KYMENLAAKSON AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma / Ohjelmistotekniikka

Jose Hirvonen

VERKKOPELI

Opinnäytetyö 2014

TIIVISTELMÄ

KYMENLAAKSON AMMATTIKORKEAKOULU
Tietotekniikka

HIRVONEN, JOSE	Verkkopeli
Opinnäytetyö	37 sivua + 2 liitesivua
Työn ohjaaja	Yliopettaja Paula Posio
Toimeksiantaja	Kymenlaakson ammattikorkeakoulu
Elokuu 2014	
Avainsanat	html, moninpeli, mysql, php

Tämän opinnäytetyön aiheena oli pelin luominen käyttäen PHP-ohjelmointikieltä ja MySQL-tietokantaa. Tavoitteena oli luoda verkkoselaimessa monen pelaajan pelattava peli.

Opinnäytetyö aloitettiin ideoimalla peli ja tekemällä yksinkertainen suunnitelma pelin toiminnallisuudesta. Tämän suunnitelman pohjalta suunniteltiin MySQL-tietokanta, jota peli käytti toimiakseen. Peli rakennettiin Linux-palvelimelle ja apuna oli lukuisia eri ohjelmia muun muassa tarvittavan grafiikan tuottamiseen. PHP-koodin kirjoittamiseen käytettiin yksinomaan Notepad++-ohjelmaa ja sen toimintoja.

Opinnäytetyön tuloksena oli peli, joka saavutti kaikki sille asetetut tavoitteet. Peli pystyttiin asentamaan toiseen järjestelmään, joka täytti kaikki tarvittavat tekniset vaatimukset. Pelaajaa pystyi luomaan oman käyttäjätilin sekä pelihahmon ja pelaaja pystyi läpäisemään pelin.

Pelin rakentaminen opetti paljon PHP:n luokkaominaisuuksien käytöstä sekä MySQL:n käytöstä kehittyneemmällä tasolla. Suunnitelmia tehtiin vielä pelin päivittämiseen tulevaisuudessa ja mahdollisten uusien pienten ominaisuuksien lisäämisestä peliin.

ABSTRACT

KYMENLAAKSON AMMATTIKORKEAKOULU
University of Applied Sciences
Information Technology

HIRVONEN, JOSE	Online Game
Bachelor's Thesis	37 pages + 2 pages of appendices
Supervisor	Paula Posio Principal Lecturer
Commissioned by	Kymenlaakso University of Applied Sciences
August 2014	
Keywords	html, multiplayer, mysql, php

The subject of this thesis was creating a game using PHP programming language and MySQL database. The objective of this thesis was to create a multi-player game that can be played in a web browser.

The thesis was started by thinking up the game and creating a simple plan of the game's functionality. Based on that plan a MySQL database was designed for the game to use. The game was built on a Linux server and numerous tools were used to, among other things, create necessary graphics. Notepad++ and its functions were used exclusively to write the PHP code.

The result of this thesis was a game that met all the goals that were set for the game. The game could be installed to another system that fulfilled all the technical requirements. The player could create their own user account and character, and the player was also able to complete the game.

Building the game taught a great deal about class properties of PHP and also about the advanced usage of MySQL. Plans were made to update the game in the future and to possibly add new minor features to the game.

SISÄLLYS

TERMIT

Apache	Apache HTTP Server, yleisesti pelkkä Apache, on avoimen lähdekoodin HTTP-palvelin, joka huolehtii verkkosivujen näyttämisestä käyttäjille (1).
Bittilippu	Muuttuja, johon on tallennettu useita totuusarvoja. Verrattavissa riviin kytkimiä, jotka ovat joko on- tai off-asennossa.
Debug, debuggaus	Debuggauksella tarkoitetaan ohjelmassa olevien mahdollisten virheiden etsimistä ja tuhoamista. Debug-tila on ohjelmaan erityisesti ohjelmoitu testaamista helpottava tila.
Html	Verkkosivujen luontiin tarkoitettu merkintäkieli, jonka internetselain tulkitsee käyttäjälle (2).
Lokalisointi	Tuotteen kääntäminen toiselle kielelle. Mahdollisuus käyttää useita eri kieliä.
MySQL	Oracle Corporationin omistama avoimeen lähdekoodiin perustuva tietokantaohjelmisto (3).
PHP	Eli PHP Hypertext Preprocessor on verkkosivuilla käytetty palvelinpohjainen ohjelmointikieli, jolla voidaan luoda dynaamista eli muuttuvaa sisältöä sivulle (4).
SQL	Eli ”structured query language”, suomeksi ”standardoitu kyselykieli” on kieli, jolla tietokannasta voidaan hakea, lisätä tai muuttaa tietoa (5). On tietokantaohjelmistokohtaista ymmärtääkö tietokanta SQL-kieltä vai ei.
Tekstuuri	Kaksiulotteinen kuva, joka kierretään kolmiulotteisen objektin päälle.

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena oli tutkia, voidaanko PHP:tä ja MySQL-tietokantaa käyttämällä luoda interaktiivinen verkossa selaimella pelattava peli, jossa on monta pelaajaa.

Inspiraatio peliin on saatu vuoden 1997 elokuvasta ”Cube”, jossa viisi toisilleen tuntematonta henkilöä heräävät kuution muotoisesta sokkelosta ja yrittävät löytää tietä ulos. Vaaroina ryhmällä ovat kuolettavasti ansoitetut huoneet sekä erityisesti ryhmän hajoaminen sisältäpäin. Jälkimmäistä ei ole oikein mahdollista jäljitellä pelimaailmassa, joten pelissä ansat ovat pelaajien kohtaama suurin vaara.

Idea toteuttamisesta PHP:tä ja MySQL:ää käyttäen lähti puhtaasta mielenkiinnosta ja haluun vastata kysymykseen ”olisiko tämä mahdollista toteuttaa?” Lisäksi peliä tekemällä oli mahdollista harjoittaa omia MySQL- ja PHP-taitoja, joista olisi todennäköisesti suuresti hyötyä tulevaisuudessa.

Pelille asetettiin seuraavanlaiset vaatimukset: PHP:stä oltava vähintään versio 5 ja MySQL:stä vähintään versio 5.0. Peli pitää olla helposti siirrettävissä, eli se pitää voida kopioida toiselle palvelimelle, jolloin se on yksinkertaisen asennuksen jälkeen helposti käytettävissä. Edellyttäen tietenkin, että järjestelmästä löytyvät tarvittavat PHP- ja MySQL-versiot. Peli ei saa toimiakseen käyttää mitään ulkoisia ohjelmia, kuten järjestelmän ajastettuja toimintoja. Tämän lisäksi pelin pitää olla helposti lokalisoitavissa, eli käännettävissä toiselle kielelle.

Pelille olisi voinut rakentaa myös erillisen asiakasohjelman, jota kautta pelaajan tekemät toiminnot ja komennot välitettäisiin palvelimelle, sen sijaan että peliä pelattaisiin puhtaasti web-selaimen kautta. Tämän katsottiin kuitenkin vievän liikaa aikaa ja tässä projektissa haluttiin muutenkin keskittyä erityisesti PHP-koodin tuottamiseen.

PHP:llä työskennellessä vastaan tuli myös erinäisiä rajoituksia. PHP on suunniteltu verkkosivukäyttöön, eli PHP-koodi suoritetaan aina sivua ladattaessa. Tässä projektissa se tarkoittaa sitä, että tiedot ladataan tietokannasta aina uudestaan verkkosivua ladattaessa, mikä aiheuttaa ainakin pientä lisäkuormaa palvelimelle.

2 OHJELMAT, TYÖKALUT JA KIELET

2.1 Web-projektin toteutus

Ei ole olemassa yhtä tiettyä oikeaa mallia, kuinka web-projekti pitäisi toteuttaa. Monia erilaisia malleja on kuitenkin kehitetty helpottamaan työtä, kuten seuraava kuusivaiheinen web-projektin toteuttamisprosessi. (6.)

Ensimmäinen vaihe, ennen kuin mitään konkreettista tehdään, koostuu tiedon keräämisestä. Tässä vaiheessa selvitetään sivuston tarkka tarkoitus, kohdeyleisö, sivuston sisältö ja tavoitteet eli tarjoaako sivusto tietoa, toimiiko se kohtauspaikkana jollekin yhteisölle ynnä muuta.

Toisessa vaiheessa aloitetaan sivuston suunnittelu ja luodaan sivukartta, johon listataan kaikki sivuston pääalueet. Samalla päätetään, otetaanko käyttöön jokin jo olemassa oleva sisällönhallintateknologia vai luodaanko oma.

Kolmannessa vaiheessa keskitytään ulkonäön suunnitteluun. Sivuston ulkonäön pitäisi olla kohderyhmänsä näköinen, eli asiasivustoilla käytettäisiin hillittyjä värejä ja kirjasimia ja niin edelleen. Tässä vaiheessa luodaan myös graafinen malli sivustosta, käyttäen yleensä kuvankäsittelyohjelmaa.

Neljännessä vaiheessa aloitetaan sivuston ohjelmointi ja puretaan mallit käyttökelpoiseksi sivustoksi. Yleensä kehitys aloitetaan etusivusta, jota valmistuttuaan käytetään pohjana muille sivuille.

Viidennessä vaiheessa aloitetaan sivuston testaus ja korjataan kaikki mahdolliset viat joita esiin tulee. Kun sivustoon ollaan tyytyväisiä, siirretään se lopulliseen sijaintiinsa ja julkaistaan yleisölle.

Kuudes vaihe on ylläpito. Web-projektit harvoin päättyvät sivuston julkaisuun, vaan aina löytyy päivitettävää, korjattavaa ja lisättävää.

Projektin voidaan katsoa päättyneen lopullisesti sitten, kun sivustoa ei enää päivitetä ja/tai aloitetaan uuden projektin tekeminen. Tässä projektissa ei kuitenkaan päästy noudattamaan aivan samaa kaavaa, vaan esimerkiksi ohjelmointi alkoi jo projektin alkupuolella, ulkonäön suunnittelu oli yksi viimeisimpiä vaiheita ja testausta tapahtui koko projektin ajan.

2.2 Tietokanta

Tietotekniikassa tietokanta, tai relaatiotietokanta, on mekanismi, jonka avulla voidaan sekä säilyttää että hakea tietoa. Tietokanta koostuu tauluista, joita tietokannasta löytyy vähintään yksi kappale. Taulu koostuu sarakkeista ja riveistä, taulukkolaskentaohjelmien tyyliin ja se sisältää usein vähintään yhden avainsarakkeen, jonka perusteella tietokannan tauluja voidaan hauissa yhdistellä toisiinsa. (7.)

Tietokantojen yhteydessä on kaksi yleistä käytäntöä, joita olisi hyvä noudattaa. Yksi on, että nimeämiskäytäntö on selkeä. Taulujen nimet kuvaavat niiden sisältöä ja nimet ovat yksikkömuodossa. Toinen käytäntö on, että tietokanta suunnitellaan siten, että tallennettavat tietueet ovat ainutlaatuisia, eikä samaa tietoa tallenneta kahta kertaa.

Esimerkkinä vaikkapa yksinkertainen tietokanta, johon listataan kirjoja. Tietokantaan luodaan kaksi taulua: yksi kirjoille ja toinen kirjailijoille. *Kirja*-taulussa (kuva 1) voisi olla vaikkapa sarakkeet *Avain*, *Nimi*, *Kustantaja* ja *Painovuosi*.

Kirja			
Avain	**Nimi**	**Kustantaja**	**Painovuosi**
1	Hakkeroinnin alkeet	HaxBoox	2011
2	Kissanhoito-opas 3	Elukka	2014

Kuva 1. *Kirja*-taulu.

Kirjailija-taulu (kuva 2) voisi sisältää sarakkeet *Avain*, *Etunimi* ja *Sukunimi*.

Kirjailija		
Avain	**Etunimi**	**Sukunimi**
1	Ahto	Simakuutio
2	Lea	Vähälaama
3	Tane	Yrmiö

Kuva 2. *Kirjailija*-taulu.

Miksi ei ole vain yhtä taulua jossa ovat kaikki tiedot yhdessä? Siksi, koska, että yhdellä kirjalla voi olla useampiakin kirjoittajia ja yhdellä kirjailijalla voi olla useampi kirja. *Kirja*-taulun suunnittelu niin, että siellä olisi vaikkapa Kirjailija1-, Kirjailija2- ja Kirjailija3-sarakkeet useampien kirjailijoiden varalta tekisi taulusta hankalan käyttää, ja se rikkoisi yleisiä käytäntöjä.

Periaatteessa myös kustantajista pitäisi tehdä oma taulunsa, mutta esimerkin vuoksi tietokannan rakenne pidetään mahdollisimman yksinkertaisena. Tässä esimerkissä rikotaankin käytäntöä, jonka mukaan jokaisen tietueen on oltava ainutlaatuinen. Useammalla kirjalla voi olla sama kustantaja, joka tässä tietokannassa pitäisi kirjoittaa *Kustantaja*-sarakkeeseen useamman kerran, sen sijaan. että tieto haettaisiin omasta *Kustantaja*-taulusta.

