Jere Karppinen

SELAINPELIN KEHITYS PHP/MYSQL-TEKNIIKALLA

Opinnäytetyö
Tietojenkäsittely

Marraskuu 2012

MIKKELIN AMMATTIKORKEAKOULU
Mikkeli University of Applied Sciences

KUVAILULEHTI

MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences	**Opinnäytetyön päivämäärä** 30.11.2012
Tekijä(t) Jere Karppinen	**Koulutusohjelma ja suuntautuminen** Tietojenkäsittelyn koulutusohjelma
Nimeke Opinnäytetyö	

Tiivistelmä

Noin 10 vuoden iässä syttynyt innostus osoita ja klikkaa -tyylisiä seikkailupelejä kohtaan ei ole laantunut vieläkään. Luovuuden ja ohjelmointiosaamisen yhteisvoimin halusin ottaa selvää, kuinka hyvin PHP/MySQL-tekniikka soveltuu selainpohjaisen seikkailu/roolipelin tekemiseen. Työkalut olivat tuttuja, mutta haaste oli uusi ja mielenkiintoinen. Otin projektiin mukaan luokkatoverini Harri Lappalaisen.

Ensin syvennyn tutkimaan sanaa ”peli” ja otan selvää, mitä sillä tarkoitetaan. Sen jälkeen kerron pelisuunnittelusta ja esittelen erilaiset videopelityypit. Tämän jälkeen siirryn eri pelialustoihin ja valotan niiden elinkaarta aina synnystä nykypäivään. Eri alustoista käyn läpi tietokoneen, konsolit, mobiililaitteet ja selaimen.

Pelimoottorin suunnittelu –osuudessa esittelen pelimoottoria terminä ja selitän, millaista peliä olin tekemässä. Sen jälkeen siirryn käytännön osuuteen, jossa esittelen ja perustelen käyttämäni tekniset ratkaisut kuvien kera.

Päätännössä selitän, miten projekti mielestäni eteni ja millaisia ajatuksia parin kanssa työskenteleminen herätti. Kerron, miten pelimoottoria mielestäni voisi parantaa. Mietin tulevaisuuden näkymiä pelin konseptin jatkuvuudesta; millaisilla välineillä eteenpäin?

Asiasanat (avainsanat)

PHP, MySQL, JavaScript, pelialustat, pelisuunnittelu, pelimoottori, PC, mobiililaitteet, konsolit

Sivumäärä	**Kieli**	**URN**
53	suomi	

Huomautus (huomautukset liitteistä)

Ohjaavan opettajan nimi	**Opinnäytetyön toimeksiantaja**
Janne Turunen	Mikkelin ammattikorkeakoulu

DESCRIPTION

MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences	**Date of the bachelor's thesis** 30.11.2012
Author(s) Jere Karppinen	**Degree programme and option** Business Information Technology

Name of the bachelor's thesis

Development of a browser-based game using PHP/MySQL technology

Abstract

I was around 10 years old when my love for point and click adventure games caught fire and has not ceased. Through the co-operation with creativity and programming skills I wanted to find out how well PHP/MySQL technique would work in the creation of a browser-based adventure game. I was familiar with the tools, but the challenge was new and interesting.

The study started by investigating the word "game" and its meaning. It introduced game design and different kinds of video game types. After that I moved on to gaming platforms and ventured through PCs, consoles, mobile devices and web browers.

The part that concentrated on the game engine, introduced the concept of the word "game engine" itself. After that I explained how I programmed the engine in practice and what kind of solutions I came up with.

In the end of the bachelor's thesis I summarized how I experienced the whole project. I tell about the improvements I could implement and speculate the future of the concept; what kind of tools should be used in the future

Subject headings, (keywords)

PHP, MySQL, JavaScript, Gaming Platform, PC, Console, Mobile, Browser, Game Design

Pages	Language	URN
53	Finnish	

Remarks, notes on appendices

Tutor Janne Turunen	**Bachelor's thesis assigned by** Mikkeli University of Applied Sciences

SISÄLTÖ

1 JOHDANTO

Kipinä ”klikkaa ja osoita”-tyylisiin seikkailupeleihin minulla syttyi n. 10-vuotiaana, kun tuttavani esitteli minulle *The Curse of Monkey Island*:in. Kiehtova pulmanratkonta vei mukanaan ja englannin taidon karttuessa myös dialogien huumori alkoi avautua. Myöhemmin tutustuin moneen muuhunkin samantyyppiseen peliin. Tämän opinnäytetyön tarkoituksena on ottaa selvää, kuinka hyvin PHP/MySQL-tekniikka sopii selainpohjaisen rooli/seikkailupelin alustaksi.

Idea opinnäytetyöhön syntyi vuoden 2011 marras-joulukuussa. Tässä vaiheessa tunsin mielestäni PHP-ohjelmointikielen hyvin ja pohdin, mitä kaikkea muuta sillä voisi saada aikaiseksi kuin perinteisiä Internet-sivuja. Oman pelin kehitys oli kuplinut takaraivossa jo jonkin aikaa, aina juonen kehittämisestä tekniseen toteutukseen. Kun sain vielä luokkakaverini puhuttua mukaan, asia oli selvä. Otimme tutut työkalut mukaamme ja yritimme saada niillä aikaiseksi jotain uutta ja luovaa. Tästä syntyi projekti pelille: A Viking Tale: The Runes of Memory.

Luvussa 2 käsittelen peliä terminä syvällisemmin ja haen pohjaa ihmisten erilaisille käsityksille peleistä. Kerron hieman pelisuunnittelun filosofiasta ja esittelen myös erilaiset pelityypit sekä niiden tunnetuimmat ja uraa uurtavimmat pelit.

Luvussa 3 kerron yleisesti eri alustoista, joilla pelejä voi pelata. Käyn läpi tietokoneen, konsoleiden, mobiililaitteiden ja selaimen roolin pelialustoina. Käsittelen niiden historiaa sekä evoluutiota aina nykypäivään saakka.

Luku 4 kertoo omasta visiostani kehittämäni pelin suhteen. Kerron, miten etenin ja toteutin asiat teknisesti. Esittelen pätkiä ohjelmakoodista ja kuvakaappauksia selaimesta hahmottaakseni lopputulosta.

Päätännössä kokoan asiat yhteen ja esitän johtopäätökseni projektin luonteesta ja kulusta. Visioin jatkokehitys- ja korjausmahdollisuuksia. Kerron myös, miten aion tämän jälkeen konseptia jatkaa.

2 PELI KÄSITTEENÄ

Filosofi Ludwig Wittgenstein pyysi lukijoitaan määrittelemään sanan ”peli”. Joka kerta, kun esille nousi jokin välttämätön ominaisuus (pelin on oltava kilpailuhenkinen, pelin on oltava viihdyttävä jne.), hän osoitti jotain tunnettua peliä, joka rikkoi tätä sääntöä. Yllätyksellisesti hän ei ehkä ollutkaan vastauksen perässä, vaan tahtoi osoittaa että, tiukat määritelmät eivät toimi, koska ihmiset eivät luonnollisesti käytä niitä. Käytämme kommunikoidessa joustavia kategorioita tiukkojen määritelmien sijaan. (Rabin 2009, 63.)

Peliksi voidaan kutsua mitä tahansa toimintoa, joka tuottaa mielihyvää, eikä sillä ole tietoista päämäärää. Tämän määrittelyn perusteella kaikkea, mitä ihmiset tekevät saadakseen mielihyvää, voidaan kutsua peliksi - kuten tanssia, musiikkia tai näyttelemistä. (What is a Game? 2000.)

Pelit muodostuvat säännöistä ja ainesosista. Useimmissa peleissä säännöt ovat ainesosia tärkeämpiä. Aineosat edustavat ikään kuin laitteistoa ja säännöt ohjelmistoa. Molempia tarvitaan pelin muodostamiseen. Ne voivat olla olemassa erikseen, mutta vain yhdessä muodostavat pelin. Arkeologit ovat löytäneet muinaisia pelilautoja, mutta säännöt ovat jääneet tuntemattomiksi. Emme luultavasti saa koskaan selville, kuinka näitä pelejä pelattiin. (What is a Game? 2000.)

2.1 Pelin suunnittelu

Ammattimainen pelinkehitys oli melkeinpä tuntematon ala vielä noin kolmekymmentä vuotta sitten. Seitsemänkymmenluvun lopun ja tämän päivän välillä mikrosiruilla toimivat videopelit ovat muuttaneet kaiken. Taitaville pelisuunnittelijoille tuli tarvetta, kun markkinat räjähtivät isoksi kaupanteoksi. Nyt olemme tilanteessa, jossa pelialan koulutus on eturivissä. (Rabin 2009, 61.)

Pelintekijä on jatkuvasti avun tarpeessa. On mietittävä esimerkiksi, miten pelaajat käyttävät teleporttia tai saavat onnistumisen tunteen uuden kokemustason karttuessa. Suunnittelijat harvoin kuitenkaan tekevät kaikkea yksin ja vaikka tekisivätkin, he yleensä silti konsultoivat muita ihmisiä. Pelikehitysryhmän jokainen jäsen astuu

jossain välissä pelisuunnittelijan saappaisiin. He arvioivat peliä eri näkökulmista samalla, kun työstävät ja muokkaavat sitä, kuten taiteilija, joka maalaa uuden maalikerroksen vahvistaakseen jälkensä voimaa tai ohjelmoija, joka kirjoittaa ensin koodin suunnitellusti, mutta testattuaan parantaa sitä hieman. Jokainen jäsen on täten ”kaappisuunnittelija”, joka parantaa peliä jollain tavalla itse tiedostamatta sitä. Pähkinänkuoressa voidaan todeta: pelissä kaikki ominaisuudet, jotka parantavat pelaajan kokemusta, ovat vaikuttaneet pelin suunnitteluun. (Rabin 2009, 62.)

Peleihin suunnitellaan koukkuja, jotka saavat ihmiset pelaamaan. Pelaajia on erilaisia. Joillakin voi olla suojeluvaisto: jos peliin heitetään söpöjä pingviinejä ja susi ajaa niitä takaa, niin pelaajalle voi tulla tarve pelastaa pingviinit. Joillakin on tarve voittaa kaikki muut. Jotkut ovat todella uteliaita ja he eivät jätä yhtään kiveä kääntämättä. Joillakin on ”hamsterigeeni” eli, kun pelissä saa tavoitteeksi kerätä tietyt asiat palkintoa vastaan, niin kerääjät ottavat haasteen vastaan. Verkkoroolipeleissä sosiaalinen status voi merkitä joillekin paljon. (Ukko 2011.)

Pohjimmiltaan pelaamisessa on kyse oppimisesta. Koulussa oppilas saa palautetta harvakseltaan tunneilla, yleensä vain kurssiarvosanan tai koulutodistuksen. Peleissä palautetta tulee kuitenkin jatkuvasti, oli se sitten pisteitä tai räjähdyksiä. Palaute kertoo, miten pelaaja suoriutui tehtävästään ja voiko hän pelata tämän kohdan vieläkin paremmin. Ihminen saa hyvänolontunteen oppimisesta, joten se on tärkeää. Esimerkiksi *Super Mario Bros* –pelissä aloitusruutu on tehty helpoksi: jos kävelet eteenpäin, niin vastaan tulee vihollinen, joka tappaa hahmon saman tien. Tämän jälkeen pelaaja painaa ohjaimen nappia, jolloin Mario hyppää. Nyt on opittu hyppääminen ja sitten pelaaja miettii, että mitä hyppimisellä voidaan tehdä. Nyt pelaaja osaa hypätä vastaan kävelevän vihollisen päälle. Seuraavaksi on vuorossa opitun taidon ”masteroiminen” eli maksimaalinen hyödyntäminen. Pelaaja miettii, mitä kaikkea hyppäämisellä voi saada aikaan. Peleissä pelaajan saama palaute on siis todella tärkeää. (Ukko 2011.)

2.2 Videopelityypit

Videopelityyppejä käytetään kategorisoimaan pelejä perustuen pelattavuuteen, ei niinkään graafisiin tai kerronnallisiin eroihin. Videopelityyppi määritellään pelattavuuden haasteiden perusteella. Esimerkiksi toimintapeli on toimintapeli huolimatta siitä, sijoittuvatko sen tapahtumat fantasiamaahan tai ulkoavaruuteen. (Video game genres 2012.) Seuraavaksi luettelen erilaisia pelityyppejä.

2.3 Tasohyppely

Tasohyppely-pelit ovat konsolimaailman tavaramerkki. *Donkey Kong* –klassikosta (kuva 1) aina *Ratchet and Clank*:iin, nämä pelit tarjoavat täydellisyyttä hipovan pelikokemuksen ja niitä pelataan paljon edelleenkin. (Schwab 2008, 143.)

KUVA 1. Donkey Kong (Nesretrco.com 2012)

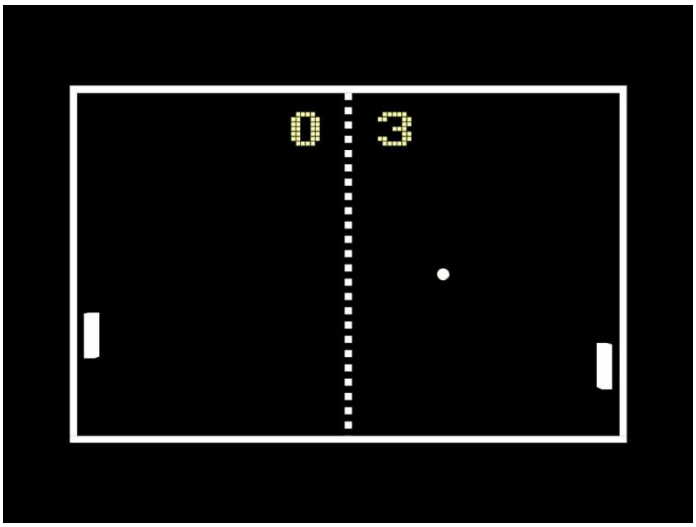
Tasohyppelyt olivat aluksi kaksiulotteisia *Super Mario Bros* (kuva 2*)* –tyylisiä kokonaisuuksia laitteiston suorituskyky- ja muistirajoitteista johtuen. Päähahmo aloittaa ruudun alalaidasta, josta hänen täytyy edetä pääasiassa hyppimällä tasolta tasolle. Nämä pelit olivat todella suosittuja entisaikojen peliluolissa, koska ne esittelivät uuden haasteen pelaajille: ajoituksen. Ennen tasohyppelypelejä *arcade*-pelien haasteet liittyivät lähes kokonaan kuvioiden ja muotojen tunnistamiseen; viholliset lähestyivät säännöllisesti erilaisissa kuvioissa, jolloin niitä piti osata väistellä tai tuhota liikkumalla oikein. (Schwab 2008, 143.)

KUVA 2. Super Mario Bros (Nintendookies 2012)

Super Mario Bros on oiva esimerkki sivuttain avautuvasta pelimaailmasta, jossa pelihahmo juoksee eteenpäin ja pelaaja kokee kokonaisen pelimaailman. *Super Mario Bros* –pelin lisäksi myös *Sonic the Hedgehog* ja *Mega Man* hyödynsivät tätä konseptia onnistuneesti. Nämä pelit olivat vaikutusvaltaisia tässä genressä ja poikivat monia jatko-osia ja jäljitelmiä. (Schwab 2008, 144.)

2.4 Toimintapelit

Ensimmäiset kotikoneille ilmestyvät pelit olivat käytännössä kopioita olemassa olevista *arcade*-peleistä, kuten *Pong* (kuva 3), *Pac-Man* ja *Space Invaders*.



KUVA 3. Pong (Animatedpng.net 2012)

Toimintapelit ovat ensimmäinen tunnistettava videopelityyppi. Toimintapelit painottavat nopeita refleksejä, keskittymistä ja pelitapahtumien nopeaa etenemistä. Grafiikat olivat aluksi hyvin alkeellisia laitteiston rajoitusten vuoksi. (Lecky-Thompson 2007, 24.)

2.5 Taistelupelit

Taistelupelit ovat sekoitus toimintaa ja vihollisiin keskittyvää pulmanratkontaa. Ne olivat iso juttu peliluolissa. Yksinkertaisella, sivuttain vierivällä ruudulla liikkui pelin sankari. Aseena eivät olleet ammukset, vaan kamppailulajit. (Schwab 2004, 203.)

Street Fighter 2: The World Warrior –pelin (kuva 4) myötä taistelupelit saavuttivat suosionsa huipun 1990-luvun alussa. *SF2* muodosti tästä pelityypistä uuden mallin, johon sisältyivät liikeyhdistelmät, torjunnat, erikoisliikkeet ja kaksinpelin. (Schwab 2004, 203.)

