopissa valikkotekstit ja taustaväripohjan, jotka sitten toin Unityyn png-muodossa.

Päävalikon taustan ja tekstiobjektien pohjana käytin Unityn omaa Plane-objektia. Tekstiobjekteja varten loin kooditiedoston, johon loin OnMouseEnter- ja OnMouseExit-metodit. Niiden avulla napit vaihtavat väriä sen mukaan, onko hiiren osoitin napin päällä vai ei. Tämä auttaa pelaajaa paremmin hahmottamaan, minkä toiminnon hän on valitsemassa. Lopuksi lisäsin OnMouseDown-metodilla toiminnallisuuden, joka lataa pelitason tai tekijätiedot sisältävän ikkunan. Koska prototyypin pääkohdealusta on toistaiseksi pelkkä selain, ei sovelluksen lopetuksen mahdollistavan toiminnon ohjelmoiminen ole tarpeen.

Tekijätiedot sisältävän ikkunan loin suurimmaksi osaksi Photoshopissa, minkä jälkeen toin sen png-tiedostona Unityyn. Unityssa loin sitä varten plane-objektin sceneen, johon raahasin tekstuurin. Totesin, että kameran ja eri objektien sijoittelun sijaan toiminnallisuus olisi nopeampi toteuttaa niin, että kaikki objektit pidetään kameran edessä samassa paikassa ja ohjelmistokoodi määrittää mitkä objektit piirretään ruudulle milloinkin.



Kuva 12. Pelin valikkorakenne.

5.2 Pelaajahahmon luominen

Päätin jo projektin alussa, että haluan luoda projektin, jossa käytetään Sprite-grafiikkaa 3D-mallien sijaan. Päädyin menetelmään muun muassa siksi, ettei projektissa ollut mukana ketään, joka olisi osannut tehdä 3D-malleja. Toinen

syy oli se, että halusin itse suunnitella myös pelin graafisen ulkoasun. Tehtävä ei tuntunut millään tavalla ylivoimaiselta, sillä olen piirtänyt koko ikäni ja jo aiemmin toteuttanut Sprite-grafiikkaa eri pikkuprojekteihin. Lähestymistapani teki toteuttamisesta aluksi myös vaikeaa, sillä minulla ei ollut paljoa aiempaa kokemusta 2D-pelin tekemisestä Unitylla. Unityn Asset storesta löytyi 2D-pelien rakentamiseen erilaisia ratkaisuja, mutta halusin kokeilla, pystyisinkö toteuttamaan vaaditun toiminnallisuuden ilman ylimääräisiä työkaluja.

Onnistuin 2D-hahmon rakentamisessa kohtalaisen hyvin, mutta päädyin lopulta muuttamaan toteutustapaani Unityn saadessa yllättäen päivityksen 4.3-versioksi. Tämä päivitys toi mukanaan uusia 2D-pelinkehitykseen suunniteltuja työkaluja (Unity 2013b). Päivitys toi vihdoin mukanaan virallisen tuen sprite-tiedostoille ja mahdollisuuden myös tarvittaessa käsittelemään 2D-objekteja ilman kolmiulotteisuutta. Uusien työkalujen käyttöönotto kannatti niin aikataulullisesti kuin laadullisestikin, sillä niiden avulla kykenin työskentelemään paljon nopeammin ja uusi Sprite-tuki teki tekemistäni kuvista visuaalisesti Unityyn tuodessa laadukkaamman näköisiä.

5.2.1 Hahmon ominaisuudet

Luodakseni roolipeliin sopivan pelihahmon minun tuli lisätä hahmolle ominaisuuksia, joita hahmonkehitysjärjestelmä voisi kehittää. Näin hahmolla olisi attribuutteja, kuten terveys, nopeus, ja voima, joiden arvoa voitaisiin kasvattaa myöhemmin pelin aikana. Lisäksi terveysarvon avulla pelaaja voisi myös hävitä viholliselle terveysarvon mennessä taistelussa nollaan.

Aloitin luomalla luokan, johon lisäsin ominaisuuksia edustavia muuttujia. Määritin muuttujat tarkoituksella julkisiksi, jotta ominaisuusluokkaa voisi käyttää myös vihollisobjekteissa uusilla arvoilla. Itse arvot määritin Unityn editorin puolella. Suunnittelin järjestelmän niin, että muut luokat joutuvat tarvittaessa hakemaan tarvitsemansa arvot tästä luokasta.

5.2.2 Hahmonkehitysjärjestelmä

Aloitin hahmonkehitysjärjestelmän toteuttamisen tekemällä sen ikkunalle oman tekstuurin Photoshopilla. Tämän jälkeen loin Unityssa tyhjän peliobjektin toiminnallisuudelle ja raahasin sen pelaajahahmon objektin alle. Näin toiminto olisi helppo paikantaa pelihahmon sisältä, sillä hahmonkehitys on sidottu osaksi pelattavaa hahmoa. Päätin jo aluksi, että sisällyttäisin ajan säästämiseksi järjestelmään vain välttämättömimmän, joten pelaaja voi kehittää kaiken kaikkiaan kolmea eri hahmonsa ominaisuutta kasvattamalla näiden arvojen kokonaispistemäärää(Taulukko 3).

Taulukko 3. Kehitettävät ominaisuudet, niiden vaikutukset ja hyödyt.

Pistetyyppi	Vaikutus	Hyöty
Terveyspisteet	kasvattavat hahmon energiamäärää	Mahdollistaa taistelussa pitempään selviytymisen
Voimapisteet	Kasvattavat hahmon aseilla tuottaman vahingon määrää	Hahmo kykenee voittamaan helpommin kohtaamiansa vihollisia
Nopeuspisteet	Kasvattavat hahmon liikkumisnopeutta	Hahmo pääsee nopeammin paikasta toiseen ja kykenee helpommin pakenemaan vihollisilta

Mikäli pelaaja onnistuu kukistamaan vihollisen taistelussa, ottaa vihollisen terveyttä hallinnoiva kooditiedosto yhteyden hahmonkehitysjärjestelmään ja kasvattaa tuhottujen vihollisten määrää yhdellä. Hahmonkehitysjärjestelmä puolestaan päättää, milloin pelaaja on tuhonnut tarpeeksi vihollisia ansaitakseen uuden tason. Pelaajan ansaitessa uuden hahmonkehitystason nollautuu tuhoutuneiden vihollisten määrää laskeva muuttuja ja aloittaa laskemisen alusta. Tämän lisäksi hahmonkehitysjärjestelmä kasvattaa joka tason kohdalla määrää, kuinka monta vihollista tulee tuhota ansaitakseen uuden tason.

Pelaajan ansaitessa uuden tason ilmestyy ruudulle GUI-ikkuna, jonka avulla pelaaja voi kasvattaa haluamiaan arvoja (kuva 13). Lisäksi pelaaja saa tason ansaitsemishetkellä kaiken menettämänsä terveyden takaisin, mikäli hän on haavoittunut taistelussa. Kun pelaajan voiman määrä kasvaa, saa hän vihollisen

nopeammin tuhottua. Lisäämällä terveyttä pelaajan hahmo kestää enemmän vahinkoa. Nopeuden kasvattaminen puolestaan mahdollistaa nopeamman liikkumisen paikasta toiseen ja helpottaa vihollisilta pakenemista.

