Paavo Pärssinen 356644
Joonas Humina 473721
Pertti Tuorila 481399
Henri Huotari 348241

Dungeon Crawler ryhmä 3/

Luolaluovinta

1. Yleiskatsaus

Tavoitteena oli tehdä yksinkertainen Dungeon Crawler -tyylinen peli, jossa pelaaja etenee satunnaisgeneroiduissa luolissa etsien uloskäyntiä ja taistellen vihollisia vastaan. Taisteluista pelaaja saa kokemuspisteitä ja hänen kyvyt kehittyvät taso tasolta. Pelaajan on myös mahdollista löytää parempia aseita ja haarniskoita, joiden avulla pelaaja kykenee voittamaan aina vaikenevat uudet viholliset.

Pelissä on SFML:llä luotu yksinkertainen käyttöliittymä. Peli-ikkunan alalaidassa sijaitsee HUD, josta pelaaja saa koko ajan tietoa kaikista häntä kiinnostavista asioista. Pelaajan hahmon statistiikat näkyvät vasemmalla alhaalla peli-ikkunassa. Keskellä alalaidassa on teksti-ikkuna johon peli tulostaa lokia pelissä tapahtuvista asioista. Näihin kuuluvat muun muassa taistelut ylipäätään, monstereiden huudot, pelaajan tason nousemiset ja luolasta toiseen siirtymiset. Oikeassa alalaidassa vuorostaan on pelaajan inventory, jossa hänen kantamansa omaisuus, miekat ja haarniskat, näkyvät. Pelaaja voi kantaa yhteensä kahdeksan itemiä, joista hänellä on päällänsä yksi haarniska ja yksi ase.

Pelaaja sijoitetaan pelin alussa satunnaisgeneroituun luolastoon, jossa hänen tehtävänsä on löytää tiensä ylemmälle luolaston tasolle ja lopulta ulos luolastosta. Luolissa kulkiessaan, pelaajan kimppuun käy erillaisia vihollisia, joista hänen tulee selviytyä. Pelaaja voi tehdä valinnan siitä että taisteleeko hän tiensä ulos vai koittaako hän juosta ja väistellä vihollisia. Pelaaja saa taisteluista kokemuspisteitä, jotka tekevät hänestä voimakkaamman ja näin hän kykenee taistelemaan seuraavilla tasoilla voimakkaampia vihollisia vastaan. Jokaisen luolaston tason jälkeen pelaaja parannetaan täysiin elämiinsä, jotta hän voisi selviytyä seuraavan tason.

1.1 Tiedetyt bugit

Uuden tason latautuessa pelaajan päälle voi syntyä vihollinen (bugi vai ominaisuus?)

Taistelun jälkeen joskus ensimmäisellä askeleella jonkun toisen vihollisen animaatio voi töksähtää. Ilmeisesti tämä bugi ilmenee, kun liikkumisnäppäintä pidetään painettuna pohjaan taistellessa.

Viholliset eivät osaa seurata kääntyessä ylös/alas. Tarkoittaa sitä, että kun pelaaja menee kulman taakse, vihollinen jää seisomaan kunnes pelaaja on poistunut vihollisen kuuloetäisyydeltä. Tämän jälkeen vihollisen jatkaa vapaata vaeltamistaan.

Viholliset saattavat mennä päällekkäin. Tällöin pelaaja voi vain lyödä päälimmäistä vihollista, mutta ottaa kaikilta vihollisilta vahinkoa (taas bugi vai ominaisuus?). Tällöin myös ilmenee ongelma että yhden vihollisen health bar peittää toisten vihollisten health barit.

Jos pelaaja katsoo esineen tietoja siirtyessään pelitasolta toiselle, voi tummeneva ruutu peittää tiedot alleen siirtymisen ajaksi.

Emme osanneet toteuttaa Item-luokkaa niin kuin aluksi meinasimme. Yritimme toteuttaa sitä niin että Armor- ja Weapon luokat ovat Item-luokan aliluokkia, joiden alkioiden pointterit tallennettaisiin samaan vectoriin. Näin saimme tehtyä, mutta kaikista alkioista tulee nyt Item-alkioita, eikä voida tehdä erottelua Armorin ja Weaponin välillä niin kuin haluaisimme.