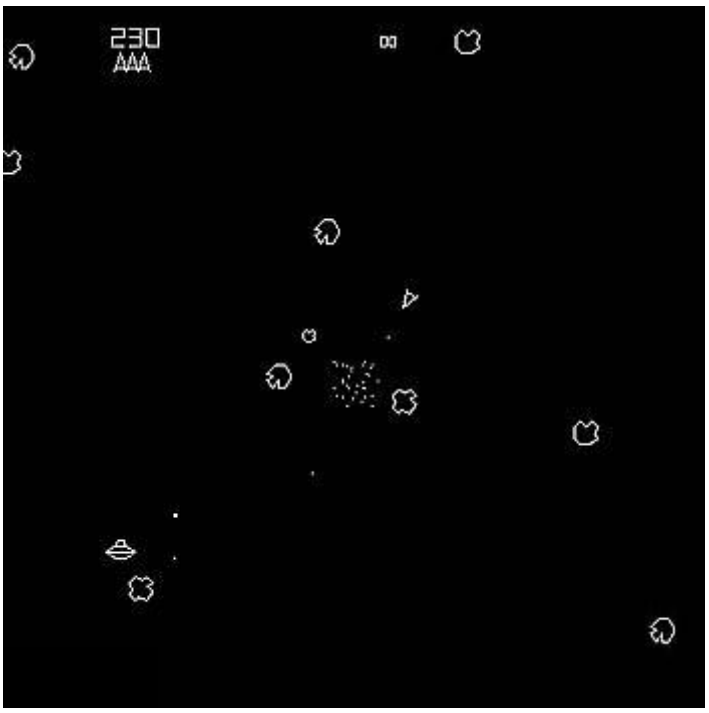
KUVA 4. Street Fighter 2: The World Warrior (Hark.com 2012)

2.6 Ammuntapelit

Ammuntapelit lähtivät liikkeelle peliluolista, mutta eivät saaneet paljoa jalansijaa PC-maailmasta. Kuitenkin huomattava määrä ammuntapelejä julkaistiin kotikonsoleille. Ammuntapeleissä yleensä pelaaja ohjaa jonkinlaista alusta, joka kohtaa kerralla suuria määriä vihollisia. Pelaaja tuhoaa mahdollisimman monta vihollista, samalla väistellen vihollisten luoteja. Matkan varrelta voi noukkia tehokkaampia aseita ja taistella loppuvastustajia vastaan, mitkä ovatkin haastavia tapahtumia. (Schwab 2004, 145.)

Ammuntapeleissä aseet ovat tärkeitä, koska niiden avulla peleissä voidaan edetä. Useimmat ammuntapelit aloitetaan hyvin pienitehoisella aseella, mutta pelin edetessä saadaan haltuun voimakkaampia aseita, joilla voidaan tuhota kestävämpiä vihollisia. (Pardew 2004, 252.)

Vuonna 1979 Atari Inc julkaisi *Asteroids*-pelin (kuva 5), josta tuli eräs suosituimmista ammuntapeleistä. Peliä myytiin 70 000 kappaletta peliluoliin. Pelaaja ohjaa avaruusalusta asteroidivyöhykkeellä. Pelin tavoitteena on ampua ja tuhota asteroidit sekä lautaset, joihin alus ei saa osua. (Wolf 2008.)

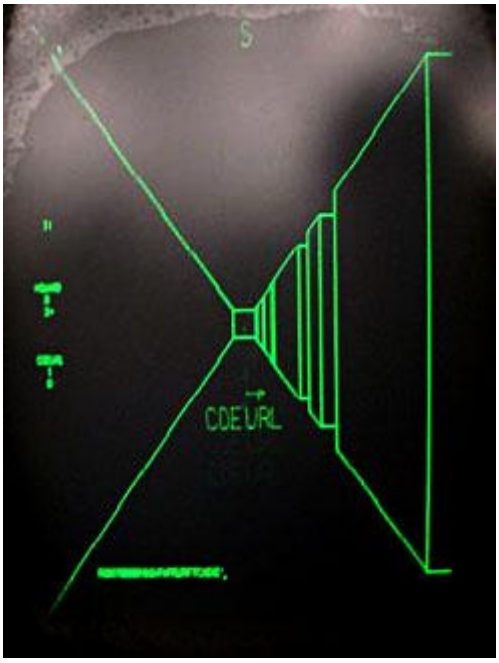


KUVA 5. Asteroids (Images4 2012)

2.7 Ensimmäisen persoonan ammuntapelit

Ensimmäisen persoonan ammuntapeleissä pelaaja pelaa hahmon silmin, eikä näe itseään. Pelin ideana on tuhota vastustaja erilaisilla aseilla. Odotetusti tämän peligenren edustajat ovat saaneet huomattavasti kritiikkiä väkivaltaisuudestaan. Tästä huolimatta (tai sen ansiosta) nämä pelit ovat suosituimpia pelejä markkinoilla. (Garmon 2005.)

Maze War (kuva 6) ja *Spasism* kehitettiin 1970-luvulla ja niitä pidetään ensimmäisinä FPS-peleinä. Molemmissa maailma nähdään hahmon silmin ja tavoitteena on ampua kanssapelaajien hahmoja. Näistä kahdesta *Maze of War*:lla on eniten yhteistä modernien FPS-pelien kanssa: hahmot harhailevat 3D-labyrintissä ja ampuvat toisia, jotka näkyvät silmämunina. (Garmon 2005.)



KUVA 6. Maze War (Old-computers.com 2012)

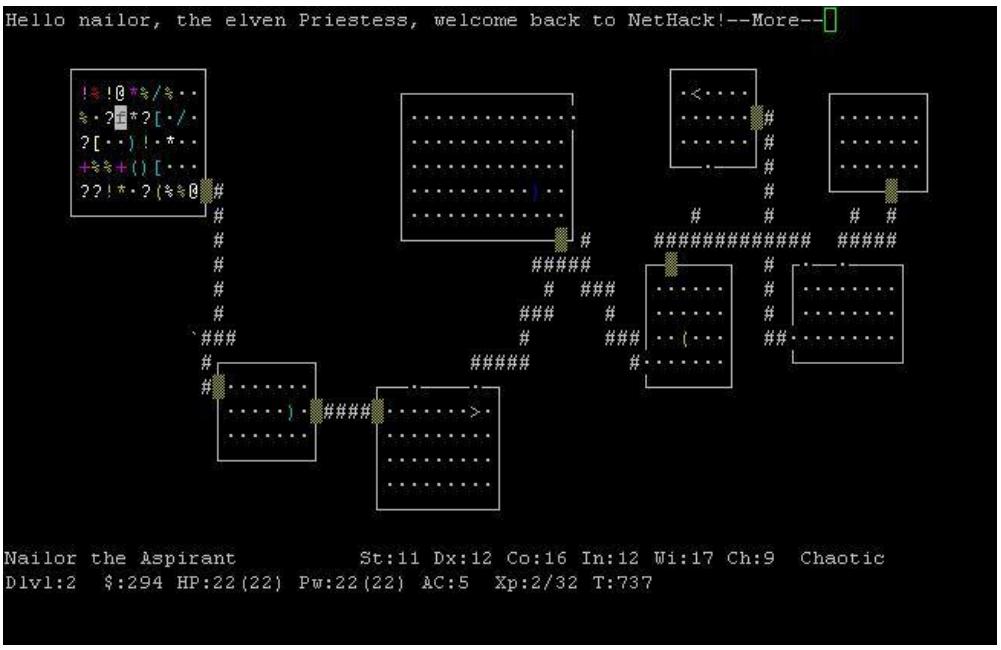
PC:lle id Softwaren julkaisema *Wolfenstein 3D* (kuva 7) laukaisi FPS-pelien vyöryn tietokoneelle. Hahmon silmin nähtävä pelimaailma oli jo esitelty aikaisemmissa peleissä. Myös 3D-maailmoiden esittämiseen oli omat pelimoottorinsa. *Wolfenstein 3D* kuitenkin yhdisti ampumisen, uuden tyyppisen pelimoottorin ja hahmon silmin koettavan pelikokemuksen uudella tavalla, jonka ansiosta sen suosio räjähti. (Garmon 2005.)

KUVA 7. Wolfenstein 3D (Fix-x.com 2012)

2.8 Roolipelit

Ensimmäiset roolipelit olivat paperisia lautapelejä, parhaiten tunnettuna *Dungeons and Dragons*. (Hallford 2002, 25). Kun tietokoneet yleistyivät kotikäytössä, alkoi sadella pelejä, jotka hyödynsivät näppäimistöä pelimaailman ohjauksessa sauvaohjaimen ja nappulan sijaan. Roolipelit olivat ensimmäisten joukossa, joita oli mahdollista tehdä PC:lle. Sitä ennen pelit olivat suunniteltu peliluoliin tuottamaan mahdollisimman paljon rahaa, ja siksi pelisessioiden kestot olivat lyhyitä. (Schwab 2004, 67.)

Ensimmäiset videoroolipelit olivat joko puhtaasti tekstipohjaisia (kuten *Adventure* tai *Hunt the Wumpus)* tai grafiikka oli tehty ASCII-merkeillä (esim. *Nethack,* kuva 8). Uudemmissa konsoleissa oli enemmän suorituskykyä ja ehkä jopa muistikorttipaikka, jolloin pelin saattoi tallentaa. Konsoli-roolipelit alkoivat haarautua. Ne olivat suurimmaksi osaksi enemmän reaaliaikaisia ja taisteluun suuntautuneita, koska konsoleissa on toimintaa suosivat ohjausmekanismit. Näiden pelien pelaajakunta oli keskimäärin nuorempaa kuin PC-pelien pelaajat, joten pelien sankaritkin olivat noin 15-vuotiaita, eivätkä karaistuneita sotureita. Nykyisin PC:n ja konsoleiden pelit ovat hyvin samankaltaisia. Suosittuun PC-peliin *Diabloon* on tehty yksinkertainen ja konsolimainen käyttöliittymä ja uusissa online-konsolipeleissä on melkein identtiset ohjausmekanismit PC-versioihin verrattuna. (Schwab 2004, 67.)



KUVA 8. Nethack (Asteriski.fi 2012)

2.9 Reaaliaikainen strategia

Reaaliaikaisessa strategiassa pelaaja ohjaa useita yksiköitä voittaakseen vastustajansa strategisesti. ”Reaaliaikainen” tarkoittaa, että kaikki toiminto pelissä tapahtuu itsenäisesti, kuten tapahtumat oikeassa elämässäkin. Moninpelissä pelaajat rakentavat omia tukikohtiaan. Jos pelaajien hahmot hyökkäävät toistensa kimppuun, joku ulkopuolinen saattaa hyökätä molempien kimppuun. Tämä pelityyppi muodostuu komentojärjestelmästä, yksiköistä ja tekoälystä. (Pardew 2004, 248.)

Komentojärjestelmä on tapa, jolla pelaaja voi hallita montaa yksikköä kerralla. Joissain peleissä on satoja itsenäisiä yksiköitä ja pelaajan on kyettävä kontrolloimaan jokaista yksikköä. Hyvän komentojärjestelmän suunnitteleminen on taidetta jo itsessään. Jos pelaajan on hallittava jokaista yksikköä erikseen, pelistä tulee helposti pitkäveteinen. Parhaiten suunnitelluissa peleissä on tapa komentaa suuria joukkoja kerralla. (Pardew 2004, 248.)

Jokaisella yksiköllä on oma erikoistaitonsa. Joidenkin voimakkaiden yksiköiden rakentamiseen voi kulua pitkä aika, koska heidän arvonsa taistelukentällä on suurempi. Yksiköiden voimakkuus pitää tasapainottaa, jotta pelistä tulee mielekäs. Tästä kannattaa tehdä kaavio, joka esittää jokaisen yksikön ja sen kyvyn, kuten myös

miten se reagoi muihin yksiköihin. Tällä tavoin pelaaja ymmärtää, miksi mikäkin yksikkö on pelissä mukana. (Pardew 2004, 248.)

Yksikön tekoälyn luominen on osa pelin kaikenkattavaa tekoälyä. Hyökkääkö yksikkö aina vastustajaa kohti, vaikka vihollisyksiköitä olisi paljon enemmän? Miten määritetään moraalinen tai strateginen käyttäytyminen? Voiko pelaaja määrätä tahtoessaan yksittäisen yksikön toimista? (Pardew 2004, 248.)

2.10 Vuoropohjainen strategia

Tässä pelimuodossa pelaajat tekevät siirtonsa vuorotellen. Loistava esimerkki tästä pelityypistä on Shakki. Kaksi pelaajaa siirtää pelinappuloitaan omalla vuorollaan, eikä siirtoon käytettävää aikaa ole rajoitettu. Joissakin shakkipeleissä yksi vuoro voi kestää useita tunteja, koska pelaaja laskelmoi mielessään jokaisen siirron vaikutuksen ja voi jopa laskea monta siirtoa eteenpäin. (Pardew 2004, 248) Vuoropohjaisissa peleissä pelaajalla on aikaa tehdä oma siirtonsa, joten tekoälyn kannattaa olla mahdollisimman haastava. Tämä pätee vain silloin, kun pelaaja pelaa tietokonetta vastaan. (Pardew 2004, 250.)

Shakissa yksikön ominaisuus on tapa, jolla se voi liikkua ruudukossa. Muissa strategiapeleissä voi olla muitakin ominaisuuksia, kuten hyökkäys- tai puolustusmekanismit. Kyvyillä ja ominaisuuksilla on suuri vaikutus pelin luonteeseen. (Pardew 2004, 249.)

Vuoropohjaiset strategiapelit kannustavat harkitsevaisempiin siirtoihin kuin reaaliaikaiset strategiapelit, mutta niillä on paljon yhteistä. Tärkeät ominaisuudet vuoropohjaisessa strategiapelissä ovat maasto/pelilauta, yksikön kyvyt ja vihollinen. Shakissa pelilauta on neliö, joka on täynnä ruutuja. Lauta muodostaa pelialueen ja rajoittaa yksiköiden liikkumismahdollisuuksia. Sama asia pätee myös muissa vuoropohjaisissa peleissä, mutta pelialueena voi olla maasto, jossa on esteitä, kuten vettä tai rakennuksia. (Pardew 2004, 249.)

2.11 Seikkailupelit

Seikkailupelit ovat sidottuja tarinaan. Pelaaja pelaa hahmoa samalla kun tarina etenee. Tarinan avautuminen riippuu osaksi pelaajan toimista ja osaksi pelin tapahtumista. Kaikista pelityypeistä juuri seikkailupelit ovat eniten sidoksissa tarinankerrontaan. (Pardew 2004, 238.) Hahmo kulkee huoneesta toiseen keräten esineitä ja ratkoen ongelmia, jotka auttavat häntä eteenpäin tarinassa ja antavat pääsyn uusille alueille (Schwab 2004, 87).

Ensimmäiset seikkailupelit olivat tekstipohjaisia kuten roolipelitkin (*Zork*-sarja). Pelaajalle kuvailtiin kulloinenkin huone, jossa hän oli ja loput jäi mielikuvituksen varaan. Alussa komennot kirjoitettiin tekstin jäsentäjään, joka tulkitsi komennot ja antoi palautetta. Jos jäsentäjä tunnisti pelaajan antaman komennon, se antoi asiaan kuuluvan vastauksen, muussa tapauksessa ilmoitti, ettei ymmärrä mitä pelaaja tahtoo tehdä. (Schwab 2004, 87.)

Ajan myötä pelintekijät lisäsivät seikkailupeleihin kuvia, mainittavina esimerkkeinä *King's Quest* –sarja (kuva 9) ja LucasArts:n mullistava *Day of the Tentacle* ja *Monkey Island* sekä *Leisure Suit Larry*. LucasArts luopui tekstin jäsentäjistä ja loi käyttöliittymään kuvaavat avainsanat ja ikonit. (Schwab 2004, 87.)

KUVA 9. King's Quest (Bp.blogspot.com 2012)

Vuonna 1993 pelitalo nimeltä Cyan julkaisi pelin *Myst*, josta oli poistettu melkein koko tarina, mutta lisätty erittäin korkealaatuinen maailma ja paljon ongelmia ratkottavaksi. Tässä pelissä hahmo ei voinut kuolla, mutta se ei myöskään sisältänyt mitään apua pelissä etenemiseen. Vaikka idea kuulostaa yksitoikkoiselta, *Myst* oli silti uraauurtava ja se myi yli kaksitoista miljoonaa kopiota. (Schwab 2004, 87.)

2.12 Urheilupelit

Jos haluat pelata kenttäpelaajana tai valmentajana, niin urheilupelit antavat siihen mahdollisuuden. Kyvykkyys tosielämän urheilulajeissa ei ole välttämätöntä pärjätäkseen sitä simuloivassa videopelissä, mutta se on tavallaan pelien tarkoituskin. Voimme unelmoida huippu-urheilijan taidoista ja videopelit voivat auttaa meitä saavuttamaan ne. (Bates 2004, 9.)

Kaikkien urheilupelien päätarkoitus on luoda mahdollisimman todentuntuinen kokemus ko. urheilulajista, joten niiden on pakko seurata sääntöjä ja pelistrategioita millintarkasti (Scattergood 2005, 189). Yhdellä pelisessiolla voi pelata yhden ottelun, turnauksen tai kokonaisen kauden. Jotkin pelit keskittyvät simuloimaan urheilijan yksilösuorituksia. Toiset lähestyvät pelimaailmaa joukkueen valmentajan tai toimitusjohtajan silmin valvoen pelejä ja tehden vaihtokauppaa muiden joukkueiden kanssa. (Bates 2004, 9.)

Urheilupelit ovat olleet mukanamme pelinkehityksen alkuhämäristä asti: *Pong* simuloi tennistä. Yhdistelmä helposti opittavaa pelin kulkua ja kilpailu ystävien kanssa antaa urheilupeleille viehätystä, jota useimmilla pelityypeillä ei ole. Kiihkeä fanilauma, joka vuodesta toiseen ostaa uuden urheilupelin, on tuottava rahasampo pelitaloille, jotka onnistuvat vangitsemaan ihmisten huomion. (Schwab 2004, 157.)

