Opinnäytetyö (AMK / YAMK)

Tietotekniikan Koulutusohjelma

NTIETS12P

2016

Oskari Rautala

TIETOKANTOJEN KÄYTTÖ PELEISSÄ

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK / YAMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka / Peliteknologia

2016 | 30

Reetta Raitoharju ja Mika Luimula

Oskari Rautala

TIETOKANTOJEN KÄYTTÖ PELEISSÄ

Tämä opinnäytetyö tarkastelee, miten ja miksi tietokantoja voidaan käyttää pelisovelluksissa. Työssä käydään läpi tiedon tallennustapoja peleissä ja näihin liittyviä yleisiä standardeja. Työssä tutkittiin myös, mitä pelisuunnittelussa pitää ottaa huomioon, kun halutaan suunnitella pelejä käyttäjäkohtaisesti.

Käytännön työssä rakennettiin tekeillä olevaan peliin järjestelmä, jossa peli hakee erilaisia pelikriteereitä palvelimelta ja muokkaa pelikokemusta näiden pohjalta. Tämän jälkeen peli lähettää pelin tulokset takaisin palvelimelle. Työssä tehtiin myös käyttöliittymä, josta näitä tietoja voidaan muokata ja tuloksia katsoa.

Tämä järjestelmä rakennettiin Linux palvelimelle, johon oli asennettu MySQL tietokanta tietojen tallennukseen. Peli otti yhteyden tähän palveluun PHP-skriptien kautta. Tämä järjestelmä toimi prototyyppinä, jota halutessa voitaisiin jatko kehittää.

Tämän kaltainen järjestelmä toimii hyvin erikoistilanteissa, esimerkiksi jos peliä käytetään osana tieteellistä tutkimusta. Järjestelmä on kuitenkin erittäin vaativa rakentaa kokonaan tietokonepohjaisesti, ja helpoin tapa käyttää sitä on ihmisten ohjatessa.

ASIASANAT:

tietokannat, pelit, palvelimet, tietoturva, pelisuunnittelu

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology / Game Technology

2016 | 30

Reetta Raitoharju and Mika Luimula

Oskari Rautala

USING DATABASES IN GAMES

This thesis looks into how and why databases could be used to save data in games. The theoretical part of the thesis examines different methods of data saving in games and general standards of data saving. In addition, it studies how to design user-centric games and what needs to be considered when designing such games.

The first objective of the practical work in this thesis was to design a system that receives player-specific criteria from a server and changes the game according to these criteria. After this, the game sends its results back to the server for analysis. A second objective of the practical work was to create a graphical web-based UI, so this data can be viewed easily.

The System designed for this work was a Linux server, with a MySQL database for storing the data. The game connected to the server through custom-made PHP scripts. This system was functional as a prototype, which was the project requirement.

In conclusion, this type of system works well in special cases, for example using a game as a part of a scientific study. However, this kind approach is resource- intensive, unless the designers can create a program that can customize a game experience thoroughly without human assistance.

KEYWORDS:

databases, games, servers, security, game design

SISÄLTÖ

KUVAT

KUVIOT

KÄYTETYT LYHENTEET JA SANASTO

API	API, tai Application Programming Interface, on kasa erilaisia rutiineja, protokollia ja työkaluja mitä ohjelmoija voi käyttää oman sovelluksensa tukena. Nämä ovat yleensä jossain verkkopalvelimella (mutta niiden ei ole pakko olla), jolloin niitä voivat käyttää kuka vaan, tai tietty rajattu yleisö.
Hex	Hex on tiedostoformaatti, jota käytetään yleisesti pelien tallennukseen.
HTTP(S)	HTTP, tai Hypertext Transfer Protocol on teknologia, jota käytetään pohjana erilaisiin verkkoteknologioihin. Se on määritelmä, miten viestejä lähetetään internetin välityksellä. Sen toinen muoto, HTTPS, toimii samalla tavalla, mutta siihen on lisätty myös salaus.
MySQL	MySQL on eniten käytetty tietokanta ohjelma. Se on ilmainen ohjelmisto, joka vastaanottaa tietoja käyttäen SQL-kyselykieltä ja tallentaa ne tietokannan muotoon.
OAUTH	OAUTH on yleinen standardi verkossa tunnistautumiseen. Tätä standardia käytetään yleensä johonkin palveluun kirjautumiseen käyttäen yleisempiä tunnuksia (esim. Google tili).
PlayFab	Playfab on yritys, joka tarjoaa erilaisia palveluita pelinkehittäjille. Suurimpina palveluina ovat erilaiset verkkopalvelut, joita käyttäen pelinkehittäjät voivat säästää paljon omaa aikaansa ja rahaa.
PHP	PHP: Hypertext Preprocessor, yleinen verkkoteknologiassa käytettävä palvelinpohjainen ohjelmointikieli.
SQL	Structured Query Language, kyselykieli, jota käytetään kommunikointiin tietokantojen kanssa.
Steam	Digitaalinen pelien jälleenmyyjä. Steamillä on isoin markkinaosuus tietokoneella digitaalisesti myydyistä peleistä.
XML	”Extensible Markup Language”, merkintäkieli, jota käytetään erilaisiin tiedonsiirron sovelluksiin.

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena on tarkastella tietokantojen käyttöä pelinkehityksessä. Tietokantoja ei käytetä peleissä usein, mutta niitä voidaan tarvita tietynlaisissa peleissä. Käytännön työssä tehtiin kehityksen alla olevaan peliin systeemi, jolla pelikokemusta voidaan muokata pelin ulkopuolella toimivan tietokannan kautta. Toimeksiantaja tälle työlle toimi Turun Ammattikorkeakoulu. Työ tehtiin peliä kehittävän ryhmän kanssa yhteistyössä, ja peliin ohjelmointiin osia, joilla peli voi lähettää ja vastaanottaa palvelimelta haluttuja tietoja. Työhön tehtiin myös verkkokäyttöliittymän, jonka avulla näitä tietoja voidaan muokata. Työssä tutkittiin lisäksi, miten tämän kaltainen järjestelmä voidaan turvata.

Opinnäytetyön teoreettisessa osuudessa käydään läpi mitä peleissä tallennetaan ja millä tavoilla se toteutetaan. Työssä tarkastellaan myös erilaisia huomioitavia poikkeustilanteita liittyen näihin tietoihin, ja miten tietoturva tulee huomioida tällaisten tietojen käsittelyssä. Tämän jälkeen työssä esitellään miten näitä erilaisia tietoja voidaan käyttää hyväksi pelinkehityksessä, ja mitä se vaatii pelisuunnittelulta. Työssä käydään myös läpi erilaisia tapoja pelin muokkaamiseksi reaali-ajassa vaikeustason tai tunnelman suhteen.

Lopuksi opinnäytetyö tarkastelee, miten hyvin tällaisen järjestelmän yhdistäminen peliin toimii ja milloin on kannattavaa harkita tällaista järjestelmää. Tämän lisäksi tuodaan esille tällaisen järjestelmän suurimmat heikkoudet ja vahvuudet.