Toimivan deconstructorin tekeminen Item-listan pointereille. Koitimmepa miten päin tahansa, emme saaneet kyseistä rakennetta toimimaan. Halusimme siis pointerit poistettua muistista kun peli sulkeutuu.

Bugi joka ilmenee joskus pelaajan edetessä pitkälle pelissä: peli kaatuu segmentation faulttiin, syy vielä tuntematon. Virhe viittaa esimerkiksi vihollisten tai itemien varastoinnin ongelmiin.

1.2 Halutut ominaisuudet tai säädöt, joita emme kerenneet toteuttamaan

Pelin taistelumekaniikan tasapainoittaminen. Olisimme halunneet saada pelin mahdollisimman hyvin tasapainoon. Tällä tarkoitamme sitä että peli skaalautuisi sopivalla nopeudella lineaarisesti vaikeammaksi. Tällä hetkellä peli voi olla aluksi todella helppo ja yhtä äkkiä. Alun vaikeuteen vaikuttaa eniten se, jos satut löytämään viholliselta jonkun pelin parhaista esineistä. Jos näin tapahtuu ei vihollisilla ole oikein mahdollisuutta tappaa sinua. Näin pääset etenemään useita tasoja kunnes viholliset skaalautuvat yhtäkkiä sen verran vaikeammiksi ettei sinulla ole oikein mahdollisuutta pärjätä niille. Viholliset tällä hetkellä skaalautuvat exponentiaalisesti. Tämä tarkoittaa sitä että muutaman tason jälkeen vihollisen puolustus kasvaa samalle tasolle kuin pelaajan hyökkäys arvo, jolloin pelaaja tekee todella vähän vahinkoa vihollisiin.

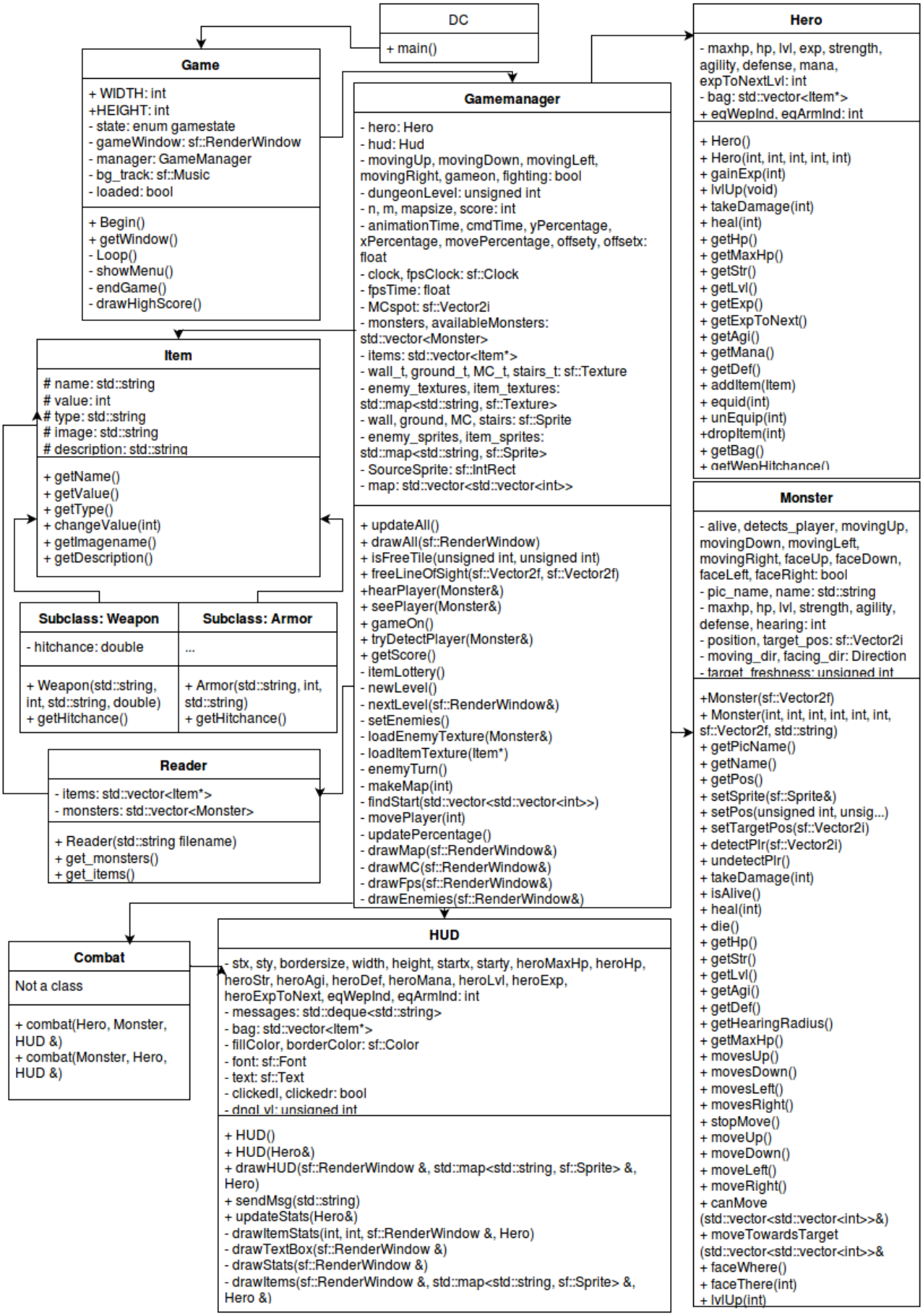
Toinen asia, johon toivoimme kerkeävän tutustumaan, oli Potion-tyyppisten esineiden toteuttaminen. Parantavat ja heron kykyjä parantavat potionit olisivat tuoneet lisää syvyyttä peliin. Näillä potioneilla olisi myös mahdollisesti pystynyt korjaamaan pelin taistelumekaniikan tasapaino ongelman. Potioneita olisi tippunut vihollisilta tai mahdollisesti olisimme myös voinut kehittää peliin kauppiaan jolta potioneita olisi voinut ostaa.