Urheilupelit jakautuvat yleisesti kahteen kategoriaan: sulavaan pelattavuuteen ja nollautuvaan pelattavuuteen. Sulavan pelattavuuden pelejä (*Fluid gameplay sports*) ovat esim. jalkapallo, jääkiekko tai koripallo, missä peli on nopeaa ja dynaamista. Pelien luontoon kuuluu, että jatkuvasti muuttuvat pelikentän olosuhteet vaikuttavat pienempiinkin strategisiin muutoksiin; on valittava pelityyli, joka johtaa pisteen saamiseen. (Schwab 2004, 157.) Nollautuvan pelattavuuden pelit *(Resetting gameplay*

sports) pysähtyvät ja nollautuvat tietyn ajan tai tapahtuman jälkeen. Tällaisia pelejä ovat amerikkalainen jalkapallo ja pesäpallo. (Schwab 2004, 157.)

2.13 Ajopelit

Ajopelit jakautuvat kahteen luokkaan: *arcade*-ajopelit ja simulaatio-ajopelit.

Arcade-ajopeleissä yleensä ajetaan kovaa rentojen fysiikan lakien puitteissa: pelaajat voivat ajaa kovempaa, hypätä pidemmälle ja luisua paremmin kuin oikeassa elämässä olisi mahdollista. Tämän pelityypin pelaajia kiinnostaa itse ajaminen ja kisaaminen muita pelaajia vastaan kuvitteellisilla tai jäljitellyillä radoilla. (Scattergood 2005, 190) Japanilaisen *Polyphony Digital*- peliyhtiön Gran Turismo –pelit (kuva 10) ovat rallipelien joukosta tunnetuimpia. Sarjan ensimmäinen peli julkaistiin vuonna 1997 *PlayStation*-konsolille.

KUVA 10. Gran Turismo (Bp.blogspot.com 2012)

Simulaatio-ajopelit yrittävät tarjota pelaajille todellisen ajotuntuman. Nämä pelit noudattavat tiukasti fysiikan lakeja, jotka vaikuttavat ajotuntumaan oikeassa elämässä. Tällaiset pelit antavat pelaajalle paljon mahdollisuuksia rassata menopelejään, kuten säätää vaihteistoa tai muuttaa renkaiden ilmanpainetta. (Scattergood 2005, 191.)

2.14 Simulaatiopelit

Simulaatiopelit saapuivat markkinoille kolmessa muodossa. Ensin tulivat kilpa-ajo-simulaatiot (ralli, moottoripyörät, Formula 1), jotka olivat toimintapelien malliin nopeatempoisia, mutta keskittyivät enemmän kilpailuun. Merkittävä ero toimintapeleihin on siinä, että toimintapeleissä tavoitteena on aina tuhota vastustaja. (Lecky-Thompson 2007, 25.)

Kilpa-ajosimulaatiopelit toivat kilpailukonseptin, yksi pelaaja vastaan monta pelaajaa. Toiset pelaajat saattoivat olla tietokoneen ohjaamia, mutta niillä kaikilla oli sama tavoite. Tämä viitoitti tietä urheilumoninpeleille, joissa pelataan ihmistä vastaan. (Lecky-Thompson 2007, 25.)

Toinen simulaatiopelityyppi harppaa takaisin ”pelaaja vastaan vihollinen” –teemaan, tällä kertaa realistiseen sotatyyliin. Tästä esimerkkinä peli nimeltä *Steel Talons* (kuva 11*)*. Idea näiden pelien takana on laittaa pelaaja esimerkiksi helikopterin ohjaimiin ja antaa mahdollisimman realistinen kuva siitä, millaista helikopterilla taistelu on. Pelattavuus ei rajoittunut enää ylös, alas, oikealle tai vasemmalle –ohjausmahdollisuuksiin. Tällä kertaa ohjaaminen mahdollisti nopeuden tai siipien kulman säätämisen. (Lecky-Thompson 2007, 26.)

KUVA 11. Steel Talons (Arcade-history.com 2012)

Kolmantena esittelen strategia-simulaatiopelit. Nämä pelit vaativat pitkäjänteisyyttä ja ajattelua. Pelien teemat vaihtelevat urheilusta sairaalan ylläpitoon ja kaupunkien rakentamiseen. Ideana pelien taustalla on saada aikaiseksi kannattava liiketoimi. Kaupungin rakennuspeleistä eräs tunnetuimpia on SimCity 2000 (kuva 12), joka on pelisarjan toinen peli. (Building Simulation Games 2012.)

KUVA 12. SimCity 2000 (Myabandonware.com 2012)

3 PELIALUSTAT

Eri tahot tarjoavat jatkuvasti kuluttajille erilaisia pelialustoja ja pelejä, joista valita. Jos tarkoituksena on vain pelata pelejä, tulee tietokone kalliimmaksi kuin pelkkä pelikonsoli, vaikka konsoleiden elinkaaret ovat lyhyempiä. Jos taloudessasi on tietokone, useat suositut konsolipelit ovat saatavilla myös tietokoneelle. Pelit kuitenkin testataan molemmilla alustoilla. (Choosing The Video Game Platform Best For You 2012.)

Pelialustoja on erilaisille pelaajille ja peleille. Uudet pelit tulevat yleensä konsoleille ensin. Toimintapelit kuten *Splinter Cell* ja *Grand Theft Auto* saivat suosiota ensin konsoleilla, ennen kuin tulivat PC-maailmaan. Urheilupelit ovat konsoleilla suosituimpia, sillä samanlaiseen pelikokemukseen ei pääse tietokoneilla. Niiden viehätys perustuu kilpailuun ja toista ihmistä vastaan pelaamiseen. Tietokone taas päihittää konsolit strategia- ja FPS-peleissä, joissa tarvitaan nopeatahtista hiirellä klikkailua. Moninpelaaminen Internetissä on monessa pelissä mahdollista. Nykyisin käytännössä jokaisessa tietokoneessa on Internet-yhteys, joten tietokoneella pelattavat

moninpelit ovat suositumpia kuin konsolivastineensa. Konsolipuolella on tapahtunut hidasta edistymistä Xbox:n ollessa trendin kärjessä. Huono puoli konsolilla Internet-pelaamisessa on siitä koituvat lisämaksut. (Choosing The Video Game Platform Best For You 2012.)

3.1 Tietokone

Tietokone on vanhin elektroninen alusta, jolle on luotu pelejä. Tietokoneen osat ovat helposti päivitettävissä, pelien ohjausmahdollisuudet ovat alustoista laajimmat, grafiikan suorituskyky on huippuluokkaa ja äänimaailma todentuntuista. Vanhat tietokoneet, kuten Commodore 64 ja Atari 2600 sisälsivät ohjainportin, johon saattoi kytkeä erillisen peliohjaimen. Useat pelit tukivat sekä näppäimistöä että *joystick*-ohjainta. Koska hiiriohjauksesta on tullut tietokoneille standardi, ne ovat myös peleissä suosittu ohjaustapa. Monet yritykset suunnittelevat hiiriä nimenomaan pelikäyttöön. *Joystick*-ohjaimia käytetään vieläkin USB-portteihin kytkettynä. (Giantbomb.com 2012.)

Tietokonepelaamisen ensimmäisiä askeleita olivat yksittäisten ohjelmoijien tekemät *arcade*-peli-kloonit. Pelien julkaisijat tai jopa itse ohjelmoijat jakoivat tuotoksensa diskettien avulla. Disketit laitettiin Minigrip-pussiin vaatimattoman käyttöohjeen kanssa. Suosituimmat pelityypit, jotka ovat kehittyneet roimasti tällä alustalla, ovat roolipelit, reaaliaikaiset strategiapelit ja ensimmäisen personaan ammuntapelit. (Giantbomb.com 2012.)

Roolipelit saivat runsaasti vaikutteita *Dungeons and Dragons* –paperiroolipeliltä. Roolipelien maailmaan ilmestyi nopeasti sellaisia nimekkäitä tulokkaita kuten ASCII-merkkiseikkailu *Rogue* ja toimintaroolipeli *Diablo*. Ensimmäiset pelit olivat suoraviivaisia, joissa edettiin tasolta tasolle. Liikkumisen ja seikkailun vapaus oli seikka, joka kehitti niitä nopeasti eteenpäin. (Giantbomb.com 2012.)

RTS-pelit (Reaaliaikainen strategia), kuten *Starcraft* ja *Command & Conquer*, edustavat pelityyppiä, joka on vahvasti sidottu PC-pelaamiseen. RTS vaatii nopeatempoista hiirityöskentelyä, joka tekee konsoliversioiden kehityksestä todella vaikeaa. Pelityyppi on kokenut suuren muutoksen, mutta peruselementit ovat aina

läsnä: resurssien kerääminen, armeijan kasaaminen ja sen lähettäminen taisteluun. (Giantbomb.com 2012.)

FPS-pelien suosiota ei voi kiistää. Id Softwaren Wolfenstein 3D räjäytti FPS-pelimarkkinat kukoistukseen. Tätä pelityyppiä on tehty myös konsoleille, mutta tietokoneella pelaaminen on suositumpaa, koska hiiri mahdollistaa tarkemman tähtäämisen. Maamerkkejä FPS-pelien viidakossa ovat Doom, Quake, Half-Life ja Counter-Strike. (Giantbomb.com 2012.)

3.2 Konsolit

Vaikka tietokoneet ovatkin helposti päivitettävissä loistaviksi pelikoneiksi, on suuri osa niistä perustoimistokäytössä, eikä pelien sulavaa toimivuutta välttämättä oteta huomioon. Tässä mukaan astuvat konsolit. Konsolit ovat televisioon kytkettäviä laitteita, jotka toistavat videopeliä. Videopelikonsoli-termiä käytetään kuvaamaan laitetta, jonka ihmiset ostavat yksinomaan pelaamista varten TV-ruudun kautta. (Console game 2012.)

Ensimmäinen kaupallinen konsoli, *Magnavox Odyssey,* oli jotakin uutta ilmestymisvuonnaan 1971. Se käytti alkeellisia grafiikoita, ja värillisiä kalvoja, jotka hoitivat taustagrafiikan virkaa. Tälle konsolille tehtiin urheilu-, opetus-, sota-, seikkailu-, pulmanratkonta-, uhka- ja ajopelejä. Pelaajat valitsivat pelin asettamalla laitteeseen ”tyhmän kortin”, joka uudelleen ohjelmoi laitteen käyttämällä hyppylankoja. Jotkin pelit vaativat useita kortin vaihtoja ja pelaajien tuli tietää, milloin kortti piti vaihtaa. (Bergeron 2006, 12.)

Nintendo, joka oli alkujaan japanilainen pelikortteja valmistava yhtiö, julkaisi oman konsolinsa vuonna 1985. Yhdysvalloissa se julkaistiin nimellä Nintendo Entertainment System ja Japanissa nimellä Famicom. Jälleenmyyjät olivat aluksi epäileväisiä uuden konsolin suhteen, koska videopelimarkkinat olivat hiljattain pudonneet rajusti. Nintendo-pelit *Super Mario Bros, Metroid* ja *The Legend of Zelda* kuitenkin palauttivat kuluttajien luottamuksen ja Nintendosta tuli eräs parhaiten myyneistä konsoleista. (A History of Video Game Consoles 2012.)

Nintendo Entertainment Systemin julkaisusta oli kulunut neljä vuotta, kun yhtiö toi markkinoille Game Boy -käsikonsolin vuonna 1989. Siinä oli 8-bittinen suoritin kuten aikaisemmassakin konsolissa ja mustavalkoinen LCD-näyttö. Pian Game Boy:lle julkaistiin oma Tetris-klooni, joka nosti Game Boy:n myynnin pilviin. (A History of Video Game Consoles 2012.)

Sega-yhtiön Genesis-konsoli oli varteenotettava kilpailija Nintendolle 1980-luvun lopulla. Se sisälsi 16-bittisen tietokonesirun, joka kykeni käsittelemään kaksi kertaa enemmän tietoa kuin sen aikaiset kilpailijansa. Se tarjosi enemmän värejä, nopeampaa toimintaa ja paremman äänimaailman. Konsolille tehty *Sonic the Hedgehog* löi itsensä läpi ja Sega hallitsi pitkään 16-bittisten konsolien markkinoita. (SEGA Corporation 2012.)

Segan 16-bittisen iskun jälkeen Nintendo pääsi takaisin jaloilleen Super Nintendo Entertainment System –konsolinsa myötä. Hitaasta alusta huolimatta Nintendo otti Segan Genesiksen kiinni. SNES oli teknisesti hieman kehittyneempi ja valmiit pelinimikkeet *Super Mario Bros* ja *The Legend of Zelda* takasivat niiden 16-bittisille versioille valmiin kohdeyleisön. 1990-luvun puoliväliin mennessä Nintendo oli suurin valmistaja 16-bittisten konsoleiden markkinoilla. (A History of Video Game Consoles 2012.)

32-bittisiin alustoihin siirryttäessä Sony tuli mukaan markkinoille vuonna 1995 PlayStation-konsolillaan, jonka pelit korostivat 3D-maailmaa. CD-ROM-teknologian myötä pelien hinnat putosivat rajusti. Suosituiksi peleiksi nousivat *Resident Evil, Gran Turismo, Tekken, Crash Bandicoot, Metal Gear Solid* ja Electronic Arts –pelitalon urheilupelit. (A History of Video Game Consoles 2012.)

Kasettipelien viimeinen suuri myyntivaltti oli Nintendo 64. Vaikka kasettien valmistus oli CD-ROM-levyjä kalliimpaa, ne latautuivat nopeammin ja näin PlayStation-peleistä tutuissa latausruuduista päästiin eroon. Kaseteille voitiin myös tallentaa peliä pelatessa, joten erillistä muistikorttia ei tarvittu. Nintendo 64 hävisi PlayStationille pelien määrässä, mutta onnistui silti myymään *Super Mario 64- ja The Legend of Zelda: Ocarina of Time* –pelejä kiitettävästi. (A History of Video Game Consoles 2012.)

Vuonna 2000 oli aika Sonyn 128-bittiselle PlayStation 2 –konsolille. Sillä oli mahdollisuus pelata myös ensimmäisen PlayStationin pelejä ja toistaa DVD-elokuvia. Suositut pelit *Grand Theft Auto, Metal Gear Solid ja Final Fantasy* nostivat PlayStation 2:n suosituimmaksi 128-bittiseksi konsoliksi. (A History of Video Game Consoles 2012.)

Seuraavana vuonna Microsoft tuli markkinoille ensimmäisellä omalla konsolillaan – Xbox:lla. Microsoft käytti tietokoneissa käytettyä teknologiaa konsolinsa rakentamiseen, mikä mahdollisti 128-bittisiä konsoleita paremman suorituskyvyn. Tästä huolimatta Xbox ei yltänyt Sonyn PlayStation 2:n myyntilukuihin. Xbox-konsolin suosituimmaksi peliksi nousi *Halo: Combat Evolved*. (A History of Video Game Consoles 2012.)

Samaan aikaan Xbox-konsolin kanssa Nintendo julkaisi GameCuben, joka oli Nintendon ensimmäinen konsoli, joka ei käyttänyt kasetteja pelien lataamiseen. Pelit ladattiin 7,6 cm halkaisijaltaan olevilta levyiltä, joiden kapasiteetti oli 1,5Gb. GameCube oli mahdollista kytkeä Nintendon käsikonsoleihin, jolloin niiden pelejä pystyi pelaamaan television ruudulta. Nintendolla oli omat menestyspelinsä (*Mario, Zelda ja Metroid*), mutta ulkopuoliset pelivalmistajat siirtyivät lopulta Sonyn ja Microsoftin alustoille. (A History of Video Game Consoles 2012.)

2000-luvun käsikonsolitaistelussa nähtiin ensin Game Boy Advance, jolla pystyi pelaamaan sekä Game Boy- että Game Boy Color –konsoleiden pelejä. LCD-ruutu näytti runsaat 32 000 väriä. Vuonna 2004 Nintendo DS –konsoli yritti tuoda kannettaviin konsoleihin enemmän laskentatehoa. DS:ssä on kaksi ruutua ja ruudun kosketus-ohjaus -mekanismi. Sonyn vastaus näihin kahteen konsoliin oli PlayStation Portable alkuvuodesta 2005. Nintendo DS:n tavoin PSP tuki langatonta tiedonsiirtoa ja sisälsi korkealaatuiset grafiikat. (A History of Video Game Consoles 2012.)

3.3 Mobiililaitteet

Mobiililaitteet eivät ole pelkästään enää vain soittamista varten; niistä on tullut valmiita viihdekeskuksia sisäänrakennettujen musiikkisoitinten ja pelien kera. Olemme nyt tulleet pitkän matkan siitä, mistä *Snake* aloitti vuonna 1997. (Mobile Game Development 2011.)

Kasvaneen suoritustehon ansiosta mobiilipeleistä on tullut suosittua ajanvietettä. Vuonna 1997 Nokian 6110-puhelimessa julkaistu *Snake*-peli aloitti uuden aikakauden pelaamisessa. Nykyään silmiämme hemmotellaan korkean resoluution grafiikalla, mutta alkuperäisessä *Snake*:ssa oli vain muutama musta pikseli, jotka liikkuivat vihreiden pikseleiden päällä. Tämä tapahtui kuitenkin ennen kuin WAP (Wireless Application Protocol) esiteltiin, joka oli seuraava askel mobiilipelien evoluutiossa. WAP mahdollisti yhteyden Internet:iin ja näin ollen pelaajat pystyivät pelaamaan puhelimellaan moninpelejä. WAP:ia ei optimoitu nopea-tahtisiin peleihin, joten ihmiset siirtyivät pelaamaan vuoropohjaisia pelejä. (History of Mobile Gaming 2011.)

