

Avoimet ohjelmistot pelikehityksessä

Janne Laukkarinen

Helsinki 9.9.2008

Avoim ohjelmistokehitys seminaari

HELSINGIN YLIOPISTO

Tietojenkäsittelytieteen laitos

Tiedekunta/Osasto — Fakultet/Sektion — Faculty		Laitos — Institution — Department	
Matemaattis-luonnontieteellinen		Tietojenkäsittelytieteen laitos	
Tekijä — Författare — Author			
Janne Laukkarinen			
Työn nimi — Arbetets titel — Title			
Avoimet ohjelmistot pelikehityksessä			
Oppiaine — Läroämne — Subject			
Tietojenkäsittelytiede			
Työn laji — Arbetets art — Level		Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Seminaari		9.9.2008	12 sivua
Tiivistelmä — Referat — Abstract			
<p>Pelikehitys avoimen ohjelmistokehityksen menetelmin ei ole mitenkään uusi asia. Vaikka Linux ei olekaan lyönyt itseään läpi pelien pelaamiseen tarkoitettuna alustana, on olemassa useita avoimia peliprojekteja. Avoimet peliprojektit voivat olla erittäin pitkäikäisiä mutta yhä aktiivisia. Myös kaupalliset peliprojektit hyötyvät niiden lähdekoodin vapauttamisesta eri tavoin. Jos pelikehittäjä ei ole vapauttanut pelinsä lähdekoodia, voivat aktiiviset peliharrastajat alkaa omin neuvoin tukemaan peliä avoimen ohjelmistokehityksen voimin. Avoimen pelikehityksen tukemiseen on olemassa jo useita kirjastoja ja työkaluja. Myös modifikaatioiden kehittäminen kaupallisiin peleihin on yksi avoimen pelikehityksen muoto, joka voi hyödyttää alkuperäisen pelin kehittäjiä.</p> <p>ACM Computing Classification System (CCS):</p> <p>K.6.3 [Software Management],</p> <p>K.8.0 [General]</p>			
Avainsanat — Nyckelord — Keywords			
pelit, tietokonepelit, avoin ohjelmistokehitys, avoin lähdekoodi, modit, pelimoottorit			
Säilytyspaikka — Förvaringsställe — Where deposited			
Muita tietoja — övriga uppgifter — Additional information			

Sisältö

1	Johdanto	1
2	Linux ja pelit	2
3	Esimerkki: Roguelike-pelit	2
4	Vapautetut pelit	4
5	Vanhojen pelien henkiinherättäminen	6
6	Avoimet pelimoottorit ja työkalut	7
7	Modit	9
8	Yhteenveto	10
	Lähteet	11

1 Johdanto

Tässä kirjoitelmassa käsittelen avoimen ohjelmistokehityksen ja pelien aihetta eri näkökulmista. Ensin tarkastelen Linux-projektin ja pelien suhdetta. Linux ei ole saavuttanut suurta suosiota pelaamisen alustana ja siihen on olemassa monta syytä.

Seuraavaksi käsittelen pelityyppiä, joka on pysynyt hengissä erityisesti avoimen pelikehityksen ansiosta jo lähes kolmenkymmenen vuoden ajan. Roguelike-peleinä tunnetun peligenren juuret ovat avoimessa kehityksessä ja elinvoimaisimmat genren edustajat ovat kehittyneet monipuolisesti avoimin menetelmin tähän päivään asti.

Kirjoituksen seuraavassa osassa tarkastelen miten pelinkehittäjä voi hyötyä pelinsä suljetun lähdekoodin avaamisesta. Hyviä syitä koodin vapauttamiseen on useita, ja pelinkehittäjä voi hyötyä päätöksestä myös taloudellisesti. Esimerkkinä käytän id Software -pelitaloa, joka on vapauttanut jo useamman pelinsä lähdekoodit.