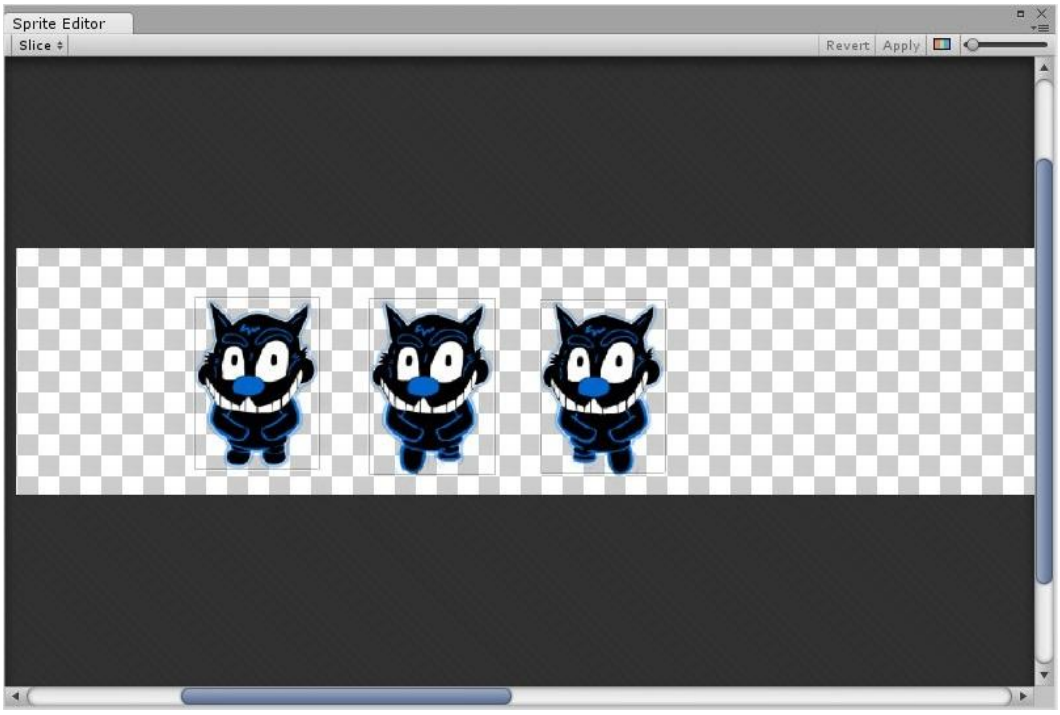
Kuva 13. Kuvakaappaus hetkestä, jolloin pelaaja saavuttaa uuden tason.

5.2.3 Hahmon animointi

Unityn uudet 2D-työkalut tekivät sprite-pohjaisten hahmojen animoinnista helppoa. Piirsin ensin hahmosta animaatiota varten kuvasarjan, jossa tämä kävelee alas. Tämän jälkeen toin kuvasarjan Unityyn, jossa muutin tuodun grafiikan tietotyypiksi Spriten. Tämän jälkeen leikkasin kuvat erillisiksi Unityn omalla Sprite Editor -työkalulla (kuva 14), ja tein niistä oman animaatiotiedostonsa valitsemalla jokaisen leikatun kuvan projektin hierarkia -ikkunasta ja raahaamalla ne scene hierarkia -ikkunaan. Tämän jälkeen loin pelaajahahmo-objektin alle toisen objektin animaation hallitsemista varten ja lisäsin siihen Animator-komponentin. Tämä komponentti hallinnoi yksittäisten animaatioiden toistoa, sitä kuinka kauan niiden toistamisessa menee aikaa ja sitä, mitkä animaatiot olisivat linkitettyinä toisiinsa. Lisäksi loin Animatoriin Boolean parametrejä, joiden avulla voisin vaihtaa toistettavasta animaatiosta toiseen. Lopuksi loin tämän Animatorin pelaajan komentoja hallitsevaan kooditiedostoon ja loin metodit, joi-

den avulla lähettää Animatorissa määritetyille Boolean-muuttujille haluamiani arvoja oikealla hetkellä.

Tehdäkseni pelaajahahmosta elävämmän oloisen toteutin pelaajalle myös lepotilaan sopivan animaation. Tämä toimii niin, että jos pelaaja ei liikuta pelihahmoa mihinkään suuntaan vähään aikaan, muuttaa pelihahmo ilmettään toiseksi säännöllisin väliajoin.

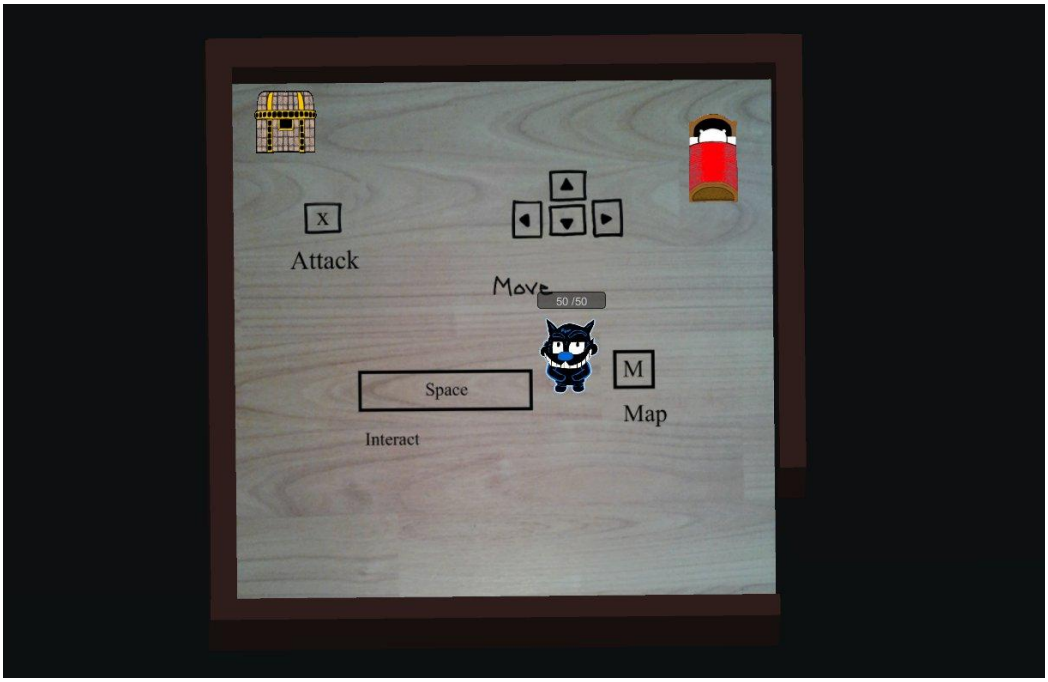


Kuva 14. Kuvakaappaus Unityn Sprite Editorista.

5.3 Pelimaailman eri alueet

Aloittaessani suunnittelemaan pelimaailmaa otin lähtökohdaksi pelimaailman, joka soveltuisi erilaisten roolipelielementtien testaamiseen. Pelimaailmassa pitäisi pystyä liikkumaan, keskustelemaan, keräämään esineitä, taistelemaan ja käymään kauppaa. Päätin luoda maailman, joka koostuu kolmesta eri osasta: pelihahmon kodista (kuva 15), sokkelosta tämän ulkopuolella (kuva 16) ja kaupasta (kuva 17).