WAP:n rajoitteiden takia kaksi uudempaa teknologiaa vastasivat pelaajien kysyntään: J2ME (Java 2 Micro Edition) ja BREW (Binary Runtime Environment for Wireless). J2ME menestyi hyvin Euroopassa ja BREW Pohjois- ja Etelä-Amerikassa sekä Aasiassa. Nämä kaksi alustaa vaikuttivat erittäin paljon mobiilipelaamiseen ja samoin aikoihin, vuoden 2001 paikkeilla, peleihin ilmestyivät myös värit. (History of Mobile Gaming 2011.)

Vuonna 2003 Nokia näki potentiaalia pelaamiseen erikoistuneilla puhelimilla ja julkaisi N-Gage-mallin, joka oli samaan aikaan puhelin ja käsikonsoli. Kokeilu ei kuitenkaan tuottanut tulosta ja se oli Nokialle suuri taloudellinen katastrofi. On spekuloitu, että Sonyn samoihin aikoihin julkaisema PSP tai N-Gage:n muhkea hinta karkottivat kuluttajat. Kriitikot tyrmäsivät puhelimen täysin ja kuluttajatkin pilkkasivat sitä, ja lopulta myös itse Nokia hylkäsi mallin. (History of Mobile Gaming 2011.)

Ennen Nokian N-Gage-puhelinta markkinoilla ei ollut julkaistu yhtään pelaamiseen suunniteltua puhelinta. Perinteisesti parhaat pelaamisen soveltuvat puhelimet olivat niitä, joissa oli paras suorituskyky. Vuonna 2008 Nokia julkaisi N-Gage-pelialustan, joka oli valmiiksi asennettu ohjelmisto useissa Nokian puhelimissa. Tästäkään

projektista ei tullut menestystä, vaan se jäi Applen sovelluskaupan varjoon. (History of Mobile Gaming 2011.)

Vuonna 2007 Apple julkaisi kosketusnäytöllisen iPhone-puhelimen, joka yhdessä App Store:n kanssa laukaisi massahysterian kuluttajien keskuudessa. Yhtäkkiä ihmisillä oli helppo väline ladata pelejä suoraan Internet-sivuilta. (A Brief History of Mobile Games 2009.) Apple muutti mobiililaitteiden tulevaisuudennäkymät. iPhone:n iOS-käyttöjärjestelmä on optimoitu peleille ja Applen markkinoille tulon jälkeen mobiilipelaaminen lähti nousuun. Apple Game Center mahdollistaa pelaamisen sekä ystävien että tuntemattomien kanssa. Pelejä on saatavilla lukematon määrä lautapeleistä toimintaan. Apple tarjoaa kuluttajille mahdollisuuden ostaa pelit suoraan pelintekijöiltä. Pelintekijät voivat julkaista pelin suoraan App Store:n, jossa kuluttajat voivat tehdä ostoksensa. Uusi ostamismalli suosii siis molempia osapuolia, kun välissä ei ole erillistä julkaisijaa. (History of Mobile Gaming 2011)

3.4 Selain

Kaikki alkoi 1990-luvulla, kun World Wide Web avasi ovet rajoittamattomiin sovellusmahdollisuuksiin ja selainpelit olivat yksi niistä (Browser Games: Basics and History, 2012). Selainpeli on nimensä mukaisesti peli, jota pelataan selaimella. Selainpelejä alkoi ilmestyä 1990-luvun lopulla, jolloin ne perustuivat DHTML-merkintäkieleen. Merkintäkieltä käytettiin yleisesti alasvetovalikoiden ja kuvien esittämiseen. (History of Web Browser Games, 2009.)

Selainpelit erottuivat muista alustoista käytännössä niin, etteivät ne vaatineet asiakkaan puolelta muita ohjelmia kuin itse selaimen. Nykyisin pelejä on suunnaton määrä, ja osa niistä vaatii selaimeen liitännäisen, kuten *Adobe Flash*:n tai *JavaScript*:n. Liitännäiset ovat kuitenkin helposti saatavilla ja asennettavissa. Pelit ovat yleensä maksuttomia, mutta niissä on kuitenkin mahdollisuus ostaa harvinaisia tavaroita tai vauhdittaa pelin etenemistä. (History of Web Browser Games, 2009.) Selainpeliä ei erota muista peleistä pelin tekniset yksityiskohdat, vaan se, millaisilla puitteilla sitä voi pelata. (What is a Browser-Based Game? 2012).

Selaimella voi pelata useita erityyppisiä pelejä. Yksinkertainen peli voi olla yksin pelattava eikä vaadi juurikaan aikaa. Toisaalta laajemmat pelit voivat olla sosiaalinen

tapahtuma muiden pelaajien kanssa ja vaatia aktiivisuutta sovittuina aikoina. On tärkeää ymmärtää, etteivät moninpelit ja selainpelit kuitenkaan ole synonyymejä. (What is a Browser-Based Game? 2012.)

Vanhin selainpeli on vuoropohjainen strategia *Earth: 2025*, jonka kehitti Mehul Patel vuonna 1996. Tällä pelillä oli kolme pääserveriä vuonna 2000: perus, turnaus ja vapaa kaikille. Se kuitenkin lakkautettiin vuonna 2009. (Browser Games: Basics and History, 2012) Tunnetuin vanhempi selainpeli on kuitenkin *Utopia* vuodelta 1999, joka on sekin vuoropohjainen strategia. Vuonna 1999 julkaistiin kaksi peliä, *Hattrick* ja *Runescape*, joita pelaa vieläkin satojatuhansia ihmisiä. Peli nimeltä *Evony* oli ensimmäinen suurempi selainpeli, jossa täytyi kerätä resursseja: louhia kaivoksia, kaataa metsää ja kasvattaa viljaa. Kun resurssit olivat kunnossa, pelaaja pääsi rakentamaan omaa kaupunkia ja kokoamaan armeijaa. (History of Web Browser Games, 2009.) *Evony:*a pelaa yhä yli 18 miljoonaa ihmistä 162:sta eri maasta (Browser Games: Basics and History, 2012).

Selain on alusta, jonka valitsin pelin kehitykseen tuttujen työkalujen takia. Seuraavaksi siirryn pelimoottorin suunnitteluun ja tähän nimenomaiseen peliin, jota työstin.

4 PELIMOOTTORIN SUUNNITTELU

Projektina oleva *A Viking Tale: The Runes of Memory* on selainpohjainen seikkailu/roolipeli, jota olemme ideoineet ja kehittäneet opiskelun ohessa. Aihe tuntui sopivalta myös opinnäytetyöksi. Se kuuluu esittelemistäni pelityypeistä johonkin seikkailupelin ja roolipelin välimaastoon. Peligrafiikoita emme ole tehneet itse. Rautalankamallit ja luonnokset piirsin Windows-käyttöjärjestelmän Paint-ohjelmalla.

Pelimoottorin käsite on häilyvä. Se on kuin auton moottori: se saa pelin käyntiin. Kuitenkin pelin sisällön ja pelimoottorin raja on joskus vaikeasti hahmotettava. Esimerkiksi, onko auton ilmastointi osa moottoria vai ei? Yleisesti ottaen pelimoottorin konsepti on yksinkertainen. Se kokoaa kaikki ”konepellin” alla tapahtuvat toiminnot, kuten pelimaailman renderöinnin, fysiikkamoottorin ja pelaajan syötteiden käsittelyn. Tämän ansiosta artistit ja käsikirjoittajat voivat keskittyä tekemään yksityiskohtia, jotka tekevät pelistä ainutlaatuisen. (Ward 2008.)

Alusta oli selvää, että tahdoin toteuttaa pelin PHP-ohjelmointikielellä MySQL-relaatiotietokantaa hyväksi käyttäen. Ajatuksena oli laittaa omat taidot likoon ja ottaa selvää, kuinka hyvin nämä työvälineet soveltuivat pelin tekoon. Tahdoin tuoda selaimeen 1990-luvun seikkailupelihenkeä hieman erilaisella käyttöliittymällä. Taustat suunniteltiin ikään kuin maalauksiksi tai tilannekuviksi, joissa olisi hiirellä klikattavia kohteita. Klikkauksen aiheuttamat toiminnot vaihtelivat taisteluista kaupantekoon, joihin kaikkiin piti suunnitella omat käyttöliittymänsä ja tiedon kuljetus. Omana lisämausteenaan mukana kulki juonen ja dialogien suunnittelu sekä esineiden ja hahmojen nimeäminen.

Lähdin tekemään peliä kuin mitä tahansa nettisivua, ilman sen suurempaa rakenteellista suunnittelua tai mietteitä olio-ohjelmoinnista. Ainoa asia, josta tein ennakkosuunnitelmia, oli SQL-taulut ja niiden keskinäiset suhteet. Ilman aiempaa pelintekokokemusta asiat lähtivät käyntiin kronologisessa järjestyksessä. Loin kirjautumisen ja rekisteröitymisen, ja testasin ne toimiviksi; useampia samoja käyttäjänimiä ei saanut luoda ja salasanan tuli olla vähintään 8 merkkiä pitkä. Tietoturvan otin huomioon jo hyvin varhaisessa vaiheessa.

HTML 5:n mukana tullut *Webstorage* ei tullut mukaan projektiin, koska halusin pelin toimivan myös vanhemmilla selaimilla. Sama syystä myös *canvas*-elementti puuttuu. Pelin tekninen rakenne ei ollut aluksi ollenkaan selvä, joten en rakentanut ”ydintä” luokkaan. Projektin edetessä toimintoja kuitenkin kertyi ja ajatus yhdestä luokasta ei tuntunut enää huonolta idealta. Sijoitin kirjoittamani toiminnot funktioiksi luokan sisään ja loin oliosta ilmentymän sisään kirjautumisen yhteydessä.

4.1 Rekisteröityminen

Roolipelien luonteeseen kuuluu, että pelisessiot ovat mahdollisesti useiden tuntien mittaisia ja edistyminen on hidasta. Tästä syystä rekisteröityminen on välttämätöntä, että hahmon edistyminen saadaan helposti yksilöityä. Seuraavaksi näytän, kuinka aluksi tein rekisteröitymisen. Loin *MySQL*-tietokantaan *users*-nimisen taulun, jonka esittelen kuvassa 13.

user_id	user_nick	user_password	user_email	user_money	active_quest_id	quest_status
4	seppo	Seppo678	seppo@testimaili.fi	220	2	1

KUVA 13. *users*-taulu MySQL-tietokannassa

Sarakkeen *user_id* arvo lisätään jokaiselle pelaajalle automaattisesti *AUTO INCREMENT* -valinnalla. Tämä mekanismi yksilöi jokaisen tietueen eli taulun rivin. Oletusasetuksilla arvo alkaa aina ykkösestä. (W3Schools.com 2012.) Tässä tapauksessa olin tehnyt etukäteen jo muutaman testikäyttäjän, jotka kuitenkin poistin myöhemmin. Tästä syystä seppo-nimimerkillä on *user_id*-arvona numero 4.
Tietokantayhteys avataan PHP:ssa esim. seuraavalla koodilla.

```
$yhteys=mysql_connect("localhost","root","")
or die("Cannot connect to database");
mysql_select_db("vikingtale",$yhteys)
or die ("Cannot select database.");
```

KUVA 14. Tietokantayhteyden avaaminen

Kuvan 14 mukaisen ohjelmakoodin laitoin aluksi *globals.php*-tiedostoon. Kyseisen tiedoston pystyy liittämään toiseen tiedostoon *include(”globals.php”);* -komennolla. Näin tietokannan avaus saadaan mukaan jokaiseen kooditiedostoon mukaan. Vastoin koulussa opittuja tietoturvakäytäntöjä, en määrittänyt kotikoneeni SQL-tietokantaan salasanaa, koska pidin tietoturvariskiä mitättömänä.

Käyttäjän syöttämiä rekisteröintitietoja varten kirjoitin lomakkeen, joka näkyy kuvassa 15.

```
<form method="post" action="<?php echo $_SERVER["PHP_SELF"]; ?>">
<tr>
    <td width=160>Username:</td>
    <td><input name="user"></td>
</tr>
<tr>
    <td width=160>Password:</td>
    <td><input name="password"></td>
</tr>
<tr>
    <td width=160>Password again:</td>
    <td><input name="password2"></td>
</tr>
<tr>
    <td width=160>E-mail:</td>
    <td><input name="email"></td>
</tr>
<tr>
    <td width=160><input type="submit" name="tallenna" value="Rekisteröidy"></td>
</tr>
<tr>
    <td colspan="2">
        <?php
            if(isset($_SESSION["varoitus"]))
            {

                echo "Error! ".$_SESSION["varoitus"];
                $_SESSION["varoitus"] = null;

            }
        ?>
    </td>
</tr>
</form>
```

KUVA 15. Rekisteröitymislomake

Lomakkeelle käyttäjän tulee antaa käyttäjänimi, salasana, salasanan varmistus ja sähköpostiosoite. Se, että vaadittiin sähköpostiosoitetta, perusteltiin aluksi varmistusviestin lähetyksen takia, mutta sitä en koskaan ehtinyt toteuttaa. Salasana vaaditaan kahdesti kirjoitusvirheiden välttämiseksi.

Lomakkeen tyyppi on POST ja toimintakohde PHP-tiedosto itse. Tämä tarkoittaa sitä, että Rekisteröidy-nappia painaessa lomake lähettää tiedot ”itselleen”. Koska ohjelmakoodi suoritetaan ilman erillisiä ohjausrakenteita ylhäältä alaspäin, sijoitin lomaketta ennen koodin, joka reagoi näihin lähetettyihin tietoihin. Koodin ensimmäinen lohko on kuvassa 16.

```
if(isset($_POST["tallenna"]))
{
$user = mysql_real_escape_string($_POST["user"]);
$password = mysql_real_escape_string($_POST["password"]);
$password2 = mysql_real_escape_string($_POST["password2"]);
$email = mysql_real_escape_string($_POST["email"]);

if($user == "" || $password == "" || $password2 == "" || $email == "")
{
    $_SESSION["varoitus"] = "Enter info to all input fields!";

}
```

KUVA 16. Rekisteröitymisen käsittelemisen ensimmäinen lohko

Eri ohjelmointikielissä lohkoille on syntaksista riippuen erilaisia esitystapoja. PHP:ssa lohkot eritellään { } –merkein (Johdatus PHP-kieleen 2012). Ensimmäinen *if*-lause tarkistaa, onko lomakkeen *submit*-nappia painettu. Tämän napin yksilöin antamalla sille nimen ”tallenna”. Käyttäjän syöttämät tiedot tarkistin *mysql_real_escape_string*-funktiolla. Se poistaa merkkijonosta erikoismerkit ja ottaa käytetyn merkistön huomioon. Tämä on yksi tapa estää SQL-injektio. (PHP.net 2012.)

Seuraavassa *if*-lauseessa tarkistin, että jokaiseen kenttään on varmasti syötetty jokin arvo. Jos jokin arvo puuttuu, päätin käyttää SESSION-muuttujaa virheilmoituksen näyttämiseen, koska sessiomuuttuja säilyttää arvonsa, vaikka käyttäjä siirtyisi dokumentista toiseen. Näin voin käyttää yhtä sessiomuuttujaa kaikkien virheilmoitusten näyttämiseen, eikä jokaiselle tarvinnut määrittää omaa muuttujaansa. SESSION-muuttuja vaatii toimiakseen session_start(); -komennon jokaisen sessiota käyttävän PHP-tiedoston alkuun. Se joko jatkaa aiemmin aloitettua sessiota tai aloittaa uuden perustuen session tunnistukseen GET ja POST –pyynnöissä. (PHP.net 2012.)

Kun edellisestä *if*-lauseesta selvittiin eteenpäin, kirjoitin *sql*-lauseen tarkistamaan, onko samalla nimimerkillä jo tietokannassa käyttäjää. Jos hakutuloksissa palautui yksi tai useampi rivi, sessiomuuttuja nimeltä ”varoitus” saa uuden arvon. Muussa tapauksessa siirrytään *else*-lohkoon, johon kirjoitin funktiokutsun salasanan validointiin. Koodi esitellään kuvassa 17.

```
else
{
    $sql_lause = "SELECT * FROM users WHERE user_nick = '".$user."'";
    $haku = mysql_query($sql_lause);
    if(mysql_num_rows($haku) >= 1)
    {
        $_SESSION["varoitus"] = "The username is unavailable.";

    }
    else {
    $_SESSION["varoitus"] = passwordCheck($password, $password2);
    }
} //else
```

KUVA 17. Käyttäjänimen tarkistaminen ja salasanan validoinnin kutsu

PasswordCheck on PHP-tiedoston ulkopuolinen funktio, jonka lisäsin koodiin tiedoston alussa aikaisemmin esittelemälläni *include*-komennolla. Tällä kertaa parametrina on funktiot sisältävä *functions.php*-tiedosto. Valmiin koodin löysin Eukhost.com-sivuston foorumilta, joten sitä ei tarvinnut kirjoittaa alusta asti itse (Eukhost.com 2006). Koodi tarkistaa, että salasana sisältää vähintään yhden ison- ja yhden pienen kirjaimen, on vähintään 6 merkkiä- ja enintään 20 merkkiä pitkä ja sisältää vähintään yhden numeron. Jos salasana läpäisee seulan, funktio palauttaa arvon 1, muussa tapauksessa se palauttaa virheilmoituksen, jonka sijoitin sessiomuuttujaan.