Kerron kirjoituksessani myös joistain avoimista projekteista, joiden tarkoitus on säilyttää jo unohdetut pelit elinvoimaisina. Vanhoilla peleillä on yhä aktiivisia pelaajia, ja jotkut ovat valmiita uhraamaan aikaa ja vaivaa varmistaakseen että pelejä voidaan pelata jatkossakin, vaikkeivat niiden alkuperäiset kehittäjät niistä enää välitä.

Seuraava osa aineesta käsittelee avoimia pelimoottoreita. Pelikehitys pitää sisällään muutakin kuin pelin sisällön suunnittelua ja toteutusta. Avoimen lähdekoodin pelien kehittämiseksi olemassa on oltava tarvittavaa pohjateknologiaa. Haasteeseen on herätty ja avoimen pelin kehittäjä voi valita useammasta vaihtoehdoista.

Päätän kirjoitukseni tarkastelemalla kuinka pelimodit, eli harrastajien tekemät modifikaatiot kaupallisiin peleihin liittyvät avoimeen ohjelmistokehitykseen. Kuten suljetun koodin avaamisella, on modikehityksen tukemisella monia hyviä puolia myös kaupalliselle pelikehittäjälle.

2 Linux ja pelit

Yleisistä avoimen lähdekoodin projekteista tunnetuin lienee Linux-käyttöjärjestelmä kaikkine siihen liittyvine sivuprojekteineen. Vaikka Linuxilla on yhä enemmän käyttäjiä ja käyttötarkoituksia, se ei ole saavuttanut suurta suosiota pelaajien keskuudessa. Syitä tähän on useita [Mace01]. Puutteellinen laitteistotuki ja puuttuvat ajurit ovat vaikeuttaneet varsinkin 3D-pelien pelaamista. Pelifirmat eivät ole lähteneet mukaan tukemaan Linuxia, mikä on ymmärrettävää taloudellisista syistä. Yhä harvemmat PC-pelit kun julkaistaan Mac-alustalle, ja myös Windowsin merkitys pelialustana on kutistunut pelikonsolien suosion kasvaessa. Lisäksi asiaan liittyy ”muna vai kana”-tyyppinen ongelma: Linux ei toimi pelialustana ilman pelejä tai pelinkehittäjiä. Tutkittaessa Linux-kehittäjien työpanoksia näihin liittyvän metadatan perusteella todettiin, että vain suhteellisen pieni osa työpanoksista liittyi peleihin [DWJG02]. Tutkimuksessa pääteltiin, että Linux-kehittäjät ovat yleensä luonteeltaan vakavia, eikä heitä kiinnosta kehittää pelejä. Tutkimus ei tietenkään anna koko kuvaa avoimen lähdekoodin peleistä tai niiden kehittäjistä, joten on tarkasteltava tarkemmin joitain tällaisia projekteja.

3 Esimerkki: Roguelike-pelit

Hyvä esimerkki avoimen lähdekoodin peliprojekteista ovat ns. ”roguelike”-pelit, eli Roguen kaltaiset pelit. Tämä on yleisnimitys yksinkertaiseen merkki- tai tiiligrafiikkaan perustuville rooliseikkailupeleille. Genren peleille yhteisiä piirteitä ovat karu ulkoasu, monimutkasiin näppäinkomentoihin perustuva ohjaus, satunnaisesti generoidut kentät, korkea oppimiskynnys sekä erityisesti se, että pelihahmon kuolema on lopullinen, eli pelaajan on tällöin aloitettava koko peli alusta. Näiden piirteiden takia pelityyppi ei ole useimpien pelaajien saati kaupallisten pelinkehittäjien

näkökulmasta erityisen houkutteleva. Kaikki roguelike-pelit eivät perustu avoimeen lähdekoodiin, mutta genren tunnetuimpien ja pitkäikäisimpien edustajien kehitys on vahvasti liitoksissa avoimen lähdekoodin kehitykseen.

