Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Kerkko Rasilainen

Rainyday-tietokonepelin kehitys ja julkaisu

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

15.12.2018

Tekijä Otsikko Sivumäärä Aika	Kerkko Rasilainen Rainyday-tietokonepelin kehitys ja julkaisu 35 sivua 15.12.2018
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikka
Ammatillinen pääaine	Ohjelmistotuotanto
Ohjaajat	Lehtori Simo Silander

1980-luvulla alkanut seikkailupelibuumi loi puitteet uudenlaisille pelisovelluksille, joissa pelaajien ohjaamat protagonistit ratkaisivat ongelmia ja keskustelivat muille pelihahmoille ennennäkemättömän graafisia pelimaailmoja tutkien.

Insinöörityön tavoitteena oli toteuttaa itsenäisesti point-and-click-tyylinen, laajennettavissa oleva, episodipohjainen seikkailupeli. Työssä oli tarkoitus tutkia, minkälainen prosessi tietokonepelin ohjelmointi nykään on ja miten se kannattaa toteuttaa. Valmis pelisovellus oli tarkoitus julkaista jossain sopivassa jakelualustassa niin, että sitä voisi jatkossa päivittää ja jatkaa pelaajien tarpeiden mukaisesti.

Työ sisälsi tietokonepelien historiaan tutustumista, pelikehitykseen tutustumista, pelisovelluksen suunnittelun ja ohjelmoinnin Javalla NetBeansilla libGDX-sovelluskehystä hyödyntäen, pelikehityksessä käytettäviin työkaluihin perehtymistä, pelissä käytettävien ominaisuuserien luomisen grafiikka-, kartta- ja äänisuunnittelutyökaluilla sekä niiden jalkauttamisen peliohjelmaan. Tämän lisäksi työhön sisältyi myös valmiin pelin julkaisu Steam-jakeluverkossa prosesseineen, julkaistun sovelluksen päivittämistä ja lisäosien kehittämistä.

Insinöörityön lopputuloksena saatiin valmis, Steam-jakelualustassa myytävä pelisovellus sekä vahva käsitys siitä, minkälainen prosessi tietokonepelin ohjelmointi ja toteuttaminen on. Peli toteutettiin ja julkaistiin suunnitelman mukaisesti kaikkine suunniteltuine ominaisuuksineen. Se on edelleen jatkuvan kehityksen kohteena ja siihen on julkaisun jälkeen lisätty huomattava määrä ominaisuuksia, päivityksiä ja episodeja ilmaisina päivityksinä pelin omistajille.

Rainyday Steamissa:
https://store.steampowered.com/app/712980/Rainyday/

Avainsanat	Rainyday, libGDX, Java, tietokonepeli, Steam

Author Title Number of Pages Date	Kerkko Rasilainen Development and Release of The Rainyday Computer Game 35 pages 15 December 2018
Degree	Bachelor of Engineering
Degree Programme	Information and Communication Technology
Professional Major	Software Engineering
Instructors	Simo Silander, Senior Lecturer

The adventure game boom that began in the 1980's created a ground for a new kind of video games where protagonists, controlled by players, solved problems and talked with other game characters while exploring in graphical worlds never seen before.

The goal of this thesis was to solo develop a point-and-click like expandable episode based adventure game. The purpose was to study the process of the development of a modern computer game and how it is supposed to be executed. The completed computer game was supposed to be released in a suitable distribution platform so it could be updated and developed further by players' needs.

The project included getting to know the history of computer games and to the game development, designing the game software and programming the game with Netbeans using Java's libGDX framework, orientation of the tools used in the game development, creation of the game's assets using graphics-, map- and sound design software and implementation of these assets to the game software. Addition of these, the work included the release of the completed game in a Steam distribution network with it's processes, updating the released software and the development of the expansions for the software.

The final result of the engineering was a completed game, sold in the Steam's distribution platform, and a strong opinion of what kind of process a programming and a implementation of a computer game is. The game was implemented and released as planned with all features, it is still under constant development and there has been a notable amount of content, upgrades and episodes that have been added as free updates for the owners of the game.

Rainyday in Steam:
https://store.steampowered.com/app/712980/Rainyday/

Keywords	Rainyday, libGDX, Java, computer game, Steam

Sisällys

Lyhenteet ja käsitteet

1 Johdanto 1

2 Point-and-click -seikkailupelit osana pelialaa 1

3 Rainyday-pelisovellus 4

3.1 Yleisesti 4
3.2 Käytetyt tekniikat ja ohjelmistot 6
3.2.1 libGDX 6
3.2.2 Git 8
3.2.3 Tiled 9
3.2.4 Paint.net, Photoshop ja Audacity 10
3.2.5 Launch4j 13
3.2.6 Steamworks SDK 15
3.3 Pelin tekninen toteutus 15
3.3.1 Oleelliset luokat ja metodit 15
3.3.2 Ruudukko ja liikkuminen 20
3.3.3 Vuorovaikuttaminen pelimaailman kanssa 23
3.3.4 Tallentaminen 25
3.3.5 Steam-rajapinta 26

4 Julkaisu Steamissa 27

4.1 Steam Direct 29
4.2 Steamworks 30
4.3 Julkaisu ja jatkuva kehitys 31

5 Yhteenveto 33

Lähteet 35

Lyhenteet

AGI	Adventure Game Interpreter. Sierra On-Linen kehittämä pelimoottori, jota käytettiin 1980-luvun seikkailupeleissä.
EGA	Enchanted graphics adapter. Vuonna 1984 kehitetty näyttöstandardi, joka sisältää 16 värin paletin sekä 640x350 pikselin resoluution.
EXE	Executable file. Suoritettavan tietokoneohjelman tiedostomuoto. Käyttöjärjestelmä suorittaa tiedostossa olevan ohjelmakoodin ajettaessa.
GIT	Pääosin lähdekoodin hallinnassa ja ohjelmistokehityksessä käytetty versionhallintaohjelmisto, joka mahdollistaa hajautetun kehityksen.
GUI	Graphical user interface. Kuviin, tekstiin ja käyttöliittymäelementteihin perustuva graafinen käyttöliittymä.
IDE	Integrated development environment. Kokonaisuus, joka yhdistää ohjelmistokehityksessä käytettävät perustyökalut koodin editoimiseen, ajamiseen ja virheenkorjaamiseen.
JAR	Java archive. Pakkausformaati, jota käytetään yleensä pakkaamaan Java-tiedostoja, niihin liittyvää tietoa sekä resurssitiedostoja.
JRE	Java Runtime Environment. Ajoympäristö ja minimivaatimus Java-ohjelmistojen suorittamiseen. Sisältää Java Virtual Machinen, ydinluokat sekä tukitiedostot.
NPC	Non-playable-character. Videopeleissä käytetty nimitys automatisoidusta ei-pelaaja-hahmosta.

SDK	Sofware development kit. Työkalupaketti, jolla voidaan luoda sovelluksia, rajapintoja, alustoja, käyttöjärjestelmiä tai pakkauksia johonkin tiettyyn tietokoneohjelmistoon.
TMX	Tile Map XML. Karttaformaatti, jolla voidaan kuvailla ruutupohjainen kartta ominaisuuksineen, kerroksineen ja objekteineen.

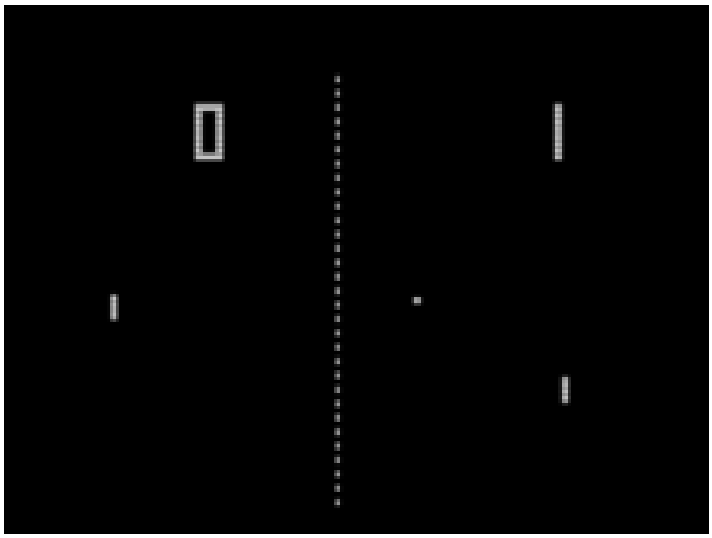
1 Johdanto

Tämän insinöörityön tavoitteena oli pelisovelluksen ohjelmointi ja sen julkaiseminen. Tarkoituksena oli tutkia, minkälainen prosessi oman tietokonepelin ohjelmointi kokonaisuudessaan on sekä miten peli kannattaa nykypäivänä julkaista. Tässä työssä kuvataan seikkailupelien taustojen lisäksi pelin tekemisen eri työvaiheet, oleelliset käytetyt tekniikat ja ohjelmistot sekä julkaisuun liittyvä prosessi.

Projekti aloitettiin tutustumalla tietokonepelien historiaan, peleihin yleisesti, seikkailupeligenreen, lajityypin kuuluisimpiin edustajiin sekä ideoimalla toteutettavan pelin peruslähtökohdat. Tämän jälkeen selvitettiin, mikä on parhaaksi todettu tapa ohjelmoida peli Javalla. Lisäksi tutustuttiin projektissa esiteltyihin ominaisuuserien luomiseen tarkoitettuihin ohjelmiin, Launch4j-käärimeen sekä Steamin kehittäjille tarjoamiin rajapintoihin sekä julkaisutyökaluihin. Näiden tietojen perusteella alettiin rakentamaan peliä Rainyday.

2 Point-and-click-seikkailupelit osana pelialaa

1970-luvulta lähtien, pelikonsolien massatuotannon sekä kotitietokoneiden rantautumisen myötä, pelaamisesta on tullut pysyvä osa ihmisten vapaa-aikaa. Hyvin alkeellisista arcade-peleistä (ks. kuva 1) kehittynyt viihdeteollisuuden haara on tänä päivänä yksi maailman tuottavimmista viiheen aloista. [1.]



Kuva 1. Alkuperäinen, vuonna 1972 Atarille julkaistu Pong.

Pelialan suosituimpia lajityyppejä ovat nykyään The Top Tens -sivuston mukaan roolipelit, hiekkalaatikkopelit, toiminta- ja seikkailupelit sekä ensimmäisen persoonan ampumispelit [2]. Peliala on kasvanut muutamassa vuosikymmenessä räjähdysmäisesti ja uusia lajityyppejä ja alalajeja ilmestyy vuosittain yhä enemmän. Uusimpana tulokkaana voidaan pitää käsittämättömään suosioon noussutta Battle Royale -lajityyppiä, joka on moninpelattava, usean eri lajityypin sekoitus, jossa samalle alueelle pudotettavat pelaajat taistelevat kuolemaan ja viimeiseen pelaajaan asti, aseita, esineitä ja kehittäviä parannuksia haalien.