Seuraavaksi kirjoitin *sql*-lauseen, joka lisää käyttäjän syöttämät tiedot tietokantaan ja alustaa sessiomuuttujan takaisin tyhjäksi. Rajapintafunktio *mysql_query* suorittaa *sql*-muuttujaan sijoitetun hakulauseen ja asettaa tuloksen *$tulos*-muuttujaan. Käyttäjän lisäämisen jälkeen kirjoitin hakulauseen hakemaan juuri lisätyn käyttäjän tietokannasta, että sain uuteen sessiomuuttujaan käyttäjän *user_id*:n talteen. Haun yhteydessä käytin *mysql_fetch_array*-funktiota, joka palauttaa assosiatiivisen taulukon taulun tiedoista (PHP.net 2012). Tämän jälkeen käyttäjä ohjataan seuraavalle sivulle *header(”Location: userpage.php”)* -komennolla. Koodi näkyy kokonaisuudessaan kuvassa 18.

```
if($_SESSION["varoitus"] == 1)
{
    $sql = "INSERT INTO users (user_nick,user_password,user_email)
    VALUES ('$user','$password','$email');";
    mysql_query($sql);

    $_SESSION["varoitus"] = null;

    $sql = "SELECT * FROM users WHERE user_nick='".$user."';";
    $tulos = mysql_query($sql);
    $rivi = mysql_fetch_array($tulos, MYSQL_ASSOC);
    $_SESSION["user_id"] = $rivi["user_id"];
    header("Location:userpage.php");
}
```

KUVA 18. Uuden käyttäjän lisääminen tietokantaan

Kun tein päätöksen sijoittaa funktiot luokan sisään, piti myös rekisteröityminen suunnitella uudestaan. Laitoin sen funktioksi, joka ottaa vastaan käyttäjänimen, salasanan, salasanan varmistuksen ja sähköpostiosoitteen. Funktiota kutsutaan seuraavasti:

$reg = $_SESSION["GameEngine"]->Registration($_POST["user"], $_POST["password"], $_POST["password2"], $_POST["email"]);

Jos rekisteröityminen onnistuu, funktio palauttaa arvon *true*. Samalla koodi tekee *gear*-tauluun pelaajalle viisi riviä, joihin tallennetaan hänen aloitusvarusteensa. Tämä oli seikka, joka tuli mieleen vasta projektin puolivälissä.

4.2 Kirjautuminen

Kirjautumisestakin tein kaksi erilaista versiota. Tässä on ensimmäinen. Loin samantyyppisen lomakkeen, mutta tällä kertaa kysytään vain käyttäjänimeä ja salasanaa. Myös tämä *html-form* kutsuu itseään *submit*-nappia klikatessa. Kirjautumisen käsittelevän koodin alku on kuvassa 19.

```php
if(isset($_POST["kirjaudu"]))
{
    $user = mysql_real_escape_string($_POST["user"]);
    $password = mysql_real_escape_string($_POST["password"]);

    if($user != "" && $password != "")
    {
        $sql = "SELECT * FROM users WHERE user_nick='".$user."'
        AND user_password='".$password."';";
        $vastaus = mysql_query($sql);
        $rivi = mysql_fetch_array($vastaus, MYSQL_ASSOC);
        $count = mysql_num_rows($vastaus);
```

KUVA 19. Kirjautumisen käsittelevän koodin alku

Kirjaudu-nappia painaessa koodi tarkistaa käyttäjänimen ja salasanan aikaisemmin esittelemälläni *mysql_real_escape_string*-funktiolla. Sisempi *if*-ehtolause tarkistaa, etteivät käyttäjänimi- ja salasana kentät ole tyhjiä. Seuraavaksi kirjoitin *sql*-haun kaivamaan tietokannasta tietuetta, joka sisältää kirjautumassa olevan käyttäjän käyttäjänimen sekä salasanan. Haettu rivi sijoitetaan kuvaavasti *$rivi*-muuttujaan. Samalla myös varmuuden vuoksi haetaan palautettujen tietueiden määrä *mysql_num_rows*-funktiolla, sillä niitä pitäisi palautua vain yksi. Kuvassa 20 esittelen koodin seuraavan osan.

```php
if($count == 1)
{
    if($rivi["user_password"] == $password) {
    $_SESSION["login"] = true;
    $_SESSION["user_id"] = $rivi["user_id"];
    header("Location:userpage.php");
    }
    else {
    $_SESSION["varoitus"]  = "Wrong password or username.";
    }

}
else
{
    $_SESSION["varoitus"] = "Account could not be found!";
}
```

KUVA 20. Kirjautumis-koodin toinen osa

Jos haettujen rivien määrä täsmää lukuun yksi, niin koodi tarkistaa vielä, että käyttäjän syöttämä salasana ja tietokannasta löytynyt salasana täsmäävät. Muussa tapauksessa toiminta ohjataan *else*-lohkoon, joka ilmoittaa, ettei käyttäjätiliä löydy, tai että käyttäjätunnukset ovat virheelliset. Tämän jälkeen sessiomuuttuja ”*login*” asetetaan *boolean*-arvoon *true*, jolla tarkistetaan käyttäjän sisään kirjautuminen eri kooditiedostojen alussa. Kuten kirjautumisessa, tässäkin otetaan *user_id* talteen ja ohjataan käyttäjä seuraavalle sivulle.

Luokkaratkaisuna kirjautumisfunktiota kutsutaan HTML-lomakkeen *submit*-nappulan painamisen jälkeen. Koodi on kuvassa 21.

```
if(isset($_POST["kirjaudu"]))
{
    $user = mysql_real_escape_string($_POST["user"]);
    $pw = mysql_real_escape_string($_POST["password"]);
    // Ajetaan pelimoottorin login-funktio
    $login = $_SESSION["GameEngine"]->Login($user, $pw);

    // Jos login onnistuu, ohjataan pelaaja peliin
    if($login)
    {
        $_SESSION["GameEngine"]->GetMoney();
        $_SESSION["GameEngine"]->GetGameState();
        $_SESSION["game_state"] = 1;
        $_SESSION["login"] = TRUE;

        header("Location: game.php");
    }
    else
    {
        echo "Invalid username or password.";
    }
}
```

KUVA 21. Kirjautuminen luokkaratkaisuna

Jos sisään kirjautuminen onnistuu, haetaan pelin etenemisen tiedot sessioon. Tämän jälkeen pelaaja ohjataan ensimmäiseen kohtaukseen.

4.3 Pelialueen piirtäminen

Pelialue koostuu kohtauksista (sceneistä), jotka ovat piirrettyjä tilannekuvia. Tietokannassa jokainen kohtaus on oma tietueensa. Yksi tietue koostuu kolmesta sarakkeesta: *scene_id*, *scene_name* ja *scene_content.* Pelin ensimmäinen kohtauksen piirtämiseen haetaan tietokannasta seuraava rivi (sarakkeiden nimet korostettu).

scene_id

1

scene_name

scene01

scene_content

Pelin ensimmäisessä versiossa sessiomuuttuja ”game_state” asetettiin arvoon 1 *userpage.php*-tiedostossa, joten *sql*-lause löysi oikean rivin tietokannasta. Kuvan 22 koodi näyttää, kuinka hain tiedot PHP:lla.

```
<?php
$game_state = mysql_real_escape_string($_SESSION["game_state"]);
$sql = "SELECT * FROM scenes WHERE scene_id=
'".$game_state."';";
$vastaus = mysql_query($sql);
$rivi = mysql_fetch_array($vastaus, MYSQL_ASSOC);
?>
<div class="<?php echo $rivi["scene_name"]; ?>">
<?php echo $rivi["scene_content"] ?> </div>
```

KUVA 22. Pelialueen piirtämisen koodi

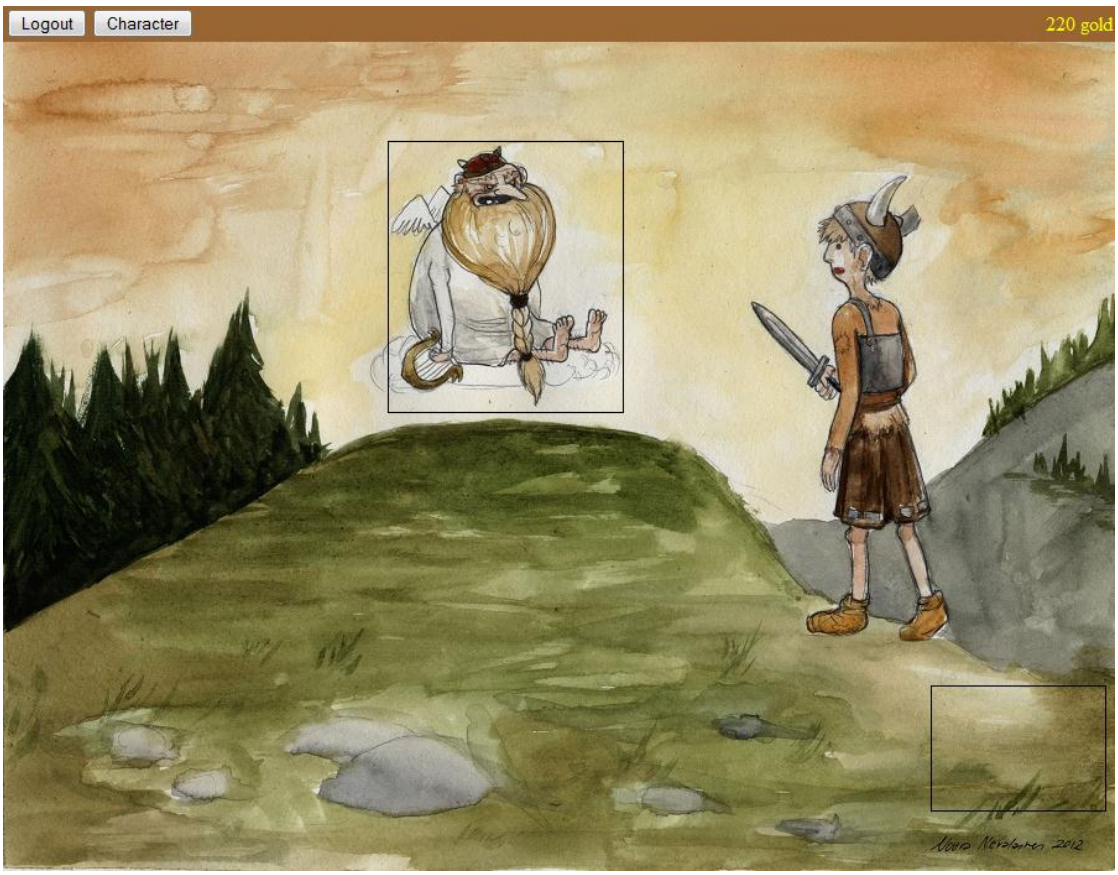
Tällöin selain saa tulkittavakseen seuraavan html-merkinnän:


```
</div>
```

CSS-tyylitiedosto sisältää seuraavat tiedot pelin aloitusruudun osalta:

```
.scene01
{
		background-image: url('../testikuvat/scene01.jpg');
		width: 900px;
		height: 670px;
		position:absolute;
}

.isan_haamu
{
		border:1px red solid;
		width:150px;
		height:120px;
		position:absolute;
		top:250px;
		left:380px;
}

.polku_01
{
		border:1px black solid;
		width:140px;
		height:100px;
		position:absolute;
		top:520px;
		left:750px;
}
```

Näin menettelemällä sain kohtauksiin omat taustansa sekä jokaiseen kohtaukseen liittyvät elementit näkyviin. Selaimessa kokonaisuus näyttää kuvan 23 mukaiselta.



KUVA 23. Pelin ensimmäinen kohtaus

Halusin aluksi, että pelissä navigointi tapahtuisi ”korostettuja” elementtejä klikkaamalla: käyttäjälle tulisi olla itsestään selvää, mitä kohteita ympäristössä voi klikata ja miten navigoida. Päädyin kuitenkin käyttämään linkkejä, joille määritin *class*-attribuutin ja tyyleihin näkyvät reunat. Tämä säästi aikaa muiden teknisten ongelmien ratkaisuun.

Kohtauksesta toiseen siirtymisen ratkaisin hyväksikäyttäen JavaScriptin jQuery-kirjaston $.post –funktiota. Kun aloitusruudussa klikataan oikealle alas vievää polkua, tietokannasta haettu *html*-merkintä aktivoi *change_scene* –funktion, jolle välitetään parametrina kyseisen kohtauksen id. Alla *change_scene* –funktio kokonaisuudessaan.

```
function change_scene(id)
{
        var url = "http://localhost/vikingtale-v02/game.php";
        id++;
        var tiedot_json = {"state":id};
        $.post(url, tiedot_json, function(data){});
```

}

Koodissa *id*-arvoa kasvatetaan yhdellä ja se välitetään *game.php*-tiedostolle, jossa *$_POST[”state”]*-muuttujasta arvo välitetään *$_SESSION [”game_state”]*-muuttujaan. Vastaavasti takaisinpäin liikkuessa *change_scene_back*-funktio vähentää arvoa yhdellä. Nyt kuvassa 22 esitelty koodi osaa hakea uuden kohtauksen tiedot kannasta ja tulostaa sen selaimeen.

4.4 Dialogi

Halusin saada peliin toiminnallisuuden pelkälle keskustelulle ja tähän työkaluksi valitsin *XML*-merkinnän (*eXtensible Markup Language*). Sovelsin tätä koulussa opittua tekniikkaa saadakseni vaihtelua SQL-tekniikan rinnalle. Keskustelutiedoston *conversations.xml*:n rakenne on kuvassa 24.

```
<keskustelut>
    <keskustelu xml:id="d1">
        <npc>Repliikki</npc>
        <player>Vastaus</player>
        <npc>Repliikki2</npc>
        <player>Vastaus2</npc>
    </keskustelu>
</keskustelut>
```

KUVA 24. Conversations.xml-tiedoston rakenne

Linkkiä class-arvoltaan *isan_haamu* klikatessa aktivoituu *start_dialogue* –funktio, joka käynnistää dialogien näyttämiseen tarkoitetun popup-ikkunan. Koska dialogit ovat xml-muodossa, täytyi ne ladata PHP:n DOM-parseriin. Latauskoodi on esitetty kuvassa 25.

```
$url = "includes/conversations.xml";
$dom = new DOMDocument("1.0");
$dom->preserveWhiteSpace = true;
$dom->formatOutput = true;
$dom->load($url);
```

KUVA 25. DOM-parserin alustus

Keskustelut on yksilöity *xml:id*-attribuutilla, jossa kirjain d tulee sanasta *dialogue* ja sen yhteydessä oleva numero on kyseisen kohtauksen id. Käytin *xpath*-hakua oikean

keskustelun hakemiseen. Kuvassa 26 on esitetty tarvittavien hakulausekkeiden määrittelyt.

```
$id = "d".$_GET["game_state"];
$juuri = $dom->getElementById($id);
//käytetään xpathia kyselyihin
$xpath = new DomXpath($dom);
//Haetaan oikea keskustelu ID:llä
$lauseke1 = "/keskustelut/keskustelu[@xml:id = '".$id."']";
//Haetaan keskustelun viimeinen virke
$lauseke2 = "/keskustelut/keskustelu[@xml:id = '".$id."']/npc[last()]";
//Suoritetaan haut
$keskustelut = $xpath->query($lauseke1);
$viimeinen = $xpath->query($lauseke2);
```

KUVA 26. Xpath-hakulausekkeet

Ensin tein id-muuttujan yhdistämällä d-kirjaimen ja sen hetkisen kohtauksen numeron *game_state* –sessiomuuttujasta. Id:n avulla pystyin määrittämään kyseisen keskustelun sijainnin xml-tiedostossa *getElementById*-funktion avulla. Dialogien käsittelijän piti myös kertoa, milloin keskustelu loppuu, jolloin se osasi tulostaa näytölle oikeanlaisen *html-formin*. Xpathin query palauttaa taulukon, joten sen käsittelin *foreach*-silmukalla.

```
foreach($viimeinen as $vika_virke)
{
        $loppu = $vika_virke->textContent;
}
```

Tämän jälkeen asetin sessiomuuttujan *$_SESSION["i"]* arvoon 0, mikäli *Next*-nappia ei ole painettu.

```
if(!isset($_POST["next"]) $_SESSION["i"] = 0;
```

Käytin tätä sessiomuuttujaa hakemaan oikeat repliikit. Ensin tein käsittelijän tapahtumalle, jossa *Next*-nappia on painettu ja haettavan vuorosanan id on suurempi kuin 0. Kyseinen *if*-lause on kuvassa 27.

```
if(isset($_POST["next"]))
{
    $_SESSION["i"]++;
    foreach($keskustelut as $keskustelu)
    {
        $npc = $keskustelu->GetElementsByTagName("npc")->
        item($_SESSION["i"])->textContent;

        $player = $keskustelu->GetElementsByTagName("player")->
        item($_SESSION["i"])->textContent;
    }
}
```

KUVA 27. Repliikkien hakemisen ensimmäinen tarkistuslohko

Jos tämä ehtolause ei toteudu, se tarkoittaa pelaajan juuri avanneen keskusteluruudun ja tällöin ruudulle pitää tulostaa ensimmäinen repliikkipari, joiden *item*-indeksi on 0. Olisin voinut käyttää $_POST –muuttujan tilalla myös muuttujaa $_REQUEST, mutta jälkimmäinen on tietoturvan kannalta heikompi, sillä se ottaa vastaan sekä $_POST että $_GET (ja $_COOKIE) –muuttujille tarkoitettua dataa (Nadeau 2010). Haluan omissa projekteissani aina olla varma siitä, millaista dataa otetaan vastaan ja mistä se on peräisin.

Ensimmäisten repliikkien haku on kuvassa 28.