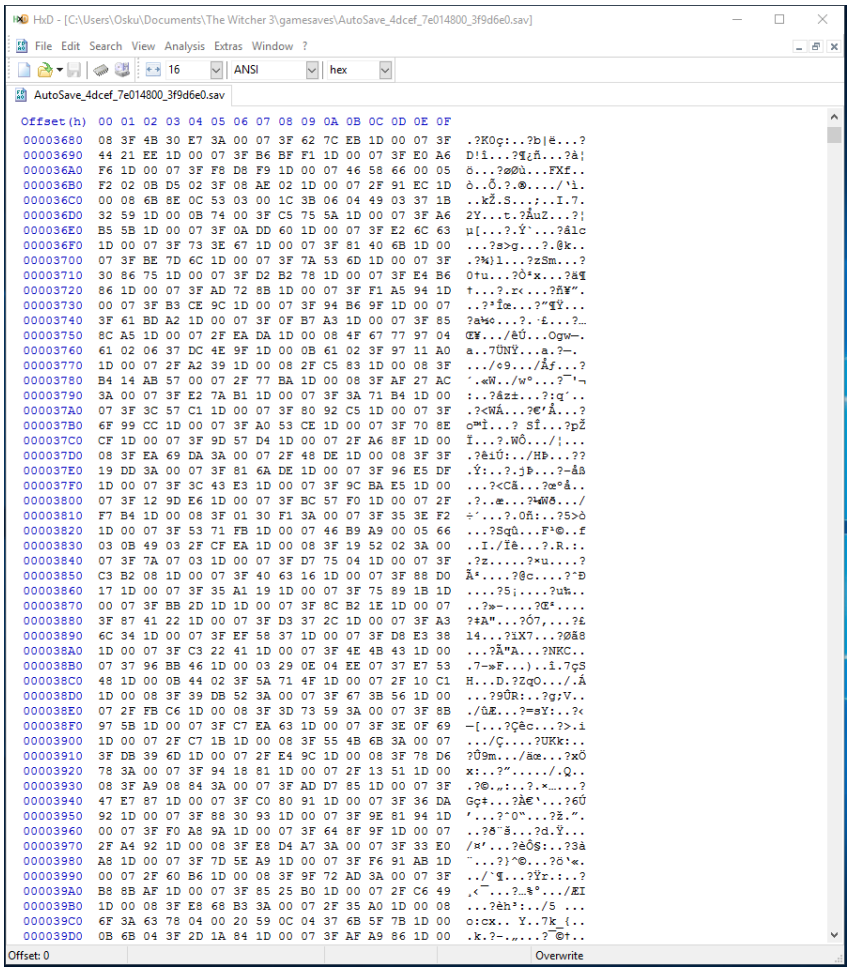
2 TIETOJEN TALLENNUS PELEISSÄ

Tietojen tallennus on tärkeä osa pelisuunnittelua, koska sillä on suuri vaikutus pelaajan mielentilaan ja/tai odotuksiin. Yleisesti tietyn tyyliseltä peliltä odotetaan tietynlaista tapaa tallentaa, kuten esimerkiksi ajopelin oletetaan yleensä tallentavan, kun ajettu kisa on ohi tai kun pelaaja on lopettanut autonsa muokkailun. Pelaaja eivät kuitenkaan yleisesti odota, että peli on tallentanut kisan puolenvälin tuloksen, jos peli sammutetaan keskellä kisaa. Tätä oletusta on yleensä hyvä seurata, jottei pelaaja koe pettymyksen tunnetta, jos peli rikkoo tällaista yleissääntöä. Peleissä jotka eivät seuraa yleisiä olettamuksia tallennuksen suhteen, on pelaajalle hyvä esitellä pelin tallennustapa pelin alussa, jottei tällaista pettymystä synny.

Peleissä on myös yleensä oletus, että jos pelaaja lopettaa pelaamisen tallentamatta tilannetta, peli unohtaa kaiken saavutetun viime tallennuksen jälkeen. Kun pelaaja lataa pelin uudestaan, peli jatkuu taas edellisestä tallennuskohdasta. Tämä voi olla huono asia pelaajalle, jos tallentaminen on pelaajan vastuulla, ja pelaaja oli unohtanut tehdä sen. Mutta se voi myös olla tapa antaa pelaajien pelata huolehtimatta tekojensa seurauksista, koska he tietävät voivansa palata turvalliseen pelitilanteeseen lataamalla vanhan tallennuksensa. Jotkin nykypelit ovat myös alkaneet tehdä kokeiluja tällä konseptilla johtaen tilanteeseen, jossa peli voikin kommentoida pelaajalle asioita joita on tapahtunut tallentamisen jälkeen. Yksiä huomattavampia nykyesimerkkejä tästä on peli nimeltä Undertale, jossa peli muistaa paljon erilaisia asioita, jotka ovat tapahtuneet pelaajan tallennuksen jälkeen. Yleisimpiä esimerkkejä mitä tästä pelistä esitellään, on pelin ensimmäinen isompi taistelu vaikeampaa ja mahdollisesti merkittävämpää vastustajaa vastaan, jossa peli on suunniteltu niin, että pelaajan on helppo tappaa tämä vastustaja vahingossa. Jos he tulevat katumapäälle nähdessään mitä tapahtuu, voivat he päättää ladata tallennuksensa pisteestä ennen kuin he tappoivat tämän pomo hirviön, ja yrittää tätä tappelua uudestaan. Jos he onnistuvat, pian tämän tapahtuman jälkeen toinen hahmo kyseenalaistaa pelaajan toiminnan, kysymällä pelaajalta katuuko hän tappamistaan, kun ei hänelle kelvannutkaan sen lopputulos. Tämänkaltainen pelaajien olettamuksien rikkominen voi olla erittäin vaikuttava tapa saada pelaajan kiinnostus, mutta se voi myös huonosti suunniteltuna olla vaarallista pelin illuusion säilymiselle.

2.1 Pelien eri tallennustarpeet

Kun suunnitellaan pelin tallennussysteemiä, pitää myös miettiä mikä on pelin kannalta olennaista tallentaa. Yleensä nykyjärjestelmissä on vaikea luoda liian isoa tallennustiedostoa, mutta pienempi tiedosto on kuitenkin nopeampi käsitellä pelissä, jotta tallennukseen ei kulu liian paljon aikaa. Tallennuksen kestoon vaikuttaa myös tiedon määrä, joka ei ole suoraan pelin muistissa. Mitä enemmän tietoa pitää johtaa tai laskea pelin muistissa olevista asioista, sitä pidempi tallennusprosessi on. Tallennustiedosto myös yleensä tallennetaan hankalasti luettavassa muodossa, jotta tietojen muokkaaminen ei ole liian helppoa. Kuvassa 1 nähdään esimerkin tästä, Witcher 3 -pelin tallennustiedoston, joka on tallennettu vaikeammin luettavana hex-tiedostona.



Kuva 1. Malliesimerkki tallennustiedostosta Witcher 3 -pelistä.

Tallennustiedoston sisältö vaihtelee paljon riippuen pelin tyypistä. Peleissä, joissa on jonkinlainen pysyvä pelimaailma eli maailma jossa pelaajan teoilla on pysyviä vaikutuksia, kuten esimerkiksi seikkailu- ja toimintapeleissä, niin tallennuksessa pitää huomioida pelaajan paikka, jotta peli jatkuu samasta paikasta, jossa pelaaja tallensi pelinsä. Tämä voi olla vaikka tallennuspiste, jossa pelaaja viimeksi tallensi tai tarkka pelimaailman tilanne, jos pelissä voi tallentaa milloin vain (tai peli tallentaa itse automaattisesti ajan mukaan). Pelitilanteeseen sisältyy myös esimerkiksi mahdollisesti kuolleet vihollliset sekä elävien vihollisten tilanne ja paikka, kerätyt esineet, mutta se voi myös olla täydellinen kopio pelitason tilanteesta, jos sellaista pelissä tarvitaan. Toisena vaihtoehtona näiden pysyvien pelimaailmojen peleille ovat erilaiset valikkopohjaiset pelit. Tällaisissa peleissä pelikentät palautetaan aina samaan tilaan, ja tärkeimmät muutokset tapahtuvat pelien jälkeen tai valikoissa, kuten esimerkiksi ajo- ja urheilupelit. Näissä peleissä pelimaailman tilanne ei ole niin tärkeä asia, jolloin peli tallentaa yleensä vain, kun pelaaja tekee asioita valikoissa tai lopettaa jonkin tapahtuman. Esimerkiksi autoilupeli voi tallentaa pelaajan tilanteen ennen kilpailun alkamista ja kilpailun päättymisen jälkeen. Poikkeuksena voi olla sellainen tilanne, jossa pelaaja voi poistua kilpailusta kesken, ja palata siihen myöhemmin, jolloin peli tallentaa juuri senhetkisen pelitilanteen.

Peleissä, joissa on erilaisia keskusteluja, pitää pelin yleensä myös huomioida mitkä keskustelut pelaaja on jo käynyt, ja tarvittaessa myös, minkälaisia valintoja pelaaja teki näissä keskusteluissa. Tähän samaan kategoriaan voidaan myös ottaa erilaiset välivideot, jotka pelaaja on jo nähnyt, jotta pelaaja ei tahtomattaan joudu näkemään samaa videota joka kerta kun hän lataa tallennuksensa. Tämän kaltaisen tiedon tallennus on yleensä vaikeimpia tietoja tallentaa, sillä tämän kaltaisissa tiedoissa pelinkehittäjä joutuu tarkasti miettimään mikä on paras tapa tallentaa ja ladata tällainen määrä tietoa. Tämä johtuu siitä, että jos pelissä on tuhansia erilaisia keskusteluja, ja jos kaikista niistä pitää tallentaa kaikki tieto, tulee näiden tietojen käsittelystä hankalaa ja hidasta.