Tietokantaa pitää erikseen ohjeistaa, kuinka eri taulut ovat yhteydessä toisiinsa. Jotta kirjat ja kirjailijat saadaan yhdistettyä, tarvitaan erityinen yhdystaulu (kuva 3), joka sisältää kaksi avainsaraketta.

KirjaJaKirjailija	
AvainKirja	**AvainKirjailija**
1	1
1	2
2	3

Kuva 3. Yhdystaulu, joka yhdistää *Kirja*- ja *Kirjailija*-taulut.

Yhdystaulu kertoo tietokannalle, mitkä taulujen avaimet kuuluvat yhteen. Yhdystaulusta voidaan nähdä, että kirjaan 1 on liitetty kaksi kirjailijaa, kirjailijat 1 ja 2. Kirjalla 2 on vain yksi kirjailija, eli 3.

Nyt tietokanta itsessään on käyttökuntoinen ja siihen voidaan tehdä kyselyjä, eli hakea tietoa. Jos esimerkiksi haluttaisiin listata tietokannan kaikki kirjat kirjailijoineen, voittaisin tehdä kuvan 4 mukainen SQL-kielinen kysely.

```
SELECT Nimi, Kustantaja, Painovuosi
GROUP_CONCAT(
        DISTINCT KirjaJaKirjailija.AvainKirjailija
        ORDER BY Kirjailija.Sukunimi ASC
        SEPARATOR ','
) AS Kirjailijat,
FROM Kirja
LEFT JOIN KirjaJaKirjailija ON Kirja.Avain = KirjaJaKirjailija.AvainKirja
LEFT JOIN Kirjailija ON KirjaJaKirjailija.AvainKirjailija = Kirjailija.Avain
```

Kuva 4. Tietokannan kaikki kirjat listaava kysely.

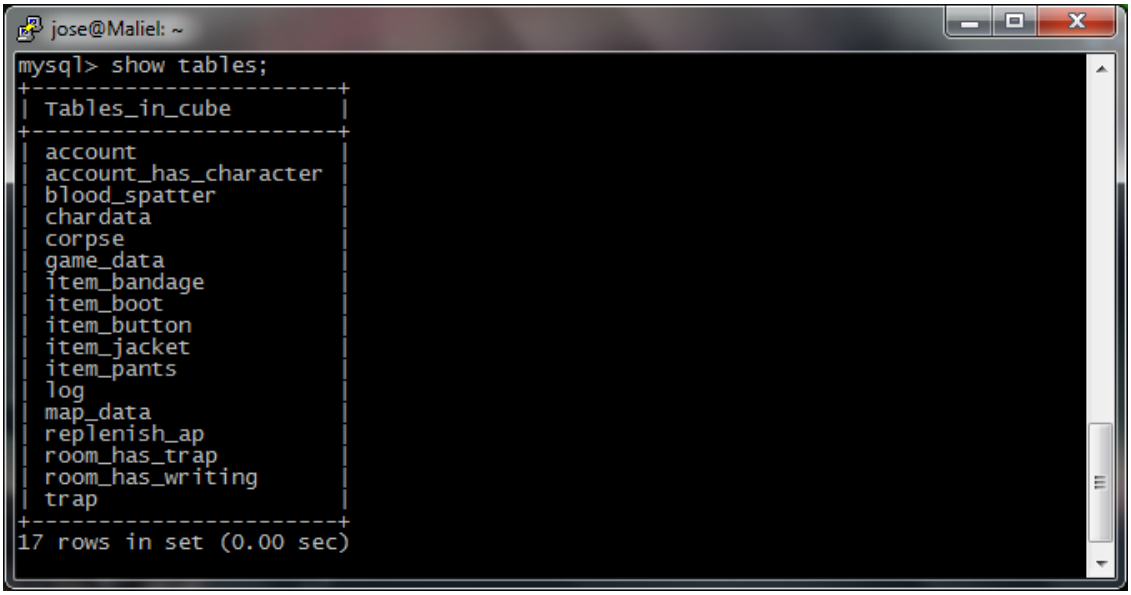
SELECT-komennolla määritetään, mitkä sarakkeet hakuun halutaan mukaan, *GROUP_CONCAT*-komennolla saadaan yhdistettyä yhteen sarakkeeseen useampia rivejä toisista tauluista, *FROM*-komento määrää, mistä taulusta tiedot ensisijaisesti haetaan ja *LEFT_JOIN*-komennoilla kaikki taulut liitetään yhdystaulun avulla yhteen. Tämä kysely tuottaisi kuvan 5 mukaisen hakutuloksen.

Hakutulos			
Nimi	**Kirjailijat**	**Kustantaja**	**Painovuosi**
Hakkeroinnin alkeet	Ahto Simakuutio, Lea Vähälaama	HaxBoox	2011
Kissanhoito-opas 3	Tane Yrmiö	Elukka	2014

Kuva 5. Haun lopputulos. Kirjailijat ovat saatu yhdistettyä pilkulla erotettuna käyttäen SQL-kielen *GROUP_CONCAT*-komentoa.

2.3 MySQL

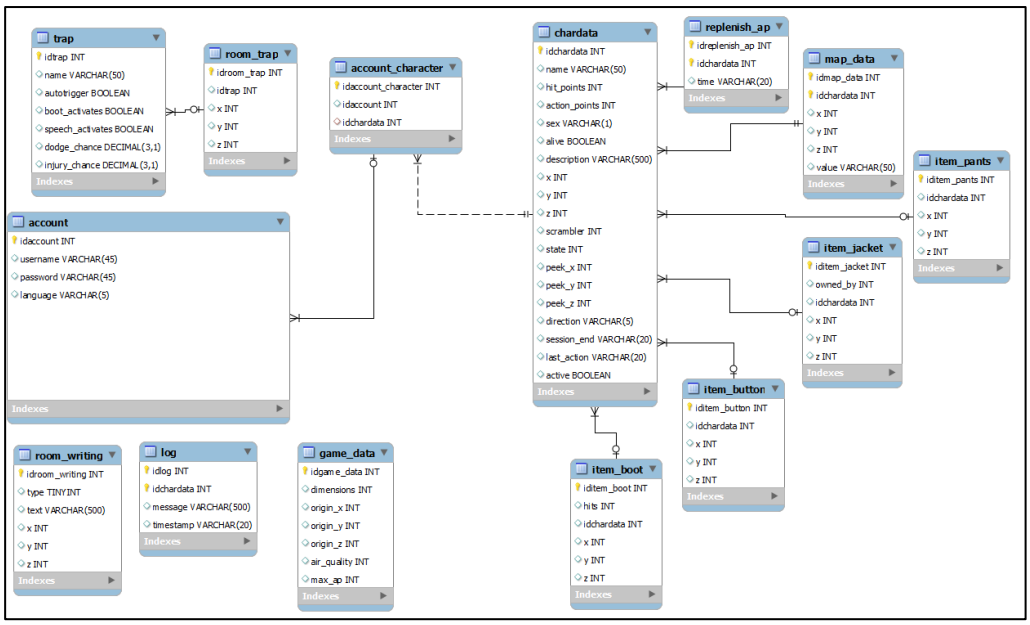
MySQL on nykyisin Oracle Corporationin omistama avoimen lähdekoodin tietokantaohjelmisto. MySQL:n hallinta tapahtuu MySQL:n oman konsolin kautta SQL-kieltä käyttäen (kuva 6), joskin graafisia työkaluja on kehitetty helpottamaan käyttöä, kuten MySQL Workbench.



Kuva 6. MySQL-konsoli.

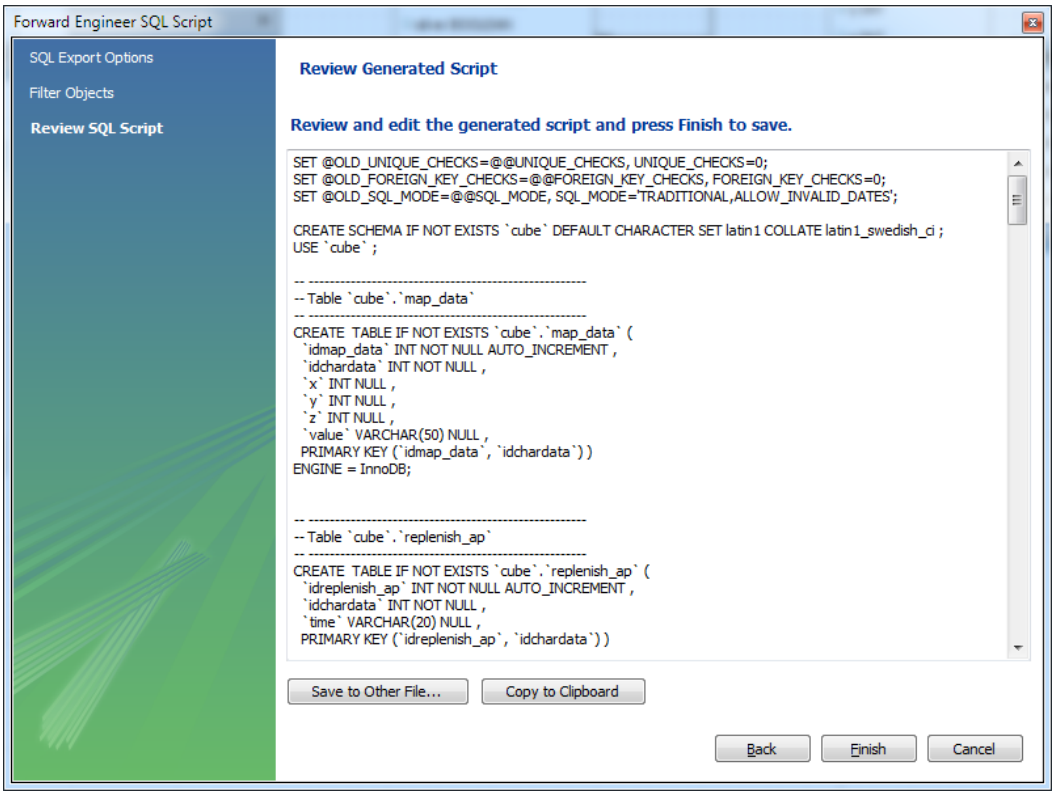
2.4 MySQL Workbench

MySQL Workbench on graafinen työkalu MySQL:n hallintaan. Workbenchillä voi suunnitella, luoda sekä muokata graafisesti tietokantoja, tauluja sekä näiden yhteyksiä. Workbenchillä tietokannasta voidaan luoda graafinen malli (kuva 7).



Kuva 7. Workbenchillä mallinnettu tietokanta.

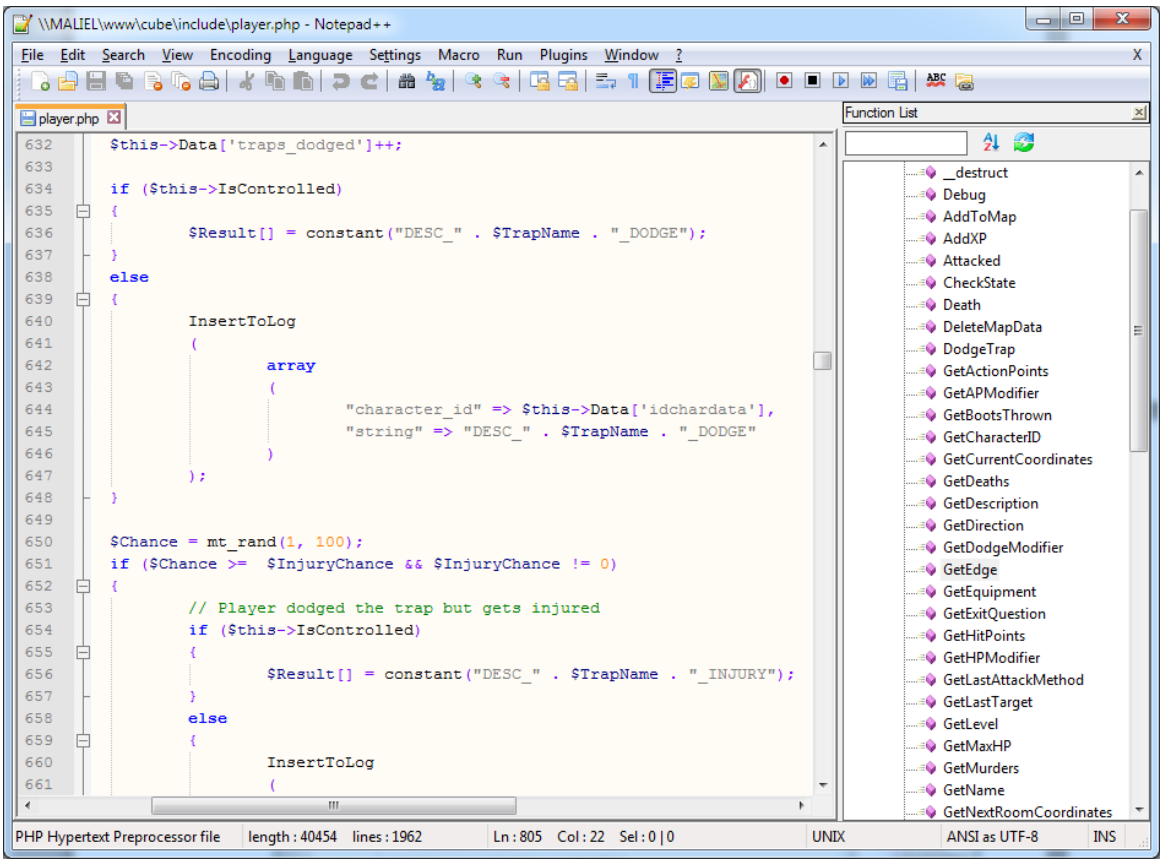
Mallinnetun tietokannan saa tallennettua Workbenchistä ”Export”-toimintoa käyttäen SQL-tiedostoksi (kuva 8). SQL-tiedosto pitää sisällään SQL-kielisiä komentoja, jotka voidaan lukea MySQL:n komentotulkkia käyttäen, jolloin jokaista komentoa ei tarvitse kirjoittaa erikseen.