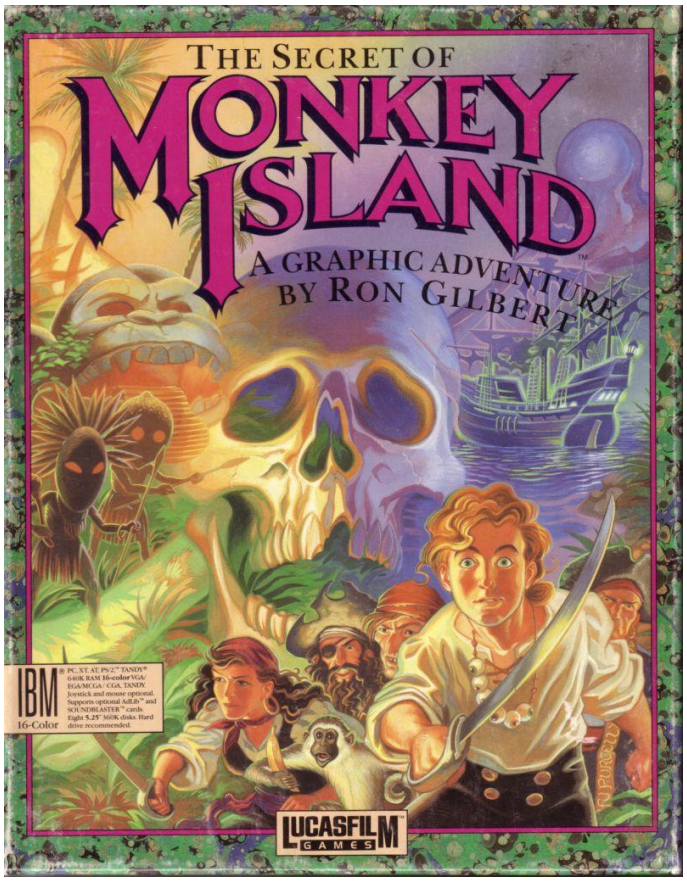
Seikkailupelillä tarkoitetaan yleensä tarinavetoista yksinpeliä, jossa pelaaja ohjaa päähenkilöä tutkimalla pelimaailmaa, keskustelemalla muiden pelihahmojen kanssa sekä ratkaisemalla erinäisiä pelin asettamia pulmia. Point-and-click -nimitys tulee seikkailupelin toiminta- ja ohjauslogiikasta, jossa pelaaja kirjaimellisesti osoittaa yleensä hiirellä liikuteltavalla osoittimella jotakin pelimaailman esinettä tai hahmoa ja klikkaa aiheuttaakseen vuorovaikutuksen.

Genren suurimpana yksittäisenä vaikuttajina voidaan pitää 1983 Sierra On-Linen julkaisemaa AGI-pelimoottorilla toteutettua King’s Quest: Quest for the Crown -peliä, joka aloitti 90-luvun loppuun asti jatkuneen seikkailupelibuumin. Peli käytti tuohon aikaan näyttävää 16-väristä EGA-grafiikkaa (ks. kuva 2) ja tekstinjäsentäjää, jolle pelaaja syötti

näppäimistöllä haluamansa pelihahmon ohjauskomennot. Kings Questille tehtiin lukuisia jatko-osia sekä vastaavaa tekniikkaa hyödyntäviä muita pelisarjoja, kuten Space Quest sekä Police Quest. [3.]

Kuva 2. Vuonna 1987 julkaistusta Space Quest II: Vohaul's Revenge -seikkailupelistä napatussa kuvankaappauksessa näkyy aikakaudelleen ominaista EGA-grafiikkaa.

Sierran lisäksi vähintään yhtä paljon genreen vaikuttanutta pelistudiota pidetään LucasArtsa (ks. kuva 3), jonka ensimmäinen, vuonna 1986 julkaistu, graafinen seikkailupeli oli Labyrinth. Tekstinjäsentäjän sijaan peli käytti toiminnoissaan tekstilaatikoita, joista toisesta pelaaja valitsi verbin ja toisesta substanttiivin. LucasArtsin tunnetuimpia tuotoksia ovat Indiand Jones and the Last Crusade (1988) sekä valtavaan suosioon 90-luvulla nousseet Monkey Island- sekä Sam and Max -pelisarjat. [4.]



Kuva 3. LucasArtsin Monkey Island -pelin kansitaidetta vuodelta 1990.

Uusien peligenrejen ilmestymisen, mobiili- sekä konsolipelaamisen sekä jatkuvasti palkitsevien moninpelien myötä klassiset, hitaat ja ajattelemista vaativat yksinpelattavat point-and-click-seikkailupelit ovat menettäneet suosiotaan huomattavasti ja niitä arvostetaan nykyään enemmänkin vain nostalgisessa merkityksessä sekä vanhemman sukupolven pelaajien keskuudessa.

3 Rainyday-pelisovellus

3.1 Yleisesti

Rainyday on ylhäältäpäin kuvattu, yksinpelattava, vanhanaikainen Point-and-click-henkinen pulma- ja seikkailupeli, joka tehtiin kunnioittamaan 90-luvun klassikkoteoksia

ja seikkailupeligenreä, jota ei sellaisenaan enää ole. Idea pelin teosta syntyi Tieto- ja viestintätekniikan tutkinto-ohjelman Pelit-teeman opintojen aikana.

Pelaaja ohjaa päähenkilöä nuolinäppäimillä. Vuorovaikutus esineiden ja NPC-pelaajien välillä tapahtuu numeronäppäimillä. Tämän lisäksi pelaaja voi tarkistaa tavaraluettelonsa sisällön i-näppäimellä. ESC-näppäin pysäyttää pelin ja tuo näkyviin Pause-valikon, josta voidaan palata takaisin peliin tai päävalikkoon.

Pelihahmo liikkuu TMX-karttojen päälle ohjelmoidun kaksiulotteisen ruudukon puitteissa ruutu kerrallaan x- ja y-koordinaatiston mukaisesti (ks. kuva 4). Vuorovaikuttaakseen kartalla näkyviin esineisiin tai NPC-hahmoihin pelaajan on liikutettava pelihahmo ensin niiden luokse. Pelin ongelmienratkaisut liittyvät aina vuorovaikutukseen esineiden tai NPC-hahmojen kanssa.

Kuva 4. Kuvakaappaus Rainyday-pelistä. Pelissä käytetään ns. Top-down-näkymää, eli kamera kuvaa pelin tapahtumia suoraan ylhäältä päin. Mahdolliset toiminnot näkyvät numeroituina kuvan vasemmassa alalaidassa.

Pelin päähenkilönä ja pelaajahahmona toimii John Rainy, alkoholisoitunut ja maailmaan väsynyt New Orleans Police Departmentin etsivä, jonka tarkoituksena on selvittää outojen lapsikatoamisten sarja lohduttomassa ja sateisessa New Orleansissa.

3.2 Käytetyt tekniikat ja ohjelmistot

Pelin ohjelmointikieleksi valittiin Java sen tuettavuuden, helppokäyttöisyyden sekä dynaamisuuden vuoksi. Toinen mahdollinen ohjelmointikielivaihtoehto oli projektin suunnitteluvaiheessa C++, mutta ajatuksesta luovuttiin, koska sille ei löydetty sopivaa sovelluskehystä peliohjelmoinnille, eikä se kielenä ollut kehittäjälle niin tuttu kuin Java.

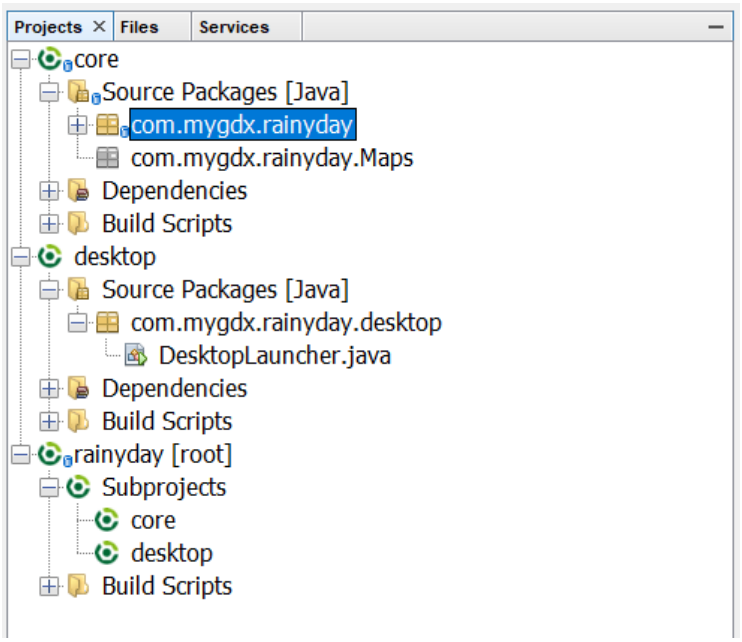
3.2.1 libGDX

Kun ohjelmointikieli oli valittu, alettiin tarkastelemaan vaihtoehtoja, joilla pelisovellus olisi mielekkäintä toteuttaa. Suosituimmiksi sovelluskehyksiksi peliohjelmointiin Javalla osoittautuivat jMonkeyEngine sekä libGDX. Tässä vaiheessa oli selvää, että pelistä tulee vanhanaikainen, kaksiulotteinen seikkailupeli, joten jMonkeyEnginen karsiutui vaihtoehdoista sen painottuessa lähinnä kolmiulotteisten pelien luomiseen.

libGDX on alustariippumaton, avoimen lähdekoodin, ilmainen Java-sovelluskehys, jolla voidaan luoda peliohjelmistoja Windowsille, Macille, Linuxille, Androidille, iOSille tai selaimille HTML5-muodossa yhdestä koodipohjasta.

libGDX vaatii toimiakseen Java Development Kit -version 8 tai uudemman sekä Netbeansin, Eclipsen tai Android Studion riippuen siitä, minkälaisen pelisovelluksen käyttäjä haluaa luoda. [5.]

Kehitysympäristöksi tässä työssä valittiin Netbeans 8.2, koska kehittäjä arvottaa sen korkeammalle kuin Eclipsen, ominaisuuksien sekä käyttömukavuuden takia.



Kuva 5. Rainyday-projektin rakenne NetBeansissa. Aliprojekteja tässä tapauksessa ovat ainoastaan desktop- ja core-projektit, joka sisältävät sovelluksen luokat.

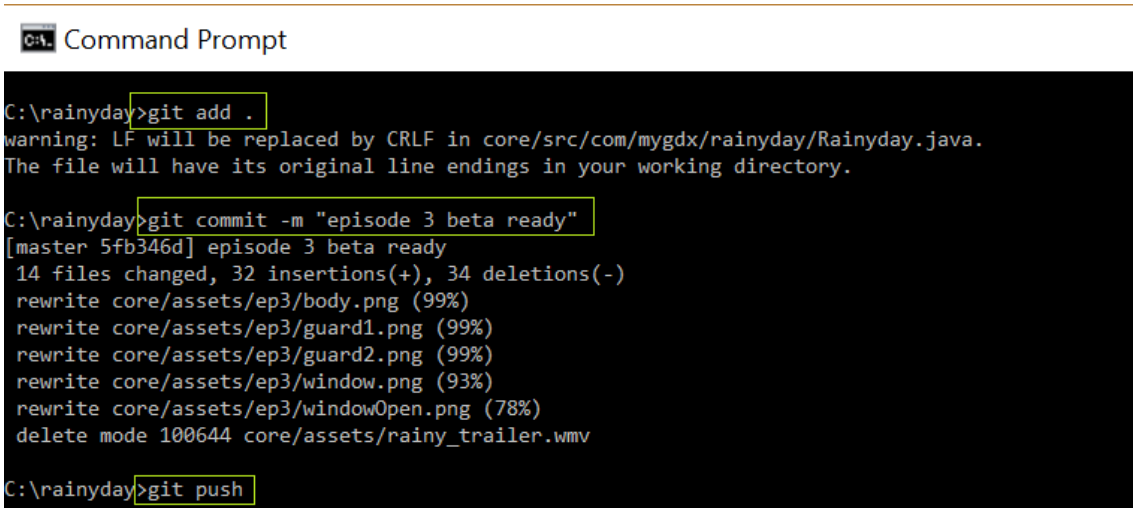
libGDX Project Generator on työkalu, jolla luodaan annettujen lähtötietojen perusteella projektirunko (ks. kuva 5). Koska libGDX on alustariippumaton, generaattorilla määritellään ennen projektin luomista ne alustat, joille projekti tullaan toteuttamaan sekä halutut lisäosat. Tämän lisäksi nimetään projekti, pakkaus, pääluokka sekä osoitetaan kohdehakemisto sekä Android SDK:n sijainti (ks. kuva 6). Kun Generator ajetaan, se luo näiden lähtötietojen perusteella libGDX gradle -projektin tietyllä ulkoasulla, tiedosto- ja kansiorakenteellaan. [6.]