Kauppiaalle olisi voinut myös myydä heron ylimääräiset itemit ja ostaa uusia itemeitä tilalle. Tällöin olisimme myös toteuttaneet raha mekaniikan peliin. Rahaa pelaaja olisi saanut myymällä löytämiään esineitä tai tappamalla vihollisia.

Lisäksi pelin sokkeloita helpottamaan olisi voitu luoda pieni kartta pelaajalle, joka täyttyy sitä mukaa kun pelaaja näkee luolastoa. Toteutus olisi ollut melko helppo, HUD-luokkaan olisi vain pitänyt tallentaa kaksiulotteista vektoria nähdyistä ruuduista ja piirtää niistä esim 180*180 pikselin kartta (3*3 pikseliä per ruutu).

Eräs idea joka olisi myös ollut mielenkiintoinen olisi esineiden hajoaminen. Kun esineet kestävät ikuisesti, ovat huonot itemit turhia pelaajalle kun hän on saanut parhaat esineet. Esineen hajoamisen olisi voinut toteuttaa toteutuneiden combat-kertojen perusteella.

2. Ohjelman rakenne



Edellisellä sivulla esitetty pelin UML-kaavio. Peli käynnistyy DC.cpp tiedostossa sijaitsevasta main-funktiosta, joka alustaa Game luokan. Game-luokka vastaa SFML-kirjastoon perustuvan peli-ikkunan avaamisesta ja sen ylläpitämisestä, sekä esittää pelin main menun pelin käynnistyttyä ja game over -screenin peli loputtua. Itse peliä pyörittää Gamemanager-luokka, joka alustetaan Game-luokassa. Gamemanagerin-luokan constructor vuorostaan alustaa kaikki muut luokat, jotka se tarvitsee pelin pyörittämiseen.