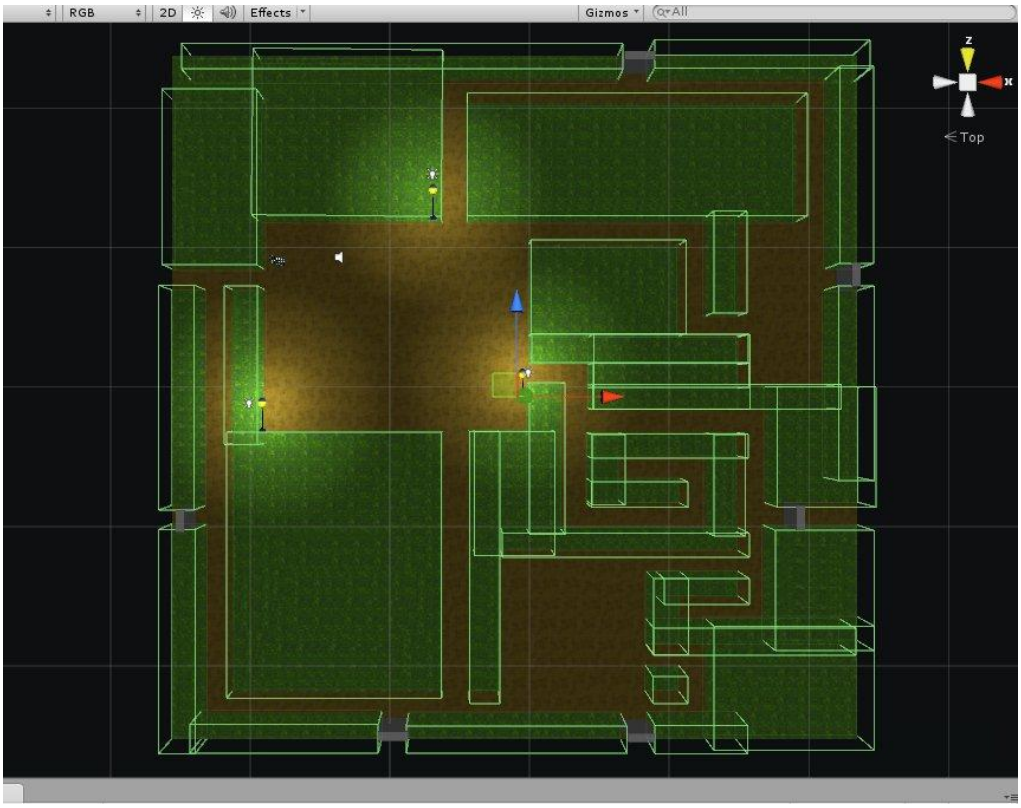
Ensimmäiseksi suunnittelin pelaajalle yksinkertaisen kotipaikan pelin aloituspisteeksi. Ohjausta neuvovien peli-ikkunoiden sijaan päätin sijoittaa ne osaksi pelimaailmaa piirtämällä kuvat käytetyistä kontrolleista aloituspisteen lattiaan. Näin pelaaja voisi kokeilla pelin kontrolleja saman tien ilman, että hänen pitäisi poistua ohjeikkunasta kokeilun ajaksi. Sijoitin aloitushuoneeseen myös sängyn ja arkun. Ensimmäisen tehtävänä on luoda kodinomaista tunnelmaa kun taas jälkimmäisestä pelaaja saa käyttöönsä aseen, jolla puolustautua kotipaikan ulkopuolella. Lisäksi loin arkkuun GUI-ikkuna-toiminnallisuuden, joka aktivoituu pelaajan avatessa arkun ja kertoo minkä esineen pelaaja löysi arkusta ja kuinka käyttää tätä esinettä.



Kuva 15. Kuvakaappaus pelihahmon kotipaikasta.

Koska peli toteutetaan kaksiulotteisena, päätin luoda suurimman osan pelimaailmasta käyttäen Unityn plane-objekteja. Näihin plane-objekteihin sitten suunnittelin ja toteutin tekstuurit. Sokkelotekstuurin pensaiden kohdalle loin cube-objektit, joista tein näkymättömiä esteitä poistamalla objektien renderöintikomponentit käytöstä (kuva 16). Näin objektit eivät näy itse pelissä, mutta estävät pelaajan kulkua niin ettei hän pääse kävelemään pensasaitojen läpi.

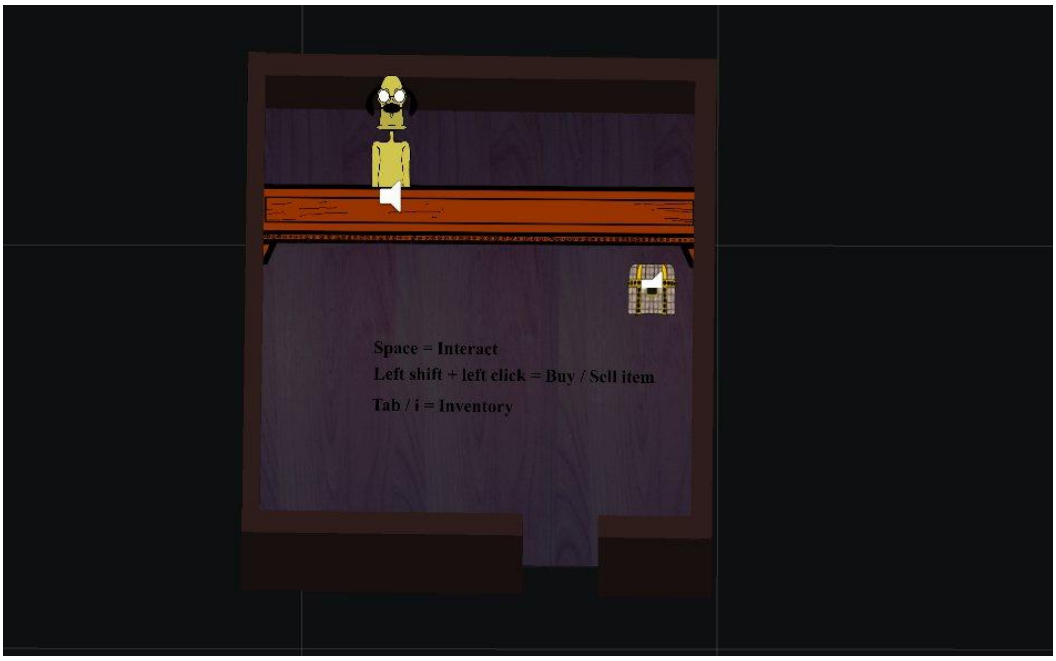
Sokkelo kaipasi kuitenkin vielä jotain, mikä saisi sen näyttämään pelaajan silmissä mielenkiintoisemmalta. Eri tietokoneroolipelien maailmat ovat usein olleet mielestäni visuaalisesti mielenkiintoisia paikkoja tutkia. Tämä sai minut ajattelemaan, että pelimaailmaan voisi lisätä valonlähteitä luomaan valoeroja, jotka voisivat parantaa pelin tunnelmaa. Mietin sokkeloon sopivia valolähteitä ja päädyin toteuttamaan katulamppuobjekteja, joiden sisään lisäsin Unityn Point Light -komponentin. Lopuksi loin lampusta Prefabin objektin uudelleenkäyttöä silmällä pitäen ja sijoittelin lamppuobjekteja muutaman lamppuobjektin sokkeloon mielestäni visuaalisesti parhaimmille paikoille.



Kuva 16. Kuvakaappaus sokkelon liikkumista rajoittavista objekteista.

Mahdollista myöhempää jatkokehitystä silmällä pitäen päätin jättää sokkeloon useita poistumisreittejä, jotka mahdollistaisivat pääsyn muualle pelimaailmaan, mikäli päättäisin jatkaa prototyypin kehittämistä vielä opinnäytetyöprosessin jälkeen. Estääkseni pelaajaa käyttämästä tyhjyyteen johtavia poistumisreittejä tein niiden eteen cube-objekteista läpikulkemisen estäviä portteja.