Nimensä roguelike-pelit ovat saaneet Rogue-nimisestä pelistä, joka sai alkunsa 1980-luvun alussa Yhdysvaltain yliopistomaailmassa [Wich97]. Ennen henkilökohtaisten mikrotietokoneiden yleistymistä yliopistoissa oli käytössä pääasiassa tekstipohjaisia päätteitä, jotka olivat yhteydessä keskustietokoneeseen. Vuoden 1980 tienoilla Berkeleyn yliopistossa Ken Arnold kehitti curses-ohjelmakirjaston niin ikään Berkeleysessä kehitettyyn BSD UNIX-ympäristöön. Kirjaston avulla päätteen näytölle voitiin piirtää merkkipohjaista ”grafiikkaa”. Santa Cruzin yliopiston opiskelijat Michael Toy ja Glenn Wichman, jotka olivat 70-luvulla kehitetyn ”Adventure”-tekstiseikkailun innokkaita pelaajia, päättivät kehittää oman, graafisen seikkailupelinsä curses-kirjaston avulla. Syntyi ”Rogue”, merkkigrafiikalla ylhäältäpäin kuvattu fantasiaroolipeli, jossa pelaaja seikkailee hahmollaan vaarallisissa luolastoissa. Pelin kenties merkittävin piirre oli se, että pelimaailma generoitiin satunnaisesti, ja jokaisella pelikerralla luolastot olivat erilaiset.

Michael Toy siirtyi Santa Cruzista Berkeleyn yliopistoon ja tapasi Ken Arnoldin, joka liittyi kehittämään Rogue-peliä. Rogue otettiin mukaan BSD UNIXin versioon 4.2, ja vapaassa levityksessä se levisi useisiin yliopistoihin ympäri maailman. Peli sai useita innostuneita pelaajia. Yksi heistä oli Jay Fenlason, joka tutustui peliin vierailulla Berkeleyssä ollessaan vasta koululainen [Bres00]. Fenlason ei onnistunut saamaan itselleen pelin lähdekoodia, joten hän alkoi kehittää omaa peliään, joka toimisi Roguen tavoin. Fenlasonin kehittämä peli sai nimen ”Hack”. Hackia alettiin kehittää eri tahoilla, ja siitä ilmestyi useita versioita eri alustoille, kunnes Mike Stephenson yhdisti eri versiot NetHack-projektiksi [Raym03]. Nimen Net-osa viittaa siihen että peliä alettiin kehittää internetin välityksellä. Pelin aktiivisimmat kehittäjät muodostivat ydinryhmän, DevTeamin. Tähän tiimiin liittyi myös Eric S.

Raymond, joka tunnetaan parhaiten merkittävän avoimia ohjelmistoja käsittelevän kirjoituksen ”The Cathedral and the Bazaar” kirjoittajana [Raym00a].

NetHackin kehitys on jatkunut 1980-luvulta 2000-luvulle saakka. Joidenkin päivitysten välillä on kulunut useampia vuosia, kun taas joskus päivityksiä on ilmestynyt useammin. Viimeisin versionumero 3.4.3 on vuodelta 2003. NetHackin kehitystiimi ei kerro julkisesti pelin kehityksen vaiheista. Pelin avoimen lähdekoodin ansiosta vuosien varrella on kuitenkin julkaistu useita epävirallisia päivityksiä ja Nethack-variantteja, joista merkittävin on Slash’EM. Monet näiden projektien lisäämät ominaisuudet ovat päätyneet osaksi NetHackia, ja jotkin niistä ovat muuttaneet peliä huomattavasti. Tämän seurauksena NetHack onkin tunnettu sen erittäin monimutkaisista säännöistä ja vaikutussuhteista. Pelin kehityksen aikana monia sen osia on jouduttu suunnittelemaan ja kirjoittamaan uudelleen [Raym03]. NetHack on silti yhä rakenteeltaan vahvasti monoliittinen, ja koodin eri osat ovat tiukasti sidoksissa toisiinsa [WaNC02]. Tutkittaessa pelin kehitystä on havaittu, että vaikka NetHackin rakenteellinen monimutkaisuus on laskenut uusien versioiden myötä, sen laskennan logiikka on monimutkaistunut [SiVL06]. Tutkimuksessa tehtiin myös se huomattava havainto, että pelin koodirivien määrä ja pakatun pelin koko ovat kasvaneet tasaisesti jokaisen version myötä, vaikka ns. ”Lehmanin lait” ennustavat ohjelman koon kasvun hidastuvan.