Yksi kiinnostava poikkeustapaus tallennustiedoissa on permadeath - pelit, joissa pelaajan kuoltua pelissä peli alkaa jälleen alusta ilman mitään jälkiä edellisestä suorituksesta tai hyvin vähäisellä tiedolla edellisestä suorituksesta. Tämä konsepti oli jo varhaisimmissa videopeleissä käytössä, sillä vanhat arcade-pelit suunniteltiin pitkälti tältä pohjalta, johtuen siitä että pelaajat vaihtuivat koneella jatkuvasti, ja kone keräsi myös paremmin kolikoita kun jokainen kerta aloitettiin peli alusta. Myös suurin osa lautapeleistä on suunniteltu samalta pohjalta, jossa oletuksena on, että edellisellä pelikerralla ei ole

mitään vaikutusta seuraavan kerran tulokseen. Näissä molemmissa on sama ajatus siitä että on vain yksi asia joka muuttuu peliä pelatessa, ja se on pelaajan taitotaso. Tämä johtaa helposti kilpailutilanteeseen, sillä kaikki pelaajat aloittavat samasta tilanteesta, ja vain taidolla peli voidaan suorittaa onnistuneesta. Tämän tyyliset pelit kuitenkin vähentyivät pelien kehittyessä, sillä peleihin alettiin saada tapoja tallentaa pelaajan eteneminen ja tällä tavoin saatiin luotua pidempiä ja monimutkaisempia pelitilanteita. Kuitenkin tämän kaltaiset pelit ovat saaneet uutta eloa nk. roguelike- tai roguelite-genren ansiosta. Tämä termi tulee vanhasta pelistä Rogue (Toy, et al., 1980), jossa käytettiin juuri tätä pysyvän kuoleman vaikutusta ja pelikenttiä, jotka arvottiin joka pelikerran aluksi. Nykyään näitä roguelike pelejä kehitetään markkinoille paljon, kiitos erinäisten pienten pelistudioiden, jotka ovat aiheesta innostuneet (esimerkiksi Spelunky, The Binding of Isaac ja Rogue Legacy). Näissä uudemmissa varianteissa kuitenkin tätä ideaa on kehitetty vähän eteenpäin, ja peleissä saattaa olla tiettyjä asioita, jotka pelaaja voi avata seuraavia yrityksiä varten, sen sijaan että peli alkaisi aina täysin tyhjältä pöydältä. Esimerkiksi The Binding of Isaacissa pelaaja voi avata erinäisiä käytettäviä esineitä tappamalla tiettyjä vihollisia tietyllä pelihahmolla seuraavia pelikertoja varten. Peli voi myös lisätä uusia kenttiä, kun pelissä on suoritettu tarpeeksi asioita. (Doull 2009.)

2.2 Muuttuvien tietojen käsittely

Pelaajan tilannetta ei yleensä tallenneta jatkuvasti suoraan tallennustiedostoon, vaan peli pitää tämänhetkisen tilanteen pelilaitteen muistissa, ja tallentaa sen tiedostoon vasta, kun pelaaja tai pelikehittäjä sen sallii. Tähän on muutamia erilaisia syitä. Yksi näistä on se, että pelitilanteen jatkuva tallentaminen rasittaisi yleisesti pelikonetta turhan paljon, johtaen pelin pyörimiseen hitaammin kuin kehittäjä halusi. Toisena voi olla huijauksen estäminen. Jos pelaaja voi jatkuvasti ladata pelitilanteen uudestaan ja kokeilla erilaisia vaihtoehtoja ilman seuraamuksia, voi se johtaa pelaajan turhautumiseen, tai pelikehittäjän suunnitteleman vaikeustason muuttumiseen (Novak, 2011). Kolmantena voi myös olla se mahdollisuus, että pelimaailman tämän hetkinen tilanne ei vaikuta mitenkään pelaajan yleiseen tilanteeseen, kuten esimerkiksi autoilupelissä pelaajan sijoituksella ei ole kisan keskellä väliä, koska vain kisan lopputulos huomioidaan pelaajan tiedoissa. (Moran 2010.)

Tämä johtaa siihen, että jos pelaaja haluaa muokata näitä tietoja pelin ollessa käynnissä, täytyy hänen lukea ja muokata laitteen muistia. Tällaisen huijauksen vaikeus riippuu pelinkehittäjän tekemistä suojauksista, ja näiden tietojen tarkastelusta pelissä. Tähän tarkoitukseen on erilaisia työkaluja, joiden avulla huijaaja voi selvittää, mitä tietoa peli tallentaa muistiinsa. Tämä voi myös vaatia pelin koodin kääntämistä konekielestä takaisin luettavaan muotoon, jotta huijaajan on helpompi tarkkailla minne peli tallentaa eri tietoja. Tähänkin tarkoitukseen on tarjolla monia ilmaisia ja maksullisia työkaluja. Niiden tehokkuus riippuu siitä onko pelinkehittäjä varautunut tähän mahdollisuuteen, ja suojannut koodiaan jollain tavalla. Siltikin nämä kaikki ovat aina lopulta kierrettävissä, kun huijareilla on tarpeeksi aikaa ja halua. (Hoglund & McGraw 2007.)

2.3 Verkkoyhteyksiä käyttävät pelit

Peleissä joissa pelataan muiden pelaajien kanssa yhdessä verkon kautta, tulee tiedon käsittelyyn uusia ongelmia. Yleensä yksin pelattavissa peleissä tallennustiedoston suojaus ei ole tärkeää, koska jos pelaaja haluaa huijata itse ostamassaan pelissä, se ei ole haitallista kenellekään, tai korkeintaan vain itse pelaajalle. Kun pelataan muiden ihmisten kanssa, huijaaja saa etulyöntiaseman muita pelaajia kohtaan, ja tämä tekee se pelistä epäreilun muita pelaajia kohtaan.

Edellä kerrottu tiedonkäsittelystä itse pelissä vaarallista tiedon kannalta, koska taitava huijaaja voi manipuloida pelin tietoja lähdelaitteessa, ja lähettää palvelimelle vain tietoja jotka antavat hänelle edun muita pelaajia kohtaan. Tästä johtuen verkkoyhteyksiä käyttävissä peleissä lähetettyjen tietojen mahdollisuus pitää yleensä tarkistaa pelinkehittäjän palvelimella, jotta peli ei voi lähettää mahdotonta tietoa. Tämän lisäksi on suositeltavaa suorittaa kaikki tärkeät päätökset palvelimella, joka sitten vain antaa lopputuloksen lähdelaitteelle. Kun laitteen lähettämien tietojen mahdollisuus tarkistetaan pelinkehittäjän hallitsemalla koneella, ja päätökset niistä myös tehdään siellä, on huijaaminen huomattavasti vaikeampaa. Tällaista pelaajan ja pelinkehittäjän vastakkainasettelua voi hyvin kuvata lauseella ”The client is in the hands of the enemy.” (Koster 2008).

Vaikka kaikki tiedot tarkistettaisiin palvelimella, huijaajat voivat silti käyttää palvelimelta saatuja tietoja hyväkseen. Esimerkiksi kun pelaaja kävelee pelissä päin seinää, voi palvelin lähettää pelille tiedon, että pelaajan edessä on seinä, eikä pelaaja voi liikkua tämän seinän läpi. Huijaava pelaaja voi halutessaan estää tämän tiedon saapumisen omaan peliinsä, ja jatkaa liikkumista seinän lävitse, kunnes palvelin hyväksyy pelaajan

sijainnin, kun pelaaja on kävellyt kokonaan seinän läpi. Jos tämän liikkeen mahdollisuutta ei tarkisteta kokonaan palvelimella, voi huijaajat tällä tavoin kiertää joitakin pelinkehittäjien rajoituksia. Osaavat huijaajat voivat myös manipuloida muita palvelimelta saatuja tietoja. Yleensä nopeasti liikkuvissa peleissä palvelin lähettää paljon tietoja muista pelaajista ja objekteista, koska pelaaja voi törmätä muihin pelaajiin hyvin nopeasti. Jos tätä tietoa yritetään piilottaa pelaajalta, kunnes tämä objekti tai pelaaja tulee hänen näkökenttäänsä, voi se johtaa palvelimen hidastamiseen, koska näkökenttien laskeminen jokaiselle pelaajalle on hyvin raskasta. Tämä voi myös johtaa siihen, että pelaajat voivat ilmestyä tyhjästä toistensa eteen jos yhteyksissä on pientä hidastelua. Tästä johtuen hakkerit voivat saada muiden pelaajien sijainnit palvelimelta, ja käyttää niitä hyväkseen. He voivat myös muuttaa esimerkiksi seinien tekstuurit läpinäkyviksi, jolloin he näkevät muiden pelaajien tarkat sijainnit kartassa. (Koster 2008)

Joissakin peleissä on suojaksi pelin mukana asennettu valvontaohjelma, joka yrittää estää tietojen manipuloinnin tarkkailemalla laitteen muistia ja mitä muita ohjelmia laitteella pyörii. Tämä voi estää perinteiset hyökkäystavat, mutta koska tämä ohjelma on pelaajan laitteella, voi hän halutessaan manipuloida tätä valvontaohjelmaa. Taitava hyökkääjä voi esimerkiksi katsoa mitä tämä valvontaohjelma tarkkailee, ja muuttaa omaa hyökkäystapaansa kiertämään tämä suojaus tai lähettää valvontaohjelmalle vääriä signaaleja, jotta se uskoo että kaikki tapahtuu oikein. Tämä johtaa taas siihen tilanteeseen, että pelaajan koneelta tuleviin tietoihin ei voida luottaa. (Koster 2008)

2.4 Tietojen etätallennus

Pelitiedot voidaan myös tallentaa pelintekijän tai jonkun muun tahon palvelimelle. Yleisin syy tehdä tämä on tallennustiedostojen varmuuskopiointi. Tällä tavoin vaikka pelaaja menettäisi tietokoneensa, voi hän jatkaa tallennustaan muualla. Näin pelaajat voivat myös pelata samalla tallennustiedostolla eri tietokoneilla, joutumatta manuaalisesti siirtämään tiedostoja. Yleensä tällaista palvelua ylläpitävät erilaiset virtuaaliset pelikaupat (esimerkiksi Steam, Playstation Store), jos pelinkehittäjä on tehnyt sopimuksen asiasta tämän kaupan kanssa.