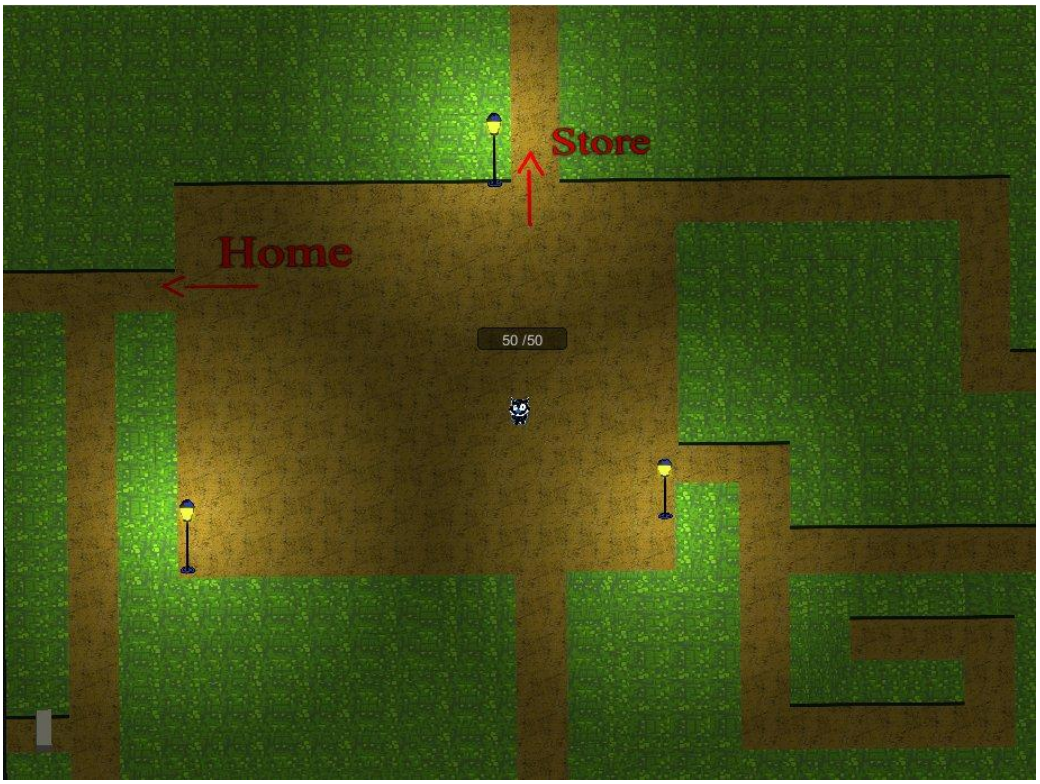
Suunniteltuani sokkelon aloin suunnitella pelin sisäistä kauppaa. Ensimmäisenä loin kauppatilan Unityn scenen sisälle hyödyntämällä aiemmin rakentamani pelihahmon kotipaikan objekteja ja luomalla kaupanmyyjälle ja tämän myyntipöydälle omat tekstuurinsa. Lopuksi toin tekstuurit Unityyn ja rakensin kaupan hyödyntämällä näitä tekstuureja ja Unityn omia cube- ja plane-objekteja (kuva 17). Kaupan lattian tein teksturoidusta plane-objektista ja seinät cube-objekteilla.



Kuva 17. Kuvakaappaus pelin sisäisestä kaupasta.

5.4 Kartta

Pelin karttatoiminnallisuuden toteuttavat kooditiedostot on kytketty pelin pääkameraan. Jos pelaaja painaa karttatoiminnallisuuden laukaisevaa nappia kotipaikan ulkopuolella, niin kamera ottaa etäisyyttä pelimaailmaan antaen pelaajalle mahdollisuuden nähdä enemmän ympäröivää aluetta (kuva 18). Lisäksi karttatoiminnallisuus pysäyttää pelin sisäisen ajankulun, jotta viholliset eivät kykene käymään pelaajan kimppuun hänen lukiessa karttaa, eikä pelaaja pysty liikkumaan karttanäkymässä. Karttatoiminnallisuus poistaa myös vihollisen ja NPC:n näkymästä kartalta ja vaihtaa sokkelon tekstuurin kartan tekstuuriksi ohjelmakoodista käsin. Metodit karttatoiminnallisuuden toteuttamiselle kutsuu pelaajaobjektin sisällä oleva, pelaajan antamia komentoja hallinnoiva luokka.



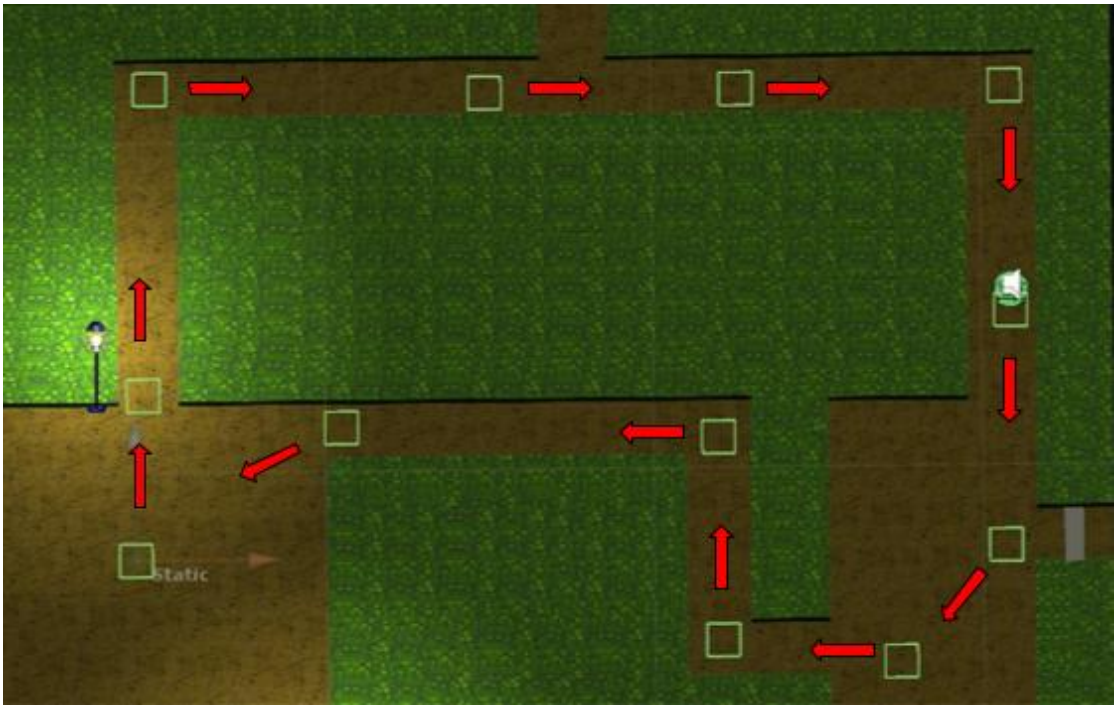
Kuva 18. Pelimaailma karttatilassa, jonka aikana peli on pysähdyksissä.

5.5 Viholliset

Peli sisältää kaksi erilaista vihollistyyppiä. Toinen on aggressiivisesti pelaajan kimppuun käyvä ja aiheuttaa vahinkoa lyhyeltä matkalta, kun taas toinen pyrkii aiheuttamaan vahinkoa pitkän matkan päästä. Lyhyen matkan päästä hyökkäävä vihollinen liikkuu oletusarvona sokkelossa ennalta arvattua reittiä pitkin, ja pitkän matkan päästä hyökkäävä pysyy samassa paikassa koko ajan. Mikäli pelaaja onnistuu voittamaan vihollisen taistelussa, jättää tämä jälkeensä pelaajalle pelin sisäisenä valuuttana toimivaa kultaa. Lisäksi pelaaja saa vihollisten voittamisesta kokemuspisteitä, joiden avulla hän voi parantaa ominaisuuksiaan saatuaan tarpeeksi suuren määrän kokemusta. Voitetut viholliset syntyvät uudelleen pelaajan siirtyessä pelialueelta toiselle. Näin pelaaja voi lisätä kokemuspisteitään päihittämällä samoja vihollisia useamman kerran.