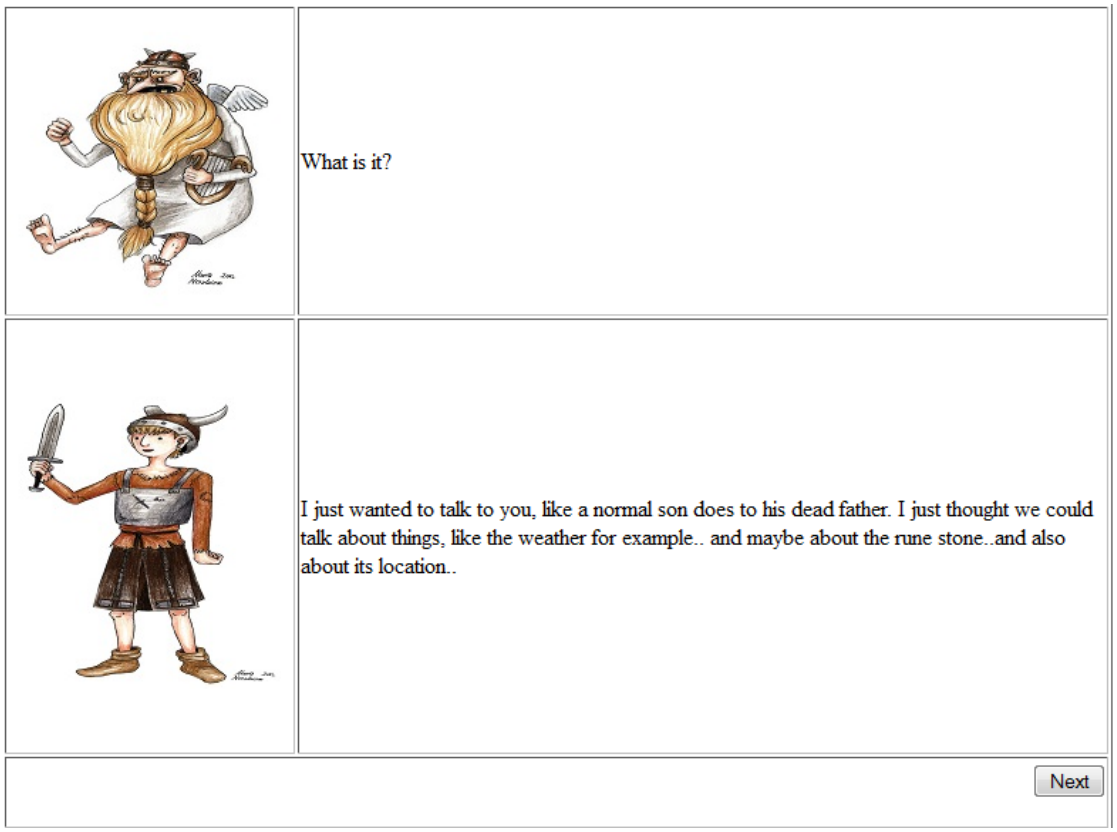
```
else
{
    foreach($keskustelut as $keskustelu)
    {
        $npc = $keskustelu->GetElementsByTagName("npc")->
        item(0)->textContent;

        $player = $keskustelu->GetElementsByTagName("player")->
        item(0)->textContent;
    }
}
```

KUVA 28. Ensimmäisen repliikkiparin hakeminen

Repliikkien näyttämisen muotoilin *HTML:n table*-elementillä. Sen sisään sijoitin hahmojen kuvat repliikkien viereen. Dialogin aloitusruudun näkee kuvasta 29.

Ruutujen ero tehtiin seuraavalla koodilla, joka tulostaa joko Next- tai Done-nappulan sen mukaan, ollaanko viimeisessä repliikkiparissa vai ei.

```
<tr height="50px"><td align="right" colspan="2"><form method="post"><input type="submit" name="next"
<?php if($loppu == $npc) echo "value='Done'"; else echo "value='Next'";?>
<?php if($loppu == $npc) echo "onclick='window.close()'";?>></form></td></tr>
```



KUVA 29. Dialogin ensimmäiset repliikit

4.5 Tehtävien suorittaminen

Halusin saada tähän peliin samankaltaisia tehtäviä kuin muissakin roolipeleissä. Päällimmäisenä mieleen tuli erinäisten esineiden kerääminen maastosta tai vihollisilta. Tehtävien jakamiseen piti suunnitella toiminnallisuus. Ensimmäisellä kerralla, kun ikkunan avaan, niin tehtävää tarjotaan. Toisella kerralla koodi tarkistaa aktiivisen tehtävän ja pelaajan inventaarion siihen liittyvien tavaroiden osalta. Kun tehtävä on suoritettu, tehtävän antaja kiittää pelaajaa avusta.

Tehtävien jakamista ja suorittamista varten suunnittelin kaksi muuttujaa *users*-tauluun: *active_quest_id* ja *quest_status.* Kun käyttäjätunnus luodaan, tietokantaan laitetaan pelaajalle *active_quest_id*:ksi arvo 1. Tämä on ensimmäisen tehtävän *id*, joka on tietokannassa *quest*-taulussa. Muuttuja *quest_status* määritetään arvoon 0 ja se kertoo, että tällä hetkellä pelaajalla ei ole mitään tehtävää aktiivisena.

Kun tehtävän antajan luo tullaan ensimmäistä kertaa ja hahmo saa tehtävän, se hyväksytään AcceptQuest-funktiolla, joka muuttaa pelaajan *quest_status*-muuttujan arvoon 1.

Koodissa tämä tarkoittaa sitä, että ensin piti tehdä tarkistus *quest_status* –muuttujan varalle ja tarkistaa, onko pelaaja tällä hetkellä tekemässä tehtävää vai ei. Ensimmäistä tehtävää hakiessa kuvan 30 koodi toteutuu, sillä *quest_status* on määritetty arvoon 0 aikaisemmin. *Active_quest_id*:llä on arvo 1, joten *sql*-lause hakee sitä vastaavan tehtävän *quest*-taulusta. Funktio palauttaa tässä tapauksessa tehtävän tiedot ja *html*-lomakkeen *submit*-nappulan, jolla pelaaja voi hyväksyä tehtävän.

```
if($_SESSION["quest_status"] == 0)
{
    $active_quest = mysql_real_escape_string($_SESSION["active_quest"]);
    $sql_lause = "SELECT * FROM quest WHERE quest_id = ".$active_quest.";";
    $tulos = mysql_query($sql_lause);
    $rivi = mysql_fetch_array($tulos, MYSQL_ASSOC);

    $return = "Quest: ".$rivi["quest_name"]."<br>";
    $return .= "Goal: ".$rivi["quest_content"];

    $return .= "<form method='post' action='quest.php'>";
    $return .= "<input type='submit' name='accept' value='Accept'/>";
    $return .= "</form>";

    return $return;
}
```

KUVA 30. Tehtävän aktiivisuuden tarkastaminen

Mikäli muuttujalla *quest_status* on arvo 1, pelaaja on jo käynyt hakemassa tehtävän itselleen. Ensin hain tehtävän tiedot kannasta ja otin talteen *quest_item_id*- ja *quest_item_count* -arvot. Sijoitin ne $this-> -alkuisiin lokaaleihin *private*-tyypin muuttujiin, jotka ovat käytössä vain tämän olion sisällä (PHP.net, 2012). Muuttuja *quest_item_id* yksilöi tarvittavan esineen ja muuttuja *quest_item_count*:n sisältämä

arvo kertoo, kuinka monta esinettä kyseessä olevaan tehtävään tarvitaan. Esittelen koodin kuvassa 31.

```
elseif($_SESSION["quest_status"] == 1)
{
  //Haetaan questin tiedot tarkistuksia varten
  $active_quest = $_SESSION["active_quest"]
  $sql_lause = "SELECT * FROM quest WHERE quest_id = ".$active_quest.";";
  $tulos = mysql_query($sql_lause);
  $rivi = mysql_fetch_array($tulos, MYSQL_ASSOC);

  $this->quest_item_id = $rivi["quest_item_id"];
  $this->quest_item_count = $rivi["quest_item_count"];

  // Tarkastetaan pelaajan inventory
  $user_id = mysql_real_escape_string($_SESSION["user_id"]);
  $sql_lause = "SELECT * FROM inventory_slot WHERE parent_id = ".$user_id." and item_id = ".$this->quest_item_id.";";
  $tulos = mysql_query($sql_lause);
  $rivi = mysql_fetch_array($tulos, MYSQL_ASSOC);

  // Jos tarvittavia itemeitä ei ole inventoryssä tai niitä on liian vähän
  if(mysql_num_rows($tulos) > 1 || $rivi["item_count"] < $this->quest_item_count)
  {
      return "You don't have enough items...";
  }
  // Jos tarvittavia itemeitä on tarpeeksi, niin kiitetään ja poistetaan kamat pelaajan inventorystä
  elseif($rivi["item_count"] >= $this->quest_item_count)
  {
      $return = "You seem to have these items I desire. Could you give them to me?<br/>";
      $return .= "<form method='post' action='quest.php'>";
      $return .= "<input type='submit' name='give' value='OK. I give them to you!'/>";
      $return .= "</form>";

      return $return;
  }
}
```

KUVA 31. Pelaajan inventaarion tarkistaminen tilanteessa, jossa tehtävä on aktiivisena

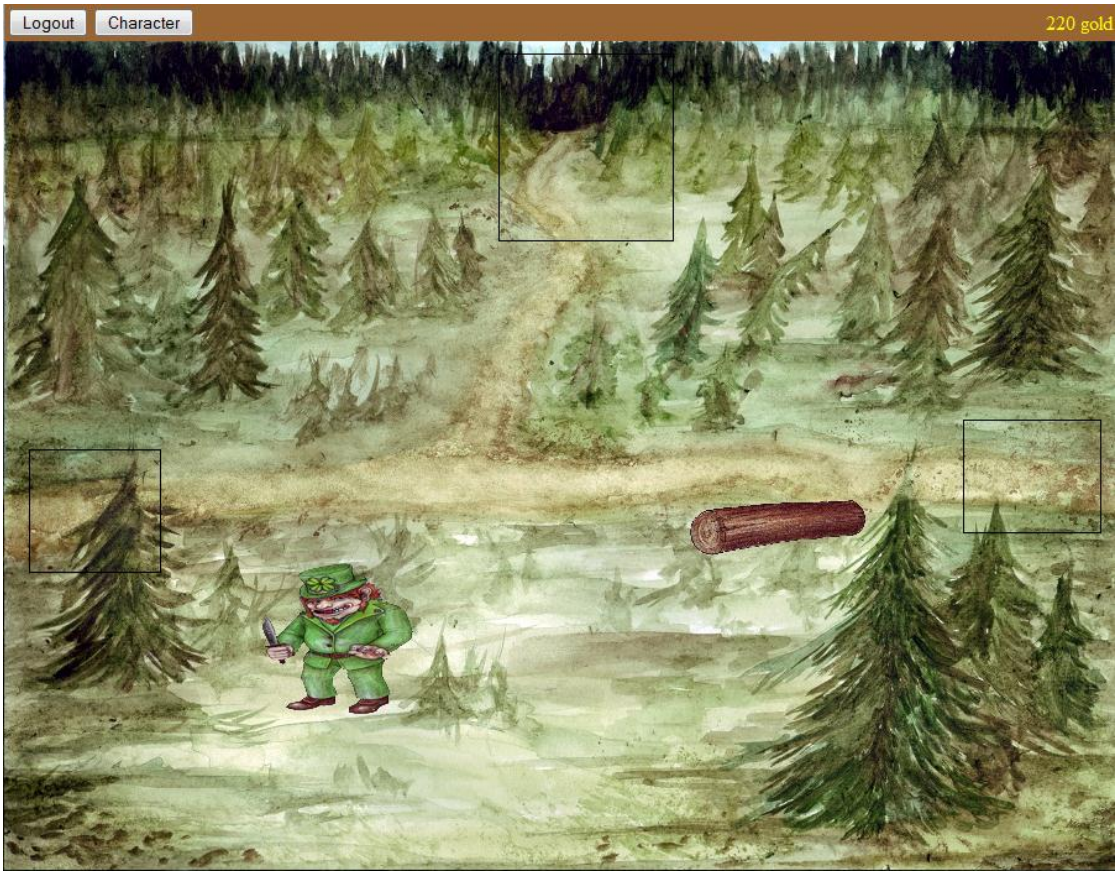
Tehtävän lopettamisen yhteyteen kirjoitin *FinishQuest*-funktion, joka tyhjentää pelaajan taskut tehtäviin tarvittavista tavaroista. Tässä funktiossa myös *active_quest_id*-muuttujan arvo kasvaa yhdellä, jonka perusteella hahmo saa seuraavan tehtävän, ja *quest_status*-muuttujan arvo asetetaan 0:ksi.

Pelin tarinassa ensimmäisen tehtävän antajan majatalo on kylmillään ja hän tarvitsee kipeästi lisää polttopuita. Ajattelin, että tämä on helppo toteuttaa teknisesti. Luon <a>-linkkielementin haluamaani kohtaukseen ja määrittelen sille tyylit tyylitiedostoon. Kun polttopuuta klikkaa, se siirtyisi hahmon inventaarioon.

Polttopuun *html*-merkinnän piirtää seuraava rivi:

<a href='' id='14' class='halko4' onclick="pickup_item(this.id)">

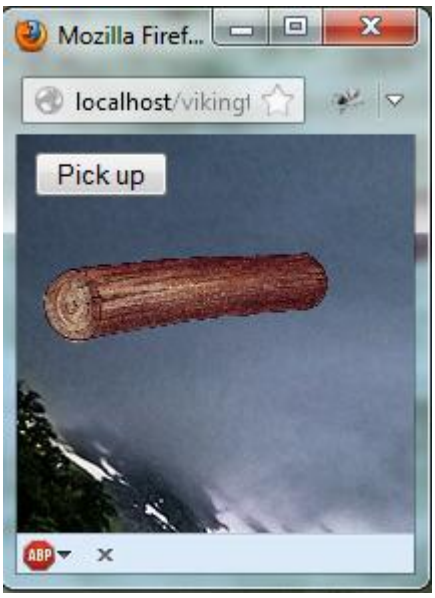
Halko esiintyy ensimmäisen kerran kohtauksessa neljä, joka on kuvassa 32.



KUVA 32. Polttopuu neljännessä kohtauksessa

Tietokannassa *items*-taulussa *Firewood*-esineen id on 14, joten se määritettiin myös tässä kohtaa kyseessä olevan linkin *id*:ksi. *Class*-attribuutissa oleva halko4-tunniste on yksilöllinen, koska sen sijainti ruudulla tulee tyylitiedostosta. Halkoja on siis useammassa eri kohtauksessa ja, jos olisin nimennyt ne kaikki samalla nimellä, niiden absoluuttinen sijainti olisi jokaisessa ruudussa sama. Halusin kuitenkin vaihtelua, joten se pakotti minut määrittämään jokaiselle halolle oman tyylinsä.

Halon klikkaaminen aktivoi *pickup_item* –funktion, jolle annoin parametrina halon id:n. JavaScript-funktio avaa popup-ikkunan (kuva 33).



KUVA 33. Pickup_item –funktion popup-ikkuna

Aluksi tein esineen inventaarioon sijoittamisen suoraan *pickup_item.php*-tiedostoon, jolle JavaScript-funktio välitti esineen *id*:n. Kirjoitin tarkistuslauseen odottamaan *url*-kentässä kulkevaa tietoa. Koodi on kuvassa 34.

```
//Haetaan itemin id:n perusteella nimi
//ja item_class_id mahdollista inventoryyn lisäämistä varten.)
$item_id = mysql_real_escape_string($_SESSION["item_id"]);
$sql_lause = "SELECT item_name, item_class_id FROM
items WHERE item_id = ".$item_id."";
$tulos = mysql_query($sql_lause);
$rivi = mysql_fetch_array($tulos, MYSQL_ASSOC);
$_SESSION["item_name"] = $rivi["item_name"];
$_SESSION["item_class_id"] = $rivi["item_class_id"];
```

KUVA 34. Esineen *id*:n vastaanottava koodi

Ensimmäinen tehtävä oli hakea tietokannasta esineen *id*:tä vastaavat tiedot: tässä tapauksessa esineen nimi ja *class*. Sijoitin kaikki tiedot omiin sessiomuuttujiinsa myöhempää käyttöä varten.