4 Vapautetut pelit

Avoimen ohjelmistokehityksen kasvava suosio on saanut monet kaupalliset pelikehittäjät julkaisemaan peliensä lähdekoodeja. Edelläkävijänä tällä saralla on toiminut id Software -pelitalo ja sen pääohjelmoija John Carmack. Carmack, joka on vastuussa id Softwaren pelimoottoreiden kehityksestä, on julkaissut merkittävimpien pelimoottoriensa lähdekoodit GPL-lisenssin alla. Tämä suuntaus alkoi 1990-luvun

puolivälissä id Softwaren alkaessa tukea pelaajien ja muiden kolmansien osapuolten tekemiä lisäyksiä Doom-peleihin avaamalla pelien speksejä [Raym00b]. Tästä oli id Softwarelle myös taloudellista hyötyä. Kun Doomiin alettiin tehdä epävirallisia lisäosia, id saattoi sopia näiden kaupallisesta julkaisemisesta. Lopulta Doomien lähdekoodi julkaistiin vuonna 1997. Eric S. Raymond näkee yhtenä syynä tähän päätökseen sen, että Doomien moninpeliominaisuudet nousivat tärkeimmäksi syyksi pelin jatkuvaan suosioon. Koska moninpelikoodi vaati kehittäjiltään yhä enemmän huomiota ja tietoliikenteeseen liittyvä koodi on omiaan kehitettäväksi avoimesti, id Softwaren kannatti tehdä lähdekoodista avointa. Toinen tärkeä syy Raymondin mukaan olivat markkinapaineet kilpailijoilta. Jos id Software ei olisi julkaissut pelinsä lähdekoodia tarpeeksi ajoissa, olisi jokin vastaava kilpaileva projekti saattanut viedä huomion ja sen mukana kehittäjät avoimelta Doom-projektilta.

Seuraavaksi id Software vapautti Quake-pelin lähdekoodin vuonna 1999, mutta pian John Carmack kohahdutti avoimen lähdekoodin yhteisöjä nostamalla esiin ongelman: pelin avoin lähdekoodi mahdollisti huijaamisen moninpeleissä asiakasohjelman koodia muokkaamalla. Ratkaisuksi Carmack esitti, että asiakaspäässä käytettäisiin suljetun lähdekoodin ohjelmaa, joka varmistaisi asiakasohjelman olevan hyväksytty ja hoitaisi kommunikoinnin pelipalvelimen kanssa. Eric S. Raymond kommentoi ehdotusta huomauttamalla, että syy huijausmahdollisuuden olemassaoloon oli se, että pelin tietoturvaa oli heikennetty suorituskyvyn parantamiseksi, ja että mikäli Quake olisi ollut alunperin avoin projekti, ongelman aiheuttaneen tekniikan käyttöä ei olisi edes harkittu [Raym99]. Tämä on Raymondin mukaan esimerkki siitä, että avoin lähdekoodi parantaa tietoturvaa pakottamalla kehittäjät käyttämään todistettavasti varmoja menetelmiä.