5.5.1 Lähitaisteluun erikoistunut vihollinen

Pelin lähitaisteluun erikoistunut vihollinen on vihreä pallonmuotoinen objekti, joka kulkee pitkin sille määrittämääni reittiä(kuva 19). Reitti muodostuu objekteista, jotka olen asettanut pelissä itsessään näkymättömiksi. Jokaisessa näistä objekteista on kooditiedosto, jossa on OnTriggerEnter-metodi. Tämä metodi tarkistaa, osuuko vihreä pallo kooditiedoston isäntäobjektiin. Jos näin tapahtuu, metodi ottaa yhteyden GetComponent-metodilla vihollisen tekoälyä hallitsevaan kooditiedostoon ja vaihtaa vihollisen seuraavaksi määränpääksi toisen näkymättömän objektin. Jokaisessa näistä näkymättömistä objekteista on sama määränpäätä vaihtava kooditiedosto. Näin vihollinen päätyy kiertämään loputtomasti näiden objektien välillä, ellei satu havaitsemaan pelaajaa. Mikäli pelaaja tulee liian lähelle vihollista, huomaa vihollinen pelaajan ja käy tämän kimppuun. Pelaaja voi päättää, taisteleeko hän vihollista vastaan vai pakeneeko paikalta. Pelaaja voi taistella vihollista vastaan vain, jos hänellä on ase tavaraluettelossa ja aktivoituna. Mikäli pelaaja päättää paeta viholliselta, voi hän tehdä sen ottamalla tarpeeksi etäisyyttä viholliseen tai poistumalla sokkelosta pelin muille alueille. Pelaajan paetessa viholliselta onnistuneesti palaa vihollinen takaisin kulkureitilleen.



Kuva 19. Vihollinen kulkee pitkin näkymättömien objektien määrittämää reittiä.

Toteutin viholliselle kaksi erillistä animaatiota: oletusanimaation ja hyökkäysanimaation. Oletusanimaatio koostuu yhdestä kuvasta, jossa vihollinen pitää silmiään kiinni kulkiessaan oletusreittiä pitkin. Hyökkäysanimaatiossa vihollisen silmät pyörivät ja vihollisen suu on avoinna.

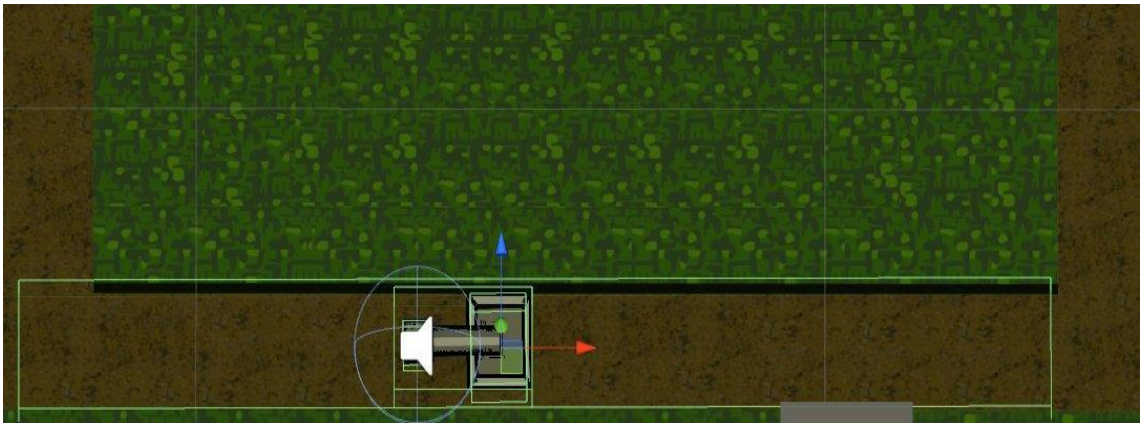
Hyökätessään vihollinen pitää myös huomiota herättävää ääntä, mikä tekee hyökkäävän vihollisen tunnistamisesta pelaajalle helpompaa. Vihollisen äänimaailmaa toimi tyttöystäväni, jonka suorituksen nauhoitin omalla nauhurillani ja miksasin Cubasella. Hyökätessään vihollinen poistuu oletuskulkureitiltään ja kohdistaa suuntansa pelaajan sijaintiin. Saadakseni vihollisen liikkumaan vain tietyillä alueilla lisäsin siihen Unityn NavMesh-komponentin jonka avulla määritin ne kulkualueet, joita pitkin vihollinen voi kulkea (kuva 20). Vihollinen ei voi liikkua sokkelon ulkopuolella, joten jos pelaaja poistuu sokkelosta kesken hyökkäyksen, palaa vihollinen takaisin oletuskulkureitilleen.

Kuva 20. Navmesh-komponentin määrittämä vihollisen kulkureitti.

5.5.2 Kaukotaisteluun erikoistunut vihollinen

Pelin kaukotaisteluun erikoistunut vihollinen on eräänlainen pelimaailmaan sijoitettu tykkitorni, joka ampuu kohti pelaajaa, mikäli tämä tulee liian lähelle. Vihollisen hyökkäysalueen määrittää vihollisobjektin sisällä oleva pelitilanteessa näkymätön objekti, joka on skaalattu peittämään vihollisen etu- ja takaosaa kentästä. Mikäli pelaaja tulee hyökkäysalueelle vihollisen takaa, kääntyy vihollinen ympäri ja alkaa ampua kohti pelaajaa(kuva 21).

Vihollisen kääntyminen on toteutettu erillisen metodin sisällä niin, että vihollisen läsnäolon tunnistava OnTriggerEnter kutsuu kääntymismetodia tykkitornin toiminnoista vastaavasta kooditiedostosta. Tämä metodi puolestaan sisältää jo valmiiksi tiedon pelaajan sijainnista ja määrittää oman rotaatio-arvonsa kääntymään kohti pelaajaa käyttämällä Quaternion-luokan Slerp-metodia. Tämä mahdollistaa sen, että siirtyminen haluttuun kulmaan tapahtuu pikku hiljaa, eikä tykkitorni ole heti osoittamassa pelaajaan. Vihollisen ammukset ovat Unityn sphere-objekteja, jotka tunnistavat pelaajan Unityn sisäisen tag-nimen avulla. Mikäli ammus osuu pelaajaan, menettää tämä osan terveydestään.



Kuva 21. Tykkitorni ampuu pelaajan hahmoa, mikäli tämä tulee liian lähelle.

5.6 Taistelujärjestelmä

Päätin jo ennen varsinaisen työn aloittamista, että haluaisin toteuttaa roolipelin, joka sisältäisi reaaliaikaisen taistelujärjestelmän vuoropohjaisen sijaan. Pelaaja kykenisi hyökkäämään neljään eri suuntaan ja määrittämään haluamansa

suunnan hahmonsa menosuunnan mukaan. Järjestelmä toimisi näin pintapuolisesti samalla tavalla kuin esimerkiksi vanhoissa Zelda-peleissä.

Aloitin järjestelmän toteuttamisen luomalla pelihahmon ympärille neljä eri seinää, yhden hahmon eteen, toisen taakse ja yhden molemmille sivuille. Tämän jälkeen lisäsin näihin seiniin Box Colliderin trigger-asetuksella niin, että objektit tunnistaisivat törmäyksen haluamieni objektien kanssa. Tämän jälkeen loin kooditiedoston ja raahasin sen jokaiseen seinäobjektiin. Kooditiedostoon itseensä loin OnTriggerEnter-metodin, jonka sisään lisäsin vahingon aiheuttamislogiikan. Metodi toimii niin, että joutuessaan kosketuksiin peliobjektin kanssa, joka on tag-nimellä merkitty viholliseksi, metodi hakee pelihahmon liikkumisesta vastaavasta kooditiedostosta tiedon, onko pelihahmo kasvotusten vihollisobjektin kanssa. Jos pelihahmo on kasvotusten vihollisen kanssa ja painaa hyökkäysnappia, niin metodi hakee vihollisen terveysarvon ja pienentää sitä. Jos taas pelaajan hahmo ei täytä kumpaakin näistä ehdoista, niin metodi hakee pelaajan hahmon terveysarvon ja pienentää sen arvoa.