Kuva 6. libGDX Project Generator. Projektin runko rakentuu generaattoriin syötettyjen lähtötietojen perusteella. Jokaisesta valitusta alustasta rakentuu oma aliprojekti.

3.2.2 Git

Pelisovellusta kehitettiin usealla eri tietokoneella, joten oli loogista järjestää koodille versionhallinta ja säilytyspaikka, johon pääsisi käsiksi eri päätteiltä. Versionhallinnalla tarkoitetaan ohjelmistotuotannossa menetelmiä, joilla pidetään muistissa tiedostojen versioita sellaisina, kuin ne ovat olleet eri kehitysvaiheissa. Täten voidaan jälkikäteen kertoa, miten koodi ja tiedostot ovat syntyneet, kuka niitä on ollut tekemässä, missä, millä ja miten. Versionhallintaa pidetään yhtenä modernin ohjelmoinnin tärkeimmistä työkaluista, sillä se mahdollistaa hajautetun ohjelmoinnin, varmuuskopioinnin sekä palaamisen edellisiin (toimiviin) versioihin ohjelmiston kehittämisvaiheessa.

Git on vuonna 2005 Linus Torvaldsin kehittämä avoimen lähdekoodin versionhallintaohjelmisto, jonka tarkoituksena oli mahdollistaa hajautettu työskentely estämään datan häviäminen ja varmistamaan tiedon yhtenäisyys. Git tukee automaattisesti suosittuja ohjelmistokehitysympäristöjä, mutta sen ominaispiirteisiin kuuluu Linux-työkaluille tyypillinen komentoriviltä käytettävä hallinta (ks. kuva 7).



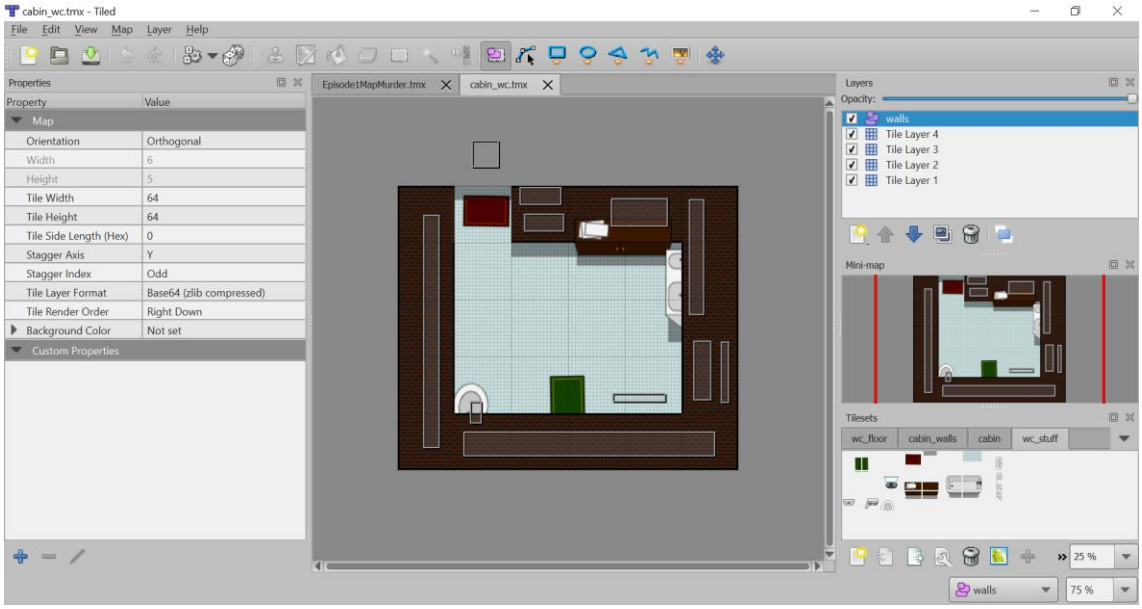
Kuva 7. Git:n peruskomennot ohjelmistovarastoon lisättäessä.

Insinöörityön versionhallintajärjestelmäksi valittiin Github. Se on maailman suurin lähdekoodin verkkopalvelu [7] ja se tukee projektissa käytettyä kehitysympäristöä Netbeansissa. Githubilla on tarjolla erilaisia jäsenyysvaihtoehtoja kehittäjille, se tarjoaa ilmaiseksi julkisia ohjelmavarastoja ja seitsemän dollarin kuukausihintaan hintaan loputtoman määrän yksityisiä varastoja.

3.2.3 Tiled

Rainydayn kartat luotiin Tiled-karttaeditorilla. Tiledillä voidaan luoda ruudukkopohjaisia karttoja, jotka tallennetaan yleensä TMX-muodossa. TMX-tiedostomuoto on yleistynyt tapa kuvailla ruudukkopohjainen kartta. Karttatiedosto sisältää tiedon mm. seuraavista ominaisuuksista: ruutujen koosta, kartan koosta, objekteista, kerroksista sekä mahdollisista mukautetuista ominaisuuksista. [8.]

Pelissä käytettiin erikokoisia, 64x64 ruuduista koostuvia karttoja, joissa jokaiseen ruutuun joko voi, tai ei voi siirtyä, riippuen siitä, onko ruudussa jokin este (ks. kuva 8). Jokaiseen karttaan lisättiin graafisten ruutukerroksien lisäksi ylimmäksi walls-esinekerros, johon merkittiin Tiledillä neliskanttisia esineitä niihin ruutuihin, joissa liikkuminen pelissä oli tarkoitus estää. Pelisovelluksen koodissa on erillinen metodi, joka tarkistaa, onko pelaajan viereisissä ruuduissa Tiled-objekteja, eli seiniä, ja pystyykö näihin ruutuihin siirtymään.



Kuva 8. Tiled-karttaeditorin näkymä. 64x64 ruuduista koostuvan kartan pituus on 6 ruutua ja korkeus 5 ruutua. Karttan walls-kerrokseen merkityt suorakaiteet merkkaavat alueita, joihin pelissä ei voi kulkea.

TMX-kartat tallennettiin libGDX-projektin määrittelemään assets-kansioon, joka sisältää kaikki peliin tuotavat ominaisuuserät. Ominaisuuserillä tarkoitetaan tässä tapauksessa kaikkia kolmannen osapuolen ohjelmilla luotuja grafiikka-, ääni- sekä karttatiedostoja.

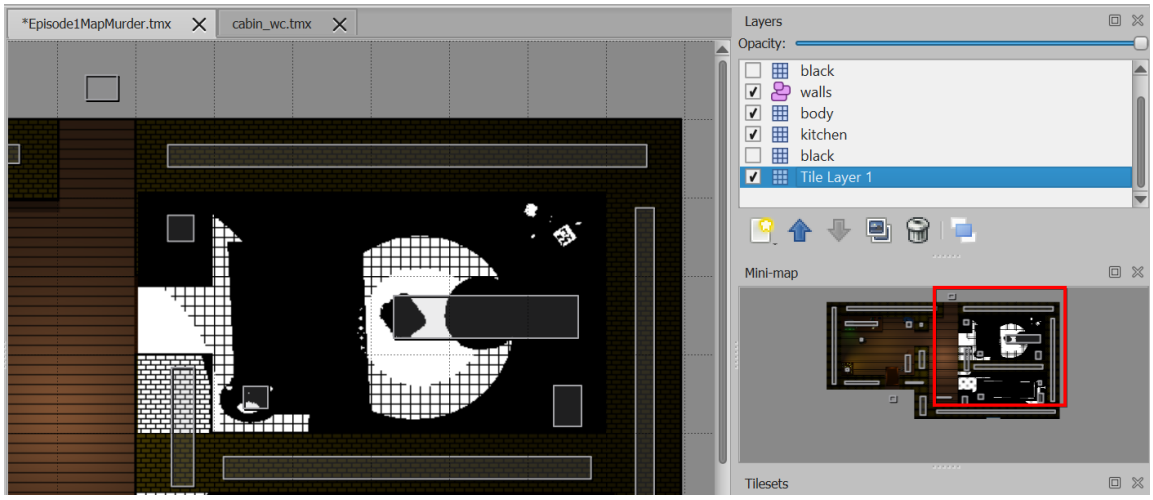
3.2.4 Paint.net, Photoshop ja Audacity

Internetin lukuisista ilmaisista ominaisuuseräpankeista huolimatta pelin graafinen ulkoasu päätettiin luoda omatoimisesti, jotta siitä varmasti tulisi kehittäjän tarkoituksen mukainen.

Paint.net on yksinkertainen ja kevyt, ilmainen kuvankäsittelyohjelma, johon on saatavilla lukematon määrä käyttäjien itse ohjelmoimia lisäosia. Käytännössä kaikki pelissä näkyvä grafiikka luotiin tällä ohjelmalla. Jokainen 64x64 pikselin ruudu piirrettiin Paint.net:llä, ja ne tallennettiin usein yhdessä suuremmassa kuvassa muiden ruutujen kanssa PNG-formaatissa pelisovelluksen assets-kansioon, Tiled-sovelluksen käyttöön. Kun kartan grafiikat oli piirretty ja koostettu yhteen Tiled:llä, lopputuloksesta tallennettiin kuva PNG-muodossa.

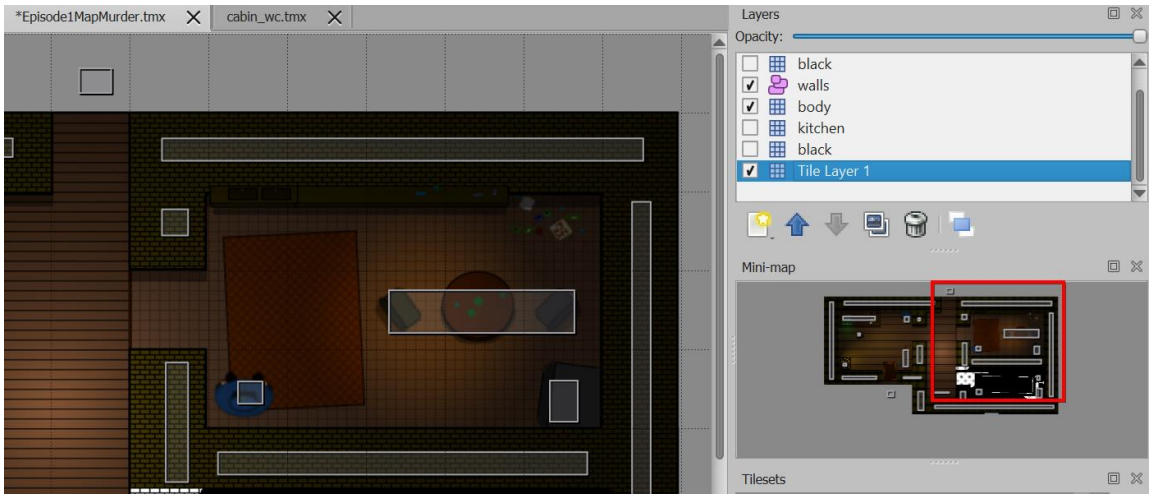
Tallennettuja kuvia käsiteltiin seuraavaksi Photoshopilla. Tässä vaiheessa PNG-kuvissa ei ollut vielä esineitä tai NPC-hahmoja, vaan ainoastaan ylhäältäpäin kuvattujen karttojen pohjat tiloineen. Nämä tilat valaistiin Photoshopin render-metodeilla sopiviksi, minkä jälkeen kuvaan sijoitettiin oikeille paikoilleen halutut esineet sekä paikallaan pysyvät hahmot Photoshopin eri kerroksiin kuin muu karttapohja. Esineet ja hahmot valaistiin samoilla render-metodin asetuksilla, jolloin ne näyttävät kuuluvan samaan tilaan karttapohjan tilojen kanssa. Valaistut esineet ja hahmot tallennettiin omiin tiedostoihinsa ja ne poistettiin karttapohjista. Valmiita valaistuja esineitä ja hahmoja lisättiin myöhemmin Tiledin karttapohjan eri kerroksiin tai ohjelmakoodin käytettäväksi Rectangle-esineiden tekstuureiksi. Samoilla asetuksilla valaistut esineet ja hahmot näyttävät istuvan saumattomasti valaistuun tilaan.