Peli voivat myös kerätä pelaajalta erilaisia tietoja pelinkehittäjää varten. Nämä tiedot voivat olla suoraa palautetta pelaajilta tai vaikka pelinkehittäjien haluamia tietoja pelaajan etenemisestä, jotta pelinkehittäjä voi tarkkailla miten pitkälle pelissä eri pelaajat ovat päässeet, tai miten eri pelaajat ratkaisivat eri ongelmia, yms.. Nämä tiedot yleensä

tallennetaan pelinkehittäjän palvelimella olevaan tietokantaan, josta heidän on helppo seurata pelin tilannetta. Tähän tarkoitukseen voidaan myös käyttää erilaisissa virtuaalisissa pelikaupoissa käytettäviä saavutuksia, joita pelaajat saavat kun he pääsevät tarpeeksi pitkälle pelissä, tai saavuttavat jotain erityisen vaikeaa. Jos tiedonkeruu on erityisen laajaa tai vaikkapa sisältää tietoja pelaajan laitteesta, kannattaa yleensä asiasta ainakin ilmoittaa pelaajille, ja myös mahdollisesti antaa vaihtoehto estää tällainen tietojen kerääminen.

2.5 Tietoturva

Tietojen tallennuksessa pitää myös huomioida näiden tietoturva, varsinkin jos pelissä tarvitaan pelaajan henkilökohtaisia tietoja tai maksutietoja. Pelinkehittäjän on tärkeää miettiä, mitä tietoja pelissä tarvitsee tallentaa, ja minne. Pelinkehittäjän pitää myös miettiä tarvitseeko tieto salata, ja millä tavoin. PlayFab-niminen yritys teetti asiasta vuonna 2015 kyselyn, joka paljasti että iso osa pelaajista ei luota siihen, että pelinkehittäjät tekevät tarpeeksi suojellaksensa näitä tietoja. Kuvassa 2 nähdään osa heidän infograafistaan edellä mainitusta kyselystä (PlayFab, 2015). Tämä epäluottamus liittyy oletettavasti ainakin osittain siihen, että muutamia isoja tietovuotoja eri pelipalveluissa on jo tapahtunut ja niistä, on uutisoitu isosti mediassa esimerkiksi Sonyn tietovuoto vuonna 2011 (Sony 2011).

Kuva 2. Lukuja PlayFabin teettämästä kyselystä (PlayFab 2015).

Pelissä, jota pelaaja pelaa yksin, voidaan kyseenalaistaa, onko tietoturvan painottaminen tärkeää pelinkehitysvaiheessa. Yleensä monet pelit käyttävät muutamia pieniä eri tapoja, jotta käyttäjät eivät voi helposti muokata tallennustietoja. Suuremmat salausalgoritmit ja muut eivät yleensä ole kehitysajan arvoisia, koska näitä tietoja muokkaavat lähinnä aktiiviset fanit. Yleensä nämä tallennustiedot tallennetaan jonnekin paikalliselle kovalevylle hex-tiedostona, joka vaikeuttaa tietojen muokkaamista, mutta ei tee siitä mahdotonta. Jotkin pelinkehittäjät ovat jopa tehneet tämänkaltaisesta tiedostojen muokkaamisesta osan peliä, ja peli voi huomata, jos se huomaa pelaajan muokanneen pelin tiedostoja.

Peleissä, joissa pelataan muiden pelaajien kanssa, tiedon suojaus on kuitenkin paljon tärkeämpää. Tässä tapauksessa verkkoliikenne palvelimen tai pelaajien välillä salataan. Paras tapa suojata tällaista liikennettä on tarkistuttaa kaikki tieto pelinkehittäjän palvelimella, jolloin huijaajien virheelliset tiedot voidaan huomata. Tämä kuitenkin vaatii paljon resursseja, joita kaikilla pelinkehittäjillä ei ole. Näissä peleissä voidaan myös käyttää erilaisia tunnistautumiskeinoja, esimerkiksi pelin oma kirjautumistunnus ja salasana, tai vaihtoehtoisesti jonkun muun palvelun tunnukset ja salasana joka on linkitetty pelin cd-avaimeen. Jos huijaus havaitaan joltain tililtä, siltä tililtä voidaan poistaa oikeus pelata peliä. Tämä johtaa ainakin siihen, että huijaaja joutuu ostamaan pelin uudestaan

jatkaakseen huijaamista, ja tällä tavoin hillitsee huijaajien intoa kokeilla eri huijauksia. (Hall & Novak 2008.)

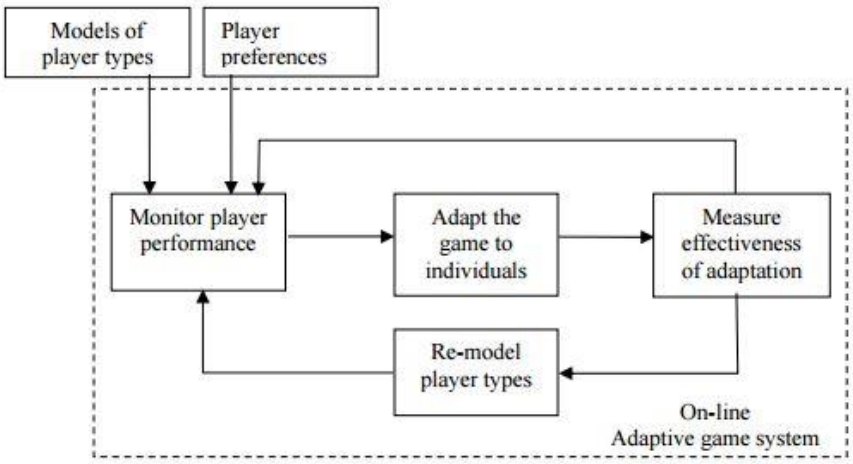
Hakkerit voivat myös tutustuttuaan pelin rakenteeseen ja sen lähettämiin viesteihin, perustaa oman nk. ”harmaan palvelimensa” pelistä, jossa osaavat pelaajat voivat pelata kyseistä peliä ilmaiseksi, tai keräten omia maksujaan pelaajilta, jotka pelaavat heidän palvelimillaan. Tällainen työ vaatii erittäin paljon selvittelyä ja kokeilua, mutta tämä tilanne on tapahtunut jo monien pelien osalta (esimerkiksi World of Warcraft ja Everquest). Sen estäminen on myös hyvin vaikeaa varsinkin, jos hakkerit onnistuvat saamaan yhden version pelistä pyörimään palvelimillaan. Myös näiden harmaiden palvelimien lopettaminen on vaikeaa, koska ne yleensä sijaitsevat fyysisesti maissa, joissa tekijänoikeus- ja tietoturvalait ovat enemmän hakkerien puolella. Erityisesti pienet pelialan yrityksen eivät pysty käymään kalliita oikeidenkäyntejä hakkereita vastaan. Viimeisin tämän kaltainen tilanne nousi uutisotsikoihin, kun Blizzard pakotti yhden suurimmista harmaista palvelimista sulkeutumaan (Polygon 2016). (Hall & Novak, 2008.)

3 PELIN MUOKKAAMINEN KÄYTTÄJÄKOHTAISESTI

Pelit ovat edelleen monille ihmisille vaikeasti lähestyttäviä, koska verrattuna muuhun mediaan peleissä on yksi iso vaatimus niiden kuluttajille. Pelaajan täytyy osata pelata peliä. Kun verrataan pelejä vaikka elokuviin, niin elokuvan katsojan täytyy vain katsoa ja kuunnella. Nautinnon määrä voi vaihdella riippuen kuinka paljon katsoja on tutustunut muihin elokuviin ja aiheeseen liittyvään teoriaan, mutta keskiverto elokuva ei edellytä katsojaltaan paljoakaan taustatietoa tai erityisiä taitoja. Peleissä sen sijaan pelaajalla täytyy olla tietty taitotaso, jotta hän voi kokea koko pelin sisällön. Tämän takia monet pelit sisältävät erilaisia harjoittelutasoja pelin alussa, joissa täysin uusi pelaaja voi opetella pelien pelaamisen perusteet. Monet pelit suunnitellaan kuitenkin jo pelejä pelanneille pelaajille. Erilaiset vaikeusvaihtoehdot voivat kuitenkin auttaa eritasoisia pelaajia saamaan haluamansa tasoisen haasteen. Vaihtoehtoisesti peli voidaan suunnitella pelaajalähtöisesti, jolloin voidaan käyttää erilaisia suunnittelutapoja pelin vaikeuden ja / tai tunnelman reaaliaikaiseen säätelyyn.