Gamemanager muun muassa alustaa Reader-luokan, joka lukee sille atribuuttina annettujen tiedostonimiä vastaavat tiedostot. Nämä tiedostot sisältävät esineiden ja monstereiden tiedot. Samalla Reader-luokka alustaa Item-luokan alkiot sekä Monster-luokan alkiot ja lisää ne omiin listoihin. Nämä kaksi listaa ovat Reader-luokalla atribuutteina koko ajan muistissa. Kun kaikki tiedot on luettu ja Reader-luokka alustanut kaikki luetut tiedot, Gamemanager voi kutsua Reader-luokan funktioita, jotka palauttavat Item-listan ja Monster-listan.

HUD-luokka vastaa peli-ikkunalle piirrettävän käyttöliittymän tulostamisesta. HUD muun muassa tulostaa kaikki tärkeimmät pelissä tapahtuvat asiat hudin teksti-ikkunaan, esimerkiksi taistelussa tehdyt ja otetut vahingot. Combat-tiedoston alla löytyy taisteluissa tarvittavat funktiot. Oikea funktio vahinkojen laskemista ja ylläpitämistä varten valitaan riippuen siitä missä järjestyksessä atribuutit ovat funktio kutsussa.

3. Ohjelman logiikat ja tärkeimpien funktioiden rajapinnat

Aiemmassa luvussa kävimme ohjelman rakennetta ja logiikkaa hieman läpi, joten emme toista kaikkea tähän. Käymme myös läpi muutaman, meidän mielestä tärkeimmän rajapinnan lyhyesti läpi.

Itse pelin piirtofunktion logiikka lienee tärkeää selittää. Ohjelmassa on määritetty tietty aika joka kuluu siirtymiseen, tällä hetkellä 0.15 sekuntia. Pelaajan liikkuessa esimerkiksi oikealle, tuleekin vihollisia, seiniä ja lattioita siirtää tällöin vasemmalle. Aina pelin käydessä läpi piirtosilmukan se mittaa aikaa joka on kulunut liikkumiskomennon antamisesta, sekä laskee siitä prosenttiluvun tarvittavasta siirtymisestä. Tämän jälkeen voidaan piirtää liikutettavat grafiikat 50*prosenttiluku pikseliä vasemmalle, jolloin ohjelma piirtää sulavan siirtymisanimaation. Lisäksi sama prosessi tulee tehdä vihollisten omalle liikkumiselle, joka onkin tämän takia asetettu tapahtumaan samaan aikaan kuin pelaajan liikkuminen, jolloin siihen voidaan käyttää samoja prosenttilukuja.

Mainitsimme tosiaan miten Gamemanager alustaa Reader-luokan jolloin tämä lukee sille attribuuttina annettujen tiedostojen tiedot, alustaa ne ja kokoaa ne listoihin. Reader-luokka siis avaa ja lukee kaksi eri tekstitiedostoa, joihin on koottu pelissä käytettävät esineet ja viholliset. Esineillä on tietoina sen tyyppi, jonka mukaan se alustetaan, sen nimi, hyökkäys arvo, osumisprosentti, kuvatiedoston nimi ja kuvaus. Enemylist-tekstitiedosto noudattaa samaa kaavaa kuin itemlist-tekstitiedosto, vain eri attribuuteilla. Monsterit vaativat tiedon statseistaan, nimestään ja kuvatiedostostaan. Alla kuvat itemlist.txt ja enemylist.txt tiedostoissa olevista rakenteista.

```
{
type:weapon
name:Lightbringer
value:20
hitchance:0.99
image:resources/flamesword.png
description:Bring the heat
}
{
type:armor
name:Fancy dress
value:3
image:resources/dress.png
description:You look fabulous
}
```

```
{
name:Gargant
health:100
strength:5
defense:3
agility:5
hearing:3
picture:resources/gargant.png
}
```

Reader-luokan constructorin ulompi while-loop lukee tiedostoa niin pitkään että löytyy aaltosulku auki, jolloin suorittaminen siirtyy sisempään while-looppiin. Sisemmässä while-loopissa luetaan atribuutteja ja tallennetaan niitä väliaikaisiin muuttujiin niin pitkään että tulee aaltosulku kiinni. Kun tämä tapahtuu, tarkistetaan tiedot mitä oltiin löydetty, ja alustetaan kyseinen alkio, mikäli kaikki alustamiseen tarvittavat tiedot on löydetty. Alustettuaan alkion, constructori puskee alkion joko items vectoriin tai monsters vectoriin riippuen missä suorituksen vaiheessa ollaan. Kun Reader-luokka on alustettu, voi Gamemanager pyytää nämä listat oikeilla funktioilla.