Valaistua karttapohjaa käytettiin usein Tiled-karttaeditorissa alimmaisena kerroksena. Tämän jälkeen karttaan lisättiin muita kerroksia, jotka esimerkiksi peittävät näkyvistä huoneita, joissa pelaaja ei ole vielä vieraillut (ks. kuva 9).



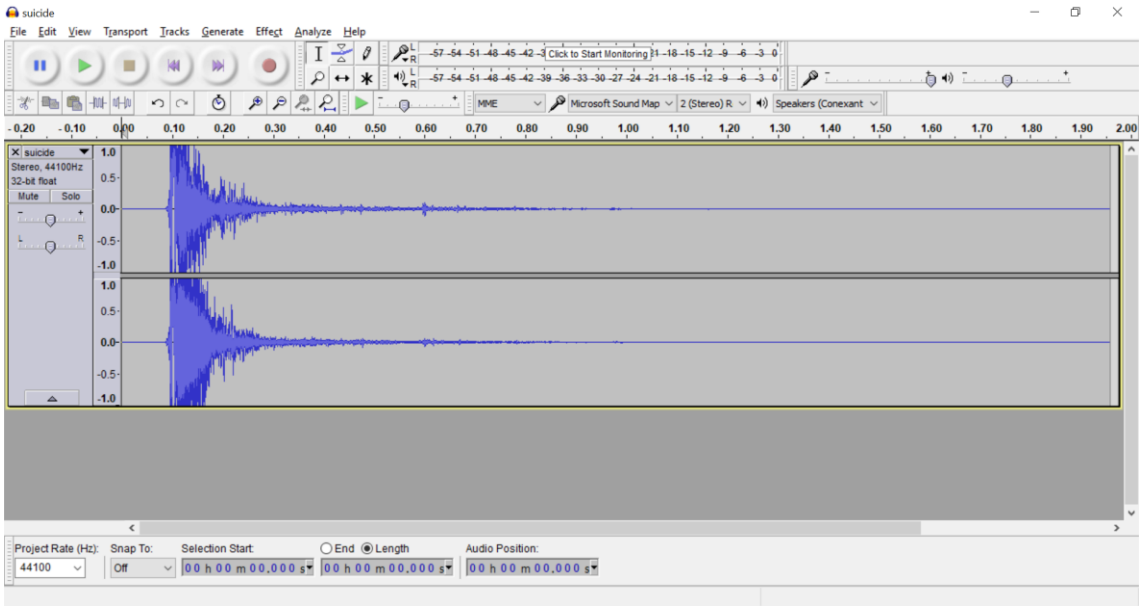
Kuva 9. Osaa karttaa peittää tummennettu alue, jossa pelaaja ei ole vielä vieraillut. Tiled-kartassa kitchen layer on päällä.

Kartassa olevat tilat, joissa pelaaja ei ole aikeisemmin vieraillut, näkyvät vaikeaselkoisina mustavalkoisina alueina. Nämä alueet lisättiin Tiled-karttaan omina kerroksinaan, joihin pelisovelluksen koodista pääsee käsiksi ja niiden asetuksiin voitiin vaikuttaa pelitapahtumien aikana reaaliajassa (ks. kuva 10).



Kuva 10. Kun pelaaja käy tummennetulla alueella, pelisovelluksen koodi muuttaa kartan asetuksia niin, että kitchen layer poistetaan käytöstä. Tummennettu alue katoaa ja pelaaja näkee piirretyn, Photoshopilla valaistun osan kartasta.

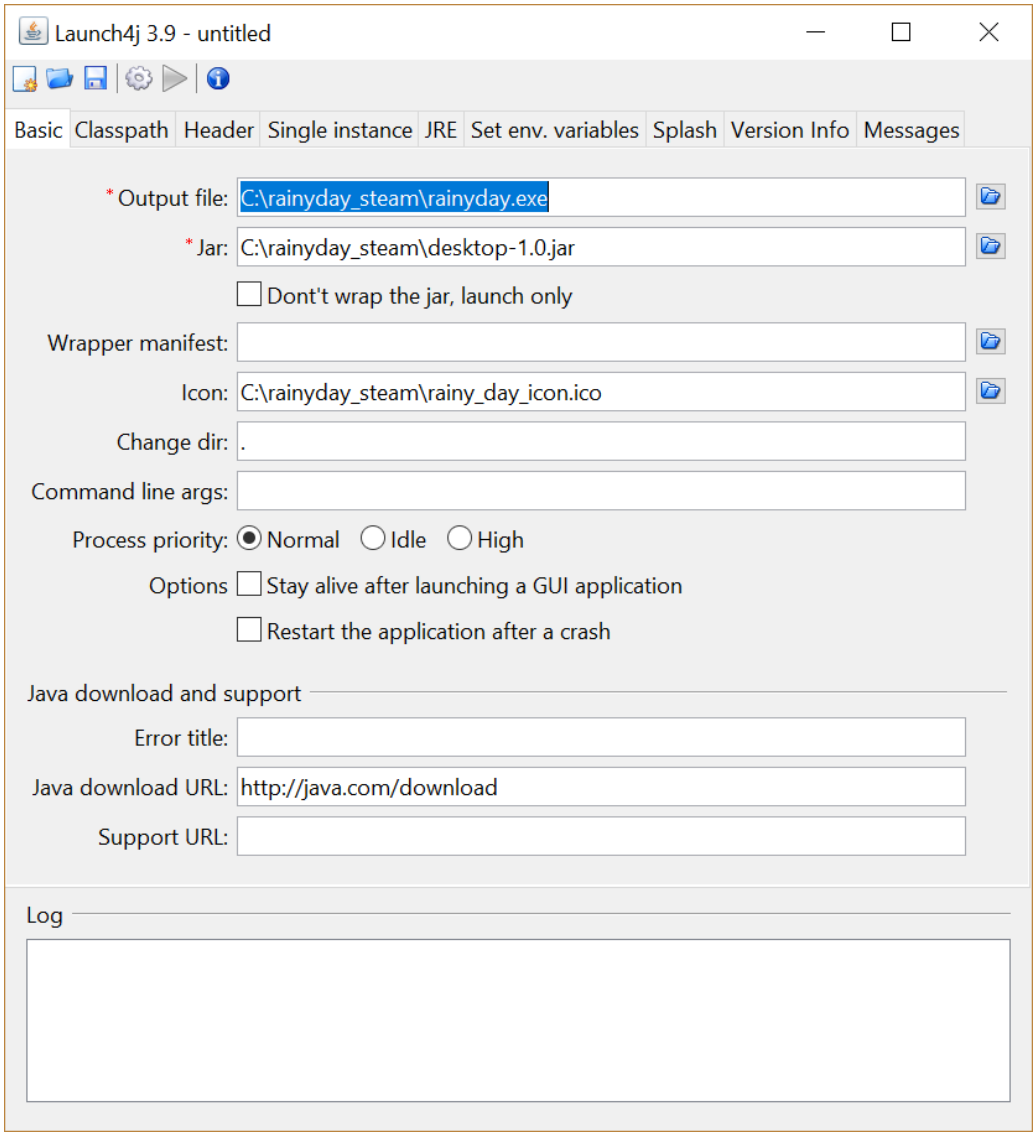
Audacity on ilmainen, avoimen lähdekoodin, alustavapaa äänitys- sekä äänen editointisovellus. Pelissä käytetyt ääniefektit ovat musiikkeja erikseen lukuunottamatta kaikki ilmaisista äänipankeista. Näitä ääniä editoitiin Audacitylla peliin sopiviksi (ks. kuva 11) ja tallennettiin projektin assets-kansioon pelisovelluksen käyttöön. Rainydayn musiikit luotiin yhteistyössä musiikkialan ammattilaisten kanssa yhteisissä nauhoitussessiossa.



Kuva 11. Äänieditori Audacityn päänäkymä.

3.2.5 Launch4j

Koska Netbeans libGDX -projekti on julkaisumuodossa JAR-formaatissa, se täytyi tässä tapauksessa muuttaa helpommin ajettavaan muotoon. Oli huomioitava, että läheskään kaikilla pelaajilla ei ole asennettuna JRE:ta, joka on edellytys pelisovelluksen ajamiseksi JAR-formaatissa. Ratkaisuksi valittiin Launch4j (ks. kuva 12), alustavapaa JAVA-suoritustiedoston käärijä, jolla JAR-paketista tehtiin natiivi Windows EXE -tiedosto, lisäämällä siihen itsenäinen JRE, jota sovelluksen JAR-tiedosto käyttää avautuessaan.



Kuva 12. Launch4j-sovelluksen päänäkymä. Pakollisiksi lähtöarvoiksi syötettiin haluttu tiedostonimi sekä alkuperäinen JAR-tiedoston osoite. Lisäksi määriteltiin mukaan pakattavan JRE:n osoite, koska pelin haluttiin toimivan myös niillä, joilla ei ole JRE:tä asennettuna.

Näin käytännössä koko pelisovellus toimii yhden suoritettavan EXE-tiedoston sisältä, viitaten tarvittaessa samassa kansiossa sijaitsevaan, pelin mukana tulevaan jre-kansion ajoympäristön tiedostoihin.

3.2.6 Steamworks SDK

Steamworks SDK on työkalupaketti, joka tarjoaa sovellukset pelin viemiseksi Steam-jakeluverkkoon. Valmis, suoritettava pelipaketti siirrettiin Steamworks SDK:n avulla jakeluverkkoon, johon pääsee käsiksi Steamin verkkosivujen kehittäjäpuolen selainsovelluksella. Steamworks SDK:ta käsitellään lisää Julkaisu Steamissa -osiossa.

3.3 Pelin tekninen toteutus

3.3.1 Oleelliset luokat ja metodit

Toiminnallisuuden kannalta tärkein luokka pelisovelluksessa on Rainyday.java-tiedoston sisältämä Rainyday-luokka. Tämä on projektin pääluokka. Siinä esitellään kaikki muut ohjelmaan liittyvät luokat ja se sisältää libGDX-sovelluksen kannalta oleelliset create-, render- sekä dispose-metodit.

Create-metodi (ks. kuva 13) ajetaan ensimmäisten toimintojen joukossa sovelluksen käynnistyessä, sen toimintatarkoitus on luoda pelin alkamiseksi tarvittavat luokat, muuttujat ja metodit sekä tuoda tarvittavat ominaisuuserät sekä aputiedostot pelisovelluksen käyttöön. Metodi myös tarkastaa, onko käyttäjällä yhteys Steamworks-rajapintaan ja lukee pelin asennuksen mukana luodusta rainy.progress-tiedostosta pelaajan edistymiseen liittyviä tietoja.

```
@Override
public void create() {

    startTime = System.currentTimeMillis();

    System.out.println(startTime);


    drownTX = new Texture(Gdx.files.internal("ep1/Player/drown.png"));

    jacketLeft = new Texture(Gdx.files.internal("guy_jacket_left_dark.png"));
    jacketLeftLeft = new Texture(Gdx.files.internal("guy_jacket_left_dark_left.png"));
    jacketLeftRight = new Texture(Gdx.files.internal("guy_jacket_left_dark_right.png"));

    jacketDown = new Texture(Gdx.files.internal("guy_jacket_dark.png"));
    jacketDownLeft = new Texture(Gdx.files.internal("guy_jacket_dark_left.png"));
    jacketDownRight = new Texture(Gdx.files.internal("guy_jacket_dark_right.png"));

    jacketRight = new Texture(Gdx.files.internal("guy_jacket_right_dark.png"));
    jacketRightLeft = new Texture(Gdx.files.internal("guy_jacket_right_dark_left.png"));
    jacketRightRight = new Texture(Gdx.files.internal("guy_jacket_right_dark_right.png"));
```

Kuva 13. Rainyday-pääluokan create-metodin alkuosa Netbeansissa. Luokan alussa esiteltyjä kuvatekstuureita tuodaan assets-kansiosta sovelluksen käytettäväksi.