Myöhemmin id Software on julkaissut peliensä ”Quake II” ja ”Quake III Arena” lähdekoodit vuonna 2001 ja vuonna 2005. Näistä teknologioista id käyttää nimiä ”id Tech 2” ja ”id Tech 3”. Joel Westin ja Scott Gallagherin artikkelin [WeGa06] mukaan

id Softwarella oli lähdekoodin vapauttamisessa kaksi tavoitetta: id haluaa ensinnäkin auttaa pelaajayhteisöä kehittämään modeja ja muita parannuksia ja toiseksi käyttää kaksoislisensiointimallia saadakseen lisensiointituloja kaupallisilta pelinkehittäjiltä. Tässä strategiassa firma tukee oman ohjelmistonsa avointa kehitystä, mutta samalla myy ohjelmistonsa kaupallisia lisenssejä maksukykyisille yrityksille.

Avoimen lähdekoodin kehitys on myös mahdollistanut id Softwaren pelien porttaamisen eri alustoille, ja nykyään Doom- ja Quake-pelit tukevat useita eri laitealustoja ja käyttöjärjestelmiä. Tämä lienee tärkeä syy Bungie Studios -pelitalon päätökseen vapauttaa Marathon 2 -pelinsä lähdekoodi vuonna 2000. Bungien tukemana syntyi Aleph One -projekti, joka mahdollistaa aiemmin lähinnä Mac-alustalle julkaistujen Marathon-pelien pelaamisen niin Windows-, Linux- kuin Mac OS X -järjestelmissä [Alep08]. Tukemalla useampia alustoja pelikehittäjä voi kasvattaa tunnettavuuttaan ja mainettaan. Avoimet projektit ovat hyvää pr:ää, varsinkin jos ne liittyvät hyväntekeväisyyteen, kuten ”One Laptop Per Child” -projekti. Electronic Arts vapautti julkaisemansa vanhan SimCity-pelin lähdekoodin, jotta peli voitiin ottaa mukaan OLPC-projektiin. EA pitää kuitenkin samalla huolen siitä, ettei se menetä arvokasta SimCity-tavaramerkkiään, joten avoimen lähdekoodin versio pelistä tunnetaan nimellä Micropolis.

5 Vanhojen pelien henkiinherättäminen

Kuten edellä mainittiin, jotkin pelinkehittäjät julkaisevat vanhempien peliensä lähdekoodeja, jotta pelejä voitaisiin pelata myös moderneilla alustoilla. Toiset kehittäjät taas joko eivät välitä asiasta, tai ovat hukanneet alkuperäisen lähdekoodin. Monet pelifirmat ovat menneet konkurssiin, tulleet ostetuiksi tai muuten lopettaneet toimintansa. Niinpä jotkin innokkaat peliharrastajat ovat aloittaneet omatoimisesti avoimia projekteja, joiden tarkoitus on saada monet kuolleina pidetyt pelit

toimimaan nykykoneilla. Tällaiset herätysprojektit saattavat onnistua vain, jos pelin ympärillä on ”omistautuneita ja innokkaita käyttäjä-kehittäjiä, jotka ovat valmiita sijoittamaan aikaa ja taitojaan pitääkseen kulttuuriperintönsä elossa” [Scac04].

Yksi tärkeimmistä tällaisista projekteista on 1980- ja 1990-luvuilla pelihalleissa pelattujen kolikkopelien emulointiin keskittyvä MAME-projekti (Multiple Arcade Machine Emulator), jonka avulla tuhansia kolikkopelejä on herätetty uudelleen henkiin [MAME08]. MAME ei ole ainoa projekti, joka on kasvanut innokkaiden kehittäjien ansiosta. Toinen merkittävä samankaltainen projekti, ScummVM oli aluksi tarkoitettu vain LucasArts-pelitalon SCUMM-skriptikieltä käyttävien seikkailupelien pelaamiseen [Scum08]. Projekti kuitenkin kasvoi sen alkaessa tukea myös muiden pelifirmojen aivan eri tekniikoihin perustuvia seikkailupelejä. Tämä on ollut pääasiassa innokkaiden vapaaehtoisten takaisinmallinnuksen (reverse engineering) ansiota, mutta jotkin kaupallisten seikkailupelien kehittäjät ovat auttaneet ScummVM-projektia julkaisemalla vanhojen peliensä lähdekoodit.