4. Ohjelman rakentaminen ja käyttö

Jotta peliä voisi pelata, tarvitsee käyttäjällä olla koneellaan g++ kääntäjä, sekä SFML-kirjastosta versio 2.4.1 tai uudempi.

Pelille on tehty oma makefile, jonka avulla pelin kääntäminen on helppoa. Makefilessä on linkattu kaikki pelin tarvitsemat tiedostot ja kääntäjän attribuutit yhden tiedoston alle. Makefilen voi suorittaa siirtymällä terminaalilla dungeoncrawler3/src/ kansioon ja suorittamalla komennon “make”. Kun kääntäjä on tehnyt tehtävänsä, pelin voi käynnistää suorittamalla terminaalissa käskyn “./main”. Tällöin peli-ikkuna avautuu ja peli siirtyy aloitusikkunaan.

Aloitusikkunasta pelaaja voi edetä itse peliin noudattamalla peli-ikkunalla näkyvää ohjetta, eli painamalla Enter-näppäintä. Pelissä pelaajan tehtävä on päästä etenemään luolastossa tasolta toiselle. Peli on vuoropohjainen, joka tarkoittaa sitä että kun pelaaja on tehnyt oman vuoronsa, vuoro siirtyy tietokoneen ohjaamille monstereille seuraavaksi.

Pelaaja voi liikuttaa omaa hahmoaan nuolinäppäimistä. Taso tasolta vaikeutuvat viholliset koittavat päästä lyömään pelaajan ohjaamaa sankaria ja lopulta kukistamaan hänet. Pelaaja voi taistella takaisin, kun hän on monsterin vieressä. Pelaajan tulee painaa nuolinäppäintä sinne suuntaan missä vierellä oleva monsteri sijaitsee. Liikkumisen sijaan, pelaajan ohjaama sankari hyökkää monsteria vastaan ja aiheuttaa tähän vahinkoa. Mikäli monsteri kuolee, pelaajan hahmo saa kokemuspisteitä ja hänellä on mahdollisuus saada jokin esine tappamaltaan monsterilta.

Esineitä pelaaja voi kerätä omaan esineluetteloonsa, joita hän voi joko kantaa mukanaan, käyttää päällään tai pudottaa maahan, jotta hänellä olisi aina tilaa uusille esineille. Esineitä käytetään siirtämällä hiiren kursori esineen päälle, jolloin esineen tiedot tulevat näkyviin. Painamalla hiiren vasenta näppäintä voidaan ottaa esine käyttöön tai poistaa käytöstä. Painamalla hiiren oikeaa näppäintä heitetään esine pois.

Kun pelaaja haluaa lopettaa pelinsä, on hänellä kaksi tapaa sulkea peli. Hän voi joko painaa peli-ikkunan raksia sulkeakseen pelin, tai hän voi painaa näppäimistöstään Esc-näppäintä. Peli loppuu myös pelihahmon kuollessa. Pelin loppuessa peli kysyy pelaajan nimeä, joka kirjoitetaan näppäimistöllä. Nimi syötetään painamalla enteriä, jolloin peli näyttää pelin historian kymmenen parasta pelaajaa. Tällöin pelistä voi poistua lopullisesti painamalla esc-näppäintä.

5. Testaus

Aivan projektin alussa, ohjelmoituja funktioita testattiin yksinkertaisesti perus tulostuskomennoilla ja suorittamalla näitä erillään muusta koodista. Mikäli koodi oli toimiva, se integroitiin ohjelmisto kokonaisuuteen. Kun suunniteltu ohjelmistorakenne oli ruvennut muodostumaan, siirryimme tutustumaan “oikeisiin testeihin” eli google-testeihin. Google-testeillä pystyimme helposti seuraamaan että kaikki luodut funktiot toimivat oikein, joka uuden funktion integraation jälkeen.