Kaikkia pelin resursseja ei kuitenkaan luoda välittömästi sovelluksen käynnistyessä create-metodilla. Esimerkiksi kartat (ks. kuva 14) luodaan tarvittaessa pelin edetessä muistinhallinnan sekä latausaikojen optimoimiseksi.

```
public void createEpisode2Sewer() {
    episode2Sewer = new Episode2Sewer(this);
    tiledMap = new TmxMapLoader().load("ep2/Episode2Sewer.tmx");
    tiledMapRenderer = new OrthogonalTiledMapRenderer(tiledMap);
    TiledMapTileLayer layer0 = (TiledMapTileLayer) tiledMap.getLayers().get(0);
    Vector3 center = new Vector3(layer0.getWidth() * layer0.getTileWidth() / 2, layer0.getHeight() * layer0.getTileHeight() / 2, 0);
    camera.position.set(center);
    pelaaja.setLocation("sewer");
    pelaaja.setPosition(new Vector2(768, 128));
    pelaaja.setFacing(1);
    layer = tiledMap.getLayers().get("walls");
    objects = layer.getObjects();
    setGrid();
}
```

Kuva 14. Pääluokan createEpisode2Sewer-metodi, joka välittää episode2Sewer-luokalle itsensä, lataa assets-kansion karttatiedoston kerroksineen ja esineineen, asettaa kameran kuvaamaan haluttua karttaa, asettaa pelaajan kuvaan oikeaan paikkaan oikein päin sekä luo kartan päälle näkymättömän ruudukon liikkumista varten.

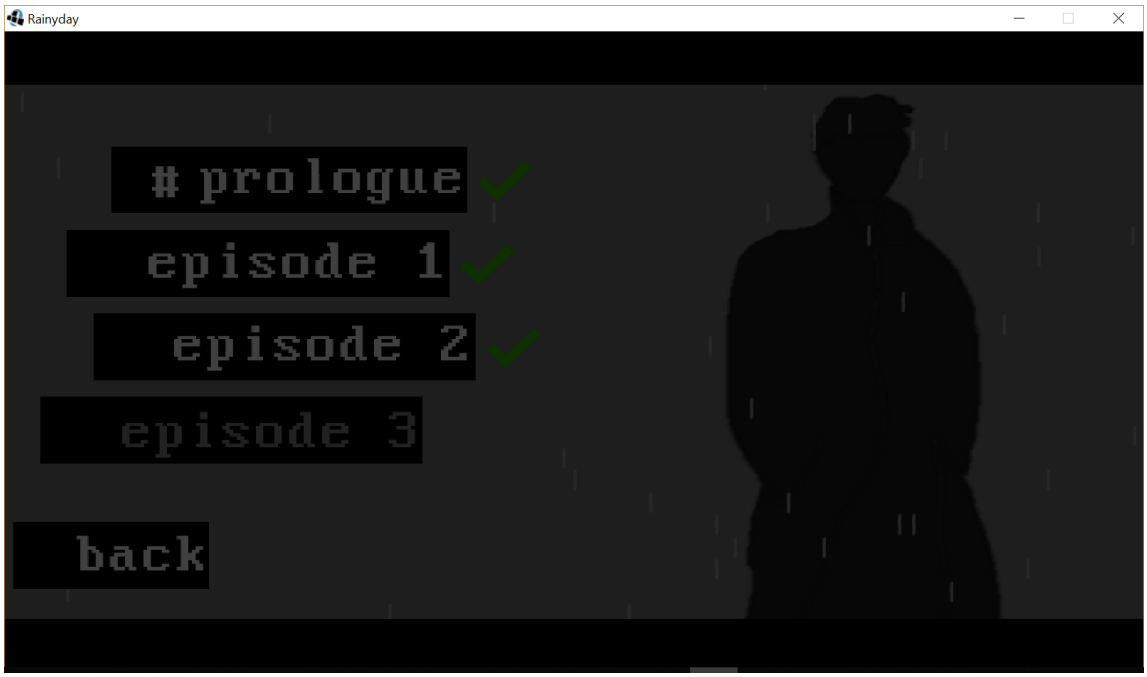
Toinen libGDX-projektin toiminnallisuuden kannalta tärkeä metodi on pääluokan render-metodi. Tämä on pelin pääsilmukka, joka käydään läpi kymmeniä tai satoja kertoja sekunnissa, pelikoneen suorittimesta riippuen. Tämä metodi sisältää kaikki interaktiot,

esimerkiksi ajastimet, näppäimistön käyttäjäsyötteet, kameran ja pelinäkymän päivityksen sekä grafiikan piirtämisen. Pelinäkymä päivittyy silmukan jokaisella kierroksella, minkä lisäksi silmukassa ikään kuin määritellään, mitä milloinkin tapahtuu.

Render-metodin alkuosassa kutsutaan Spritebatch-luokan batch-ilmentymän metodia begin ja loppuosassa end-metodia. Käytännössä kaikki karttojen päällä näkyvä grafiikka piirretään tähän batch-nimiseen tuotantoerään siten, että tekstuurien tai Sprite-luokan ilmentymät kutsuvat draw-metodiaan jossain batchin alkamisen ja loppumisen välillä, muttei koskaan sen ulkopuolella. Samalla määritellään, missä järjestyksessä grafiikat piirretään; aiemmin draw-metodiansa kutsuvat grafiikat piirretään pohjimmaiseksi.

Pelin jokaisen näkymän pohjalla on TMX-kartta ja jokaista karttaa vastaa myös kuvaavasti nimetty java-luokka. Jokaisessa luokassa on draw-metodi, jota kutsutaan Rainyday-pääluokan render-metodissa aina sen mukaan, mitä Player-luokan ilmentymä pelaajan getLocation-metodi palauttaa. Jos esimerkiksi pelaajahahmo sijaitsee vankila-kartassa, on pelaajan sijainniksi asetettu setLocation-metodilla String-muuttujan arvo ”prison”. Pääluokan render-metodissa on if-else-lohko, joka tarkastaa, onko pelaaja-ilmentymän getLocation-metodin palauttama arvo ”prison”. Jos on, suoritetaan Episode2Prison.java-luokan draw-metodia jatkuvasti render-silmukassa, jolloin sovellus käyttää tuon luokan draw-metodin sisäisiä toiminnallisuuksia.

Ensimmäinen näkymä, mikä pelissä tulee vastaan, on päävalikko, Main Menu (ks. kuva 15). Päävalikko luodaan create_MainMenu-metodilla. Teknisesti valikkonäkymä perustuu samanlaiseen TMX-karttaan kuin mikä tahansa muukin pelinäkymä. MainMenu-luokan draw-metodi ottaa vastaan käyttäjäsyötteitä nuolinäppäimistöllä liikuttaen valitsinkursoria valikko-näkymän eri kohteissa.



Kuva 15. Päävalikkonäkymän episodinvalintaruutu.

Create-metodissa luotu kameraluokan ilmentymä päivittää pelinäkymän joka kerta, kun silmukka käydään lävitse. Kamera saa pelin grafiikat piirtymään ruudulle, ja joka kierroksella uudelleen päivittyvä näkymä saa esimerkiksi sijaintia vaihtavat kappaleet liikkumaan kuvassa.

Dispose-metodi on kolmas libGDX-projektin oleellisista metodeista. Tätä kutsutaan, kun ohjelma sammutetaan, ja se poistaa kaikki create-metodilla luodut resurssit. Sitä voidaan myös käyttää pelisovelluksen ajon aikana vapauttamaan muistia ja resursseja, joita ei enää tarvita.

Toinen hyvin tärkeä luokka on sovelluksen Player-luokka. Create-metodissa luotu Player-luokan ilmentymä, pelaaja, perii Rectangle-luokan ominaisuudet. Näin pelaajailmentymä saa käyttöönsä Rectangle-luokan metodit, tärkeimpinä sijaintiin liittyvät setPosition- ja getPosition-metodit. Lisäksi pelaajailmentymällä on ArrayList, inventory, joka pitää kirjaa pelaajan omistamista esineistä, say-metodi, joka tulostaa pelinäkymään pelaajahahmon yläpuolelle tekstiä sekä setLocation-metodi, joka ottaa

vastaan String-syötteen ja jonka avulla pääohjelman render-silmukassa voidaan toimia sen mukaan, missä kartassa pelaajahahmo milloinkin sijaitsee.

Pelissä on useita aikaan sidottuja tapahtumia. Esimerkiksi putoavat sadepisarat, baarimikon edestakainen kävelyliike tai pelihahmojen välinen keskustelu tapahtuvat aina jonkun tietyn ajan kuluessa. Näiden toteuttamisen helpottamiseksi luotiin yksinkertainen Timer-luokka, jonka ilmentymiä käytetään pääluokan render-metodissa (ks. kuva 16).

```
public class Timer {

    Rainyday rainy;
    float aika;
    boolean on = false;

    public Timer(Rainyday rainy) {
        this.rainy = rainy;
    }

    public Timer() {

    }

    public void start() {
        on = true;
    }

    public void stop() {
        aika = 0;
        on = false;
    }

    public void run() {
        if (on) {
            aika += Gdx.graphics.getDeltaTime();
        }

    }
}
```

Kuva 16. Timer-luokka. Luokan ilmentymiä käytetään Rainyday-pääluokan render-metodissa. libGdx:n Graphics-rajapinnan getDeltaTime-metodi palauttaa kuluneen ajan nykyisen ja viimeisen kuvan välillä float-muodossa.

Timer-luokan ilmentymän run-metodia kutsutaan jatkuvasti läpikäytävässä pelin pääsilmukassa eli render-silmukassa. Pelinäkymä päivittyy kerran jokaisella silmukan kierroksella ja libGDX:n Graphics-rajapinnan getDeltaTime-metodi palauttaa kuluneen ajan millisekunneissa nykyisen näkymän kuvan sekä sitä edeltäneen kuvan välillä. Timerin start-metodi käynnistää ajanoton muuttamalla Timer-luokan ilmentymän ”on”-nimisen boolean-muuttujan arvon todeksi, jolloin getDeltaTime-metodi alkaa kasvattamaan ”aika” float-muuttujan arvoa jokaisella render-silmukan kierroksella. Pääohjelman create-osuudessa luodaan useita eri Timer-luokan ilmentymiä, joiden run-metodeita kutsutaan jatkuvasti render-silmukassa.

Esimerkiksi Player-luokan ilmentymällä on playerSay-metodi, joka saa parametrikseen String-muuttujan, joka ilmaisee, mitä pelaaja sanoo, tulostaen sanottavan tekstin näytölle pelaajahahmon yläpuolelle. Kun playerSay-metodia parametreineen kutsutaan, aktivoituu sayTimer-niminen Timer-luokan ilmentymä, joka alkaa kasvattaa tietyn float-muuttujan arvoa yhdellä joka sekunti edellisen kappaleen selityksen mukaisesti. Render-silmukassa piirretään näytölle playerSay-metodin määrittämää String-muuttujan arvoa niin kauan, kun sayTimer-metodin kasvattaman float-muuttujan arvo on yli 0, mutta alle 5. Kun arvo on yli 5, kutsutaan sayTimerin stop-metodia, joka pysäyttää ajanoton sekä nollaa kerrytettävän float-muuttujan. Näin ollen pelaajahahmo ”sanoo” pelissä viiden sekunnin ajan playerSay-metodilla määritellyn sanan tai lauseen.