Seuraavaksi muodostin ehtolauseen odottamaan *Pick up* –nappulan painamista. Kun esinettä lisätään inventaarioon, mieleeni tuli kaksi erilaista tilannetta:

1. Onko kyseistä esinettä jo valmiiksi inventaariossa?
2. Onko inventaariossa tilaa uudelle esineelle?

Kohtaan yksi liittyi myös aluksi tekninen seikka, joka mahdollisti ensimmäisen tehtävän polttopuita kerättävän samaan inventaarion ”taskuun” monta kappaletta. Roolipelimaailmassa esineiden ”kasautumista” kutsutaan *stack*:iksi (Item Stacking, 2012). Suunnittelin peliin inventaarion, jossa on paikka kuudelle erinäiselle esineelle. Esineen luonteesta riippuu, onko se valittu *stack*-esineeksi vai ei. Jos tehtävään tarvitaan useampi samanlainen esine, on luonnollista, että ne menevät samaan pinoon. Kohdan yksi koodi on kuvassa 35.

```
//Jos pick up -nappia on painettu
if(isset($_POST["pickup"]))
{
    $user_id = mysql_real_escape_string($_SESSION["user_id"]);
    $item_id = mysql_real_escape_string($_SESSION["item_id"]);
    // Katsotaan, onko quest-itemeitä inventoryssä jo entuudestaan
    $sql_lause = "SELECT * FROM inventory_slot WHERE parent_id =
    ".$user_id." and item_id = ".$item_id.";";
    $tulos = mysql_query($sql_lause);
    if(mysql_num_rows($tulos) == 1)
    {
        // Päivitetään olemassa olevaa riviä
        $_SESSION["itemitaulukko"][] = $_SESSION["game_state"];
        $sql_lause = "UPDATE inventory_slot SET item_count =
        item_count + 1 WHERE parent_id = ".$user_id."
        and item_id = ".$item_id.";";
        $tulos = mysql_query($sql_lause);
```

KUVA 35. Inventaarion tarkastaminen, osa 1

Sessiomuuttuja ”itemitaulukko” oli aluksi yritys pitää kirjaa, mistä kohtauksista puun halko on jo poimittu talteen, jotta selain ei enää piirtäisi sitä uudestaan samaan kohtaan. Kirjoitin *game.php*-tiedostoon tekstijäsentäjän, joka kävi tietokannasta tulevan *html*-merkinnän läpi ja poisti sieltä kyseisen polttopuun. Tämä tapa kuitenkin osoittautui tökeröksi ratkaisuksi, sillä jokaiselle pelin esineelle (joka poimittuaan hävisi maasta) olisi pitänyt kirjoittaa samanlainen jäsentäjä yksilöivän *class*-attribuutin takia. Kohdan kaksi koodi esitellään kuvassa 36.

```
else
{
    // Jos itemeitä ei ole vielä ennen poimittu, niin katsotaan onko tyhjää tilaa inventoryssä
    $user_id = mysql_real_escape_string($_SESSION["user_id"]);
    $sql_lause = "SELECT * FROM inventory_slot WHERE parent_id = ".$user_id." and item_id = 0";
    $tulos = mysql_query($sql_lause);
    if($tulos)
    {
        // Jos tyhjää tilaa löytyy, niin laitetaan itemi inventoryyn
        // Katsotaan, montako tyhjää slottia on vapaana
        if(mysql_num_rows($tulos) > 1)
        {
            // Jos tyhjiä slotteja on enemmän kuin 1, niin pistetään uusi itemi pienimpään slot_id:hen.
            $sql_lause = "SELECT MIN(slot_id) as pienin_slot FROM inventory_slot WHERE
            parent_id = ".$user_id." and item_id = 0;";
            $tulos = mysql_query($sql_lause);
            $rivi = mysql_fetch_array($tulos);
            $_SESSION["itemitaulukko"][] = $_SESSION["game_state"];
            $item_id = mysql_real_escape_string($_SESSION["item_id"]);
            $item_class_id = mysql_real_escape_string($_SESSION["item_class_id"]);
            $sql_lause = "UPDATE inventory_slot SET item_id = ".$item_id.",
            item_count = 1, item_class_id = ".$item_class_id." WHERE parent_id = ".$user_id."
            and slot_id = ".$rivi["pienin_slot"].";";
            $tulos = mysql_query($sql_lause);
        }
        else
        {
            $_SESSION["warning_desc"] = "You don't have room for new items.";
        }
```

KUVA 36. Inventaarion tarkastaminen, osa 2

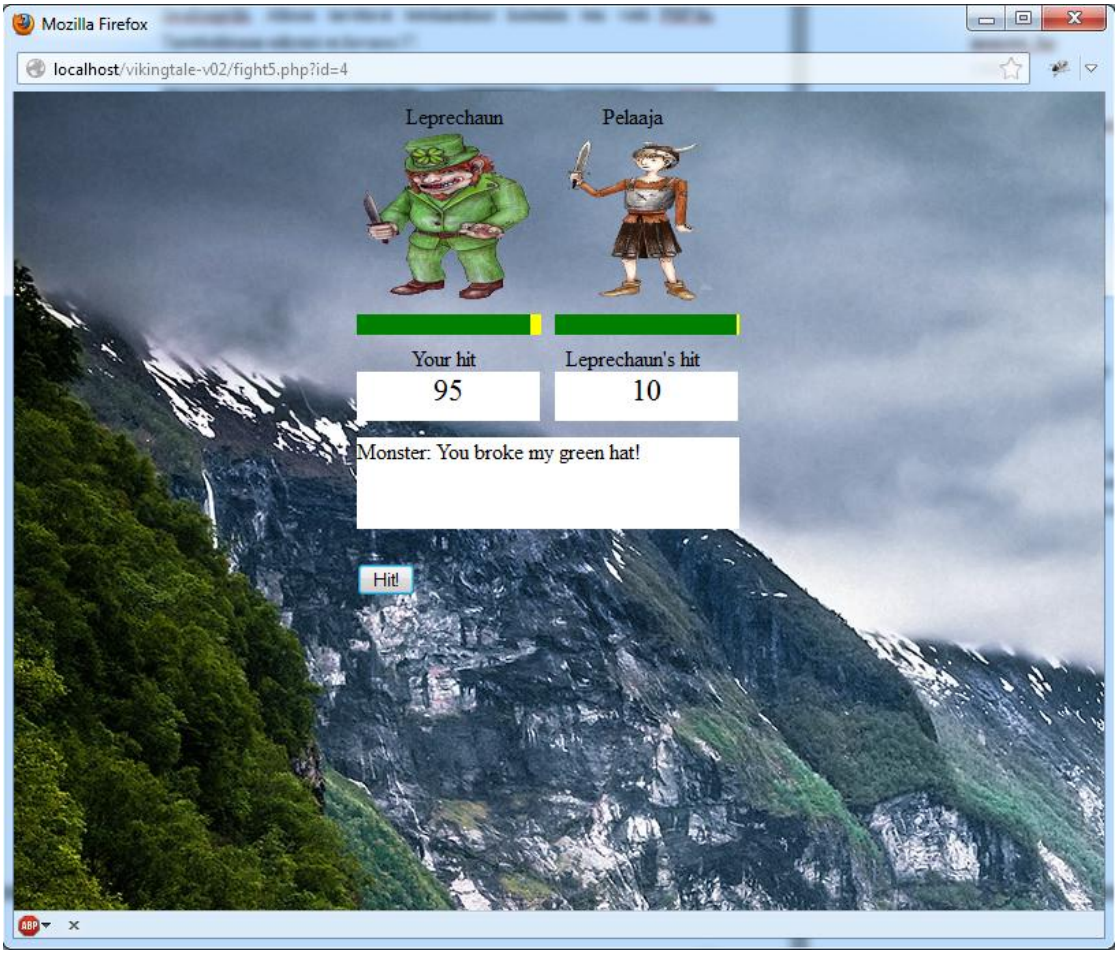
Koodi tarkastaa ensin inventaarion ja, jos tilaa löytyy, laitetaan esine riville, jossa on pienin *slot_id*. Muussa tapauksessa ikkunaan tulostetaan teksti, ettei pelaajan laukkuun mahdu enempää tavaraa.

4.6 Taistelu

Roolipeleissä kuuluu olla taisteluita. Joskus tapellaan lohikäärmeitä vastaan, joskus pahat peikot ja menninkäiset kokevat julman kohtalon sankarin miekasta. Tähän peliin halusin metsään menninkäisiä, joita klikatessa avautuisi taisteluikkuna. Samalla suunnittelin myös toiminnon vihollisen tiputtamalle aarteelle hänen kuollessaan; pitäähän voitetusta tappelusta vähän taikajuomaa tai kultaa saada.

Suunnittelin taistelun prototyypin jo loppusyksystä vuonna 2011, kun idea pelin tekemiseen tuli mieleeni. Kirjoitin sen PHP:lla, koska ajattelin sen tulevan sillä kielellä myös itse peliin. Kuvassa 31 esiintyvä *Leprechaun* on sekin klikattava kohde, josta avautuu taisteluikkuna *JavaScript*-funktion avulla. Ajattelin tekeväni taistelun yksinkertaisesti niin, että kirjoittaisin HTML-merkinnällä *submit*-nappulan, jonka painallus edustaisi yhtä miekan lyöntiä. Halusin taistelusta vuoropohjaisen, eli ensin löisi sankari ja sen jälkeen vihollinen vastaisi iskuun. Koska PHP on serveripuolen

scriptikieli, muutosten tapahtuminen tarvitsee aina sivun uudestaan lataamisen (Bradley 2012). Jos jokaisen lyönnin jälkeen sivu latautuisi uudestaan, pelattavuus kärsisi. Tähän olisi voinut käyttää ehkä myös asynkronista *ajax*-tekniikkaa, jota käytin jo aikaisemmin kohtauksen vaihtamiseen. En kuitenkaan saanut toimivaa rakennetta mieleeni tätä tekniikkaa hyödyntäen. Päädyin ohittamaan PHP-pelimoottorini ja kirjoittamaan taisteluikkunan toiminnallisuuden kokonaan selaimessa suoritettavalla JavaScript:llä. Alkuun tarvittavat tietokantahaut kuitenkin tein vielä PHP:lla. Taisteluikkunan näkymä on kuvassa 37.



KUVA 37. Taisteluikkuna

Kun taisteluikkuna avataan, PHP-koodi suorittaa tietokantahaun ja hakee vihollisen tiedot url:ssa kulkevan *id*:n perusteella. Jouduin vielä tässä käyttämään PHP:ta, koska JavaScript-kielessä ei ole MySQL-rajapintaa. Tietokannassa *monsters*-taulussa on NPC:n (Non-Player Character) nimi, hp:n (*Health Points*) määrä, vihollisen tiputtaman kullan määrä ja kolmen eri esineen *id*:n, joita vihollisen on mahdollisuus tiputtaa kuollessaan. *Leprechaun*-vihollisen tietue tietokannassa näyttää seuraavalta:

monster_id

4

monster_name

Leprechaun

monster_hp

1500

monster_money_loot

5

monster_item_loot1

6

monster_item_loot2

7

monster_item_loot3

8

Vihollisen tietojen hakemisen jälkeen hain koodilla *strength*-luvun, jonka sain summaamalla jokaisen hahmon päällä olevien varusteiden *strength*-arvon. Tämän tekevä funktio on kuvassa 38.

```
function GetGearStrength()
{
    $strength = null;
    $user_id = mysql_real_escape_string($_SESSION["user_id"]);
    $sql = "SELECT strength_value FROM items JOIN gear ON
    items.item_id=gear.item_id WHERE gear.parent_id = ".$user_id.";";
    $tulos = mysql_query($sql);
    while($rivi=mysql_fetch_array($tulos, MYSQL_ASSOC))
    {
        $strength += $rivi["strength_value"];
    }

    return $strength;
}
```

KUVA 38. Strength-arvon summaaminen pelaajan varusteista

Seuraavaksi arvoin vielä etukäteen pelaajan saaman palkinnon, mikäli hän voittaa taistelun. Koska vihollisella on kolme eri esinettä taskussaan ja päätin, että pelaaja saa vain yhden niistä, kirjoitin koodin arpomaan luvun väliltä 1-3. Luvun yhdistin merkkijonon ”monster_item_loot” perään, jolloin sain aikaiseksi viittauksen johonkin taulun kolmesta *monster_item_loot*-sarakkeesta.

Koska palkinto ilmestyy ruudulle vasta pelaajan voitettua taistelun, kirjoitin seuraavan koodin piilottamaan palkinnon sisältävän *div*-elementin.

```
$(document).ready(function() {
// piilotetaan lootit heti aluksi
$("#lootti_ikkuna").hide();
});
```

Yleensä roolipeleissä pelaajan HP-arvo tulee kokemustason perusteella ja/tai *stamina*-arvosta. Tässä pelissä ei ole mukana kumpaakaan, joten päätin sitoa pelaajan HP-arvon ainoaan käytettävissä olevaan muuttujaan: *strength* – eli voimakkuus-arvoon. HP:sta tuli *strength*-arvo kerrottuna kymmenellä. Myöskään vihollisen iskun voimakkuudelle en ollut miettinyt mitään perusteltua arvoa, joten laskin sen kaavalla: pelaajan_hp / 5.

Toteutin vahingon määrän laskevaan funktioon todennäköisyydet ohilyönnille ja erikoistapaukselle, jossa lyönnin voimakkuus 2-4-kertaistuu. Tällaista tapahtumaa kutsutaan *critical hit*:ksi – eli kriittiseksi iskuksi. Kerroin on itseni määrittelemä. Kuvassa 39 on *damage*-funktion alku, jossa on toiminto ohilyönnille ja sen arvon palauttamiselle.

```
function damage(vihun_hp, strength, x)
{
    //hitti vai huti
    var hudin_tn = Math.floor((Math.random()*100)+1);

    if(hudin_tn >= 95)
    {
        var hp_ja_hitti = [vihun_hp, 0];
        return hp_ja_hitti;
    }
```

KUVA 39. Damage-funktion alku

Muuttuja x sisältää arvon *monster* tai *player* riippuen siitä, kumman vuoro on lyödä. Käytin tätä tunnistetta kriittisen iskun vihreäksi värittämisen vuoksi (kuva 40). Kaksi muuta funktiolle välitettävää parametria ovat *vihun_hp* ja *strength.*

```
else
{
    //arvotaan tähän lyönnin voimakkuus strenan ja strena*2 väliltä,
    //tulee vähän hajontaa eikä koko ajan samaa numeroa

    var hitti = Math.floor(((strength*2)-(strength-1))*Math.random()) + strength;

    // critti
    var todnak_critille = Math.floor((Math.random()*100)+1);

    if(todnak_critille >= 95)
    {
        if(x == "monster") { $("#monster_damage").css("color","green"); }
        else if(x == "player") { $("#player_damage").css("color","green"); }

        // uuden hitin kerroin
        hitti = hitti * Math.floor((4-1)*Math.random()) + 2;
        // alle satasen crittejä ei anneta
        if(hitti < 100) { hitti += 100; }
    }
```

KUVA 40. Damage-funktion toinen osa

Kuten ohilyönnin, myös kriittisen iskun todennäköisyys on 5 %. JavaScript:n *jQuery*-kirjaston avulla väritin sen osapuolen iskun vihreällä, kumpi tämän suuren iskun löi. Tämän jälkeen määritin uuden hitin ja varmistin, ettei se ole suuruudeltaan alle 100.

```
        var vihun_hp;
        vihun_hp -= hitti;

        var hp_ja_hitti = [vihun_hp, hitti];

        return hp_ja_hitti;
    }
}
```

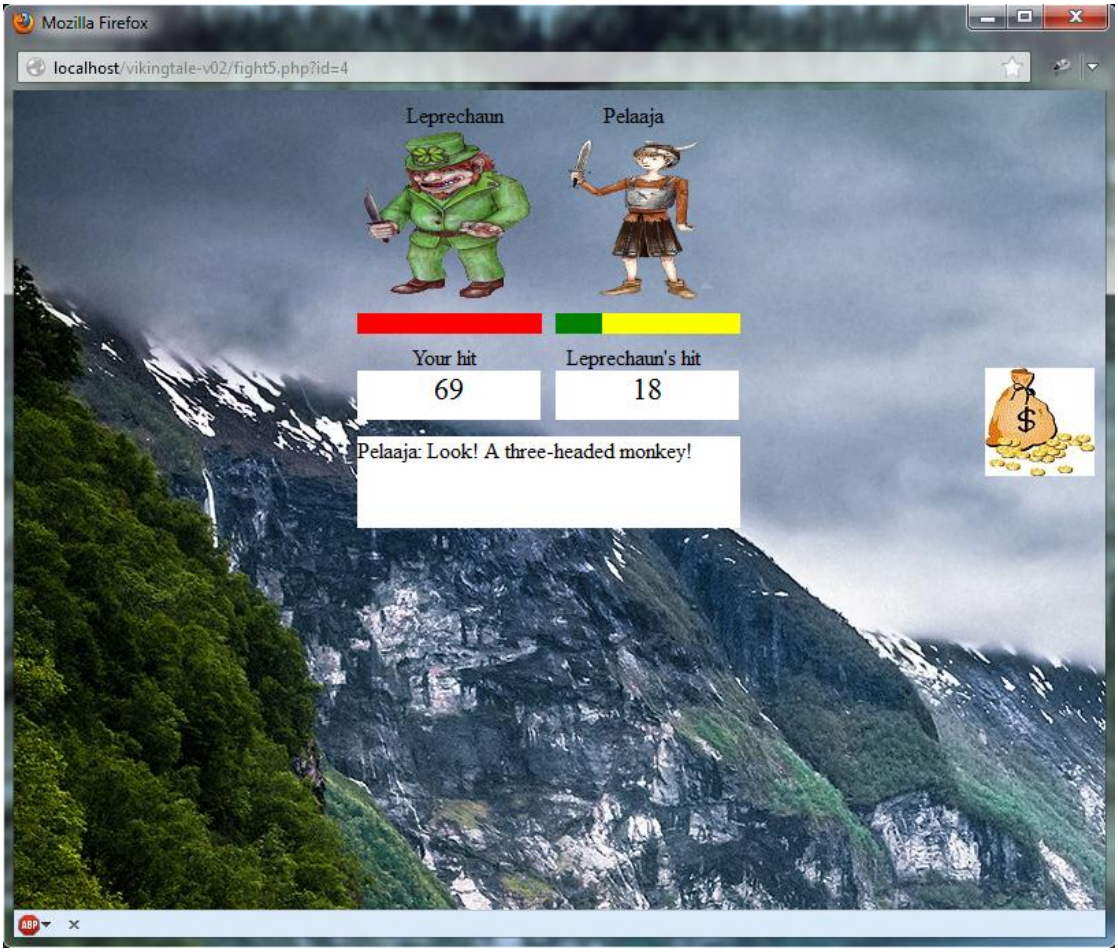
KUVA 41. Damage-funktion lopetus

Kun tarvittavat laskutoimitukset on lyönnin suhteen tehty, funktio palauttaa arvot kaksialkioisena taulukkona (kuva 41). Näiden arvojen perusteella *healthbar*:ia pystytään liikuttamaan.