Google testejä suoritettiin Googletestin versiolla 1.7.0. Google testit voi ajaa, mikäli Googletest löytyy asennettuna kansiosta $(HOME)/gtest-1.7.0. Yksikkötestit löytyvät kansiosta /dungeoncrawler3/src/unit_test. Niille on luoto oma Makefile, jonka suorittamisen jälkeen (“make” komento terminaalissa) voidaan testit suorittaa terminaalin komennolla “./test”.

Ohjelman sattuessa olemaan peli, pystyimme suorittamaan testausta myös peliä pelaamalla. Pelaamalla helposti pystyi huomaamaan mikäli uusi luotu ominaisuus ei toiminutkaan halutulla tavalla tai mikäli se rikkoi pelin kokonaan. Tällöin pystyi palaamaan koodiin ja ja pyrkiä korjaamaan pelin rikkoneen ongelma. Testaus pelaamalla sai usein helpommin ideoitua ja suunniteltua haluttavia ominaisuuksia pelille, joita ei välttämättä olisi tullut miettineeksi ilman.

Pelin graafisten funktioiden toteutusta oli helpointa testata tietenkin kokeilemalla. Usein on vaikea hahmottaa koodissa mikä piirtyy mihinkin, joten pelin käynnistäminen ja pelimaailmassa liikkuminen oli aina oleellinen osa testausta. Esimerkiksi pelihahmojen kääntyminen oli hankala hahmottaa koodissa, sillä käytettäessä SFML:n rotate-funktiota kääntyvät pelihahmot vasemman yläkulmansa mukaan. Vasta pelitestauksessa havaittiin, että kuvan originia pitää siirtää sitä mukaa kun kuvaa käännetään.

6. Työloki

Haluaisimme mainita tämän osion ensimmäisenä asiana, että ilman Paavo Pärssisen työpanosta projektiin, olisi projekti jäänyt ohueksi ja useita ominaisuuksia olisi jäänyt kehittämättä. Paavo koodasi pelin koodista muita enemmän, ja siksi haluisimme tehdä Paavon työpanoksesta erityismaininnan.

Ryhmän neljästä henkilöstä kolme oli ennen tätä kurssia tehnyt saman kaltaisen, isomman kokoluokan, ohjelmointiprojektin. Nämä projektit kuitenkin oltiin suoritettu yksin, eikä tällä tavoin ryhmässä, joten kaikille ryhmän jäsenille tämä oli uusi kokemus.

Viikko 1 (7.11 – 13.11, sekä edellisen viikon aloitusluento 2.11)

Aloitimme projektin esittely luennon jälkeen valitsemalla ryhmämme projektiaiheen. Kaikille lopulta sopi että tekisimme dungeon crawler tyyppisen pelin. Päätimme heti ensimmäisellä viikolla että tulisimme ainakin kerran viikkossa istumaan yhdessä alas kasvotusten. Koimme sen tarpeelliseksi ajatellen projektin eri vaiheiden selkeyttä.

Ensimmäinen viikko meni pelin yleisessä suunnittelussa, sekä ohjelmien asentelemisessa ja opettelussa. Kaikki siirryimme Ubuntun virtuaalikoneille tai oikeaan Debianiin, joten opettelu toimimaan uudessa ympäristössä vei hetken. Ensimmäisen viikon jälkeen meillä oli kasassa alustava kuva siitä, minkälainen luokkarakenne pelissä tulisi olemaan. Tämän pohjalta teimme projektisuunnitelman, joka sattui silloin jäämään hieman raakaversioksi.

Aika-arviot ja työn kuvaus

Yhdessä tekeminen: 5h, projektisuunnitelma

Henri: 4h, ohjelmien asentaminen ja opettelu

Paavo: 8h

Pelin yleistä suunnittelua tapaamisessa, ohjelmistojen asennus
Ensimmäiset graafiset testit SFML:llä, mm dvd-screensaver tyylinen hahmotelma, sekä pallon liikuttaminen ruudulla nuolinäppäimillä
Laajennettu liikkumistesti, jossa pelihahmoa liikutettiin tyhjällä alustalla
Hero-luokka, tärkeimmät funktiot
Yksinkertainen dungeonin luominen kaksiulotteisessa arrayssa (nollia ja ykkösiä)

Pertti: 6h

Ohjelmien asentelua, suunnittelua mm. luolien generoiminen ja project plan beta.