Kuva 8. Luotavan SQL-tiedoston esikatselu.

2.5 Notepad++

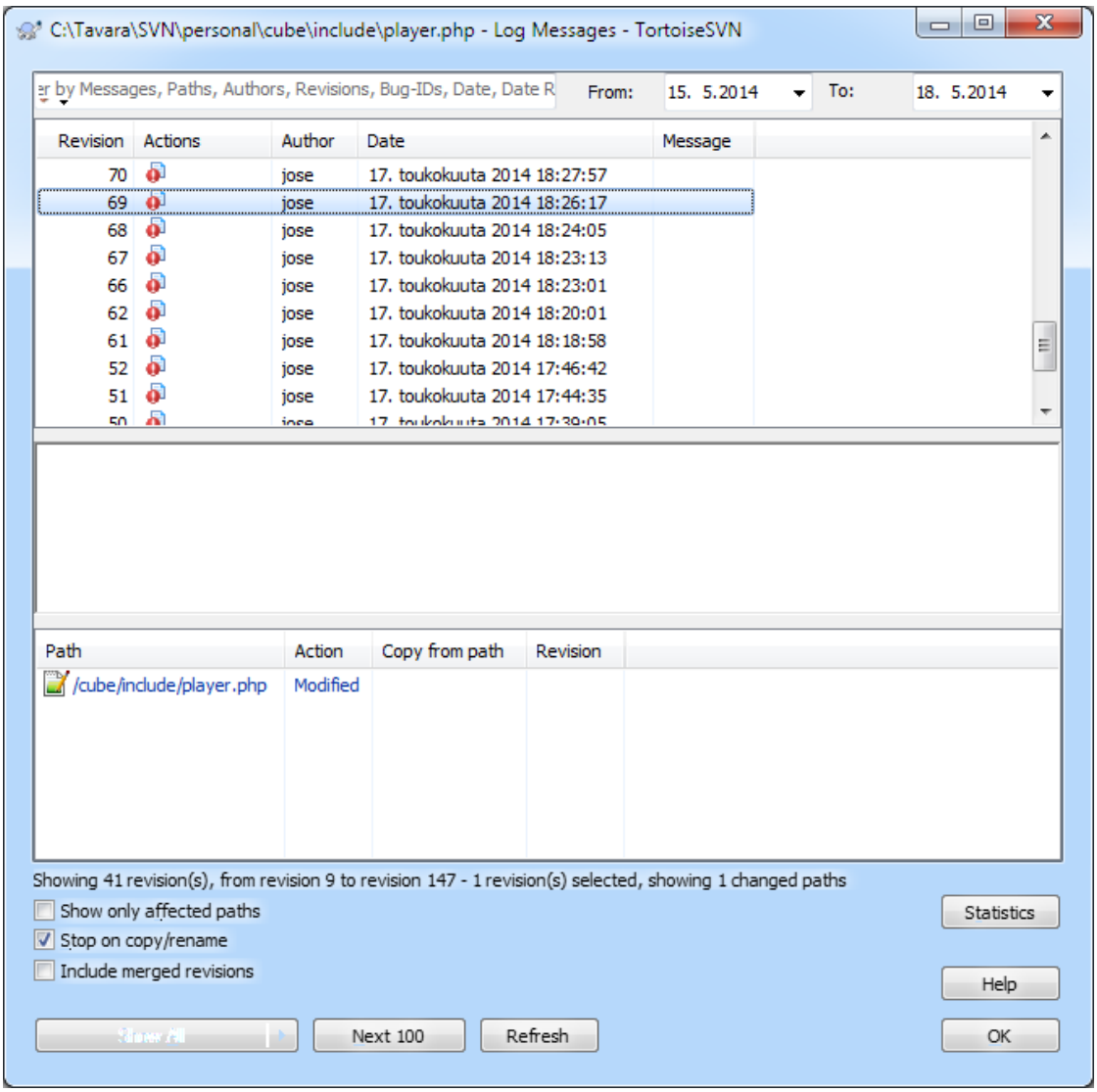
Notepad++ on yksinkertainen tekstieditori ja sitä voi pitää Windowsin muistion paranneltuna versiona. Notepad++ on erityisesti optimoitu ohjelmointitarkoituksiin (8). Ohjelma osaa esimerkiksi värjätä koodia käytetyn ohjelmointikielen kieliopin mukaan helpottamaan koodin luettavuutta ja sillä saa myös listattua kaikki tiedostosta löytyvät funktiot (kuva 9).



Kuva 9. PHP-tiedosto avattuna Notepad++:ssa. Tiedoston sisältämät funktiot ovat listattuna oikealla puolella.

2.6 Apache Subversion

Apache Subversion on Apachen kanssa yhdessä toimiva versionhallintaohjelma, jota käytetään yleisesti ohjelmistoprojekteissa (9). Versionhallinta pitää kirjaa jokaisesta versionhallintaan syötetystä tiedostosta ja niiden muutoksista. Versionhallintaa käyttäen on mahdollista saada halutusta tiedostosta ulos mikä tahansa versio (kuva 10), mikä versionhallintaan vain on tallennettu.



Kuva 10. Tiedostohistorian tarkastelua versionhallinnassa.

2.7 PHP

PHP on suunniteltu dynaamisen sisällön tuottamiseen verkkosivuilla ja sen kielioppi on paljolti lainattu C-kielestä, Javasta ja Perlistä. Se ei ole selainkohtainen kieli, kuten esimerkiksi JavaScript, jonka toimivuus riippuu käyttäjän selaimesta, vaan PHP-koodi suoritetaan palvelimella, eikä ulkopuolisen henkilön ole mahdollista tarkastella koodia murtautumatta palvelimelle. Lopputulos näytetään pelaajalle tavallisella HTML-kielellä selaimen kautta käyttäjälle. Mitään erityisesti peleille suunniteltuja HTML-elementtejä ei käytetä.

3 PROJEKTIN TOTEUTUS

3.1 Pelin yleiskuvaus

Pelin päämääränä on selvitä ulos kuution muotoisesta sokkelosta. Oletuksena sokkelon yhdessä sivussa on 26 huonetta, eli sokkelossa on oletuksena kaiken kaikkiaan 17 576 huonetta.

Jokainen huone on kuutionmuotoinen, jossa on jokaisella sivulla ovi. Pelaaja voi siirtyä siis kuuteen eri suuntaan. Aivan kuten oikeassa elämässä, jos henkilö heräisi kuutionmuotoisesta ikkunattomasta huoneesta, ei hän tietäisi missä on mikäkin ilmansuunta. Tätä simuloidakseen on jokaisella pelaajalla niin kutsuttu sekoitusarvo. joka arvotaan aina kun pelaaja luodaan tai pelaaja herää uudestaan henkiin kuoleman jälkeen. Käytännössä tämä tarkoittaa esimerkiksi sitä, että kun pelaaja 1 liikkuu pohjoiseen, niin pelaaja 2:n näkökulmasta pelaaja 1 liikkuu länteen ja päinvastoin.

Pelaajan siirtyminen huoneesta toiseen tapahtuu kahdessa vaiheessa. Ensin pelaaja valitsee suunnan, jolloin pelaaja kurkkii toiseen huoneeseen. Kurkkiessaan pelaaja voi päättää siirtyykö hän seuraavaan huoneeseen vai jääkö hän nykyiseen sijaintiinsa. Pelaaja näkee kurkkiessaan toisessa huoneessa olevat mahdolliset pelaajat ja esineet.

Sokkelossa vaarana ovat ansoitetut huoneet, joita on noin 45 % sokkelon huoneista. Jos pelaaja kuolee, joko ansaan tai toisen pelaajan tappamana, pitää hänen odottaa tunnin verran uudestisyntymistä. Uudestisyntyessään pelaaja sijoitetaan satunnaisesti valittuun uuteen, turvalliseen huoneeseen.

3.2 Palvelin ja tietokanta

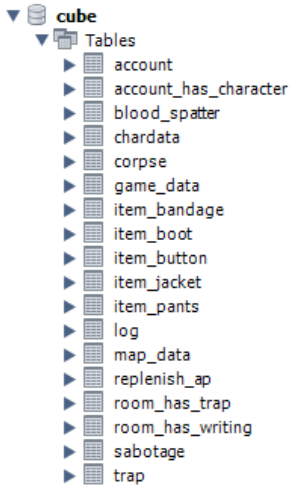
Koska projektina oli verkossa pelattava moninpeli, tarvittiin projektiin oma palvelin. LAMP-palvelin (Linux, Apache HTTP Server, MySQL, PHP), joka oli ennestään rakennettu henkilökohtaiseen käyttöön, otettiin kehitysalustaksi. Vaikka kyseinen palvelin käyttikin käyttöjärjestelmänään Linuxia, ei Linux ole millään tavalla pakollinen taikka tarpeellinen pelin käytön kannalta. Ainoastaan MySQL ja PHP olivat projektin kannalta täysin välttämättömiä.

Palvelimen toimintakuntoon asettamisen jälkeen aloitettiin pelin toimintojen ja ominaisuuksien suunnittelu peli-idean pohjalta. Näiden pohjalta laadittiin paperille yksinkertainen versio tietokannasta, joka täyttäisi tarvittavat vaatimukset. Tämän jälkeen alettiin toteuttaa tietokantaa MySQL Workbenchiä käyttäen.

Aivan ensimmäiseksi laadittiin malli käyttäen Workbenchin ”New Model” -toimintoa. Mallissa määritettiin ensimmäiseksi tietokannan nimi, jonka jälkeen aloitettiin tietokannan mallintaminen paperista versiota jäljitellen. Mallintamisen aikana tietokanta kehittyi vielä paperisesta versiosta, mutta mallintamisen jälkeen tietokannan ensimmäinen toimiva prototyyppi oli valmis (liite 1).

Malli tallennettiin SQL-tiedostona nimellä ”*cube.sql*” ja se siirrettiin palvelimelle, jolla projekti oli tarkoitus rakentaa. SQL-tiedoston sisältö luettiin tietokantaan käyttämällä MySQL-konsolia komennolla ”*source cube.sql*”.

Tietokanta oli nyt luotu tauluineen ja se oli valmiina käyttöä varten. Projektin edetessä tuli tarvetta lisätä tietokantaan vielä uusia tauluja (kuva 11) ja muokattiin jo olemassa olevia tauluja (liite 2, tietokannan lopullinen versio). Nämä muutokset tehtiin suoraan tietokantaan Workbenchiä käyttäen.



Kuva 11. Listaus pelin tauluista lopullisessa versiossa.

3.3 Toiminnallisuuden suunnittelu

Pelisovellukset yleensä toimivat siten, että pelaaja käynnistää sovelluksen, jonka jälkeen peli käynnistyy ja sen jälkeen pyörii pääsilmukassa, josta poistutaan sitten kun pelaaja lopettaa pelisessionsa. Tämä toimintapa ei toimisi tässä projektissa, sillä

PHP:tä ei ole suunniteltu erityisesti peleihin tai ohjelmiin, jotka olisivat käynnissä jatkuvasti.