Kun taistelu alkaa, määritin *fight_over*-muuttujan arvoon *false*, sillä tämän muuttujan tilaa tarkastelemalla voin päätellä, koska taistelu on ohi ja palkinto-elementti näytetään ruudulla. Toteutin pienen taistelulogin, jonne tulostin jokaisen iskun yhteydessä jonkin huudon tai solvauksen. Solvaukset tallensin taulukkoon, josta arvoin jokaisella kierroksella yhden.

Vuorovaikutteisuuden illuusion luomiseksi käytin *setTimetout*-funktiota aiheuttamaan viivettä pelaajan lyönnin ja vihollisen vastaiskun välillä. Määritin viiveeksi kaksi sekuntia. Kun pelaaja on painanut *Hit*-nappia, se katoaa näkyvistä, ettei sitä voi klikata monta kertaa peräkkäin ja näin taistelu pysyy mielekkäänä. Pelaajan ja vihollisen kuvien alle tulostin kulloisenkin lyönnin voimakkuuden. Sain aikaiseksi myös *healthbar*-animaation *jQuery*:n *animate*-toiminnolla. Tämän tuloksena *healthbar* liikahtaa menetetyn *HP*:n verran 400 ms:n aikana.

Kun taistelu on loppunut ja pelaaja on selviytynyt voittajaksi, ruutu on kuvan 42 mukainen. Oikealle on ilmestynyt rahasäkki. Palkinnon sijoittamista inventaarioon en kuitenkaan ehtinyt toteuttaa.



KUVA 42. Taisteluruutu, kun pelaaja on voittanut

5 PÄÄTÄNTÖ

Ensiksi tahdon sanoa, että työskentely näin isossa projektissa parin kanssa on opettavaista. Se kannustaa pitämään aikatauluista kiinni, kun sovitaan tekevämme tiettyinä päivinä asioita. Vaikka itse olen sellainen, että saan enemmän aikaan yksin rauhassa asioita miettimällä, on kaverin tuki ja turva välillä korvaamaton ongelmia ratkoessa. Meille muistaakseni joskus koulussa sanottiin, että tällä alalla opinnäytetyö kannattaa tehdä yksin. Nyt kuitenkin luotin pariini niin paljon, että uskalsin jakaa kokemuksen hänen kanssaan. Lisäksi projektia siivitti molempien yhteinen innostus aiheeseen; oman pelin tekeminen ja juonen keksiminen.

Työn tekoa helpotti paljon se, että minulla oli jo valmiiksi visio mielessäni lopullisesta tuotteesta, vaikka sitä emme näillä välineillä saaneetkaan aikaiseksi. Haasteellinen tehtävä oli siirtää sama visio ystävälleni ja työparilleni Harri Lappalaiselle. Joskus

yksinkertaiseltakin tuntuvia asioita on hankala selittää haluamallaan tavalla. PHP/MySQL-tekniikka oli tuttu aikaisemmista ohjelmointitehtävistä, joten halusin yrittää soveltaa niitä toisenlaiseen tarkoitukseen. Pelin idean konsepti on siirrettävissä PC:n kautta mobiililaitteisiin, sillä ohjausmekanismi on yksinkertaista ”osoita ja klikkaa”-naksuttelua. En ole vastaavaan roolipeliin vielä törmännyt, joten haluan saada tällaisen vielä joskus toteutettua mallikkaasti. Tämä oli hyvä omien siipien kokeilu pelisuunnittelussa, koska minun ei tarvinnut opetella uutta syntaksia ohjelmoidessa ja käytetty tekniikka on ajankohtainen tietojenkäsittelyn koulutusohjelmassa. Peliä tehdessä sai soveltaa monia eri kursseilla opittuja tekniikoita kokonaisuuden luomiseksi. Lappalainen työskenteli projektin ajan pääsääntöisesti selainpään ominaisuuksien, kuten kauppias-ikkunan parissa, joten jätin sen osuuden omasta raportistani pois.

Parannusideoita tuli loppua kohti paljon, sillä asioita voi tehdä monella eri tavalla väärin. Koodi meni projektin laajetessa erittäin sekavaksi ja työhön palaaminen muutaman viikon huilaustauon jälkeen ei tuntunut aina houkuttelevalta. PHP-pelimoottorin siirtäminen omaksi luokakseen helpotti koodin järjestelyä hieman, mutta silti motivaatio oli kateissa useampaan otteeseen. Tietokantaan tallennuksia tuli tehtyä melko mielivaltaisesti, enkä ollut enää loppuvaiheessa selvillä, milloin jokin sessiomuuttuja oli jo määritetty. Tämän vuoksi määrityksiä saattoi tulla useampia päällekkäin. Mietin jälkeenpäin, että ehkä koko tietokanta olisi pitänyt ladata sessioon heti kirjautuessa ja tarvittavat tallennukset tehdä uloskirjautumisen yhteydessä. Näin ollen, suuret tietomäärät olisivat siirtyneet PHP:n ja tietokannan välillä kahdessa erässä.

Questauksen – eli tehtävien tekemisen rakenne ei ole mielestäni hyvä, sillä se mahdollistaa vain tietyn tyyppisten tehtävien luomisen peliin. Tämäkin tuli mieleeni vasta, kun olin funktiot jo kirjoittanut. Mielenkiinnon ylläpitämiseksi tällaisessa pelissä pitäisi olla vaihtelevia tehtäviä ja niitä pitäisi olla motivaatiota tehdä, kuten olen luvun 2.1 pelisuunnittelua käsittelevässä osiossa maininnut; pelaaja oppii ja saa hyvänolontunteen onnistuttuaan tehtävässä. Popup-ikkunoiden käyttäminenkään ei toimi, koska ikkunaa päivittämällä saa aloitettua koko taistelun alusta. Joissain kohti pelaaja näkee osoiteriviltä, millaista tietoa milläkin hetkellä kuljetetaan. Se tarjoaa mahdollisuuden pelin manipulointiin. Halkoa en saanut sulavasti pois näkyvistä sen poimimisen jälkeen, vaan sen katoaminen vaati *jQuery*:n *hide*-funktion kanssa aina

sivun päivittämisen. Lisäksi vajaa vuosi tällaisen pelin tekemiseen on erittäin lyhyt aika, eikä jätä mahdollisuutta taisteluiden tasapainon hiomiseen. Taisteluiden pitäisi olla kuitenkin haastavia ja samalla mielenkiintoisia. Aluksi mielessä pyörineet taikajuomat ja niiden käyttö taisteluiden aikana karisivat nopeasti pois ajan puutteen vuoksi. Päädyin toteuttamaan rautalankamallin toiminnoista, jotka mielestäni olivat olennaisimpia.

Esitin johdannossa, että opinnäytetyöni tarkoituksena on ottaa selvää, kuinka hyvin PHP/MySQL-tekniikalla pystyy toteuttamaan selainpelin. Tässä tapauksessa se on teknisesti mahdollista, mutta pelattavuus on huono. Jouduin turvautumaan ajax:iin ja JavaScript:iin joissain vaiheissa, joten yksinomaan serveripuolen ohjelmointikielellä tätä ei kannata tehdä.

Seuraavaksi siirryn tutkimaan luultavasti Multimediaohjelmointi-kurssilla tutuksi tullutta *XNA*-ympäristöä. Ehkä se tarjoaisi paremmat työkalut tämän pelin kehittämiseen. Tästä projektista jäi käteen se, ettei tällaista peliä kannata tehdä serveripuolen scriptauskielellä, koska se on osittain teknisesti mahdotonta ja vaikuttaa pelattavuuteen negatiivisesti. Projekti jatkuu edelleen ja onneksi olen nyt jo paljon viisaampi.

LÄHTEET

A Brief History of Mobile Games. 2009.Pocketgamer.biz. WWW-dokumentti. http://www.pocketgamer.biz/r/PG.Biz/A+Brief+History+of+Mobile+Games/feature.asp?c=10723 Päivitetty 1.2.2009 Luettu 25.9.2012

Advanced Password Validation Using Regular Expressions. 2006. Eukhost.com. WWW-dokumentti. http://www.eukhost.com/forums/f18/PHP-advanced-password-validation-using-regular-expressions-229/ Päivitetty 2.10.2006. Luettu 5.10.2012.

A History of Video Game Consoles. 2012. Time.com. WWW-dokumentti. http://www.time.com/time/interactive/0,31813,2029221,00.html Ei päivitystietoja. Luettu 24.9.2012.

Bates, Bob 2004. Game Design, Second Edition. Boston, MA, USA: Course Technology. Päivitetty 10/2004. Luettu 21.9.2012.

Bergeron, Bryan 2006. Developing Serious Games. Boston, MA, USA: Course Technology. Päivitetty 01/2006. Luettu 22.9.2012.

Bradley, Angela. PHP is a Server Side Language. WWW-dokumentti. http://PHP.about.com/od/PHPbasics/a/PHP-Is-A-Server-Side-Language.htm Ei päivitystietoja. Luettu 13.10.2012.

Browser Games: Basics and History. 2012. Ctrustnetwork.com. WWW-dokumentti. http://www.ctrustnetwork.com/a36/gaming/browser-games/articles/growser-games-history.html Päivitetty 27.2.2012 Luettu 24.9.2012

Building Simulation Games. 2012. Compsimgames.about.com. WWW-dokumentti. http://compsimgames.about.com/od/citybuildingsims/City_Building_Simulation_Games.htm Ei päivitystietoja. Luettu 21.9.2012

Choosing The Video Game Platform Best For You. About.com. WWW-dokumentti. http://compactiongames.about.com/cs/beforeyoubuy/a/vgamePlatforms.htm Ei päivitystietoja. Luettu 25.9.2012.

Console game. Englanninkielinen Wikipedia. WWW-dokumentti. http://en.wikipedia.org/wiki/Console_game Päivitetty 10.9.2012 Luettu 22.9.2012

Garmon, Jay 2005. Geek Trivia: First shots fired. TechRepublic. Englanninkielinen WWW-dokumentti. http://www.techrepublic.com/article/geek-trivia-first-shots-fired/5710539 Päivitetty 24.5.2005. Luettu 10.9.2012.

Giantbomb.com. 2012. WWW-dokumentti. http://www.giantbomb.com/pc/60-94/ Ei päivitystietoja. Luettu 22.9.2012

Hallford, Neal LaMothe, Andre Hallford, Jana 2002. Swords and Circuitry: A Designer's Guide to Computer Role-Playing Games. Boston, MA, Usa: Course Technology.
Päivitetty 07/2002. Luettu 13.9.2012

History of Mobile Gaming. 2011. Phonearena.com. WWW-dokumentti. http://www.phonearena.com/news/History-of-mobile-gaming_id17949 Päivitetty 7.2.2011 Luettu 25.9.2012

History of Web Browser Games. 2009. Ezinearticles.com. WWW-dokumentti. http://ezinearticles.com/?History-of-Web-Browser-Games&id=2670093 Päivitetty 27.7.2009 Luettu 24.9.2012

Item Stacking. 2012. Wiki.Guildwars.com. WWW-dokumentti. http://wiki.guildwars.com/wiki/Item_stacking
Päivitetty 26.8.2012. Luettu 12.10.2012.

Johdatus PHP-kieleen. PHP:n syntaksi. http://users.jyu.fi/~kolli/ITK215_05/PHP/?sivu=syntaksi
Ei päivitystietoja. Luettu 5.10.2012.

Kramer, Wolfgang 2000. What is a Game? The Games Journal. Englanninkielinen WWW-dokumentti. http://www.thegamesjournal.com/articles/WhatIsaGame.shtml Päivitetty 12/2000. Luettu 24.8.2012.

Lecky-Thompson, Guy W 2007. Video Game Design Revealed. Boston, MA, USA: Course Technology. Päivitetty 12/2007 Luettu 18.9.2012

Mobile Game Development. 2011. Evontech.com. WWW-dokumentti. http://www.evontech.com/login/topic/17000.html Päivitetty 21.7.2011 Luettu 27.9.2012.

mysql_fetch_array. PHP.net. 2012. WWW-dokumentti. http://PHP.net/manual/en/function.mysql-fetch-array.PHP Päivitetty 5.10.2012. Luettu 6.10.2012.

Nadeau, Christopher. 2010. Devlog.info. Why PHP Request Array Is Dangerous. WWW-dokumentti. http://devlog.info/2010/02/04/why-PHP-request-array-is-dangerous/ Päivitetty 2.4.2010. Luettu 12.10.2012.

Pardew, Les Pugh; Scott Nunamaker, Eric 2004. Game Design for Teens. Boston, MA, USA: Course Technology. Päivitetty 9/2004. Luettu 10.9.2012.

PHP.net. 2012. WWW-dokumentti. http://PHP.net/manual/en/function.mysql-real-escape-string.PHP Päivitetty 5.10.2012. Luettu 5.10.2012.

PHP.net. 2012. WWW-dokumentti. http://PHP.net/manual/en/language.oop5.visibility.PHP Päivitetty 12.10.2012. Luettu 13.10.2012.

Rabin, Steve 2009. Introduction to Game Development (2nd Edition). Herndon, VA, USA: Course Technology. Päivitetty 6/2009. Luettu 4.9.2012

Scattergood, Marc 2005. Beginning Game Level Design Development. Boston, MA, USA: Course Technology. Päivitetty 2/2005. Luettu 13.9.2012

SEGA Corporation. 2006. Referenceforbusiness.com. WWW-dokumentti.
http://www.referenceforbusiness.com/history/Ro-Sh/SEGA-Corporation.html
Päivitetty 5/2006. Luettu 24.9.2012.

Scwab, Brian 2004. AI Game Engine Programming, Hingham, MA, USA: Charles River Media. Päivitetty 7/2004. Luettu 10.9.2012.

session_start. PHP.net. 2012. WWW-dokumentti.
http://www.PHP.net/manual/en/function.session-start.PHP
Päivitetty 5.10.2012. Luettu 5.10.2012.

SQL AUTO INCREMENT FIELD. 2012. W3Schools.com. WWW-dokumentti.
http://www.w3schools.com/sql/sql_autoincrement.asp
Ei päivitystietoja. Luettu 5.10.2012

Timeline of Computer History 2012. Computerhistory.org. WWW-dokumentti
http://www.computerhistory.org/timeline/?year=1971 Ei päivitystietoja. Luettu 22.9.2012

Ukko, Reko 2011. Hapen Pelitalo. Pelisuunnitteluseminaari. Helsinki. Youtube-video. Päivitetty 2.11.2011. Katsottu 13.9.2012.

Video game genres. 2012. Englanninkielinen Wikipedia. WWW-dokumentti.
http://en.wikipedia.org/wiki/Video_game_genres
Päivitetty 16.9.2012. Luettu 18.9.2012

Ward, Jeff. 2008. What is a Game Engine? WWW-dokumentti.
http://www.gamecareerguide.com/features/529/what_is_a_game_.PHP
Päivitetty 29.4.2008. Luettu 13.10.2012.

Wolf, Mark J.P 2008. The video game explosion: A history from Pong to Playstation and beyond. Westport, Connecticut: Greenwood Press. Päivitetty 2008. Luettu 10.9.2012.

What is a Browser-Based Game? 2012. Wisegeek.com. WWW-dokumentti. http://www.wisegeek.com/what-is-a-browser-based-game.htm Ei päivitystietoja. Luettu 27.9.2012

Ward, Jeff. What is a Game Engine? WWW-dokumentti. http://www.gamecareerguide.com/features/529/what_is_a_game_.PHP Päivitetty 29.4.2008. Luettu 13.10.2012.

KUVA 1. Donkey Kong. MarioHistory. Nesretrco.com. http://www.nesretro.com/mariohistory/images/DonkeyKongJR.jpg

KUVA 2. Super Mario Bros. Nintendookies.files.wordpress.com 2010. http://nintendookie.files.wordpress.com/2010/06/super-mario-bros.jpg

KUVA 3. PONG. Animatedpng.net 2012. http://animatedpng.net/wp-content/uploads/2012/02/Pong-game.jpg

KUVA 4. Street Fighter 2: The Word Warrior. Hark.com 2012. http://cdn2.hark.com/images/000/033/813/33813/original.0

KUVA 5. Asteroids. Images4.wikia.nocookie.net. http://images4.wikia.nocookie.net/__cb20081002093461/uncyclopedia/images/e/e6/Asteroids_game_screen.jpg

KUVA 6. Maze War. Old-computers.com. http://www.old-computers.com/museum/software/mazewar_ss2.jpg

KUVA 7. Wolfenstein 3D. Fiz-x.com 2012. http://fiz-x.com/wp-content/uploads/2012/05/Wolfenstein3d-4.png

KUVA 8. Nethack. Games.asteriski.fi. http://games.asteriski.fi/nethack_iso.jpg

KUVA 9. King’s Quest. Bp.blogspot.com.
http://1.bp.blogspot.com/_dJTtf9n17W0/S82Ck1Gmh9I/AAAAAAAAB9I/sd2bL0D4wCs/s1600/kingsQuest.jpg

KUVA 10. Gran Turismo. GBp.blogspot.com.
http://3.bp.blogspot.com/-OaCt78aTBKM/T9iP0Mm7Q_I/AAAAAAAADkY/o2dMcZYNf6Q/s1600/wpid-granturismo21.jpg

KUVA 11. Steel Talons. Arcade-history.com
http://www.arcade-history.com/images/game/2630_1.png

KUVA 12.SimCity 2000. Myabandonware.com.
http://www.myabandonware.com/media/captures/S/simcity-2000/simcity-2000_1

.