Muonitus

Joonas: 6h

Tietokoneelta puuttui kokonaan ohjelmointiympäristö, joten sen asentaminen ja SFML tutkailu teorian tasolla. Debian ei asentunut koneelle ensimmäisen viikon aikana.

Viikko 2 (14.11 – 20.11)

Periaatteessa koodausviikko. Aloitimme suunnitelman mukaan kaikista yksirkertaisimmista asioista. Ensimmäisinä loimme käytettävien luokkien pohjat, esimerkiksi item-luokan pohjatyö. Myös ensimmäiset testaukset SFML-kirjaston käyttöön tehtiin tällä viikolla ja Makefilen käyttöön tutustuttiin myös tarkemmin. Paavo toteutti pelistä jo alkeellisen grafiikka kokeilun.

Aika-arviot ja työn kuvaus

Yhdessä tekeminen: 4h, suunnittelu ja yhdessä koodaaminen

Henri: 5h, Item-luokan suunnittelu ja toteuttaminen

Paavo:6h

- Dungeon generatorin yhdistys grafiikkatestiin, hero liikkuu luolastoissa
- Grafiikkatestin ikkunan suurentaminen, portaiden kuvan lisääminen, dungeon generator array→vector
- Pelaajan HUDin perustoteutus
- Hero-luokan header-tiedosto

Pertti: 6h

- Filereader beta.

Joonas: 8 h

- Linux asennettuna, mutta koneen verkkokortti lakannut toimimasta täysin debianissa. Pelkkää koneen kanssa taistelua vähintään 6h, siinä sivussa taistelujärjestelmän alkuperäinen suunnitelma. Ensimmäisessä koodaus-sessiossa syntyy projektin Makefile.

Viikko 3 (21.11 – 27.11)

SFML-kirjaston graaffiset kokeillut siirrettiin pelin source tiedostoihin ja lisää peliin tarvittavia luokkia luotiin tällä viikolla. Peliä pyörittävät luokat myös jaettiin uudestaan helpommin käsitettäviin ja ylläpidettäviin osiin (DC, Game ja Gamemanager rakenne). Pieniä lisäyksiä pelin grafiikoihin ja vanhojen ominaisuuksien korjailua.

Aika-arviot ja työn kuvaus

Yhdessä tekeminen: 3-4h, suunnittelu, korjailu ja koodaaminen

Henri: 3h, reader-luokan suunnittelu

Paavo: 4h

Grafiikkatesti jaettu kunnolla luokkiin, ei enää pelkkä testi. Luokat: Gamemanager, HUD,Game.
Game käynnistää managerin, joka pyörittää liikettä ja grafiikoita
Tarpeellisia lisäyksiä HUDiin, pelaajan statsit tulostetaan, tekstipalaute

Pertti: 0h

Joonas: 0h

Viikko 4 (28.11 – 4.12)

Perus koodailu viikko, hyvin samanlainen kuin edelliset viikotkin. Lisää muokkauksia pelin HUD:iin ja Gamemanager-luokkaan. Reader-luokan luonti vastaamaan pelissä olevien esineiden lukemiseen tesktitiedostosta. Korjailua aiempaan koodiin ja viimeisten luokkien suunnittelu.

Aika-arviot ja työn kuvaus

Yhdessä tekeminen: 6h, projektin yhdessä korjailu, uusien ominaisuuksien suunnittelu ja koodaaminen

Henri: 4,5h, Reader-luokan suunnittelu ja toteuttaminen, dokumentoinnin aloittaminen

Paavo:6h

Liikkumisanimaatiot vihollisille, vihollisluokan yleinen säätäminen
HUD-säätöjä
Vihollisten kääntyminen ja liikkuminen animoitu paremmin
HUDissa hero:n tavaroiden piirtäminen
Sprite:jen ja textuurien tallettaminen managerissa mappiin
Itemien käyttö HUDissa, voi pukea päälleen armorin ja weaponin
Header-guardeja luokille, ei enää ongelmallisia virheilmoituksia
Readerin säätö

Pertti: 6h

Aloin luomaan vihollisluokkaa.