Tietokannan suunnittelun jälkeen mietittiin pelin toimintalogiikka ja siinä päädyttiin seuraavanlaiseen ratkaisuun:

1. Pelaaja lataa sivun internetselaimessa
2. PHP:llä luetaan tarvittavat tiedot tietokannasta
3. Käsitellään luetut tiedot sekä pelaajan tekemät mahdolliset syötteet
4. Tallennetaan mahdolliset muutokset tietokantaan
5. Näytetään lopputulos pelaajalle

Peli myös suunniteltiin niin, että iso osa siitä olisi muokattavissa käyttäjän tarpeiden mukaan, joten esimerkiksi tiedostoon *”_constants.php*” (kuva 12) tallennettiin pelin käyttämät vakiot. Näitä arvoja voidaan helposti muuttaa tekstieditorilla pelin ulkopuolelta.

```
define("DISABLED",              " disabled='disabled'");
define("GAME_CHARACTER_LIMIT",  1);
define("GAME_DIMENSION",        26);
define("GAME_EXIT_ANSWER1",     1);
define("GAME_EXIT_ANSWER2",     0);
define("GAME_EXIT_ANSWER3",     0);
define("GAME_EXIT_ANSWER4",     1);
define("GAME_EXIT_ANSWER5",     1);
define("GAME_EXIT_QUESTIONS",   5);
define("GAME_ADDITIONAL_AP",    1);
define("GAME_ADDITIONAL_HP",    2);
define("GAME_MAX_AP",           50);
define("GAME_MAX_HP",           30);
define("GAME_MAX_LEVEL",        11);
define("GAME_MAX_LEVEL_STEP",   25);
define("GAME_MAX_XP",           250);
```

Kuva 12. Tiedoston ”*_constants.php*” sisältö. Kuvassa näkyy esimerkiksi arvo ”*GAME_DIMENSION*”, joka määrittää sokkelon koon. ”*GAME_MAX_HP*” määrittää pelaajan terveyspisteiden enimmäismäärän ja niin edelleen.

3.4 Ohjelmointi

PHP-ohjelmointikieli oli projektin olennaisin osa. Tietokantana olisi voitu käyttää lähes mitä hyvänsä tietokantaohjelmistoa, mutta PHP:n osaaminen oli projektin onnistumisen kannalta kaikkein tärkein.

Koska peli on ominaisuuksiltaan varsin monimutkainen, tehtiin päätös, että ohjelmointi toteutetaan käyttäen luokkia. Luokkien käyttö tekee koodista selkeämpää ja ohjelmointi helpottuu.

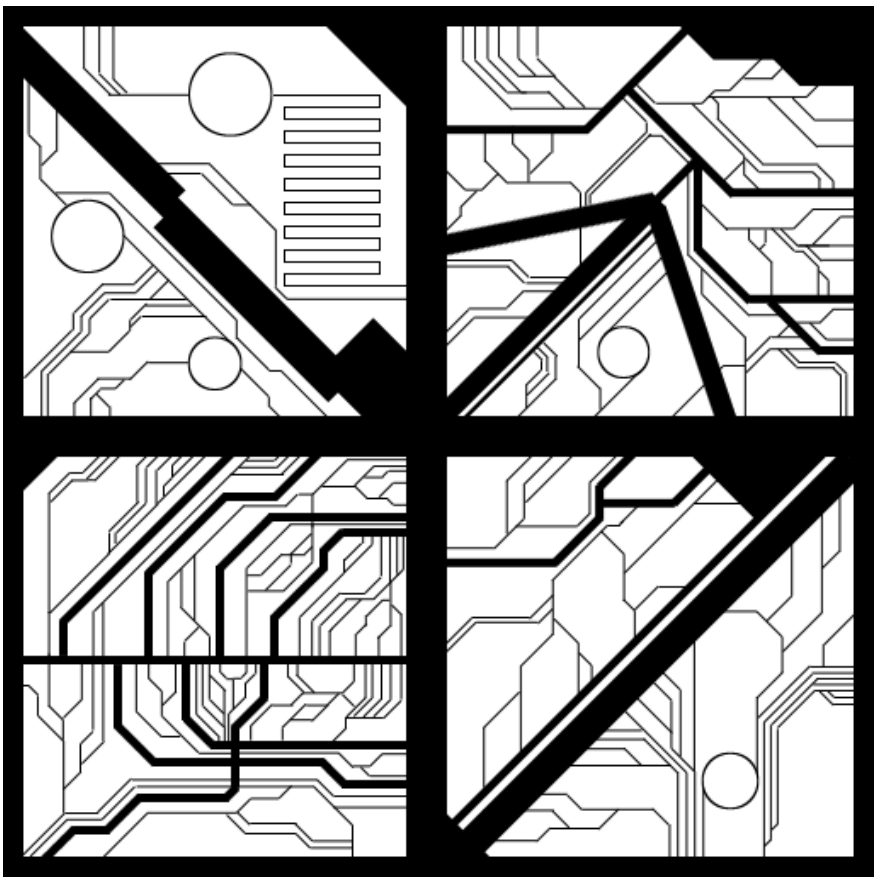
Ohjelmointi aloitettiin luomalla ensin MySQL-workbenchiä käyttäen testipelaaja ”*chardata*”-tauluun (kappale 4.2). Tämän jälkeen luotiin hyvin yksinkertainen prototyyppi *Player*-luokasta (kappale 3.7), jolla pelaajan sijaintikoordinaatteja pystyi muuttamaan. Ohjelmointi eteni käyttäen apuna pelin tietokannan mallia, miettien, kuinka mikäkin ominaisuus toimisi ja kuinka se tulisi toteuttaa.

3.5 Grafiikan tuottaminen

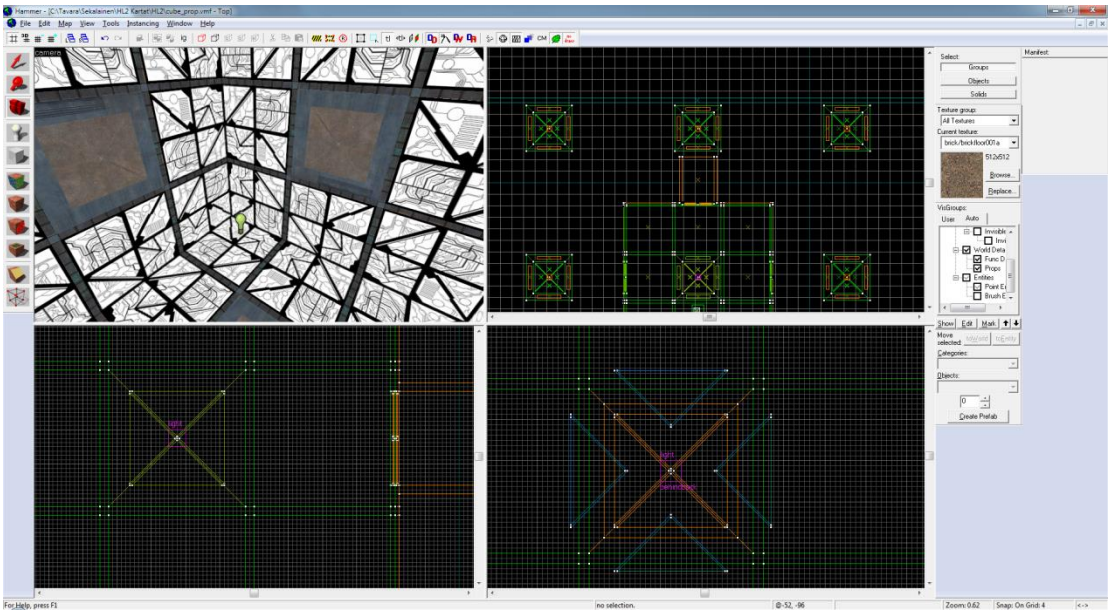
Vaikka projektissa ei paljon grafiikkaa tarvittukaan, oli silti tarvetta antaa pelaajalle edes hieman visuaalista palautetta pelin ja pelaajan tilasta. Graafikkoa ei ollut saatavilla, eivätkä taidot riittäneet 3D-mallinnukseen, mutta jostain oli saatava grafiikkaa projektiin, vaikka grafiikka olisikin staattista ja yksinkertaista. Tässä avuksi tuli Adobe Photoshop, Valve Hammer Editor sekä Valven Half-Life 2 -peli.

Käyttäen Adobe Photoshop -kuvankäsittelyohjelmaa piirrettiin peruspiirtotyökaluilla yksinkertainen tekstuuri (kuva 13), jota muokaten luotiin vielä toinen projektissa tarvittava tekstuuri. Kaikki muut tarvittavat tekstuurit löytyivät Half-Life 2:sta ennestään.

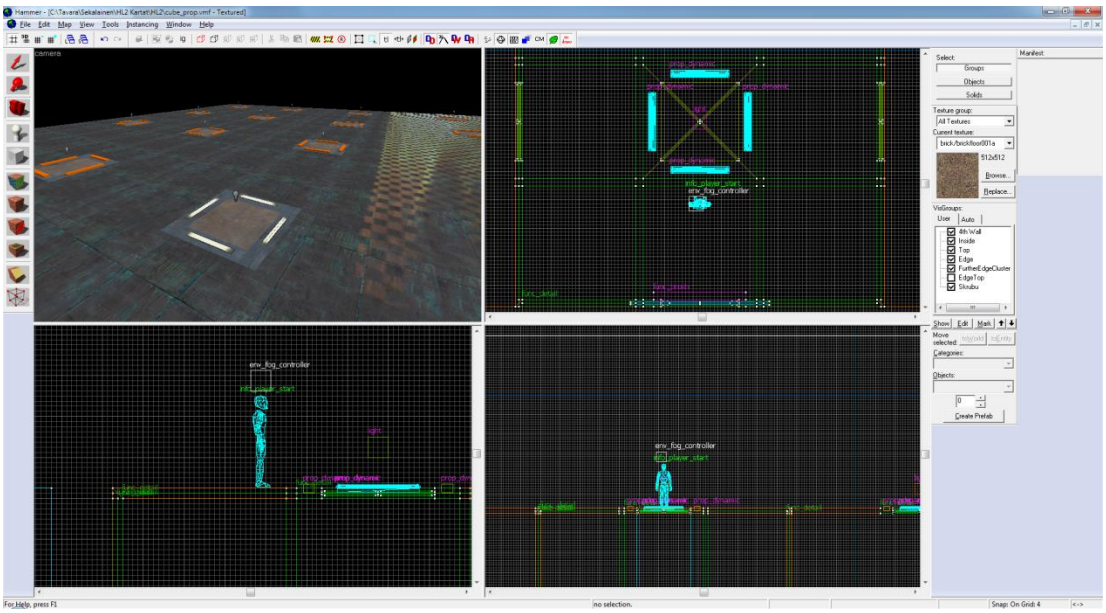
Valve Hammer Editor on kartanluontityökalu Valven Half-Life 2 -peliin. Hammer Editorilla luotiin kolmiulotteiset lavasteet, jotka esittivät yksittäistä huonetta (kuva 14) ja sokkelon kattopuolta (kuva 15). Kun kenttä lavasteineen oli valmis, avattiin kenttä Half-Life 2 -pelissä ja valmiista tuotoksesta otettiin kuvakaappaus (kuva 16), jota projektissa sitten saattoi käyttää.



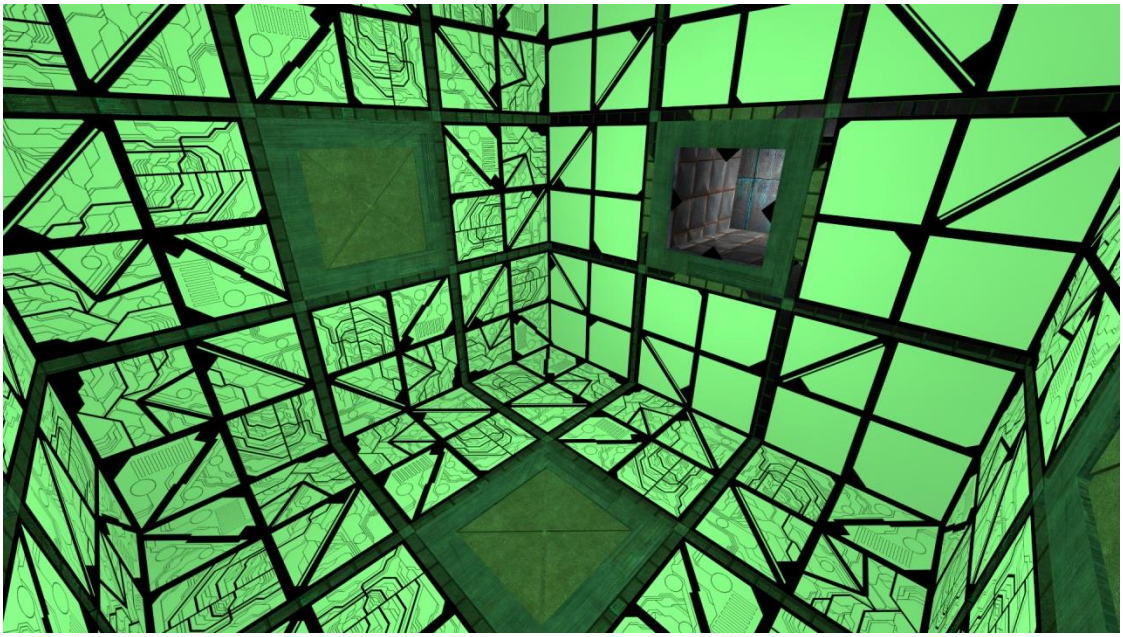
Kuva 13. Photoshopilla luotu pelissä käytetty tekstuuri.