3.3.2 Ruudukko ja liikkuminen

Peli koostuu pääosin 64x64-kokoisista ruuduista, jotka muodostavat ruudukon eli gridin. Pelaajahahmoa ympäröi näkymätön neljän ruudun ruudukko, joka tarkkailee ympäristön esineitä ja seiniä (ks. kuva 17). Koska pelissä voidaan liikkua ylös, alas, vasemmalle sekä oikealle, jokaisessa näistä suunnista on oltava oma ruutunsa ja tämän ruudukon on liikuttava pelaajahahmon mukana.

```
public void setGrid() {
    gridUp.setPosition(pelaaja.getX(), (pelaaja.getY() + 64));
    gridRight.setPosition((pelaaja.getX() + 64), (pelaaja.getY()));
    gridDown.setPosition(pelaaja.getX(), (pelaaja.getY() - 64));
    gridLeft.setPosition((pelaaja.getX() - 64), (pelaaja.getY()));
}
```

Kuva 17. setGrid-metodi, jota kutsutaan aina uuden kartan luomisvaiheessa. Metodi asettaa pelaajan ympärille näkymättömän ruudukon, jonka sijainti riippuu pelaajahahmon omista koordinaateista.

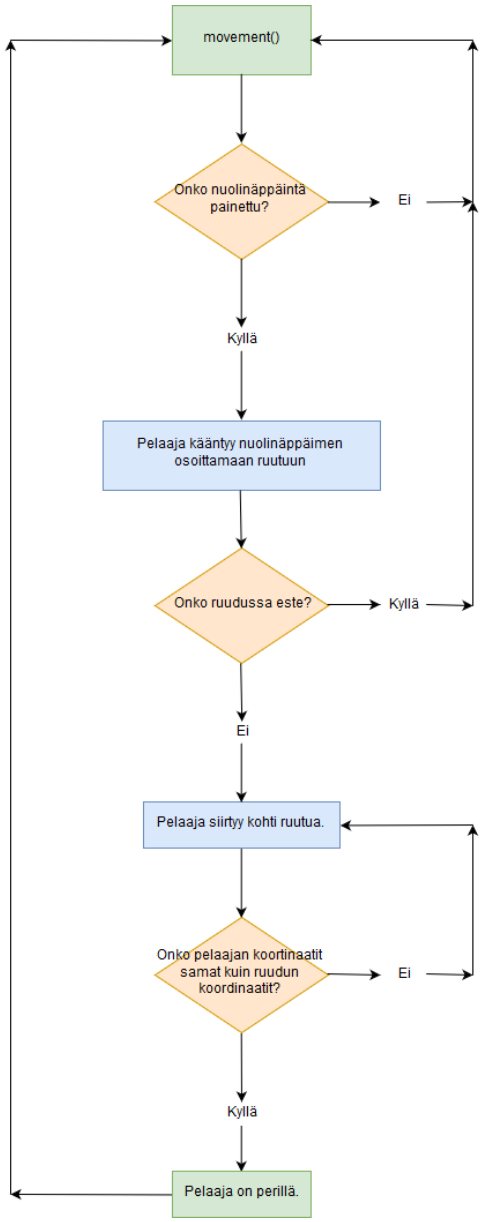
Pelissä liikkuminen on toteutettu render-silmukassa kierroksittain läpikäytävällä movement-metodilla. Metodissa tarkastellaan aluksi, onko käyttäjä painanut jotain nuolinäppäimistä. Jos on, nuolinäppäintä vastaava arvo tallennetaan keyDown-integer-muuttujaan. Tämän jälkeen boolean-muuttujilla tarkastetaan, onko pelitilanne sellainen, että pelaaja ylipäätään voi liikkua (ks. kuva 18).

```
//PELAAJAN LIIKE
if (keyDown == (Input.Keys.LEFT) && move == true && moveLeft == true) {
    passLeft = true;
    pelaaja.setFacing(1);
    for (RectangleMapObject rectangleObject : objects.getByType(RectangleMapObject.class)) {
        Rectangle rectangle = rectangleObject.getRectangle();
        if (rectangle.overlaps(gridLeft)) {
            passLeft = false;
        }

    }
    if (passLeft == true) {
        active = false;

        if (pelaaja.x > gridLeft.x) {
            pelaaja.x -= 200 * Gdx.graphics.getDeltaTime();
            if (pelaaja.x < gridLeft.x) {
                pelaaja.x = gridLeft.x;
                gridLeft.x = gridLeft.x - 64;
                gridRight.x = gridRight.x - 64;
                gridUp.x = gridUp.x - 64;
                gridDown.x = gridDown.x - 64;
                passLeft = false;
                keyDown = 5;
                active = true;
            }

        }

    }
}
```

Kuva 18. Movement-metodin osuus, joka tarkkailee, onko käyttäjä painanut nuolinäppäimistön vasenta painiketta.

Oletetaan, että pelaaja haluaa liikuttaa hahmoaan vasemmalle. Ensimmäisenä pelaajahahmo kääntyy ja for-silmukka käy lävitse kartalla olevat seinät. Jos pelaajan vasemmalla puolella olevassa ruudussa on seinä, passLeft-muuttuja saa arvon false, eikä pelaajahahmon sijainti muutu kääntymisestä huolimatta. Jos seinää ei ole, hahmo liikkuu vasemmalle niin kauan, kun pelaajan x-koordinaatti on suurempi kuin pelaajahahmon vasemmalle puolella olevan ruudun x-koordinaatti. Kun pelaajan x-koordinaatti on pienempi kuin ruudun x-koordinaatti, ruutua siirretään 64 pikseliä liikkeen suuntaan, eli vasemmalle. PassLeft-muuttuja asetetaan falseksi ja keyDown-muuttujaan asetetaan joku arvo, joka ei vaikuta ohjelmakoodiin tai pelin toimintaan mitenkään. Hahmon liike pysähtyy (ks. kuva 20).

Kuva 19.



Kuva 20. Vuokaaviossa havainnollistetaan pelaajan syötteen aiheuttamaa interaktiota movement-metodissa.

3.3.3 Vuorovaikuttaminen pelimaailman kanssa

Koko idea videopeleissä on, että pelaaja vaikuttaa pelin tapahtumiin käyttöliittymän avulla. Pelaajan näppäimistöllä, hiirellä tai ohjaimella antamat komennot aiheuttavat

pelissä jonkun toiminnon, mikä luo pelaamisesta interaktiivista: pelaaja vaikuttaa pelin kulkuun.

Vuorovaikuttaminen pelimaailman esineiden ja NPC:n kanssa tapahtuu yleensä liikuttamalla pelihahmo nuolinäppäimillä halutun kohteen luokse ja valitsemalla ohjelman ehdottama toiminto numeronäppäimillä. Karttaa, jossa pelaajahahmo sijaitsee, vastaavassa java-luokassa on jatkuvassa render-silmukassa suoritettavassa draw-metodissa ehdot eri koordinaatiopisteille, joissa pelaajahahmon täytyy sijaita, jotta toimintavaihtoehdot tulostuvat näkymään.

Koodissa voi olla esimerkiksi seuraavanlaiset ehdot:

- jos pelaajahahmo sijaitsee kotonaan
- jos pelaajahahmon x-koordinatti on a
- jos pelaajahahmon y-koordinaati on b
- jos pelaajahahmo on kääntyneenä oikealle.

Kun nämä kaikki ehdot toteutuvat, päästään haluttuun ohjelmalohkoon sisälle, tulostetaan näytölle numeronäppäimillä aktivoitavat toimintavaihtoehdot ja odotetaan käyttäjäsyötettä uusilla if-ehdoilla (ks. kuva 19).

```
if (rainy.pelaaja.x == 640 && rainy.pelaaja.y == 640 && rainy.pelaaja.getFacing() == 2) {
    rainy.font.draw(rainy.batch, "1. Look at the window.", 120, 140);
    if (keyDown == 1) {
        rainy.font.draw(rainy.batch, "The window is shut, intact and locked from the inside.", 120, 80);

    }
}
```

Kuva 21. Ohjelmalohkoon päästään sisälle pelaajahahmon seistessä tietyssä pisteessä, tiettyyn suuntaan kääntyneenä.

Pelaaja voi suorittaa halutessaan sovelluksen ehdottaman toiminnon. Kun joku if-ehdoista muuttuu epätodeksi, pelaajan liikuttaessa hahmoa tai kääntyessä, ehtolohkosta poistutaan, eikä sovellus enää tarjoa toimintavaihtoehtoja.

Toimintoja ovat esimerkiksi muiden hahmojen kanssa keskusteleminen, esineiden kerääminen, ovien avaaminen sekä jo kerättyjen esineiden käyttäminen. Oikeissa paikoissa suoritetut toiminnot kuljettavat pelitapahtumia eteenpäin sekä edistävät pelin sisäistä tarinankerrontaa.

3.3.4 Tallentaminen

Vaikka käyttäjä ei voi tallentaa pelitilannetta halutessaan, sovellus seuraa pelaajan edistymistä ja tallentaa tiedon suorituksista erilliseen progress-tiedostoon käyttäjän koneelle. Tämä oli tietoinen valinta, johon päädyttiin pelin suunnitteluvaiheessa. Pelin käyttöliittymästä haluttiin mahdollisimman yksinkertainen ja episodeista niin lyhyitä, että kesken pelin tallentamista ei tarvittaisi. Kun pelaaja läpäisee episodin, saveProgress-metodi kirjoittaa progress-tiedostoon läpäistyn episodin tunnuksen (ks. kuva 20), joka välitetään String-muuttujana.

```
public void saveProgress(String content) {
    try {
        FileWriter fw = new FileWriter(file, true);
        BufferedWriter bw = new BufferedWriter(fw);
        bw.newLine();
        bw.write(content);
        bw.close();
    } catch (IOException e) {

    }

}
```

Kuva 22. FileWriterlla toteutettu pelitilanteen tallennin.

Sovelluksen käynnistysvaihessa readProgress-metodi (ks. kuva 21) tarkastaa progress-tiedostosta, mitkä episodit pelaaja on läpäissyt ja muokkaa päävalikon näkymää sen

mukaan: läpäistyn episodin nimen eteen piirretään vihreä merkintä. Pelaaja voi kuitenkin vapaasti valita aloitusvalikosta minkä tahansa episodin, eikä peliä tarvitse pelata lineaarisessa järjestyksessä.

```
public boolean readProgress(String progress) {

    try {
        fileReader = new FileReader(file);

        bufferedReader = new BufferedReader(fileReader);

        while ((line = bufferedReader.readLine()) != null) {
            System.out.println(line);
            if (progress.equals(line)) {
                System.out.println("löytyy");
                return true;
            } else {
                System.out.println("ei löydy");
            }
        }

        bufferedReader.close();
    } catch (FileNotFoundException ex) {
        System.out.println(
                "Unable to open file '"
                + file + "'");
    } catch (IOException ex) {
        System.out.println(
                "Error reading file '"
                + file + "'");
    }
    return false;
}
```

Kuva 23. readProgress-metodi, joka tarkistaa tiedostosta, mitkä episodit pelaaja on läpäissyt.

3.3.5 Steam-rajapinta

Steam tarjoaa kehittäjille ilmaiseksi pääsyn rajapintoihinsa, muttei mitään virallista Java-pakkausta toimintoineen. Pelisovellus on yhteydessä Steamiin Code-disasterin kehittämällä käärimellä, joka on tarkoitettu Java- ja C++-sovelluksien yhdistämiseen Steamworksiin rajapinnan avulla. Sovellus tarkistaa Render-metodissa joka kierroksella, onko käyttäjän Steam-sovellus auki, minkä jälkeen se ajaa runCallbacks-metodin, joka päivittää rajapintaan tehdyt muutokset Steamiin. Edistyminen pelissä avaa käyttäjälle

Steam-sovelluksessa pelikohtaisia saavutuksia, jotka aktivoidaan Steamworksin rajapinnan avulla (ks. kuva 22).

```
if ((rainy.stats.isAchieved("EP2", false)) == false) {
    System.out.println("Achievement unlocked");
    rainy.stats.setAchievement("EP2");
    rainy.stats.storeStats();
}
```

Kuva 24. Episodi 2:n läpäisyn jälkeen ajettava lohko. Koodissa tarkistetaan, onko pelaajalla EP2-niminen Steam-saavutus. Jos sellaista ei ole, pelaaja ansaitsee sen. Render-metodin sisällä jatkuvasti toimiva runCallbacks-metodi päivittää muutokset Steamiin.