3.1 Pelikokemuksen muokkaaminen käyttäjäkohtaisesti

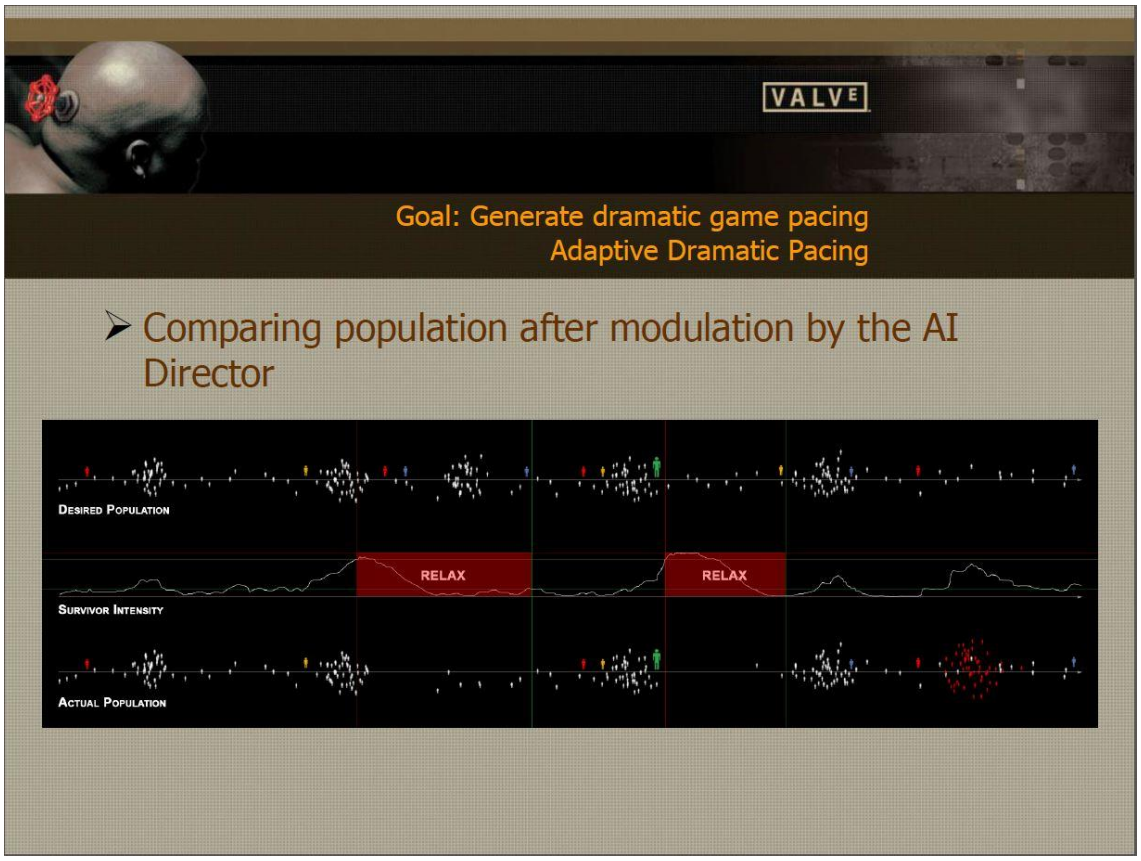
Pelikokemuksen muokkaustapa riippuu pelin tyypistä. Kuviossa 1 näemme mahdollisen pohjan adaptiivisen tai pelaajaan muokkautuvan pelisuunnittelun pohjaksi. Pelit, jotka suunnitellaan vastaavanlaisen suunnitelman pohjalta, toimivat paremmin mahdollisimman isolle yleisölle. Peleissä, jotka ovat suunniteltu pelaajalähtöisesti, voidaan käyttää erilaisia tapoja pelin vaikeuden ja/tai pelikokemuksen säätelyyn. Yksi näistä tavoista on niin kutsuttu ”auto-dynaaminen vaikeustaso”, jossa peli muuttaa pelin vaikeutta riippuen pelaajan onnistumisesta pelissä siihen asti. Tätä käytettiin esimerkiksi pelissä Max Payne, jossa peli lisäsi peliin vihollisia tai vaikeutti näiden tappamista erinäköisten statistiikkojen pohjalta, jota pelaajalta kerättiin pelin taustalla (Digital Games Research Group 2005.).



Kuvio 1. Mahdollinen pohja adaptiivisen pelisuunnittelun pohjaksi. (Digital Games Research Group 2005.)

Yksi tapa toteuttaa adaptiivista suunnittelua on käyttää jotakin toista tahoa pelin säätelyyn sen sijaan että peli säätelee itse itseään. Tällä tavoin onnistutaan parhaiten lukemaan ihmisen mielentilaa. On kuitenkin erittäin kallista, jos jokaista pelaajaa kohden pitää olla toinen ihminen, joka säätelee tämän peliä. Tällainen lähestymistapa onkin paras, jos kyseessä on esimerkiksi jonkinlainen tieteellinen testaustilanne, ja halutaan testata tiettyjä asioita peliä tai jotain muuta tarkoitusta varten. Vaihtoehtoisesti tämä voidaan toteuttaa laittamalla toinen pelaaja tähän rooliin, ja esimerkiksi suunnitella peli olemaan neljä vastaan yksi moninpeli, jossa tällä yhdellä pelaajalla on enemmän valtaa kuin hänen vastustajillaan. Tätä voidaan vaikka verrata Dungeons & Dragons – roolipeliin, jossa yksi pelaaja on ”Dungeon Master”, joka ohjaa koko pelimaailmaa muita pelaajia vastaan. Hänen tehtävänsä on yrittää pitää vaikeustason pelaajille sopivana, vaikeuttaen tai helpottaen peliä pelaajien tilanteen mukaan. Tässä pitää kuitenkin huomioida se mahdollisuus, että jos tällä yksittäisellä pelaajalla on liikaa valtaa, voi hän pilata pelin monelta ihmiseltä, varsinkin jos tästä ei seuraa mitään rankaisua. Vaihtoehtoisesti jos yksittäisellä pelaajalla on liian vähän valtaa pelissä, voi se johtaa tilanteeseen, jossa kukaan pelaaja ei halua pelata tätä roolia. Tällainen nk. asymmetrinen pelisuunnittelu on oma kiinnostava lajinsa, jota on alettu enemmän kokeilemaan joissakin nykypeleissä (esimerkiksi Evolve (2015)), Se on kuitenkin vielä pitkälti tutkimaton pelityyppi.

Tämänkaltaisen suunnittelun tarkoituksena on yleensä saavuttaa niin kutsuttu flow-tila. Tällä tarkoitetaan vaikeustason osaa, jossa peli on pelaajalle juuri sopivan vaikea, johtaen tilaan jossa pelaaja uppoutuu täysin peliin. Jos vaikeustasoa säädetään tästä pisteestä vaikeampaan; pelaajat yleensä turhautuvat. Jos taas vaikeustasoa säädetään helpompaan päin; niin pelaajat saattavat kyllästyä. Kun peli saadaan tasapainotettua pelaajalle juuri oikein, tulee pelaajan immersion eli peliin uppoutumisen taso pysymään korkealla koko pelin ajan. (Chen 2006.)



Kuva 3. Valven esitys Left 4 Dead:n teko-äly ohjaajasta (Valve 2009).

Pelejä on vaikea suunnitella tätä flow-tilaa ajatellen, koska pelaajille ei ole välttämättä hyvä pelata samalla vaikeudella joka hetki pelissä. Tätä voidaan miettiä nk. interest curve tai kiinnostuskurvin – kautta. Tämä teoria on alun perin kehitetty elokuvia varten, mutta pätee hyvin myös peleihin. Teorian mukaan pelaajan kiinnostusta ei haluta pitää tasaisena, vaan halutaan, että jännittäviä hetkiä seuraa hiljaisempia hetkiä, jotta pelaa-

ja saa aikaa rauhoittua ja käsitellä tapahtumia. Tällaisen kiinnostuskurvin saavuttamiseen voidaan käyttää hyväksi vaikeustason säätelyä. Tällaista tunnelman ja vaikeustason automaattista säätelyä kokeiltiin esimerkiksi Valve:n Left 4 Dead pelissä. Pelin taustalla oli nk. AI Director tai tekoälyohjaaja, joka muokkasi pelin tapahtumia tukeakseen haluttua tunnelmaa. Tämä tekoäly muokkasi pelin tapahtumia juuri perustuen kiinnostuskurviin, kuten voimme nähdä kuvasta 3. Tekoäly ei muokannut peliä pelkästään vaikeustason osalta, vaan arvioi myös, milloin pelin pitäisi pitää hiljaisempia hetkiä jännityksen kohottamiseksi. Tämä johti peliin, jossa oli vaikea arvata mitä seuraavaksi tapahtuu, vaikka pelaaja pelasi samoja kenttiä uudestaan ja uudestaan. Tällöin peli loi jännitystä jokaiseen hetkeen pelissä, muokkaamatta suuremmin pelin vaikeustasoa. (Extra Credits 2012.; Serviss 2013.; Valve 2009)l