Kuva 14. Huonenäkymä Hammer Editorissa.



Kuva 15. Kattonäkymä Hammer Editorissa.



Kuva 16. Kuvakaappaus huoneesta Half-Life 2:ssa.

Muu yksinkertainen grafiikka, kuten erilaiset ikonit ynnä muut, luotiin alusta asti Adobe Photoshopilla piirtäen.

3.6 Versionhallinnan käyttöönotto

Apache Subversion otettiin projektissa käyttöön vasta loppupuolella, sillä sitä ennen kohdepalvelimella otettiin joka aamu varmuuskopiot kaikista projektin tiedostoista, jolloin mahdollinen häviökään ei olisi ikinä vienyt kuin yhden päivän työt, mutta

työskentelytahdin kiihtyessä loppua kohden tehtiin päätös siirtyä versionhallinnan käyttöön.

Tämä myös mahdollisti sen, että tiedostoja ei tarvinnut muokata suoraan internet-yhteyden ylitse vaan tiedostoja saattoi muokata paikallisesti, riippumatta siitä oliko internet-yhteyttä saatavilla vai ei. Kun mahdolliset muutokset oli tehty, ja internetyhteys oli käytettävissä, siirrettiin tiedostot versionhallintaan, jonka jälkeen palvelimella sijaitsevat tiedostot päivittyivät automaattisesti.

3.7 Keskeiset luokat

Peli käyttää toimiakseen monia erilaisia luokkia, joista keskeisimmät ovat:

- Player
- Room
- Item ja ItemList
- GUI
- MessageBuffer
- User
- WorldGenerator
- ActionHandler
- Installer

Player-luokka pitää sisällään kaiken toiminnallisuuden, mikä pelaajahahmoon liittyy. Pelin toiminta rakentuu täysin tähän luokkaan. Luokkaan kuuluu keskeisiä toimintoja, kuten hahmon liikuttaminen, taistelu muiden hahmojen kanssa ja ansan torjuminen.

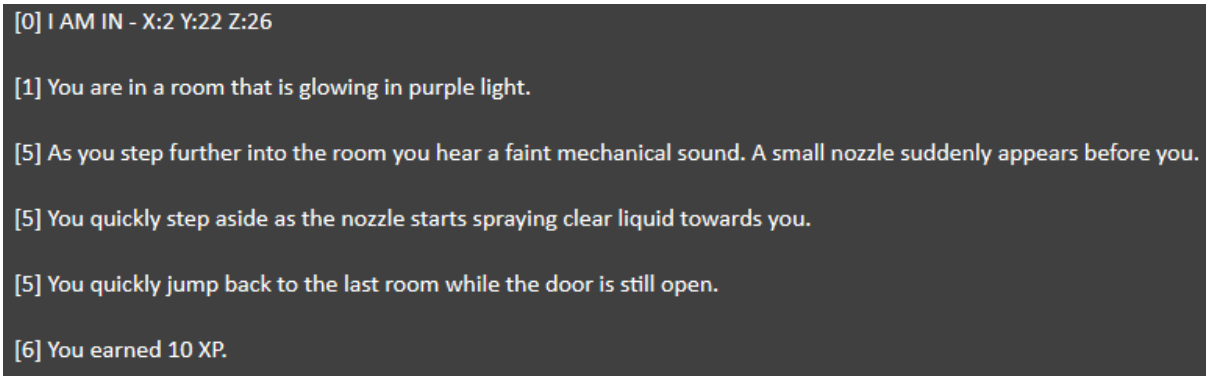
Room-luokka pitää sisällään kaiken huoneisiin liittyvät toiminnallisuuden. Luokan kautta muun muassa aktivoidaan ansat ja listataan huoneessa sijaitsevat esineet ja pelaajat.

Item-luokka on abstrakti luokka, joka sisältää pelin esineiden (kappale 4.4) toiminnallisuuden. Abstrakti luokka tässä tapauksessa tarkoittaa sitä, että pelissä ei voi olla esinettä, joka olisi pelkkä ”*Item*”, aivan kuten oikeassa elämässä ei voi olla esinettä joka, olisi vain ja ainoastaan esine, vaan on olemassa erityyppisiä esineitä (lamppu, kirja ja niin edelleen) Pelin esineet käyttävät siis omaa luokkaansa, jotka perivät *Item*-luokan.

ItemList-luokka taas toimii tavaraluettelona, sekä pelaajalle että huoneessa oleville esineille ja se sisältää *Item*-tyyppisiä luokkia.

GUI-luokka sisältää pelinäkymän käyttöliittymään liittyvän toiminnallisuuden ja huolehtii, siitä että oikeat painikkeet näkyvät pelaajalle oikeaan aikaan. Luokan nimi on harhaanjohtava, sillä GUI:n vastuulla on vain ja ainoastaan pelinäkymä itsessään, eikä esimerkiksi karttanäkymä.

MessageBuffer eli viestipuskuriluokka toimii yhdessä GUI-luokan kanssa ja sisältää kaikki pelaajalle näytettäviin viesteihin liittyvän toiminnallisuuden (kuva 17). Jokaiselle viestille voidaan määrittää oma niin sanottu kanava, joten viestit näkyvät aina oikeassa järjestyksessä pelaajalle riippumatta siitä, missä järjestyksessä asiat oikeasti tapahtuvat ohjelmassa.



Kuva 17. MessageBuffer-luokan käyttöä debug-tilassa. Numerot viestien edellä näyttävät, mihin kanavaan viestit kuuluvat.

User-luokka pitää sisällään kaikki käyttäjään liittyvät toiminnallisuuden, esimerkiksi mikä hahmo on aktiivinen, uuden käyttäjätilin luonti, salasanan vaihto sekä sisään- ja uloskirjautuminen.

WorldGenerator-luokka on vastuussa pelimaailman luonnista. Kun uusi pelimaailma luodaan, kaikki vanhat pelimaailman tiedot poistetaan ja peliin arvotaan uudet ansahuoneet, uusi poistumishuone ja niin edelleen. Kaikki pelaajat asetetaan oletustilaan ja heille arvotaan uusi sijainti.

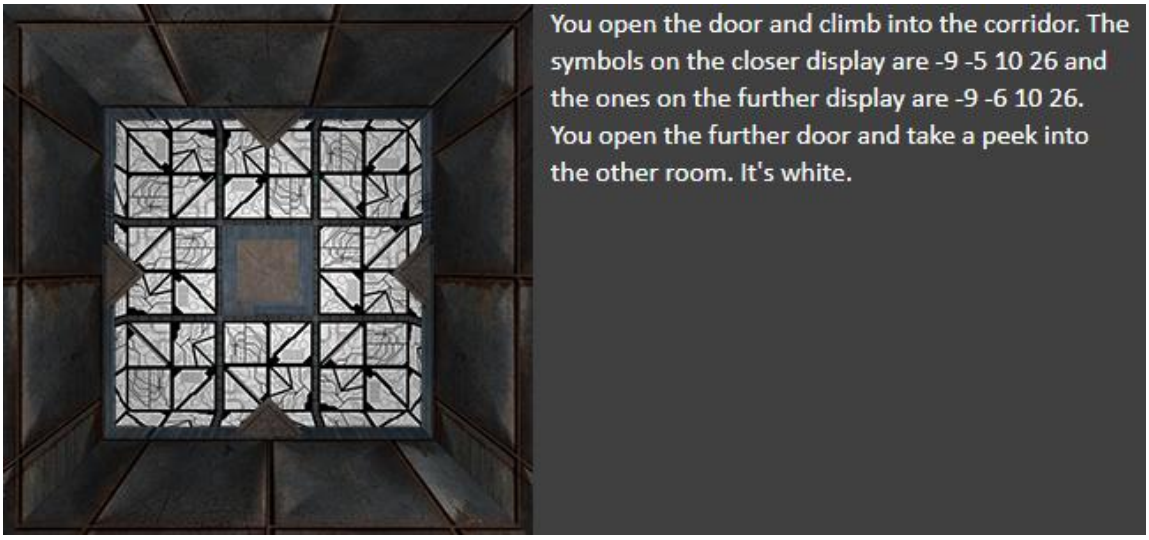
ActionHandler-luokka huolehtii käyttäjän tekemien syötteiden käsittelemisestä ja oikeellisuudesta. Sen tehtävänä on varmistaa, ettei pelaaja pääse huijaamaan pelissä ja että pelaajan toiminnot välittyvät pelille oikein.

Installer-luokan vastuulla on pelin asentaminen järjestelmään. Se huolehtii muun muassa siitä, että järjestelmästä löytyvät tarvittavat ohjelmistoversiot ja että tietokannan asetukset ovat kohdillaan.

4 PELIN KUVAUS

4.1 Tavoite

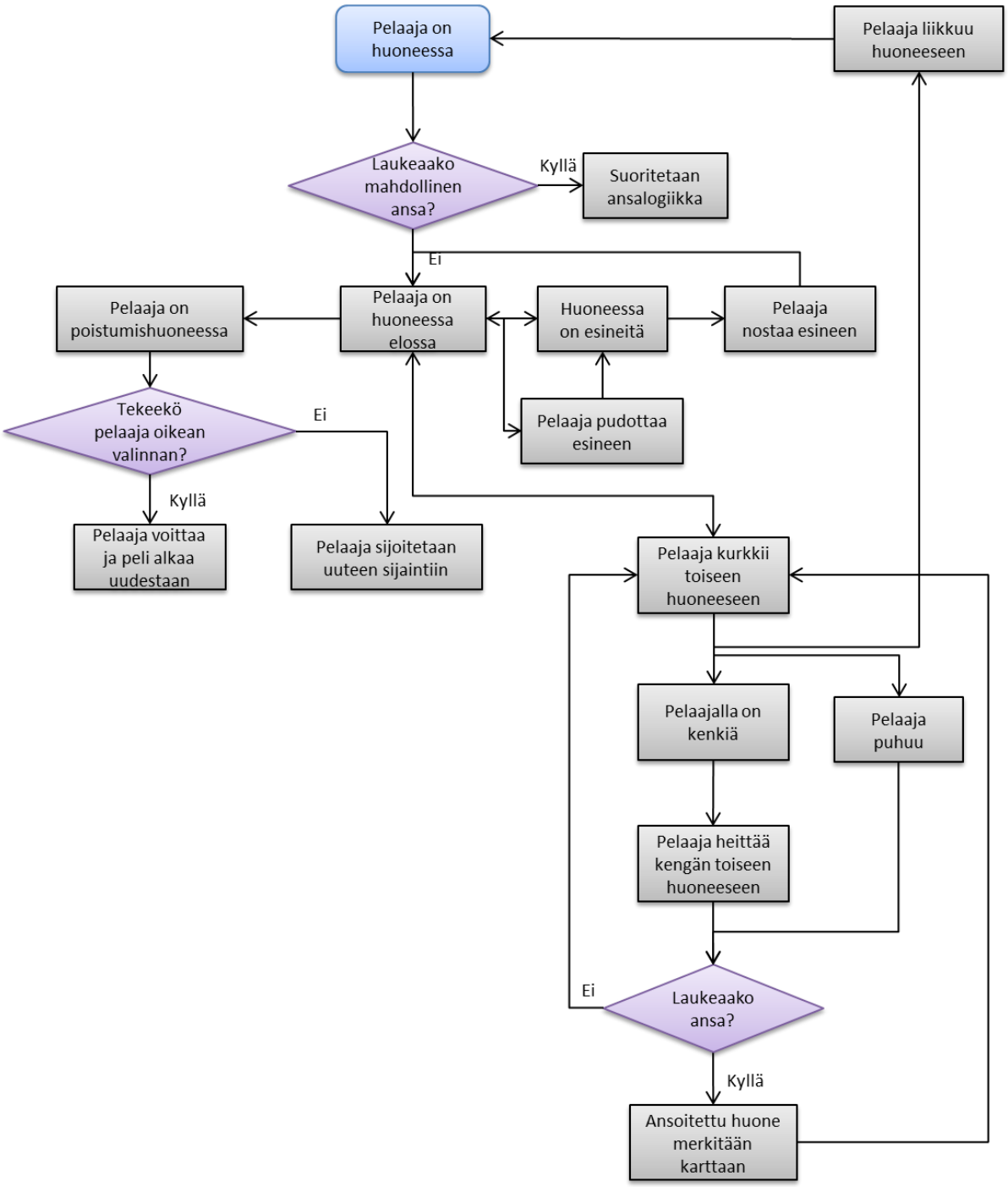
Pelin tavoite on selvitä elossa satunnaisesti arvottuun poistumishuoneeseen (kappale 4.8). Pelaaja voi nähdä oman etäisyytensä poistumishuoneeseen huoneiden välisistä koordinaattinäytöistä (kuva 18). Koordinaattinäytöt kertovat huoneen sijainnin X-, Y- ja Z-koordinaattien muodossa, ja viimeinen luku ilmaisee kuinka monen huoneen päässä pelaaja on poistumishuoneesta.