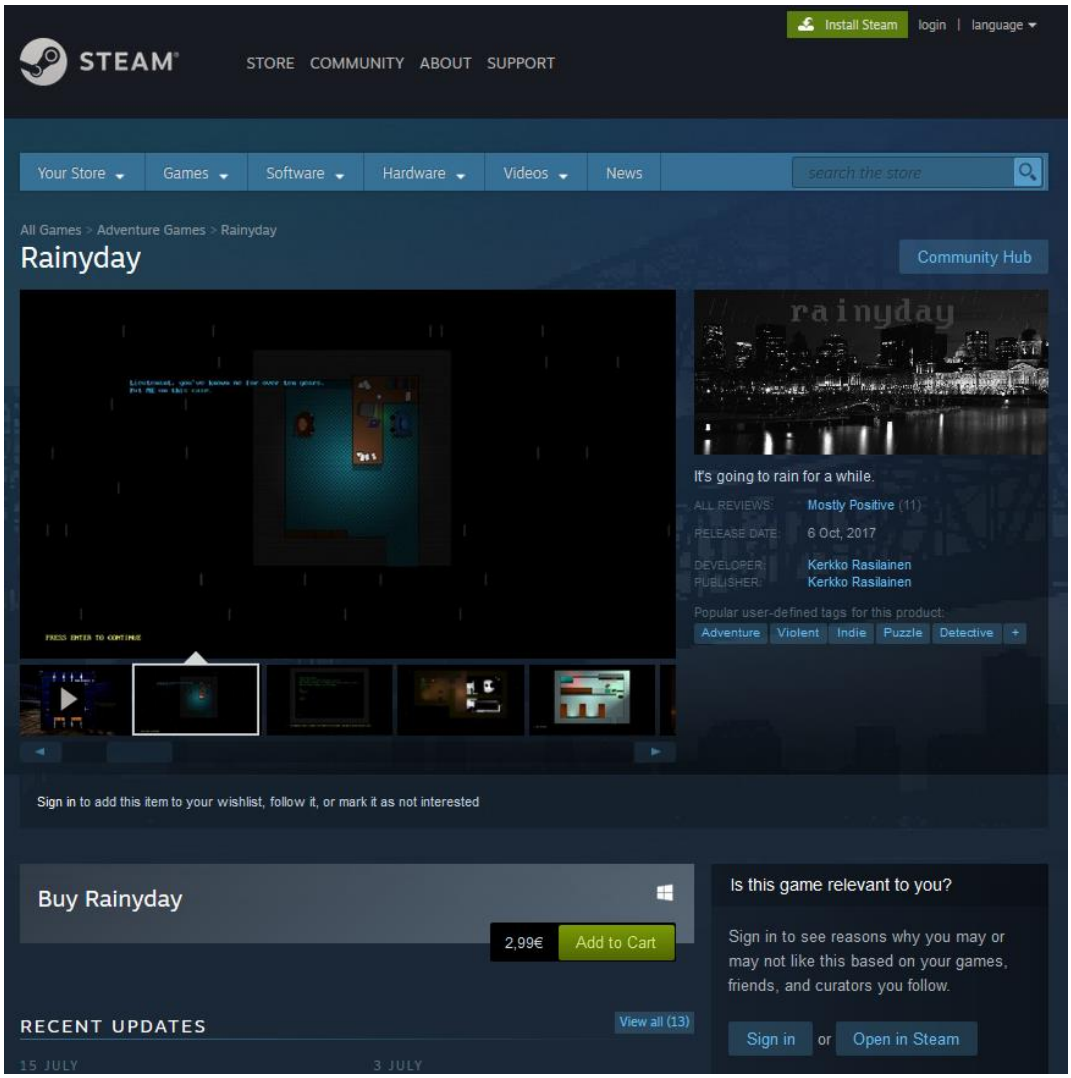
4 Julkaisu Steamissa

Suosituimpina julkaisualustoina indie-peleille voidaan pitää kolmea seuraavaa: Steam, Greenmangaming sekä GoG eli Good Old Games. Steam on näistä kolmesta huomattavasti suurin pitäen yksin hallusaan yli 70 % tietokonepelien verkkomarkkinoista. [9.] Greenmangaming on vuonna 2010 avattu verkkokauppa peleille, mikä käsittää nykyään kaupan lisäksi arvosteluja, uutisia, tilastoja sekä julkaisupalveluita. Greenmangamingilla ei julkaisijan etsimisvaiheessa ollut mitään, millä se voisi kilpailla Steamin kanssa. GoG on vuonna 2008 avattu erityisesti vanhojen ja vanhoillisten tietokonepelien julkaisijana ja uudelleenjulkaisijana profiloitunut kauppa sekä julkaisija. Rainydayn tyylin huomioiden GoG oli potentiaalinen vaihtoehto julkaisijaksi, mutta karsiutui kuitenkin pois Steamin ylivoimaisen tavoittavuuden, kehittäjätyökalujen sekä äärimmäisen monipuolisten kehittäjäpalveluiden ja -rajapintojen vuoksi.

Steam on vuonna 2003 Valven julkaisema, Windowsille suunniteltu pelijakelualusta, joka suunniteltiin alun perin tukemaan Valven omia pelejä. Ohjelman alkuperäinen tarkoitus oli toimia rajapintana Valven kehittäjien sekä pelaajien välillä, ja sitä käytettiin aluksi ainoastaan pelaajien asentamien pelien automaattiseen päivitykseen. Kaksi vuotta myöhemmin alustassa alettiin julkaista myös muiden kehittäjien sovelluksia.

Nykyään Steam tukee myös MAC- ja Linux-käyttöjärjestelmiä, pelaajat voivat jakaa tuottamaansa sisältöä, ja se sisältää pelien lisäksi myös hyötyohjelmia. Vuonna 2015 Steamilla oli yli 125 miljoonaa aktiivista käyttäjätiliä [10]. Steam on nykyään maailman suurin tietokonepelien jakeluverkko, ja se tekee pääasiallisesti tulosta ottamalla 30 % osuuden jokaisesta sen kautta myydystä sovelluksesta. Vuonna 2017 Steamin myynti ylitti 4,3 miljardin dollarin rajapyykin [11].

Jokaisella pelillä ja ohjelmistolla on oma sivunsa Steamissa. Sivulta näkyvät tuotetiedot, kuten tekijä, hinta, tyylilaji, vaatimukset sekä rajoitukset. Kauppasivulla on myös esiteltävä tuotetta kuvankaappauksin sekä vähintään yhdellä videolla (ks. kuva 23). Muut rekisteröityneet käyttäjät voivat arvioida tuotteita kauppasivuilla antamalla positiivisen tai negatiivisen palautteen ja kirjoittamalla lyhyen, tuotekohtaisen arvion.



Kuva 25. Rainyday:n Steam-kauppa sivu. Kehittäjä luo Steam Direct -rajapinnan kautta omalle sovellukselleen kauppasivun. Sivu sisältää esittelyvideoita, kuvia sekä tietoa pelistä/sovelluksesta ja sen ominaisuuksista.

4.1 Steam Direct

Steam Direct on Steamworksin jakeluohjelmaosuus, jonka avulla Rainyday lisättiin Steamin jakeluverkkoon. Sovelluksen lisääminen verkostoon maksaa aina 100 dollaria. Pelin lisäämisvaiheessa Steamin kauppaehdot täytyi hyväksyä, todistaa henkilöllisyys ja allekirjoittaa digitaalisesti sopimuspaperit, joista selviää Steamin liiketoimintamalli sekä kuinka paljon yritys tulee ottamaan jokaisen myydyn pelin hinnasta itselleen. Lisäksi oli olennaista selvittää lisääjän verotiedot sekä se, onko kehittäjänä yksityishenkilö vai

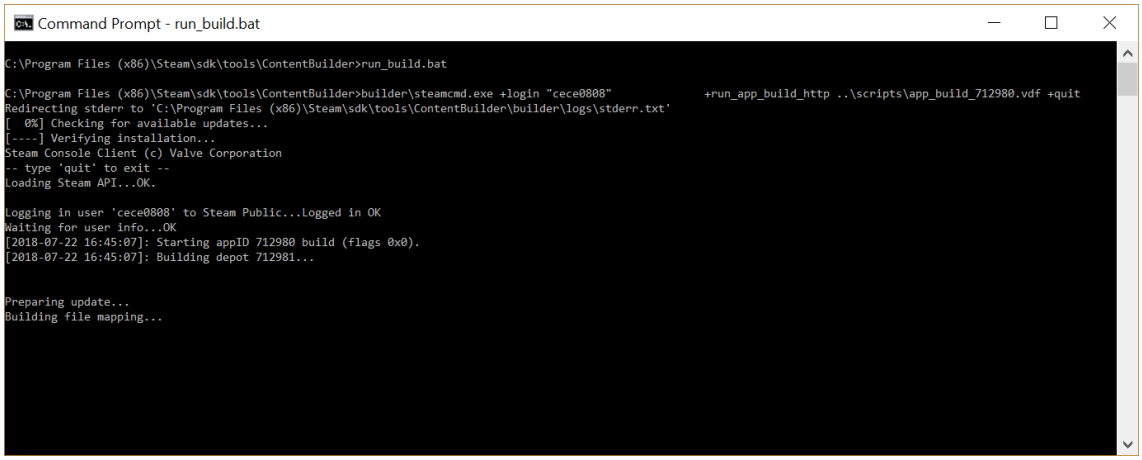
yritys. Ehtojen sekä sopimuksen hyväksymisen jälkeen Steamin edustajat todentavat lisääjän tiedot.

4.2 Steamworks

Steamworks on ohjelmistokehittäjille tarkoitettu ilmainen kokoelma työkaluja ja apuohjelmia. Steam Direct on Steamworksin osa, jolla kehittäjät voivat lisätä sovelluksiaan Steamiinn sekä hallinnoida niitä. Tämän lisäksi Steamworks tarjoaa rajapinnan, jolla sovelluksiin voidaan lisätä Steamin omia ominaisuuksia sekä palveluita, esimerkiksi huijauksen eston, moninpelien pelinhaun, tilastojen ylläpidon, pilvitallennuksen sekä tässä projektissa käytetty saavutusten päivittämisen.

Steamworksin App Admin on selaimella käytettävä verkkosivusto ja rajapinta, jolla voidaan hallinnoida pelin versioita, päivityksiä, kauppasivua, markkinointia sekä näyttää myyntiraportteja.

Steamworks SDK on Steamworksin ladattava työkalupaketti, joka sisältää ohjelmistot pelisovelluksen saattamiseksi Steamiin. Rainydayn lisääminen tapahtui käyttämällä SDK:n Content Builder -sovellusta, joka vie määritellyssä kansiossa sijaitsevan pelisovelluksen Steamworksiin App Adminin käsiteltäväksi (ks. kuva 24).