Joonas: 6h

Combat-funktion rakentelua, sekä siihen liittyvät asiat. Koneen kanssa toiminta edelleen vaikeaa.

Viikko 5 (5.12 – 11.12)

Viiikko tuntui vähän hiljaisemmalta kuin muut viikot. Puuttuvien ominaisuuksien lisäily, muun muassa combat-funktiot. Projektin dokumentointia jatkettiin ja peli sai jo oman taustaraitansa.

Aika-arviot ja työn kuvaus

Yhdessä tekeminen: 2h, puuttuvien ominaisuuksien yhdessä miettiminen ja toteuttaminen

Henri: 3h, dokumentoinnin jatkaminen ja puuttuvien ominaisuuksien suunnittelu

Paavo: 8h

Siirtyminen tasolta tasolle
HUD säädöt
Combat-algoritmin käyttö pelissä, perus taistelu liikkumalla toteutettu
Animaatio tasolta toiselle siirtymiseen, sekä järkevät funktiot siihen
Ensimmäiset unit-testit luotu
Implementoitu kuoleminen pelissä, sekä game over -screen
Lisätty kuvaukset itemeille, näkyy HUDissa
Itemien unequipping ja dropping toteutettu
Lisätty reilusti eri itemejä
Toteutettu yksinkertainen High Score -systeemi, sekä yleensäkin score-systeemi
Lisätty background-musiikki

Pertti: 6h

Lisää vihollisten kehittämistä, koodin siivousta

aJoonas: 0,5h

Ryhmän henkinen johtaminen.

Viikko 6 (12.12 – 16.12)

Kyseisellä viikolla kaikilla meistä oli jo tenttejä, mutta onneksemme projekti oli jo hyvässä vaiheessa. Puuttuvia asioita, jotka halusimme toteuttaa, ei ollut kuin muutama. Puuttuvia asioita olivat reader-luokan laajentaminen koskemaan myös monstereita. Pieni muutos aiempaan reader-luokkaan tarvittiin tämän lisäksi. Esinelista laitettiin sisältämään pointtereita esineisiin ennemmin kuin pelkät item-alkiot. Myös combat funktiot päivitettiin sisältämään tarkistus osuuko hyökkääjä viholliseen käyttämällään aseella.

Torstaina 15.12 teimme viimeiset muutokset ja lisäykset projektiin, tarkistimme työn, sekä päivitimme dokumentoinnin ajan tasalle. Vietimme koko päivän yhdessä tilassa, eli noin 10h yhteen kyytiin. Ryhmän kaikki jäsenet olivat tyytyväisiä lopulliseen versioon pelistä.

Aika-arviot ja työn kuvaus

Yhdessä tekeminen: 15+h, viimeisten puuttuvien asioiden suunnittelu ja toteuttaminen

Henri: 7h, dokumentaation täydentäminen

Paavo: 12h

Vihollisten skaalautuminen eri tasoillle lisätty

Lisätty unit-testejä

Loppuväännöt

HUD-säätöjä

High score korjauksia

Lisää vihollisia

Pertti: 6h

Korjannut Itemit, nyt suojat ja aseet toimii.

Muonitus

Joonas: 6h

Taisteluihin lisätty mahdollisuus lyödä ohi.

Lisätty erilaisia vihollisia.

Pelin raivokas testaaminen.

7. Projektin päätös

Ohjelmoinnin ryhmätyö on ollut opettavainen kokemus. Kaikilla ryhmän jäsenillä on yhteinen päämäärä, mutta eri ajatus reitistä, jota pitkin päästään päämäärään. Työn aikana tuli nähtyä päällekkäistä työtä, ristiriitaisia toteutuksia sekä muita epämääräisyyksiä.

Mitä on opittu? Ennen varsinaista toteutusta, pitää ohjelman rakennetta suunnitella mahdollisimman yksityiskohtaisesti. Todennäköisesti alkuperäiset suunnitelmat tulevat muuttumaan, mutta etukäteen tehty suunnittelu auttaa välttämään turhaa työtä, sekä myöhäisessä vaiheessa tehtyjen muutoksien aiheuttamaa ahdistusta.