Kuva 18. Koordinaattinäytöille ei ole omaa grafiikkaa vaan ne esitetään tekstikuvauksena pelaajalle. Tässä tapauksessa molemmista huoneista on noin 26 huoneen matka poistumishuoneeseen.

Jos pelaaja selviytyy poistumishuoneeseen, kysytään pelaajalta kysymys, johon pelaajan pitää vastata joko kyllä tai ei. Kysymys valitaan satunnaisesti kysymyspoolista ja jokaisella kysymyksellä on oma oikea vastauksensa. Oikein vastaamalla pelaaja voittaa pelin ja väärin vastaamalla pelaaja sijoitetaan satunnaiseen, mutta turvalliseen huoneeseen sokkelossa.

Pelilogiikka on yksinkertaistettuna vuokaaviona kuvassa 19.



Kuva 19. Yksinkertaistettu kaavio pelin logiikasta.

4.2 Pelaajan ominaisuudet

Pelaajalla on kolme erityisesti pelaajalle itselleen tärkeää ominaisuutta: osumapisteet, toimintapisteet ja kokemuspisteet. Osumapisteet kuvaavat pelaajan terveyttä ja pelaaja on elossa niin kauan kuin osumapisteet ovat yli nollan. Oletuksena osumapisteiden enimmäismäärä on 30, mutta ansaitsemalla tasoja osumapisteiden enimmäismäärä nousee.

Pelaajalla voi olla kaksi eri loukkaantumistilaa, loukkaantunut ja vakavasti loukkaantunut. Molemmissa tiloissa pelaaja jättää verijälkiä ympäristöönsä sekä pelaaja menettää osumapisteitä tehdessään jonkun toiminnon, mutta vakavasti loukkaantunut menettää enemmän osumapisteitä. Pelaaja voi poistaa loukkaantumistilan käyttämällä käärei-tä (kappale 4.4) sekä palauttaa osumapisteitään.

Toimintapisteet kuvaavat pelaajan fyysistä jaksamista. Suurin osa pelaajan tekemistä toiminnoista kuluttavat toimintapisteitä. Jos toimintapisteet ovat nollassa, ei pelaaja voi tehdä juuri mitään ja esimerkiksi ansaan astuminen ilman osumapisteitä johtaa pelaajan varmaan kuolemaan. Oletuksena pelaajalla on 50 toimintapistettä, mutta enimmäismäärä nousee ansaitsemalla tasoja. Toimintapisteet palautuvat yhden pisteen verran aina puolen tunnin välein.

Kokemuspisteet kuvaavat kirjaimellisesti pelaajan kokemuksia sokkelossa. Keräämällä 25 kokemuspistettä pelaaja ansaitsee yhden tason aina tasoon 10 asti. Yksi taso lisää aina hieman pelaajan terveyspisteiden ja toimintapisteiden enimmäismäärää sekä pelaajan ansanväistömahdollisuuksia (”*trap dodge modifier*” kuvassa 20).

Character info for Seppo	
Sex	Male
Level	8
Experience points	185 / 250
Max AP	50 + 7
Max HP	30 + 14
Trap dodge modifier	35 %
Victories	0
Deaths	14
Murders	1
Traps dodged	23
Boots thrown	84
Equipment	You are wearing a jacket with the name "Seppo" written on it. You are wearing pants. You are wearing boots. You are carrying nothing.

Kuva 20. Pelaajan tiedot pelistä käsin tarkasteltuna.

Kokemuspisteitä voi ansaita lisää väistämällä onnistuneesti ansoja, selvittämällä onko huone ansoitettu sekä taistelemalla muita pelaajia vastaan (kappale 4.5).

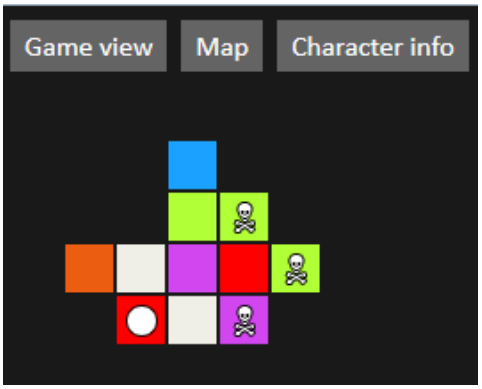
Pelaajan tiedot on tallennettu tietokantaan ”*chardata”*-tauluun (kuva 21).

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
idchardata	INT(11)	☑	☑	☐	☐	☐	☐	☑	
name	VARCHAR(50)	☐	☐	☐	☐	☐	☐	☐	NULL
description	VARCHAR(500)	☐	☐	☐	☐	☐	☐	☐	NULL
respawn_time	VARCHAR(20)	☐	☐	☐	☐	☐	☐	☐	'0'
alive	TINYINT(1)	☐	☐	☐	☐	☐	☐	☐	'1'
sex	TINYINT(1)	☐	☐	☐	☐	☐	☐	☐	'0'
hit_points	INT(11)	☐	☐	☐	☐	☐	☐	☐	'0'
action_points	INT(11)	☐	☐	☐	☐	☐	☐	☐	'0'
xp	INT(11)	☐	☐	☐	☐	☐	☐	☐	'0'
injury_counter	INT(11)	☐	☐	☐	☐	☐	☐	☐	'0'
injury_level	INT(11)	☐	☐	☐	☐	☐	☐	☐	'0'
x	INT(11)	☐	☐	☐	☐	☐	☐	☐	'-1'
y	INT(11)	☐	☐	☐	☐	☐	☐	☐	'-1'
z	INT(11)	☐	☐	☐	☐	☐	☐	☐	'-1'
scrambler	INT(11)	☐	☐	☐	☐	☐	☐	☐	'0'
state	INT(11)	☐	☐	☐	☐	☐	☐	☐	'0'
peek_x	INT(11)	☐	☐	☐	☐	☐	☐	☐	'-1'
peek_y	INT(11)	☐	☐	☐	☐	☐	☐	☐	'-1'
peek_z	INT(11)	☐	☐	☐	☐	☐	☐	☐	'-1'
direction	INT(11)	☐	☐	☐	☐	☐	☐	☐	'0'
last_action	INT(11)	☐	☐	☐	☐	☐	☐	☐	'-1'
last_target	INT(11)	☐	☐	☐	☐	☐	☐	☐	'0'
last_attackmethod	INT(11)	☐	☐	☐	☐	☐	☐	☐	'0'
exit_question	INT(11)	☐	☐	☐	☐	☐	☐	☐	'-1'
boots_thrown	INT(11)	☐	☐	☐	☐	☐	☐	☐	'0'
deaths	INT(11)	☐	☐	☐	☐	☐	☐	☐	'0'
murders	INT(11)	☐	☐	☐	☐	☐	☐	☐	'0'
traps_dodged	INT(11)	☐	☐	☐	☐	☐	☐	☐	'0'
victories	INT(11)	☐	☐	☐	☐	☐	☐	☐	'0'
flags	INT(11)	☐	☐	☐	☐	☐	☐	☐	'0'

Kuva 21. "*chardata*"-taulu MySQL Workbenchissä.

4.3 Kartta

Pelissä on sisäinen kartta, johon tallennetaan kaikki huoneet joissa pelaaja on käynyt (kuva 22). Karttaan tallennetaan huoneen väri sekä tieto huoneen mahdollisesta ansoituksesta.



Kuva 22. Karttanäkymä. Valkoinen ympyrä tarkoittaa pelaajan tämänhetkistä sijaintia, pääkallot merkitsevät ansoitettuja huoneita.

Kartta säilyy pelaajalla pelaajan kuolemankin jälkeen, mutta kartta voi hävitä pelaajalta, jos pelaaja astuu tietyn tyyppiseen ansaan. Samoin, jos joku pelaaja onnistuu voittamaan (kappale 4.8) pyyhkiytyy kartta jokaiselta pelaajalta, koska peli alkaa alusta.

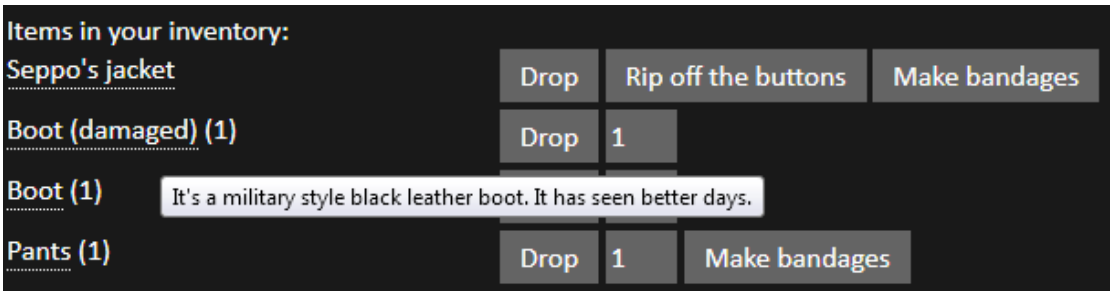
Tietokannassa karttatiedot ovat tallennettu ”*map_data*”-tauluun (kuva 23).

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
idmap_data	INT(11)	☑	☑	☐	☐	☐	☐	☑	
idchardata	INT(11)	☑	☑	☐	☐	☐	☐	☐	
x	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
y	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
z	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
flags	INT(11)	☐	☐	☐	☐	☐	☐	☐	'0'
		☐	☐	☐	☐	☐	☐	☐	

Kuva 23. Taulu "*map_data*”.

4.4 Esineet

Esineet ovat tärkeä osa pelissä selviytymistä. Oletuksena pelaajalla on yksi takki, yhdet housut sekä kahdet kengät. Pelaaja saa oletusesineet aina tavaraluetteloonsa (kuva 24) synnyttyään uudelleen sekä silloin, kun hahmo ensimmäisen kerran luodaan.



Kuva 24. Pelaajan tavaraluettelo.

Pelaaja voi tehdä sekä takista että housuista kääreitä, joilla pelaaja voi parantaa itseään, mikäli hän on loukkaantunut. Takista taas voi repiä irti napit, joilla pelaaja voi kaivertaa viestejä peliympäristöön muille pelaajille.

Takki kuuluu aina jollekin pelaajalle, ja siinä lukee aina alkuperäisen omistajan nimi. Takin voi repiä viideksi kääreeksi ja/tai siitä voi repiä neljä nappia pois. Takki myös suojaa tietyiltä ansoilta, jos pelaaja ei ansaa onnistu väistämään. Tämä johtaa aina takin tuhoutumiseen.

Housut toimivat muutoin samoin kuin takki, mutta housuissa ei lue pelaajan nimeä eikä siitä ole mahdollisesti irrottaa nappeja. Housuista saa revittyä kolme käärettä.

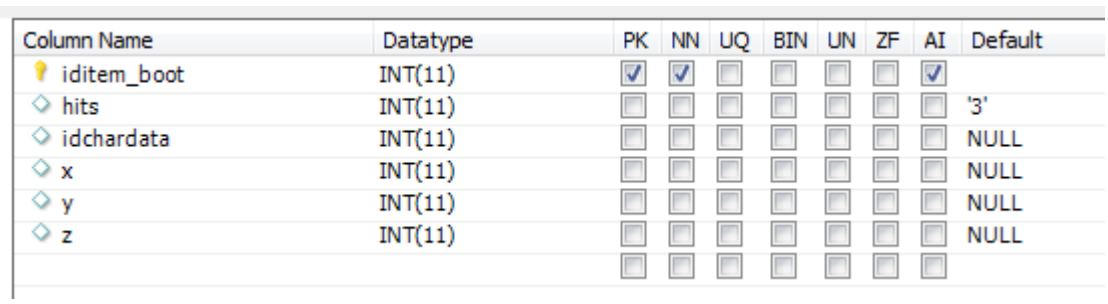
Kenkiä pelaaja voi käyttää selvittääkseen onko huone mahdollisesti ansoitettu. Pelaaja voi heittää kengän kengännauhoista kiinni pitäen toiseen huoneeseen ja mahdollisesti laukaista ansan. Jos kenkä laukaisee ansan, kenkä vahingoittuu. Yksi kenkä voi vahingoittua kolme kertaa, kunnes siitä tulee käyttökelvoton. Kenkä pysyy pelaajan tavaraluettelossa kunnes se tuhoutuu käyttökelvottomaksi, jolloin se jää siihen huoneeseen, johon se on viimeiseksi heitetty.