Pelaajalla on hyökkäämistä varten käytettävissä miekka, jonka hän voi löytää aloituspaikan sisällä olevasta arkusta. Miekan hyökkäystoiminto sisältää oman animaationsa, jonka graafisen ulkoasun toteutin Unityn ulkopuolella. Animaatioiden toimintoja hallinnoin ohjelmakoodissa Unityn oman Animator-työkalun avulla. Hyökkäyslogiikan sisältävä kooditiedosto lähettää Animatorille eri arvoja sen mukaan, mihin suuntaan pelaaja on menossa ja painaako hän hyökkäysnappia. Näiden arvojen perusteella Animator sitten toistaa määrittelemiäni animaatioita oikealla hetkellä. Animaatiot on linkitetty toisiinsa niin, että osa vaihtuu animaatiosta toiseen pelaajan liikkumisen mukaan kun taas hyökkäysanimaatio lopettaa itse itsensä tietyn ajan kuluessa, mikäli Animator ei ole saanut toistokäskyä.

5.7 Keskustelujärjestelmä

Aiemmat kokemukseni sarjakuvien piirtämisen parissa saivat minut kokeilemaan keskustelujärjestelmän luomista, jossa hahmojen välinen keskustelu ta-

pahtuisi puhekuplien välityksellä. Päädyin rajallisen ajan vuoksi tekemään yksinkertaisen keskustelujärjestelmän, joka hyödyntää koodissa Unityn GUI-luokkaa ja julkista string-taulukkoa. Tekemällä string-taulukon julkiseksi pystyin lisäämään taulukkoon tekstin Unityn inspectori-näkymässä ja varmistamaan, että pystyisin myös käyttämään luomaani koodia uudelleen toisessa tilanteessa toisen tekstin kanssa.

Piirsin Photoshopilla puhekuplalle tekstuurin, uudelleen käytin pelaajahahmon yhtä tekstuuria ja loin näistä pelimaailmaan NPC -hahmon, joka puhuu pelaajalle tämän lähestyessä häntä. Teknisesti ajatellen siis loin kooditiedostoon OnTriggerStay-metodin, joka tunnistaa lähestyvän pelaajan ja tuo näkyviin puhekuplan muuttamalla Boolean-arvoa (kuva 22).Mikäli pelaaja poistuu alueelta, puhekupla häviää näkyvistä. Kuplan hävittämisessä käytin IEnumeratoriksi määrittämääni OnTriggerExit-metodia, joka muuttaa dialogin näkymättömäksi muuttamalla Boolean-muuttujan arvoa(kuva 22). Mikäli tämän Boolean-muuttujan arvo on tosi, suorittaa OnGUI sisältämänsä komennot, ja ruudulle piirtyy puhekupla, joka neuvoo pelaajalle näppäinkomennon juoksemiselle(kuva 23).

```
void OnTriggerStay (Collider other)
{
        if (other.tag == "Player") {
                showDialog = true;
        }
}


IEnumerator OnTriggerExit (Collider other)
{
        yield return new WaitForSeconds (3);
        showDialog = false;
}
```

Kuva 22. Puhekuplan näkyvyyttä hallinnoivat metodit.

```
void OnGUI ()
{
        GUI.BeginGroup (new Rect (goScreenPos.x - offsetX, Screen.height - goScreenPos.y - offsetY, bubbleWidth, bubbleHeight));
        if (showDialog) {

                GUI.DrawTexture (new Rect (bubbleX, bubbleY, 130, 90), tex);
                GUI.color = Color.black;
                GUI.Label (new Rect (textX, textY, 120, 30), npcDialog [0]);
                GUI.Label (new Rect (textX, textY + 11.0f, 120, 20), npcDialog [1]);
                GUI.Label (new Rect (textX, textY + 20.0f, 120, 20), npcDialog [2]);
        }
        GUI.EndGroup ();

}
```

Kuva 23. Puhekuplan ruudulle piirtävä metodi.

5.8 Resurssit ja niiden hallinta

Tässä luvussa käyn läpi pelin sisäisiä resursseja ja niiden hallintaa. Opinnäytetyön peliprototyypissä voi kerätä esineitä, hallinnoida niitä tavaraluettelonäkymässä ja myydä ja ostaa niitä pelin sisäisessä kaupassa.

5.8.1 Kerättävät esineet

Pelimaailmasta voi kerätä kahta erilaista tyyppiä olevia resursseja. Näitä ovat kulta ja aseet. Keräämällään kullalla pelaaja voi puolestaan ostaa lisää tavaroita kaupasta. Pelaajan hahmo kykenee taistelemaan vain jos hänellä on ase. Mikäli pelaajalla ei ole asetta, hän ei kykene kukistamaan ympäristössä olevia vihollisia. Pelaaja saa resursseja ympäristöstä löytyvistä arkuista, ostamalla kaupasta tai kukistamalla vihollisia.

Pelin arkut loin piirtämällä ensiksi sprite-tekstuurit, joissa toisessa arkku on kiinni ja toisessa auki. Sitten loin Unityssa objektin, johon lisäsin Box colliderin ja tekstuurin, jossa arkku on kiinni. Box colliderin asetin arkun ympäri niin, ettei pelaaja pääsisi kävelemään arkun läpi. Tämän lisäksi loin toisen Box colliderin hoitamaan pelin sisäisen triggerin tehtävää. Tämä triggeri mahdollistaisi sen, että arkku tunnistaa pelaajan olevan lähellä tämän ohjaaman hahmon tagnimestä. Näin arkku reagoi pelaajaan näppäinkomentoon vain pelaajan ollessa lähellä arkkua. Seuraavaksi aloitin arkun toimintalogiikan tarkemman määrittämisen. Suunnittelin arkun niin, että se avautuisi pelaajan painaessa välilyöntinäppäintä samalla kun hän on arkun lähellä. Tämä komento puolestaan vaihtaisi arkun tekstuurin toiseen riippuen siitä, onko arkku komentohetkellä auki vai kiinni. Tekstuurin vaihtamisen lisäksi arkku lähettää pelaajalle viestin poimitusta esineestä ja lisää poimitun esineen pelaajan tavaraluetteloon.

Pelin sisäisille esineille päätin toteuttaa niin, että jokainen erillinen esine saisi oman Unityn sisäisen peliobjektinsa, ja sen sisään kooditiedoston, jossa olisi erikseen määritelty muuttujat esineen nimelle, tavaraluettelotekstuurille sekä muille ominaisuuksille. Toteutin esineiden ominaisuudet määrittävän ohjelmointikoodin niin, etten alustanut arvoja kooditiedoston itsensä sisällä vaan tein kai-

kista muuttujista julkisia ja annoin niille arvot vasta peliobjektin sisällä. Tämä mahdollisti sen, ettei minun tarvinnut luoda jokaiselle esineelle omaa kooditiedostoa, vaan pystyin uudelleen käyttämään samaa tiedostoa jokaisessa esineelle luodussa objektissa. Loin osalle objekteista kaksi tekstuuria tavaraluettteloa varten, toisen kertomaan pelaajalle, että esine on tavaraluettelossa ja toisen kertomaan, että esine on tavaraluettelossa ja aktivoituna käyttöön. Tämän järjestelmän avulla pelaaja näkisi nopeasti yhdellä vilkaisulla, onko esimerkiksi joku tietty ase aktivoituna pelaajan käytettäväksi.