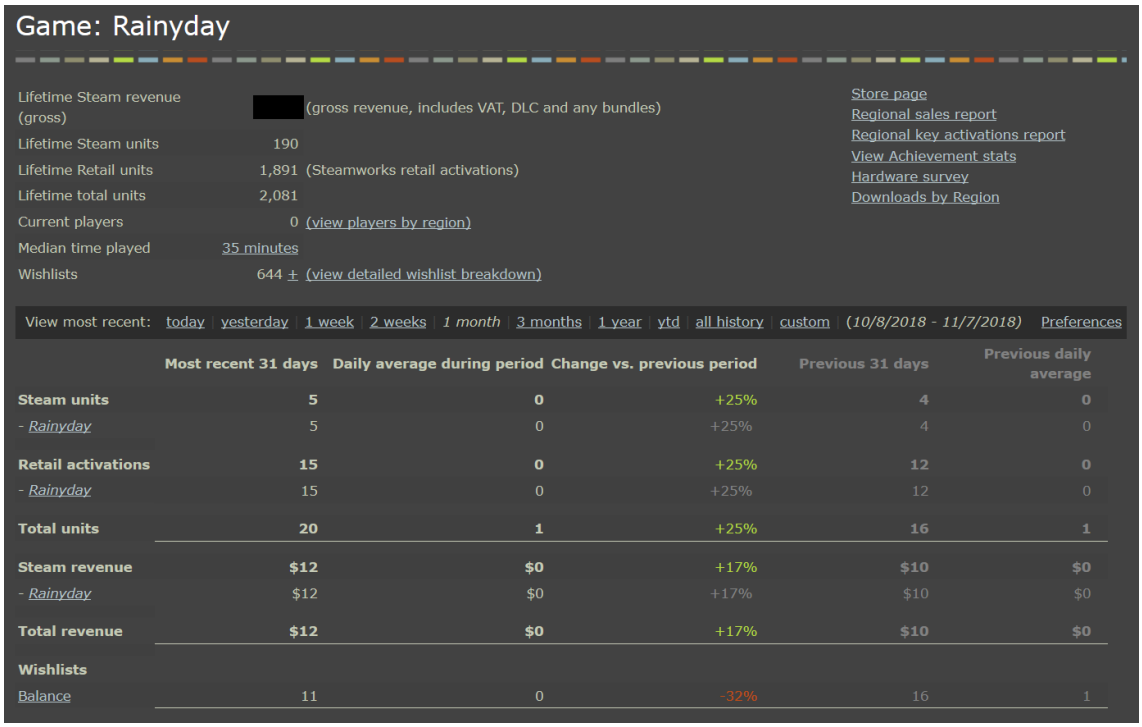
Kuva 26. Komentoriviltä ajettava SDK:n Content Builder vie valmiin pelisovelluksen Steamiin.

Content Builderin lähtöarvoiksi annetaan sovelluksen yksilöivä Steam-tunnus sekä rakenteen versionumero, joka on kehittäjän määrittelemä tunniste App Adminin käyttöön. Rakenteen versionumeroiden avulla kehittäjän on helppo pitää kirjaa sovellukseen lisätyistä päivityksistä, lisäosista sekä beta- ja julkaisuversioista.

4.3 Julkaisu ja jatkuva kehitys

Steamiin lisättävällä sovelluksella on aina oltava kauppasivu, jossa sovellus on kaikkien nähtävillä coming soon -tilassa vähintään kaksi viikkoa ennen kuin käyttäjät pystyvät ostamaan ja lataamaan sovelluksen. Rainyday-projektissa noudatettiin tätä kaavaa ja täsmälleen kaksi viikkoa kauppasivun luomisesta eteenpäin peli julkaistiin. Tässä vaiheessa peli sisälsi yhden pelattavan episodin, joka tutustutti pelaajat pelimekaniikkaan ja esitteli Rainydayn melankolisen maailman.

Steamworksin App Adminin avulla voitiin tarkkailla myytyjen sovellusten määrän (ks. kuva 25) lisäksi pelaajien edistymistä, kuinka kauan keskiverto pelaaja pelasi, kuinka pitkälle pelaajat pääsivät, mitä saavutuksia pelaajat ansaitsivat pelissä ja muuta sovelluksen käyttöön liittyvää tietoa.



Game: Rainyday

Lifetime Steam revenue (gross) [redacted] (gross revenue, includes VAT, DLC and any bundles)
Lifetime Steam units 190
Lifetime Retail units 1,891 (Steamworks retail activations)
Lifetime total units 2,081
Current players 0 (view players by region)
Median time played 35 minutes
Wishlists 644 ± (view detailed wishlist breakdown)

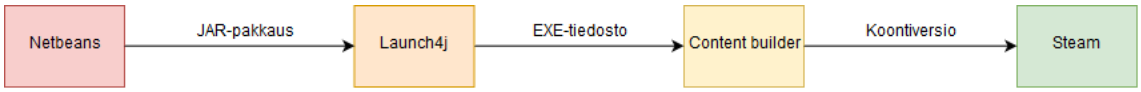
Store page
Regional sales report
Regional key activations report
View Achievement stats
Hardware survey
Downloads by Region

View most recent: today | yesterday | 1 week | 2 weeks | 1 month | 3 months | 1 year | ytd | all history | custom | (10/8/2018 - 11/7/2018) | Preferences

	Most recent 31 days	Daily average during period	Change vs. previous period	Previous 31 days	Previous daily average
Steam units	**5**	**0**	**+25%**	**4**	**0**
- *Rainyday*	5	0	+25%	4	0
Retail activations	**15**	**0**	**+25%**	**12**	**0**
- *Rainyday*	15	0	+25%	12	0
Total units	**20**	**1**	**+25%**	**16**	**1**
Steam revenue	**$12**	**$0**	**+17%**	**$10**	**$0**
- *Rainyday*	$12	$0	+17%	$10	$0
Total revenue	**$12**	**$0**	**+17%**	**$10**	**$0**
Wishlists					
Balance	11	0	-32%	16	1

Kuva 27. Steamworksin App Admin -verkkosovellus, jolla voidaan tarkastella pelaajien edistymistä sekä myyntiraportteja.

Noin kuukausi prologue-episodin julkaisusta peliin julkaistiin ensimmäinen pelattava laajennus, Episode 1: Gris-gris sekä korjauspäivitys paikkaamaan käyttäjäpalautteena saatujen tietojen perusteella esittelyepisodissa ilmenneitä bugeja. Samoin kuin esittelyepisodin kanssa uuden episodin sekä korjaavan päivityksen sisältävästä versiosta tehtiin jakelupaketti, joka muutettiin Launch4j:n avulla exe-tiedostoksi. Paketti ladattiin Steamworksin Content Builderilla Steamiin ja App Admin -rajapinnasta määriteltiin tuorein koontiversio Steam-verkossa jaettavaksi myyntiversioksi (ks. kaavio 2). Lopulta muutokset julkaistiin pysyviksi, jolloin kaikki pelin omistavat Steam-käyttäjät automaattisesti päivittivät pelisovelluksensa uusimpaan versioon. Kahden seuraavan kuukauden sisällä käyttäjille julkaistiin ilmaiseksi immersiota parantavat päivitykset, jotka toivat peliin yhden uuden kartan, korjasivat bugeja sekä lisäsivät kävelyanimaation pelaajahahmolle.



Kaavio 2. Sovelluksen saattaminen kehitysympäristöstä Steamin jakeluverkkoon.

Koska seuraavasta osasta haluttiin saada edellisiä episodeja teknisesti edistyneempi sekä pidempi, sen kehittämiseen kului aikaa. Episode 2: Kids julkaistiin päivityksen mukana lähes puoli vuotta edellisen osan julkaisusta. Tällä kertaa koontiversioon lisättiin uudistettu, realistinen sade-efekti, kaksi uutta Steam-saavutusta, uusia karttoja sekä pelissä ilmenneiden virheiden korjauksia.

5 Yhteenveto

Insinöörityössä pyrittiin saamaan valmis sovellus sekä valottamaan käsitystä siitä, minkälainen prosessi tietokonepelin ohjelmointi ja julkaisu nykyään on.

Työssä suunniteltiin ja toteutettiin seikkailupelisovellus, joka julkaistiin Steam-jakeluverkossa. Tämän lisäksi työssä tutustuttiin tietokonepelien historiaan sekä seikkailupeligenreen osana peliteollisuutta. Suunnitteluvaiheessa selvitettiin eri vaihtoehtoja, miten ja millä tietokonepeli nykypäivänä kannattaa kehittää. Tuotantovaiheessa perehdyttiin Paint.net-, Photoshop- sekä Audacity-työkaluihin, joilla luotiin pelisovelluksen grafiikkaerät sekä äänet. Kun peli oli saatu valmiiksi, selvitettiin, mikä olisi paras tapa julkaista sovellus sekä minkälainen paketti sovelluksesta kannattaa tehdä, jotta se olisi yhteensopiva mahdollisimman monen käyttäjän laitteiston kanssa.

Työn aiheena ollut peli ohjelmoitiin Javalla, libGDX-sovelluskehystä hyödyntämällä. Kehitystä tehtiin usealla eri päätteellä ja versionhallintatyökaluna käytettiin Githubia. Valmiista sovelluksesta tehtiin pakkaus Launch4j-ohjelmistolla, joka nitoo Java-sovelluksen mukaan oman ajoympäristön mahdollisimman laajan yhteesopivuuden takaamiseksi. Valmis sovellus julkaistiin maailman suurimmassa tietokonepelien jakeluverkossa, Steamissa.

Työ tarjoaa pelinkehityksestä kiinnostuneille tietoa peleistä, nykyaikaisesta peliohjelmoinnista, pelin toteutuksen eri vaiheista, ominaisuuserien luomiseen tarkoitetuista ohjelmista, Steam-jakeluverkoston ominaisuuksista ja toiminnasta sekä julkaisuprosessista.

Projekti onnistui asetettujen vaatimusten mukaisesti ja sovellukseen saatiin tuotua kaikki suunnitteluvaiheessa ideoidut ominaisuudet. Julkaisun jälkeen peliin on lisätty lukuisia lisäominaisuuksia ja episodeja, mitkä peliyhteisö on ottanut arvioiden mukaan hyvin vastaan.

Lähteet

1 Why Video Games Succeed Where The Movie And Music Industries Fail. 2013. Verkkoaineisto. Fast Company. < https://www.fastcompany.com/3021008/why-video-games-succeed-where-the-movie-and-music-industries-fail>. Luettu 17.10.2018.

2 Top 10 Greatest Video-Game Genres. 2018. Verkkoaineisto. The Top Tens. <https://www.thetoptens.com/video-game-genres/>. Luettu 18.10.2018.

3 A truly graphic adventure: the 25-year rise and fall of a beloved genre. 2011. Verkkoaineisto. Ars Tehcnica. <https://arstechnica.com/gaming/2011/01/history-of-graphic-adventures>. Luettu 20.7.2018.

4 A Brief History of Adventure Games. 2012. Verkkoaineisto. Giantbomb. <https://www.giantbomb.com/profile/gbrading/lists/a-brief-history-of-adventure-games/28894>. Luettu 20.7.2018.

5 Setting up Environment. 2013. Verkkoaineisto. libGDX. <https://libgdx.badlogicgames.com/documentation/gettingstarted/Setting%20Up.html>. Luettu 1.3.2018.

6 Creating a libgdx project. 2016. Verkkoaineisto. Github. <https://github.com/libgdx/libgdx/wiki/Project-Setup-Gradle>. Luettu 1.3.2018.

7 Microsoft acquires GitHub, the world’s largest source code platform. 2018. Verkkoaineisto. Techradar. <https://www.techradar.com/news/microsoft-reportedly-acquires-github-the-worlds-largest-source-code-platform>. Luettu 1.4.2018.

8 TMX Map Format. 2017. Verkkoaineisto. Tiled. <http://docs.mapeditor.org/en/stable/reference/tmx-map-format>. Luettu 1.4.2018.

9 Steam controls up to 70% of the PC download market and is "tremendously profitable". 2011. Verkkoaineisto. PC Gamer.<https://www.pcgamer.com/steam-controls-up-to-70-of-the-pc-download-market-and-is-tremendously-profitable/>. Luettu 17.8.2018.

10 There Are Over 125 Million "Active" Steam Accounts. 2015. Verkkoaineisto. Kotaku.<https://kotaku.com/there-are-over-125-million-steam-accounts-1687820875>. Luettu 15.8.2018.

11 With $4.3 billion in sales, 2017 was Steam’s biggest year yet. 2018. Verkkoaineisto. PCGamesN. <https://www.pcgamesn.com/steam-revenue-2017>. Luettu 16.8.2018