Mikäli pelaajalla on tavaraluettelossaan ainakin kaksi käyttökelpoista kenkää (vähintään yhden vahingoittumisen päässä käyttökelvottomuudesta), suojaavat kengät mahdollisesti tietyiltä ansoilta. Kengät eivät vahingoitu itse ansasta.

Kääreillä pelaaja voi parantaa itseään, jos pelaajan osumapisteet ovat alle enimmäismäärän ja/tai pelaajalla on erityinen ”loukkaantunut” tila.

Napeilla pelaaja voi kaivertaa viestejä ympäristöönsä muille pelaajille tai itselleen. Pelaaja näkee aina, mitkä viestit hän on itse kirjoittanut, mutta muiden viestit näkyvät vain muodossa ”joku on kirjoittanut.” Jotta huoneet eivät aivan täyttyisi pelaajien mahdollisista viesteistä, voi yhtä nappia käyttää vain kerran.

Jokaisella esinetyypillä on oma taulunsa tietokannassa. Kaikki taulut ovat identtisiä siten, että jokaisella esineeltä löytyvät arvot jotka kuvaavat X-,Y- ja Z-koordinaatteja sekä omistajaa, mutta esineestä riippuen taulusta löytyy myös erityisarvoja. Esimerkiksi kenkien tiedot tallennetaan tauluun ”*item_boot*” (kuva 25), josta löytyy esimerkiksi ”*hits*”-arvo, joka kuvaa kuinka paljon vahinkoa kenkä kestää.

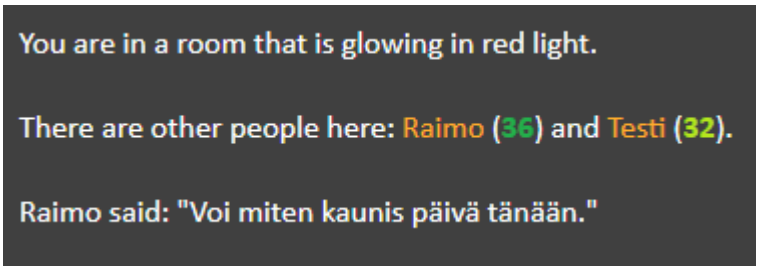


Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
iditem_boot	INT(11)	☑	☑	☐	☐	☐	☐	☑	
hits	INT(11)	☐	☐	☐	☐	☐	☐	☐	'3'
idchardata	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
x	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
y	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
z	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	

Kuva 25. Taulu *item_boot*

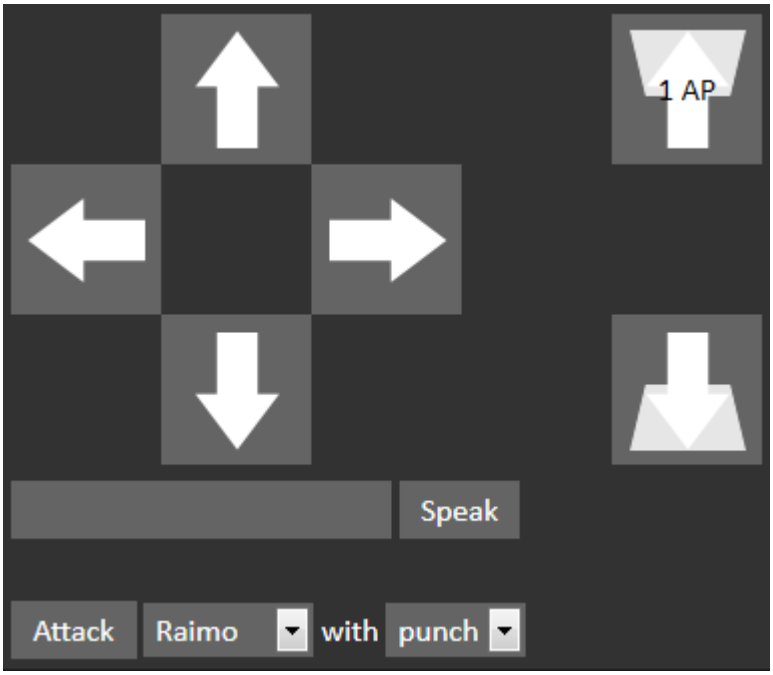
4.5 Vuorovaikutus muiden pelaajien kanssa

Pelaajat voivat jättää viestejä toisilleen kirjoittamalla niitä pelimaailmaan ja pelaajat voivat myös puhua toisilleen sekä taistella keskenään. Jos pelaaja puhuu muiden pelaajien läsnä ollessa, tallennetaan pelaajan sanoma viesti väliaikaiseen ”log”-tauluun tietokannassa. Kun nämä toiset pelaajat lataavat sivun, luetaan ”log”-taulusta pelaajan viesti ja näytetään se pelaajalle (kuva 26). Näin pelaajat näkevät, mitä toinen pelaaja on samassa huoneessa sanonut, vaikka edellisestä pelisessiosta olisi useita päiviä.



Kuva 26. Toinen pelaaja on puhunut samassa huoneessa oleville pelaajille.

Olennainen osa pelaajien kanssakäyntiä on taistelu. Jos samassa huoneessa on muita pelaajia, pelaajille näkyy erityinen taisteluvalikko (kuva 27).

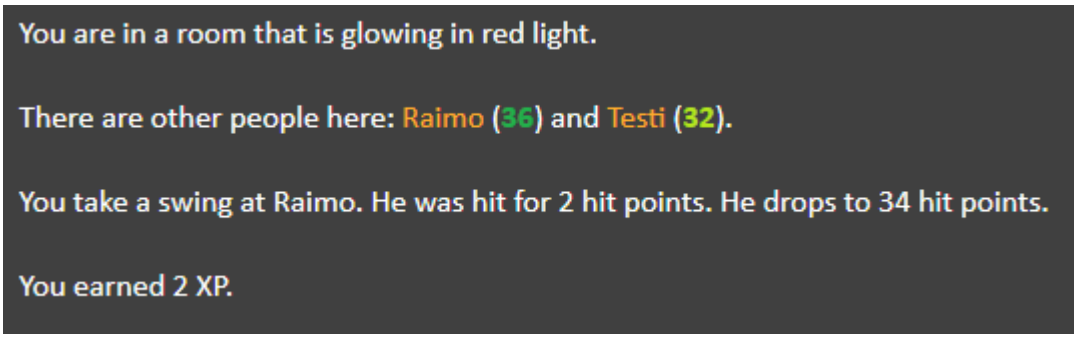


Kuva 27. Taistelukäyttöliittymä.

Pelaaja valitsee valikosta kohteen sekä tavan, jolla hyökätä. Hyökkäystapoja on kaksi, lyönti (kuva 28) ja potku. Lyönnillä on 50 %:n osumamahdollisuus, ja osuessaan se

vie kohteelta kaksi osumapistettä. Potkulla sen sijaan on 33 % osumamahdollisuus, mutta se vie 4 osumapistettä. Hyökkäys, osui se tai ei, vie aina yhden toimintapisteen.

Pelaajalla on 5 %:n mahdollisuus iskeä kriittisesti, jolloin pelaajan tekemä vahinko kaksinkertaistuu. Jos vastapelaajan osumapisteet laskevat nollaan tai siitä alle, vastapelaaja kuolee ja hyökkäävä pelaaja saa merkinnän murhasta omiin tietoihinsa.



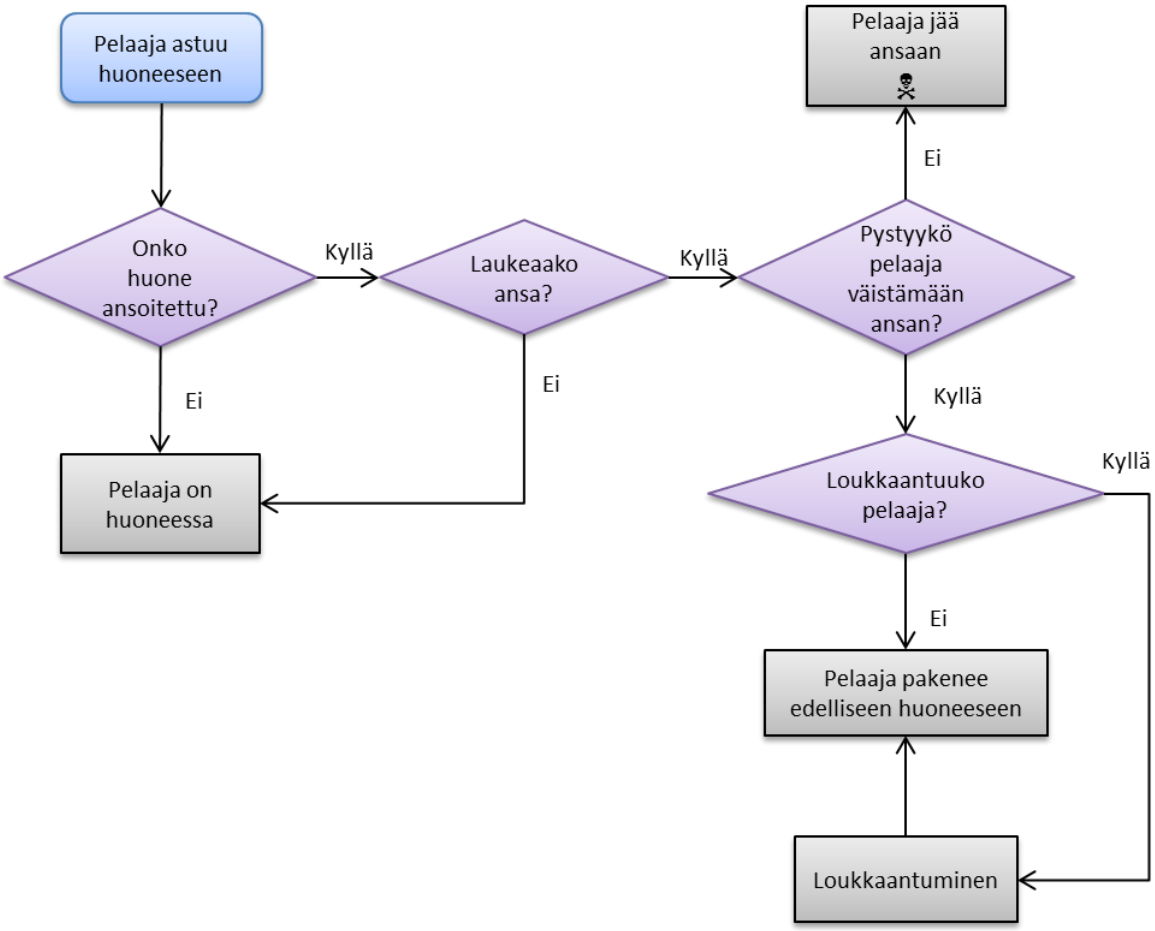
Kuva 28. Aktiivinen pelaaja lyö pelaajaa Raimo.

Puolustavan pelaajan pitää itse aktiivisesti iskeä takaisin tai paeta tilanteesta, jos pelaaja on kirjautunut pelistä ulos jää hänen hahmonsa seisomaan puolustuskyvyttömänä paikalleen.

4.6 Ansat

Ansoitettujen huoneiden sijainti on arvottu satunnaisesti ja samoin niiden sisältämä ansa. Pelaajan astuessa uuteen huoneeseen tutkitaan aina sisältääkö kyseinen huone ansan (kuva 29). Jos huone on ansoitettu, pelaaja laukaisee ansan ja tutkitaan selviääkö pelaaja ansasta.

Pelaaja voi myös laukaista tietyt ansat kurkkiessaan ansoitettuun huoneeseen ja suorittaessaan sopivan toiminnon ansasta riippuen.



Kuva 29. Ansan toimintalogiikka

Suurin osa ansoista on tappavia, mutta jotkut esimerkiksi pelkästään poistavat pelaajalta osumapisteitä tappamatta tai tuhoavat pelaajan kartan.

Pelaajan oma väistömahdollisuus ynnätään yhteen ansan väistömahdollisuuden kanssa, jolloin pelaajat, joilla on paljon kokemustasoja selviävät etenkin tietyistä ansoista hyvin usein. Koodissa on kuitenkin määrätty selviämismahdollisuuden ylärajaksi 90 %, eli vaikka ansan oma selviytymismahdollisuus olisi 30 % ja pelaajan oma selviytymismahdollisuus olisi 65 %, pyöristyisi se 90 %:iin. Sekä ansan ja pelaajan väistömahdollisuus sivuutetaan, jos pelaaja pudottautuu ansahuoneeseen yläpuolelta. Tuolloin väistömahdollisuus on 0 %.

Tietokannassa erityinen taulu ”*room_has_trap*” (kuva 30) kertoo pelille, missä koordinaateissa mikäkin ansa sijaitsee.