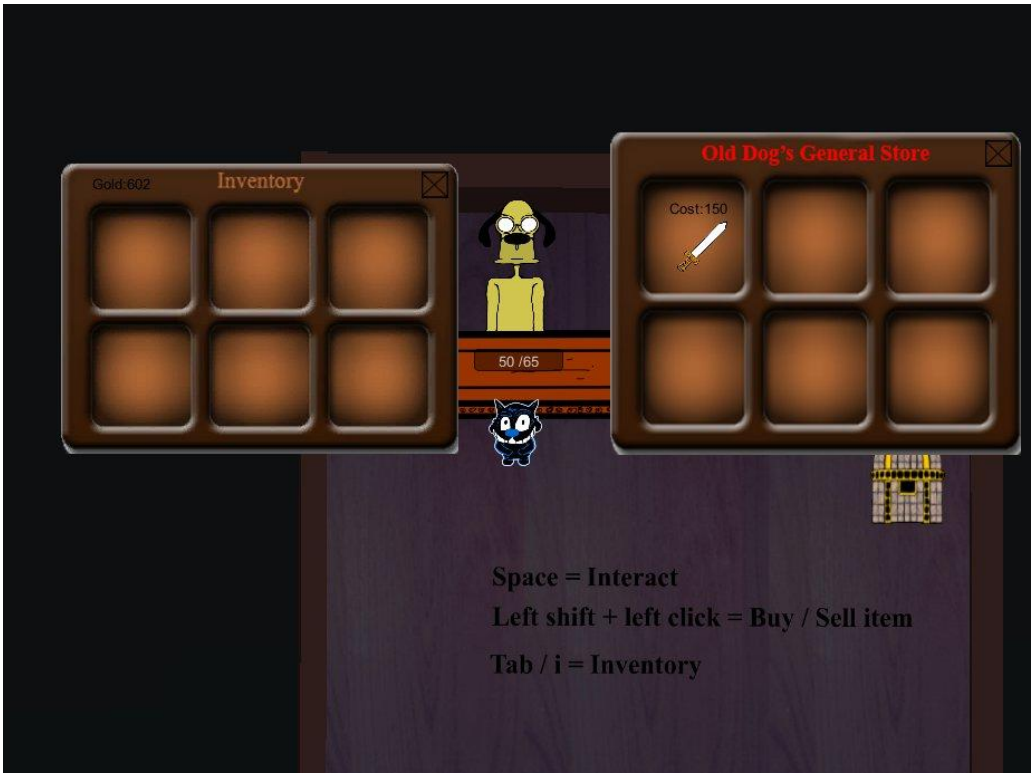
5.8.2 Tavaraluettelo

Ensimmäiseksi loin Photoshopissa tavaraluettelo-ikkunaksi soveltuvan tekstuurin, josta käsin pelaaja kykenisi helposti aktivoimaan, pudottamaan tai myymään pelimaailmasta löytämiään esineitä. Lisäksi loin erillisen tekstuurin ikkunan sulkemisnapille, jonka pystyisin sitten sijoittamaan tavaraluettelo-ikkunan oikeaan ylälaitaan koodista käsin. Toin molemmat tekstuurit Unityyn png-muodossa ja aloitin ohjelmointipuolen suunnittelemisen. Lopulta päädyin ratkaisuun, jossa tavaraluettelon logiikasta vastaava kooditiedosto sisältää julkiseksi määritetyn taulukon, johon olen raahannut jokaisen aiemmin luomani eri esinetyypin objektin. Näin pystyn helposti käsittelemään pelimaailman kerättäviä objekteja tavaraluettelossa nopeasti ilman, että minun tarvitsee ajonaikana erikseen kutsua objekteja taulukkoon.

Tavaraluettelon loin käyttämällä Unityn GUI-luokkaa. Loin sen avulla ikkunan ja sen sisälle sulkemisnapin aiemmin luomallani tekstuurilla. Saatavat esineet loin ikkunaan nappeina, jotta niiden klikkaaminen tuottaisi niihin määrittelemiäni toimintoja. Lopuksi loin jokaista esinetyyppiä kohden oman Boolean-arvon, joka määrittää, löytyykö pelaajalta kyseistä esinettä vai ei. Tämän jälkeen laitoin jokaisen esinenapin näiden totuusarvoa testaavien ehtolauseiden sisään. Näin yksikään esine ei voi näkyä tavaraluettelonäkymässä, mikäli pelaaja ei ole löytänyt kyseistä esinettä.

5.8.3 Tavaroiden ostaminen ja myyminen

Pelin sisäisen kaupankäynnin logiikan toteutin niin, että kaupan tavaraluettelo avautuu pelaajan tullessa myyjän eteen ja painaessa välilyöntiä(kuva 24). Toiminnon laukaisijana toimii OnTriggerStay-metodi, joka tunnistaa pelaajan tämän tagi-nimen avulla ja tarkistaa painaako pelaaja välilyöntiä ollessaan Colliderin määrittämällä alueella. Jos pelaaja painaa välilyöntiä, kutsuu metodi kauppaikkunaan luomaa metodia, ja ottaa samalla yhteyden pelaajan tavaraluetteloluokkaan ja kertoo sille Boolean-arvolla, että pelaaja suorittaa kaupankäyntiä.



Kuva 24. pelaajan ja kaupan tavaraluettelo kauppatilanteessa.

Jos pelaaja avaa oman tavaraluettelonsa suorittaessaan kaupankäyntiä, luo pelaajan tavaraluettelo-logiikkaa kontrolloiva kooditiedosto normaalista poikkeavan tavaraluettelonäkymän (kuva 24). Tämä tavaraluettelonäkymä eroaa normaalista sillä tavalla, että esineiden klikkaaminen myy esineen normaalin käyttöönoton sijaan. Tämän lisäksi tavaraluettelo näyttää esineestä myymällä saatavan kullan määrän. Mikäli pelaaja päättää myydä esineen, poistuu esinekuvake hänen tavaraluettelostaan ja siirtyy kaupan tavaraluetteloon. Samalla pelaaja saa esineestään ennalta määritetyn määrän kultaa omaan tavaraluette-

loonsa. Jos pelaaja päättää ostaa esineen, poistuu se kaupan valikoimasta ja siirtyy pelaajan tavaraluetteloon.

6 Pohdinta

Tässä luvussa käyn läpi tuntemuksia, joita opinnäytetyön toteuttaminen tekijässään herätti, opinnäytetyön tekemisestä saatuja tuloksia ja ideoita siitä, miten opinnäytetyön osana syntynyttä roolipeliprototyyppiä voitaisiin jatkokehittää varsinaisen opinnäytetyöprosessin jälkeen.

6.1 Toteutus

Roolipeliprototyypin toteuttaminen oli minulle täysin uusi kokemus. Ennen tätä projektia olin toteuttanut vain pienempiä projekteja muilla kehitysympäristöillä. Olin kuitenkin pohtinut isomman projektin toteuttamismahdollisuuksia jo aiemmin, mutta en muiden opintojeni ohella ehtinyt aloittaa koskaan vastaavaa. Opinnäytetyön suunnitteluvaiheen alettua päätin, että nyt olisi aika luoda projekti, joka opettaisi minulle mahdollisimman paljon uutta ja samalla toimisi työnnäytteenä siitä, mitä olen koulutukseni aikana pystynyt oppimaan.

Alun perin suunnittelin työni toteuttamista XNA-kehitysympäristöllä, sillä ajattelin sen olevan helpoin ympäristö 2D-pelin toteuttamiseen. Microsoft kuitenkin ilmoitti vielä työni ollessa suunnitteluasteella, että XNA:n kehitys lopetetaan. Tämä sai minut etsimään vaihtoehtoja muista kehitysympäristöistä. Olin aiemmin tutkinut Unreal Development Kitiä ja sen mahdollisuuksia, joten sillä prototyypin toteuttaminen tuntui aluksi hyvältä idealta. Kehitysympäristö alkoi kuitenkin tuntua jo ennen työn aloittamista liian vaikealta hallita ja tunsin, ettei UDK:n dokumentaatio ja saatavilla ollut lähdekirjallisuus pystynyt opastamaan minua tarpeeksi, jotta saisin työkalulla riittävän nopeasti tuloksia aikaiseksi.