3.2 Käyttäjäkohtaisen suunnittelun vaikeudet

Adaptiivisessa pelisuunnittelussa on kuitenkin myös paljon omia vaikeuksiaan. Ensimmäisenä ongelmana tulee vastaan se, että tällainen suunnittelu on huomattavasti perinteistä lineaarista suunnittelua vaikeampaa ja kalliimpaa. Myös tällaisten pelin testaus vaatii enemmän aikaa ja rahaa pelinkehittäjältä, koska peliä pitää testata kaikilla eri vaikeusvaihtoehdoilla. Toisena pelin sisältö pitää suunnitella muokkautumaan pelaajan taitotasoon. Siinä, missä perinteisemmässä pelissä suunnitellaan yhden huoneen tapahtumia, voi pelinkehittäjä rauhassa suunnitella tarkalleen esimerkiksi vihollisten sijainnit huoneessa. Vastaavasti, jos peli suunnitellaan toimimaan adaptiivisesti, tämän huoneen vihollismäärää voidaankin vaihtaa pelaajan taidon mukaisesti. Tällöin suunnittelijan täytyy miettiä vihollisten sijainti kaikilla mahdollisilla vihollismäärillä, jos hän haluaa suunnitella tarkasti miten pelaaja kokee huoneen. Kolmantena ongelmana on, miten pelaajat kokevat tämän vaikeustason säätelyn. Jotkt pelaajat haluavat pelata mahdollisimman vaikeaa peliä, ja jos peli laskee vaikeustasoa itsenäisesti, voivat pelaajat kokea tämän negatiivisesti. Vaihtoehtoisesti, jos peli ehdottaa vaikeustason laskemista pelaajalle, voivat pelaajat kokea tämän loukkaavaksi.

Adaptiivisessa suunnittelussa pitää myös huomioida pelaako pelaaja yksin vai esimerkiksi muita pelaajia vastaan. Tällöin adaptiivisen suunnittelun käyttö voi antaa jollekin pelaajalle etua muita pelaajia vastaan, jolloin heidän tuloksensa eivät voi olla verrattavissa toisiinsa. Yleisesti tämän kaltaisissa tilanteissa on parempi antaa muiden pelaajien luoda adaptiivinen vaikeustaso, eikä muokata pelin omia parametreja jokaiselle

pelaajalle. Tätä voidaan auttaa sisällyttämällä peliin esimerkiksi jonkinlainen systeemi, mikä vertailee pelaajien taitotasoja ja yrittää laittaa samaan peliin mahdollisimman samantasoisia pelaajia. Tämänkaltaisia systeemejä käytetään monissa kilpailullisissa verkkopeleissä (esimerkiksi League of Legends, DOTA 2, Counter-Strike: Go).

4 ASIAKAS-PROJEKTI

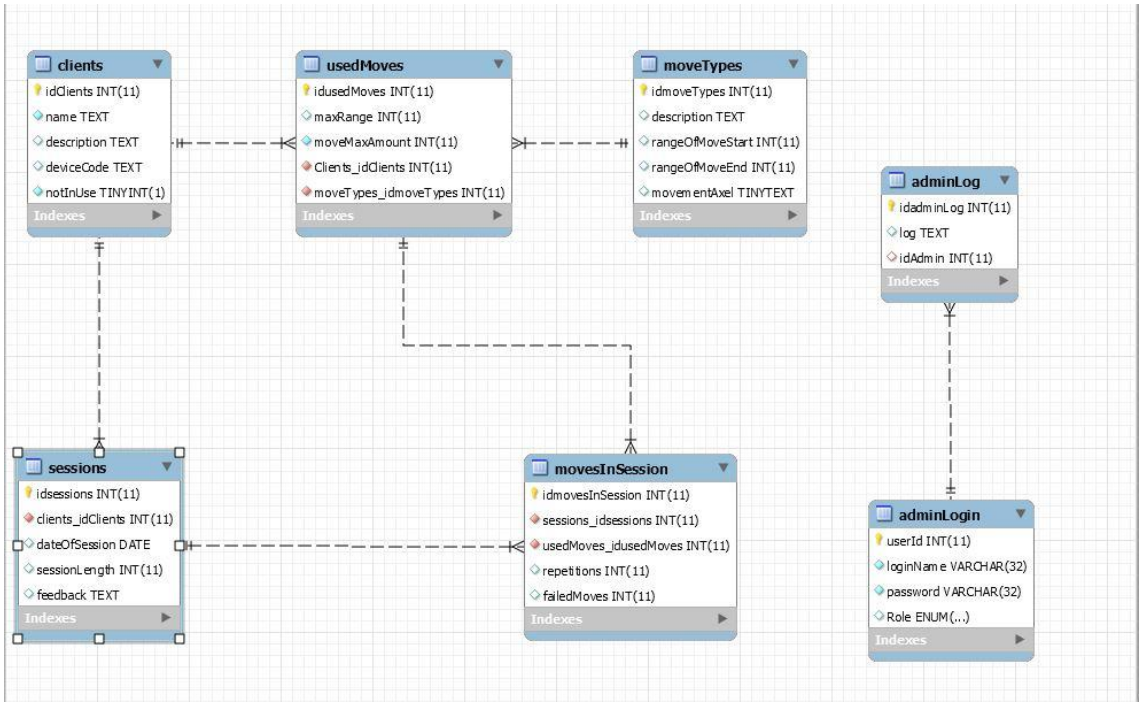
Opinnäytetyön käytännön osiona luotiin systeemi, jossa peli kommunikoi palvelimen kanssa, ja käyttää sieltä saatuja tietoja pelisisällön muokkaamiseen. Tähän sisältyi myös palvelimen koodin tekeminen, ja näiden tietojen muuttamista varten käyttöliittymä palvelimelle. Näin ollen tämä opinnäytetyön käytännön osuus voidaan jakaa neljään eri osa-alueeseen: tietokanta, jossa tiedot säilytetään, hallintasovellus jossa käyttäjä voi muokata peliin meneviä tietoja ja selata pelien tuloksia, verkkorajapinta johon peli yhdistää pyytäessään ja lähettäessään tietoja sekä kuluttajapinta eli itse peli, jossa näitä tietoja käytetään.

Työn aluksi pystytimme yhdessä The Firma-yrityksen kanssa testipalvelimen, jota tässä projektissa käytettiin testaukseen. The Firma on Turun ammattikorkeakoulun alla toimiva yritys, joka antaa opiskelijoille mahdollisuuden saada työkokemusta koulutuksen aikana. Sen jälkeen minä vastasin testipalvelimen ylläpidosta ja tarpeellisten ohjelmien hankkimisesta. Suurin osa projektiin liittyvästä työstä tehtiin pelilaboratorion tiloissa ottaen etäyhteyden palvelimeen asioiden testausta varten. Työtä testattiin suoraan pelin testiversion kanssa, jotta se toimisi mahdollisimman hyvin. Pelin omaa sisältöä kehitti samaan aikaan opiskelijaryhmä, jonka kanssa projektissa tehtiin yhteistyötä. Pidimme ryhmän kanssa viikoittain kokouksia, joissa seurattiin projektin etenemistä ja suunniteltiin työn etenemistä. Käytimme myös Trello-verkkosivua, jossa pidimme kirjaa projektin etenemisestä. Tämän lisäksi kaikki projektiin liittyvä materiaali ja lähdekoodi tallennettiin BitBucket -versionhallintapalveluun. Kommunikointiin ryhmässä käytettiin sähköpostia tai WhatsApp-palvelua.

4.1 Tietokanta

Tämän projektin sovelluksen tietokanta on Debian – palvelimelle pystytetty MySQL-tietokanta. Tähän tietokantaan saadaan yhteys palvelimen ulkopuolelta PHP-skriptien kautta. Kuvassa 4 näemme tämän tietokannan rakenteen. Tietokanta suunniteltiin etukäteen MySQL Workbench – ohjelmalla, josta myös kuva on otettu. Tässä tietokannassa on muutama keskenään linkitetty taulukko. Ensin on admin-taulukko, jossa on hallintasovelluksen kirjautumisen tunnukset ja salasanat, ja log-taulukko, johon tallennetaan, tietoja siitä mitä eri käyttäjätunnuksilla on tehty. Tämän jälkeen muut taulukot ovat linki-

tetty asiakkaan tietoihin. Asiakkaalla on ensin tietyt liikkeet, jotka ovat sallittuja tälle asiakkaalle. Näihin määritetään myös kuinka monta toistoa asiakkaalle on sallittu ja kuinka suuri kyseisen liikkeen astekulma saa maksimissaan olla. Sen lisäksi jokaisesta pelisessiosta tallennetaan tietoja tietokantaan. Nämä tiedot sisältävät pelisession pituuden, ajankohdan ja eri liikkeiden toistomäärät. Tietokanta sisältää myös yhden näkymän, joka on yhdistelmä eri taulukkoja. Tämä näkymä palauttaa taulukon, jossa on tietoja clients-, usedMoves- ja moveTypes-taulukoista. Tämä näkymä on tarkoitettu helppoon tiedon lähettämiseen pelille, yhdessä helposti luettavassa kokonaisuudessa.