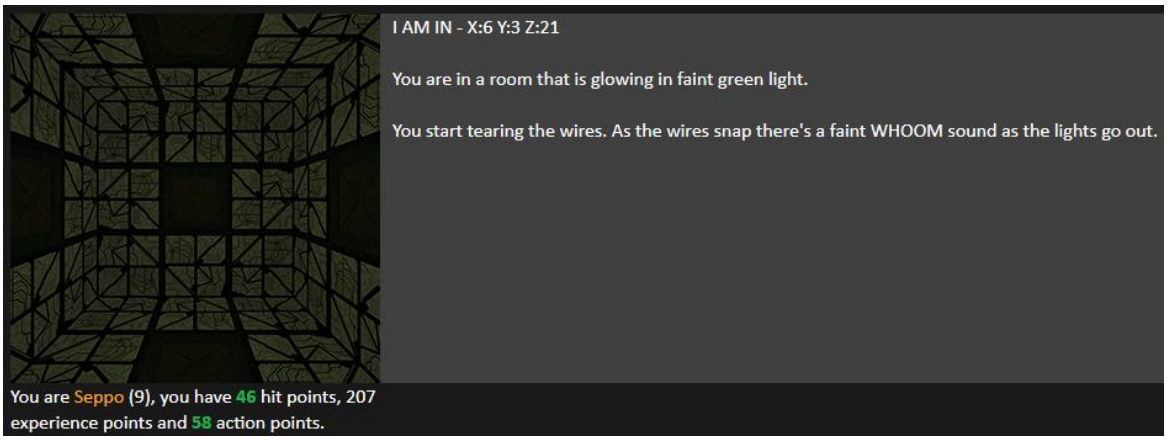
Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
idroom_has_trap	INT(11)	☑	☑	☐	☐	☐	☐	☐	
idtrap	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
x	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
y	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
z	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	

Kuva 30. Taulu "*room_has_trap*".

4.7 Sabotaasi

Pelimaailmaan luodaan WorldGenerator-luokkaa käyttäen niin kutsuttuja sabotaasihuoneita *sivu*^{2} määrä, eli oletuksena 676 huonetta (3 % kaikista sokkelon huoneista). Sabotaasihuoneessa pelaajat pystyvät halutessaan katkaisemaan sokkelosta sähköt suorittamalla sabotaasin, joka onnistuu aina, mutta voi tappaa pelaajan 50 % mahdollisuudella.

Pelaajan suoritettua sabotaasin, arpoo peli satunnaisluvun lukujen 5 ja 60 väliltä, joka on sähkökatkotilan kesto minuuteissa. Sähkökatkotilassa jokainen sokkelon huone on pimeä (kuva 31) eivätkä ansat sekä koordinaattinäytöt toimi. Sokkelossa voi siis navigoida vapaasti ilman vaaraa ansoista, mutta koska koordinaattinäytöt eivät toimi, vaikeutuu myös poistumishuoneen löytyminen.



Kuva 31. Pelaaja on suorittanut sabotaasin, aiheuttaen sähkökatkon.

Pelaaja ei voi itse tietää kauanko sähkökatkotila kestää, joten sähköjen tullessa takaisin voi pelaaja joutua suoraan ansaan.

Jos sokkelo on jo sähkökatkotilassa pelaajan yrittäessä suorittaa sabotaasia, arpoo peli satunnaisluvun lukujen -30 ja 30 väliltä, ja tämä aika lisätään sähkökatkotilan aikaan. Käytännössä se tarkoittaa, että jos sähkökatkotilaa on ollut jäljellä 45 minuuttia niin uuden sabotaasin jälkeen sitä voi olla jäljellä 15 tai 75 minuuttia.

4.8 Voittaminen

WorldGenerator-luokka luo pelimaailmaan erityisen poistumishuoneen, jonka sijainti arvotaan satunnaisesti, kuitenkin siten että poistumishuone on vähintään neljän huoneen päässä reunasta. Huone ympäröidään ansoitetuilla huoneilla, yhtä satunnaisesti valittua huonetta lukuun ottamatta, joten poistumishuoneeseen on mahdollista kulkea astumatta ansaan.

Poistumishuone on ulkoasultaan täysin erilainen kuin muut huoneet. Huoneeseen astuessa pelaaja siirretään ulos pelimaailmasta, eikä pelaaja pysty enää vuorovaikutukseen muiden pelaajien kanssa.

Pelaajalle arvotaan satunnainen kysymys, johon on tietty vastaus. Jos pelaaja vastaa väärin, pelaaja siirretään takaisin pelimaailmaan satunnaiseen, mutta turvalliseen, huoneeseen. Jos taas pelaaja vastaa oikein, pelaajalle merkitään voitto hahmon profiiliin ja pelaaja saa yhden kokemustason lisää, edellyttäen että pelaajan taso on alle 10. Pelimaailma luodaan uusiksi ja kaikkien hahmojen osuma- ja toimintapisteet palautetaan täysille.

4.9 Lokalisointi

Yksi projektin tavoitteista oli, että peli olisi helposti lokalisoitavissa. Tätä varten suunniteltiin, että pelin kaikki tekstit ladattaisiin aina yhdestä tietystä tiedostosta, riippuen pelaajan valitsemasta kielestä. Näin kaikki tekstit löytyvät helposti yhdestä paikasta ja tiedoston voi helposti antaa kielitaitoiselle henkilölle käännettäväksi, ilman että hänen täytyy tietää mitään PHP-kielestä tai ohjelmoinnista.

Vaikka projektin lopussa pelissä olikin vain yksi kieli, amerikanenglanti, ei työ ollut turhaa, sillä mahdolliset kirjoitusvirheet ja ajatusvirheet on helppo korjata, kun tarvittavat tekstit löytyvät yhdestä paikasta, eikä tekstejä tarvitse etsiä jokaisesta PHP-tiedostosta erikseen.

5 PÄÄTELMÄT JA JATKOSUUNNITELMAT

Projekti onnistui ylitse odotusten vaikka toisinaan työ oli haastavaa. Esimerkiksi esineitä hallitseva *Item*-luokka ja muut siihen liittyvät luokat kirjoitettiin projektin aikana kolme kertaa uusiksi, ja *Player*-luokasta pilkottiin suuri palanen *Room*-luokkaan. Vaikka esineiden toimintalogiikka on nyt tyydyttävä, on suunnitelmissa ollut kirjoittaa koko toimintalogiikka uusiksi vielä neljännen kerran, mutta koska kaikki toistaiseksi toimii tyydyttävästi, ei sille vielä ole akuuttia tarvetta.

Omia ongelmia projektiin toivat myös versiopäivitykset. Projektia aloitettaessa oli käytössä PHP:n versio 4, joka kuitenkin myöhemmin päivittyi versioon 5. Versiossa 5 MySQL-rajapinta oli uusittu täysin, joten kaikki tietokantaa käsittelevät komennot piti kirjoittaa uusiksi version 5 mukaan. Tämä ei onneksi ollut hankalaa, mutta vei kyllä työtunteja, jotka olisi voitu käyttää paremminkin.

Peli on kuitenkin täysin toimintakunnossa ja sen voi, kuten tavoitteena oli, asentaa kohtuullisen helposti mille tahansa laitteelle, mistä löytyvät oikeat PHP- ja MySQL-versiot. Vaikka projekti onkin saatettu loppuun tavoitteet täyttäen, ei se todennäköisesti koskaan tule saavuttamaan tasoa, jossa se olisi täysin valmis, vaan aina löytyy parannettavaa, lisättävää ja korjattavaa.

Yksi tulevaisuudensuunnitelma on avata peli yleisölle ja kehittää peliä muiden pelaajien mahdollisten ehdotusten mukaan, sekä mahdollisesti ottaa mukaan lisää kehittäjiä.

LÄHTEET

1. Apache Software Foundation. About the Apache HTTP Server Project. Saatavissa: http://httpd.apache.org/ABOUT_APACHE.html [viitattu 1.6.2014]

2. Shannon, R. What is HTML? Saatavissa: http://www.yourhtmlsource.com/starthere/whatishtml.html [viitattu 21.5.2014]

3. Oracle Corporation. About MySQL. Saatavissa: http://www.mysql.com/about [viitattu 14.5.2014]

4. The PHP Group. PHP General Information. Saatavissa: http://fi1.php.net/manual/en/faq.general.php [viitattu 14.5.2014]

5. IT Business Edge. SQLCourse – Lesson 1: What is SQL? Saatavissa http://www.sqlcourse.com/intro.html [viitattu 21.5.2014]

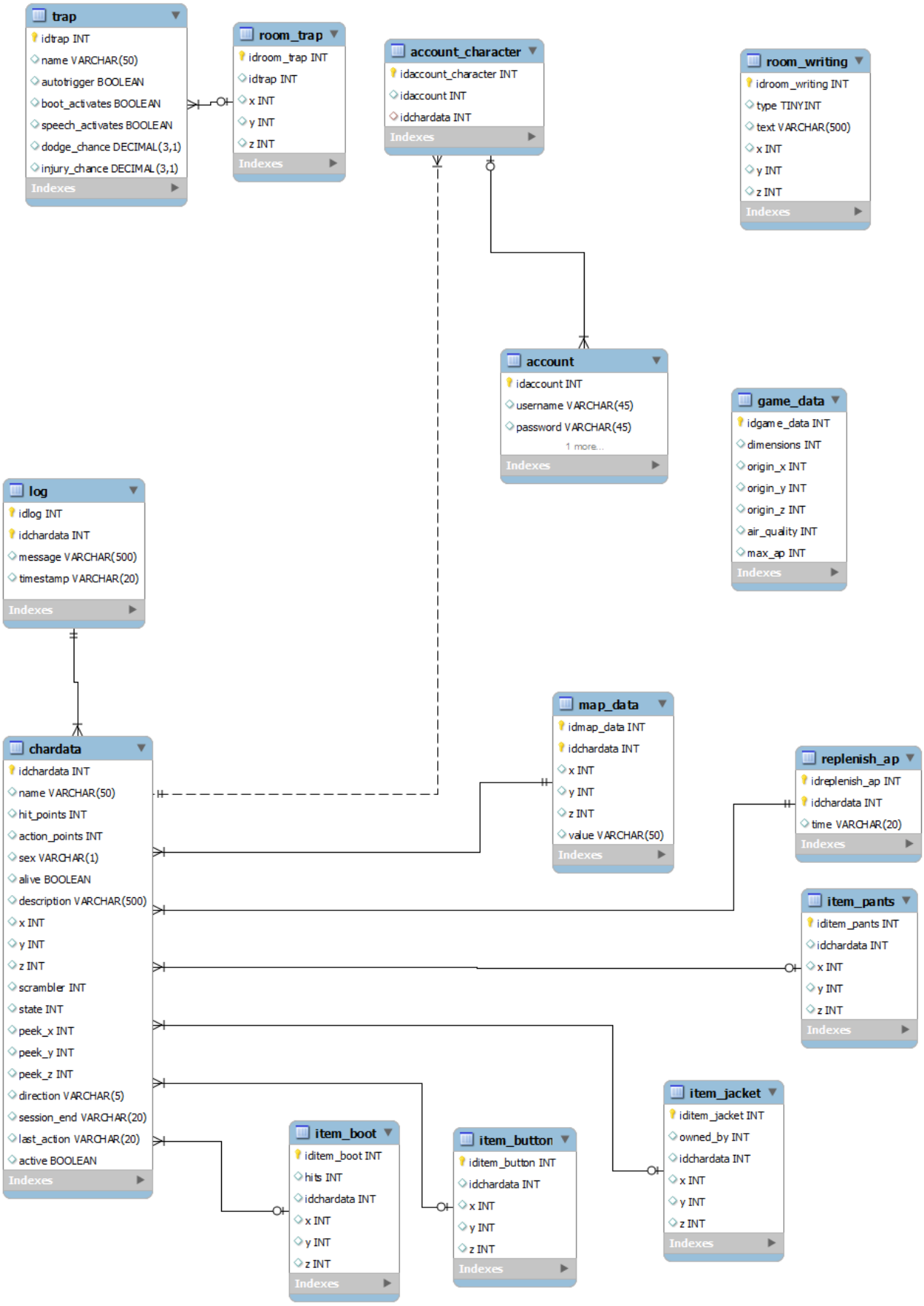
6. Bowlby, Selene M. 6 Phases of Website Design and Development Process: http://www.idesignstudios.com/blog/web-design/phases-web-design-development-process [viitattu 16.6.2014]

7. Chapple, Mike. What is a Database? Saatavissa: http://databases.about.com/od/specificproducts/a/whatisadatabase.htm [viitattu 14.6.2014]

8. Ho, Don. Notepad++ Saatavissa: http://notepad-plus-plus.org [viitattu 25.5.2014]

9. Apache Software Foundation. Apache Subversion. Saatavissa: http://subversion.apache.org [viitattu 16.5.2014]

Tietokannan ensimmäinen versio



Tietokannan lopullinen versio