Lopulta päädyin valitsemaan kehitysympäristöksi Unity 3D:n, mikä osoittautui oikeaksi ratkaisuksi. Opinnäytetyöprosessin aikana olin jo onnistunut luomaan 2D-elementtejä peliin, mutten ollut täysin tyytyväinen niiden laatuun. Onneksi Unity päivittyi prosessin aikana versioon 4.3, joka tarjosi uudet työkalut 2D-pelinkehitykseen. Etenkin tuki Sprite-tiedostomuodolle ja mahdollisuus editoida ja animoida Sprite-tiedostaja Unityssa osoittautuivat tärkeiksi uusiksi työkaluiksi projektin edetessä. Niiden avulla kykenin parantamaan työskentelytapojani huimasti alkuperäisestä, ja lopputuloksesta tuli näin niin teknisesti kuin visuaalisesti paljon laadukkaampi.

6.2 Tulokset

Opinnäytetyön tuloksena syntyi roolipeliprototyyppi, joka oli mielestäni monin paikoin onnistunut työ. Sen lisäksi, että luomisprosessi opetti minulle paljon uutta roolipelien kehittämisestä, opin myös paljon uutta Unity 3D:stä kehitysympäristönä. Prototyyppi on jo tässä vaiheessa mielestäni sen verran onnistunut, että pystyn käyttämään sitä referenssinä osaamisestani. Käytännön työn lisäksi opinnäytetyöhön kuului tämän raportin kirjoittaminen, joka myös onnistui hyvin vastaamaan niihin kysymyksiin, jotka prosessin alkupuolella määritin.

6.3 Kehittämisideat

Opinnäytetyön osana syntyneellä prototyypillä on mielestäni erittäin hyvät mahdollisuudet kehittyä oikeaksi peliksi. Vaikka suunnittelinkin prototyypin perusidean eri roolipelielementtien ympärille, voi lopputulosta kehittää muokkaamalla ja lisäämällä sisältöä. Ensimmäiseksi pelin ympärille tulisi rakentaa tavoitteet, jotka pelaajan tulee saavuttaa menestyäkseen pelissä.

Pelille pitäisi myös suunnitella tarina, jonka pohjalta olemassa olevia elementtejä voisi lähteä muokkaamaan tarinan sisällön vaatimaan suuntaan. Myös monet prototyyppiin toteuttamani elementit ovat pelkistetyn yksinkertaisia ja monia niistä voitaisiin kehittää paljon eteenpäin. Esimerkiksi hahmonkehitys sisältää tällä hetkellä vain kolme kehitettävää ominaisuutta. Lisäksi pelimaailmaan tulisi lisätä

paljon lisää löydettäviä esineitä. Esimerkiksi kaupasta ei voi tällä hetkellä ostaa kuin yhden tavaran ja viholliset pudottavat tuhoutuessaan vain kultaa. Toteutin prototyypin keskittäen ajankäyttöni ensisijaisesti ohjelmointiin ja vasta sitten ääniin ja graafiseen ulkoasuun, joten myös pelin audiovisuaalista ulkoasua tulisi parantaa nykyisestä, jotta voitaisiin lähteä puhumaan oikeasta pelistä.

Lähteet

Barton, M. 2007. The History of Computer Role-Playing Games Part 1: The Early Years(1980-1983).UBM Tech. http://www.gamasutra.com/view/feature/132024/the_history_of_computer_.php?page=1. 19.11.2013.

Barton, M. 2008. Dungeons and Desktops : The History of Computer Role-Playing Games.Natick: A K Peters, Ltd.

Bethesda Game Studios. 2011. The Elder Scrolls V: Skyrim. Bethesda Softworks.

Bioware Edmonton. 2009. Dragon Age: Origins. Electronic Arts.

BioWare. 2012. Mass Effect 3. Electronic Arts.

Bioware. 2013. About. Electronic Arts. http://www.bioware.com/en/about/. 04.09.2013.

Blizzard Entertainment. 2012. Diablo 3. Blizzard Entertainment.

CD Projekt RED. 2011. Witcher 2: Assassins of Kings. Bandai Namco Games.

Clayman, D. 2010. The History of Blizzard. IGN Entertainment. http://www.ign.com/articles/2010/10/22/the-history-of-blizzard?page=2. 03.09.2013.

Eidos Montreal. 2011. Deus Ex: Human Revolution. Square Enix.

Epic. 2013. UDK. Epic. http://www.unrealengine.com/udk/. 14.09.2013.

Epic.2012.DevelopmentKitProgramming.Epic. http://udn.epicgames.com/Three/DevelopmentKitProgramming.html. 04.01.2014.

Fog Creek Software. 2013. Instant clarity on any project. Fog Creek Software. https://trello.com/tour. 14.09.2013.

Fullerton, T. 2008. Game design workshop: a playcentric approach to creating innovative games. Burlington: Elsevier.

Google. 2013. Google Drive. Google. http://www.google.com/drive/about.html. 14.09.2013.

Handrahan, M. 2013. Richard Garriott: The Power of crowd-sourced development. Gamer Network. http://www.gamesindustry.biz/articles/2013-10-08-richard-garriott-the-power-of-crowd-sourced-development. 31.03.2014

Humble Hearts. 2011. Dust: An Elysian Tail. Microsoft Studios.

Image-Line. 2013. Introducing FL Studio 11. Image-Line Software. http://www.image-line.com/documents/flstudio.html. 11.09.2013.

Kärkkäinen, T. 2014. Andrzej Sapkowski. Kärkkäinen. http://www.saunalahti.fi/tapank/index2.html. 27.03.2014.

Kasavin, G. 1997. Final Fantasy VII Review. CBS Interactive. http://us.gamespot.com/final-fantasy-vii/reviews/final-fantasy-vii-review-2547583/. 03.09.2013.

Laaksonen, J. 2006. Äänityön kivijalka. Helsinki: Riffi.

Manninen, T. 2007. Pelisuunnittelijan käsikirja: ideasta eteenpäin.Pello: Rajalla.

Microsoft. 2013. Microsoft XNA Game Studio 4.0. Microsoft. http://www.microsoft.com/en-us/download/details.aspx?id=23714. 12.09.2013.

Monodevelop. 2013. Monodevelop. http://monodevelop.com/. 04.09.2013.
Obsidian Entertainment. 2010. Fallout: New Vegas. Bethesda Softworks.
Paananen, P. 2010. Photoshop CS5 kuvankäsittely. Jyväskylä: WSOYpro Oy.
Paananen,P. 2011.Flash CS4 & CS5-julkaisijan opas.Jyväskylä: WSOYpro Oy.
Petterson, T. 2012. Sfxr. Petterson. http://drpetter.se/project_sfxr.html. 11.09.2013.
Pyykkönen, J. 2009. Dragon Age: Origins. H-Town Oy. http://www.pelaajalehti.com/arvostelut/dragon-age-origins. 03.09.2013.
Rose, M. 2013. It's official: XNA is dead. UBM Tech. http://www.gamasutra.com/view/news/185894/. 12.09.2013.
Sarkar, S. 2012. Humble origins: the solo odyssey behind Dust: An elysian tail. Vox Media. http://www.polygon.com/2012/11/14/3554380/humble-origins-the-solo-odyssey-behind-dust-an-elysian-tail. 12.09.2013.
Square.1997. Final Fantasy VII.SCEE.
Unity. 2011. Audio Filters(Pro only). Unity Technologies. http://docs.unity3d.com/Documentation/Components/class-AudioEffect.html 12.09.2013.
Unity. 2013a. Multiplatform. Unity Technologies. http://unity3d.com/unity/multiplatform/. 04.09.2013.
Unity. 2013b. What's new. Unity Technologies. http://unity3d.com/unity/whats-new. 14.11.2013.
Vigil Games. 2012. Darksiders 2. THQ.
Ward, T. 1997. Diablo review. CBS Interactive. http://www.gamespot.com/diablo/reviews/diablo-review-2538662/ 03.09.2013.