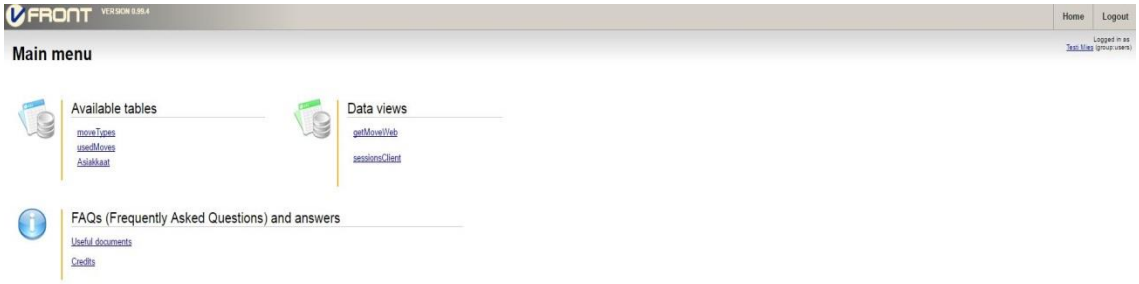


Kuva 4. Sovelluksen tietokannan rakenne.

Pelistä lähetetyt pyynnöt tietokannalle ovat lähinnä pyyntöjä saada nykyinen lista liikkeistä, joita kyseinen asiakas saa suorittaa pelissä. Peli taas lähettää tietokantaan vain jokaisen session tuloksen heti session jälkeen, jos mahdollista. Jos pelillä ei ole yhteyttä, tallennetaan session tiedot, jotta ne voidaan lähettää myöhemmin tietokantaan. Kaikkia muita tietoja voidaan vaihdella hallintasovelluksen kautta, ja myös sessioiden tuloksia voidaan katsoa tätä kautta.

4.2 Hallintasovellus

Tämän verkkoselainpohjaisen hallintasovelluksen tarkoituksena on antaa palvelun käyttäjien tarkkailla asiakkaidensa pelien tapahtumia ja antaa heille palautetta. Tämä saavutetaan verkkopalvelimelle pystytetyn VFront – käyttöliittymän kautta. VFront on tietokannan muokkaamiseen ja selaamiseen tarkoitettu valmis verkkosivupohja, johon voi muokata haluamansa ulkonäön ja sisällön. VFront toimii tässä projektissa kohtalaisesti, vaikka se onkin keskeneräinen (esimerkiksi kaikkia UI elementtejä ei ole käännetty englanniksi, vaan ovat kehittäjän alkuperäiskielellä italiaksi). Koska projektin tarkoituksena oli luoda prototyyppi, ja siihen tämä palvelu on sopiva. Palvelu pystytettiin projektin testipalvelimelle ja muokattiin prototyypille sopivaksi. Kuvassa 5. voimme nähdä esimerkin palvelun käyttöliittymästä.



Kuva 5. Verkkopalvelun käyttöliittymän aloitussivu peruskäyttäjälle.

VFront – pohjan voi ladata suoraan verkosta ja sijoittaa verkkopalvelimelleen, jonka jälkeen sivustolle mennessä, käynnistyy palvelun asennus. Tällöin käyttäjä voi osoittaa mitä tietokantaa palvelu tarkkailee, ja muita tarpeellisia tietoja. Asennuksen jälkeen näitä tietoja voi vielä muokata palvelun kautta, tai muokkaamalla konfiguraatio tiedostoja palvelimella. Palveluun voidaan tämän jälkeen luoda erilaisia ryhmiä, joihin käyttäjät voidaan sijoittaa, ja jotka vaikuttavat käyttäjien käyttöoikeuksiin. Tätä kautta eri käyttäjäryhmille voidaan antaa esimerkiksi katsomisoikeudet johonkin tiettyyn taulukkoon tai vaihtoehtoisesti täydet muokkaamis- ja poistamisoikeudet kyseiseen taulukkoon. Eri

käyttäjäryhmille voidaan myös laittaa erilaisia erikoisnäkymiä eri taulukoihin, joista voi nähdä erilaisia ristikkäisiä tietoja toisista taulukoista. Tässä projektissa luotiin kaksi ryhmää, peruskäyttäjät ja hallintakäyttäjät. Peruskäyttäjät näkevät ja voivat muokata asiakkaidensa tietoja ja voivat selata asiakkaidensa pelisessioidensa tuloksia. Hallintakäyttäjät taas näkevät kaikki tietokannassa olevat tiedot ja voivat myös muokata kaikkien ryhmien oikeuksia.

Palveluun voidaan luoda uusia käyttäjiä hallintakäyttäjien toimesta. Palvelun tili muodostuu sähköpostiosoitteesta ja siihen linkitetystä salasanasta. Kun palveluun luodaan uusi käyttäjä, voi palvelu lähettää tähän sähköpostiin tiedot tilistä, ja jos käyttäjä hukkaa salasanansa, voidaan se palauttaa tämän sähköpostiosoitteen kautta. Tämän jälkeen käyttäjä pitää sijoittaa haluttuun käyttäjäryhmään, jotta hän saa hänelle kuuluvat oikeudet.

4.3 Verkkorajapinta

Opinnäytetyössä tehty verkkorajapinta toimii itse pelin ja palvelimen välissä. Se on tehty muutamilla yksinkertaisilla PHP-skripteillä, jotka tarkistavat käyttäjän oikeudet lähettää ja vastaanottaa tietoa PlayFab -palvelun session ticketillä. Tämä session ticket saadaan kun laite kirjautuu onnistuneesti PlayFabin palvelimelle, ja sitä käytetään eri palveluissa tunnistautumiseen. Kun palvelu toteaa tämän ticketin paikkansapitäväksi PlayFabin palvelimelta, tarkistaa ohjelma vielä, että kyseinen PlayFab-tili on linkitetty tähän palvelimeen. Tämä tarkistetaan tietokannassa olevassa asiakas pöydästä, johon palvelun ylläpitäjä on lisännyt tämän PlayFab-tunnuksen. Jos nämä molemmat ehdot täyttyvät, palauttaa palvelin halutut tulokset tai vastaanottaa lähetetyt tulokset pelistä. Tällä tavoin voidaan välttää jonkin verran ylimääräistä liikennettä ja estetään turhaa tiedon lähetystä ulkopuolisille tahoille.

Tämän kaltainen rajapinta on yleisesti turvallinen satunnaisia tiedon kerääjiä vastaan. Kuitenkin jos esimerkiksi hyökkääjä saa haltuunsa käytössä olevan session ticketin, voi hän käyttää tätä hyökätäkseen järjestelmää vastaan. Tätä voitaisiin vaikeuttaa esimerkiksi salaamalla kaiken kulkevan tiedon ja lyhentämällä ticketin kestoa. Hyökkääjä eivät voi kuitenkaan saada paljoa aikaan pelkästään tällä ticketillä, vaan voivat vain pyytää tietoja käytetyistä liikkeistä tai mahdollisesti syöttää tietokantaan turhia sessio-rivejä. Jotta hyökkääjä saisi enemmän tietoja haltuunsa, pitäisi hänen murtautua itse palvelimelle, jonka turvallisuus riippuu siitä, miten palvelin on muuten suunniteltu, eikä pelkäs-

tään tästä käytetystä järjestelmästä. Kyseinen järjestelmä ei sisällä mitään henkilötietoja tai muita hakkeria kiinnostavia tietoja, joten palvelimelta saatava tieto on arvotonta mahdollisille hyökkääjille. Mahdollisesti, jos tämän sovelluksen käyttö laajenee, voidaan siihen lisätä tietoja, joita tarvitsee suojella. Silloin voidaan tarvita järjestelmän suojauksien jatkokehittämistä.

PlayFab on palvelu, joka myy itseään pelinkehittäjien käytettäväksi. Sen tarkoituksena on olla halpa vaihtoehto, johon voidaan tallentaa pelien tietoja ja statistiikkoja tarvitsematta omia palvelimia. PlayFab voi mitata monia muitakin tietoja, ja sen implementointi moniin pelimoottoreihin on hyvin tuettua. Rekisteröiminen ja peruskäyttö palvelussa on ilmaista, mutta tietyt asiat vaativat kuukausipohjaisia maksuja. Tähän projektiin riittää kuitenkin tämä ilmaisversio.

4.4 Kuluttajapinta

Viimeisenä projektin osana oli peli, jossa näitä tietokannassa olevia tietoja käytettiin. Pelissä käytetään näitä tietokannasta saatavia tietoja pelikokemuksen muokkaamiseen. Kun peli käynnistyy, lähettää se palvelimelle kyselyn pyytäen tietoja, mitä pelaaja saa pelissä tehdä. Tiedot saadaan verkkorajapinnan kautta. Tiedot ovat näkymättömiä itse pelaajalle, ja suurin osa näistä operaatioista tapahtuu hiljalleen pelin taustalla. Tämä on tehty niin, koska pelistä haluttiin mahdollisimman yksinkertainen käyttää, ja esimerkiksi kirjautumistunnukset koettiin kehittäjien puolelta turhan vaikeaksi tämänkaltaiseen peliin, varsinkin jos hyvä tietoturva saadaan toteutettua ilman sitä. Jos peli otetaan tulevaisuudessa laajempaan käyttöön, on todennäköisesti parempi siirtyä käyttämään kirjautumistunnuksia (esimerkiksi vaikka Google- ja Apple-tunnukset, jotka löytyvät melkein jokaisesta mobiililaitteesta) asiakkaan tunnistukseen. Kuitenkin tässä prototyypissä tämä käytetty järjestelmä on hyvä kompromissi turvallisuuden ja käytettävyyden välillä.

Yksi tämän lähestymistavan heikkous on laitteet, jotka eivät ole välttämättä pelihetkellä verkkoyhteydessä. Tämä johtaa siihen, että laite ei välttämättä saa haluamiaan tietoja pelin käynnistyessä, tai pysty lähettämään pelituloksia palvelimelle. Peliin suunniteltiin tämän takia erilaisia varmistusjärjestelmiä tämän kaltaisten tapauksien varalle. Heti kun peli saa ensimmäisen kerran tämän asiakkaan sallitut liikkeet, peli tallentaa ne erilliseen tallennustiedostoon, josta peli voi ladata ne, jos se ei saa yhteyttä palvelimeen. Kun taas laite saa yhteyden palvelimeen, lataa se uudet liikkeet, ja päivittää ne tiedos-

toon. Tämän lisäksi pelaajan suorittamat sessiot tallennetaan tiedostoon, jos peli ei onnistu lähettämään niitä session lopuksi. Sitten peli yrittää lähettää näitä sessiotietoja myöhemmin, kunnes onnistuu siinä, ja sitten poistaa nämä tiedot pelilaitteesta. Nämä kaikki tiedot tallennetaan helpossa XML-tiedostomuodossa. Tämä on lähinnä sen käytettävyyden helppouden takia.

Kuva 6. Tehdyn pelin päävalikko

Peliin lisättiin myös asiakkaan palautteen pohjalta pelisessioiden selaus. Tällä tavoin pelin käyttäjä voi selata edellisten sessioidensa tuloksia laitteellaan ja nähdä mahdollisen fysioterapeutin antaman palautteen. Peliin tehtiin myös muutama muu pieni muutos tämän saman palautteen pohjalta (esimerkiksi epäonnistuneiden liikkeiden mittaus). Kuvassa 6 näemme pelin päävalikon, jossa on myös edellä mainittu palvelinyhteys ikkuna, josta näitä tietoja voidaan lukea.

5 YHTEENVETO

Opinnäytetyössä tutkittiin, miten tietoa tallennetaan peleissä ja miten kerättyä tietoa voidaan käyttää pelin muokkaamiseen käyttäjäkohtaisesti. Tämä tehtiin käytännössä palvelimella sijaitsevalla tietokannalla, sillä se on parhaimpia tiedonsäilytystapoja palvelimilla. Tämä lähestymistapa tuo omia vaikeuksiaan pelinkehitykseen, mutta hyvin toteutettuna on erittäin hyödyllinen pelinkehittäjälle. Tämän kaltainen lähestymistapa ei ole käytettävissä kovin suureen määrään pelejä, mutta on hyvä vaihtoehto niissä projekteissa, joissa tällainen tiedonkeruu kehittäjälle on tarpeen. Yleisesti tämän kaltaista lähestymistapaa voidaan tarvita erilaisissa tieteellisemmissä peleissä, jossa pelin omistaja haluaa kerätä haluamaansa tietoa tai jos pelin omistaja haluaa muokata pelaajan kokemusta tallennettujen tietojen kautta.

Tämän kaltainen lähestymistapa ei ole kovin vaikea toteuttaa, mutta vaatii paljon tietoturvan miettimistä. Tässä projektissa kuitenkin rakennettiin prototyyppiä, joten tietoturva ei ole tehokas. Lisäksi tässä prototyypissä ei kulje mitään asiakastietoja, joita mahdolliset hakkerit voisivat käyttää. Myös tehdyn tietoturvan pitäisi estää joitakin mahdollisia hyökkääjiä kaatamasta palvelinta, mutta sen tehokkuus riippuu myös siitä, miten palvelin on muuten konfiguroitu. Jos palvelun kehitystä päätetään jatkaa ja palvelussa aletaan käyttää enemmän asiakastietoja tai jos palvelu päätetään yhdistää johonkin toiseen palveluun, jossa on tärkeitä asiakastietoja, tarvitsee kehittäjien suunnitella tietoturvaa eteenpäin. Pelissä itsessään oleva osa, joka lähettää tietoja palvelimelle, pitäisi myös olla helppo kehittää jatkossa, jos pelistä halutaan lisää tietoja tai peliin tarvitaan enemmän tietoja.

Jos projektia jatketaan, kannattaa myös verkkokäyttöliittymä todennäköisesti rakentaa kokonaan uudestaan käyttäen tähän tarkoitukseen kehitettyjä erilaisia PHP-kirjastoja. Projektissa käytetty VFront-pohja toimi hyvin idean demonstroimiseen, mutta todennäköisesti on kehittäjien helpompi rakentaa oma sivusto kuin ruveta muokkaamaan tätä pohjaa. Muuten projektista saatu palaute asiakkaalta on ollut positiivista ja projektia viimeistellessä ryhmä sai vielä muutaman toteutettavan idean itse peliin. Pelissä itsessään on myös paljon erilaisia hyviä toteutettavia ideoita, jos kehitystyö saa tarpeeksi aikaa ja tarpeeksi osaavan ryhmän jatkokehitykseen.

LÄHTEET

Chen, J., 2006.. *Flow in Games.* [Online]
Available at: http://www.jenovachen.com/flowingames/thesis.htm
[Haettu 22 Huhtikuu 2016].

Digital Games Research Group, 2005.. [Online]
Available at: http://107.167.189.191/~vishnu/gameResearch/AI_november_2005/digra05.pdf
[Haettu 22 Huhtikuu 2016].

Doull, A., 2009.. *Gamasutra.* [Online]
Available at:
http://www.gamasutra.com/view/news/115412/Analysis_The_Game_Design_Lessons_Of_Permadeath.php
[Haettu 29 Huhtikuu 2016].

Extra Credits, 2012.. *Pacing - How Games Keep Things Exciting.* [Online]
Available at: https://www.youtube.com/watch?v=5LScL4CWe5E
[Haettu 27 Huhtikuu 2016].

Hall, R. & Novak, J., 2008.. Game Development Essentials: Online Game Development. Teoksessa: s.l.:Cengage Learning, pp. 197 - 203.

Hoglund, G. & McGraw, G., 2007.. Exploiting Online Games: Cheating Massively Distributed Systems. Teoksessa: s.l.:Pearson Education, Inc., pp. 113 - 183.

Koster, R., 2008. *Raph Koster's Website.* [Online]
Available at: http://www.raphkoster.com/2008/04/17/how-to-hack-an-mmo/
[Haettu 7 Huhtikuu 2016].

Moran, C., 2010.. *The Fibreculture Journal.* [Online]
Available at: http://sixteen.fibreculturejournal.org/playing-with-game-time-auto-saves-and-undoing-despite-the-magic-circle/
[Haettu 29 Huhtikuu 2016].

Novak, J., 2011. Game Development Essentials: An Introduction. Teoksessa: s.l.:Cengage Learning, pp. 271 - 274.

PlayFab, 2015. *PlayFab.* [Online]
Available at: https://playfab.com/what-we-found-when-we-asked-players-about-security/
[Haettu 20 Huhtikuu 2016].

Polygon, 2016. *Polygon.* [Online]
Available at: http://www.polygon.com/2016/4/7/11384154/world-of-warcraft-nostalrius-shutdown-blizzard
[Haettu 27 Huhtikuu 2016].

Serviss, B., 2013.. *Gamasutra.* [Online]
Available at:
http://www.gamasutra.com/blogs/BenServiss/20130207/186193/The_Discomfort_Zone_The_Hidden_Potential_of_Valves_AI_Director.php
[Haettu 27 Huhtikuu 2016].

Sony, 2011. *Playstation Blog.* [Online]
Available at: http://blog.us.playstation.com/2011/04/26/update-on-playstation-network-and-qriocity/
[Haettu 21 Huhtikuu 2016].

Toy, M., Wichman, G. & Arnold, K., 1980. *Rogue.* s.l.:Epyx.

Valve, 2009. *Valve Software.* [Online]
Available at:
http://www.valvesoftware.com/publications/2009/ai_systems_of_l4d_mike_booth.pdf
[Haettu 27 Huhtikuu 2016].