6 Avoimet pelimoottorit ja työkalut

Yhtenä avoimen pelikehityksen esteenä on ollut avointen yleiskäyttöisten pelimoottorien puute. Doom- ja Quake-moottoreita on vapautettu jo 1990-luvun lopusta lähtien, mutta ne soveltuvat parhaiten lähinnä ensimmäisen persoonan ammuskelupelien toteutukseen. Nykyään onkin saatavilla useita avoimia 3D-moottoreita. Näistä OGRE on erittäin pitkälle kehitetty ja suosittu [OGRE08]. Se ei ole varsinainen pelimoottori, vaan pelkkä 3D-grafiikkamoottori. Se ei yksin riitä pelin kehitykseen, joten pelinkehittäjän on joko koodattava itse muut osat kuten tekoäly-, verkko-, ääni- ja fysiikkakoodi tai käytettävä valmiita kirjastoja. Toinen vastaava projekti on Irrlicht, jota yleisesti pidetään erittäin tehokkaana ja OGREa helpommin opittavana sen yksinkertaisemman arkkitehtuurin ansiosta, vaikkei se ehkä tarjoa yhtä

paljon ominaisuuksia [Gebh08]. Molemmat tukevat useita alustoja ja ovat helposti laajennettavia. Näiden piirteiden vuoksi niitä voidaan käyttää monenlaisiin 3D-peli- ja simulaatioprojekteihin.

Vaikka 3D-grafiikka onkin nykyään kaupallisissa peleissä usein perusedellytys, on etenkin harrastelijakehittäjien keskuudessa tilausta avoimille 2D-pelimoottoreille. FIFE-projekti vaikuttaa erittäin lupaavalta 2D-pelimoottorilta ”isometristä” kuvakulmaa käyttävien pelien toteuttamiseen [FIFE08]. Projekti oli aluksi tarkoitettu PC:lle vuonna 1997 julkaistun Fallout-roolipelin ja sen vuonna 1998 ilmestyneen jatko-osan pelaamiseen useammilla alustoilla sekä pelien laajentamiseen ja modernisointiin. Projektin tarkoitus on kuitenkin muuttunut ja nykyään se pyrkii toimimaan avoimena alustana isometrisestä kuvakulmasta kuvattujen 2D-pelien toteutuksessa. Tällainen kuvakulma on omiaan esimerkiksi roolipeleihin ja strategiapeleihin. Projekti on tällä hetkellä aktiivinen ja on jo edennyt tarpeeksi pitkälle, jotta sitä voidaan käyttää peliprojektien toteuttamiseen.

Avoimet pelimoottorit käyttävät pääasiassa avoimen lähdekoodin kirjastoja ja työkaluja. Esimerkiksi FIFE käyttää toteutuksessaan avointa SDL-kirjastoa ja Python-skriptikieltä. Pelikehittäjä voi kuitenkin usein halutessaan käyttää projektissaan kaupallisia tai suljetun lähdekoodin kirjastoja. Esimerkiksi OGRE-perustaisessa projektissa voidaan käyttää avoimen fysiikkamoottorin sijasta vaikkapa nVidian PhysX-fysiikkamoottoria, josta on olemassa ilmainen versio, joka ei sisällä lähdekoodia, sekä kaupallinen, lähdekoodin sisältävä versio [OGRE08]. Avoimia pelimoottoreita, kirjastoja, väliohjelmistoja ja työkaluja käytetään myös useissa kaupallisissa peleissä. Jotkin avoimet kirjastot ovatkin saaneet alkunsa kaupallisten kehittäjien tarpeista ja saavat näiltä jatkuvaa tukea. Tällä hetkellä vaikuttaa melkein siltä kuin avoimia pelimoottoreita olisi enemmän kuin tarpeeksi, mutta niitä hyödyntäviä avoimia ja aktiivisia peliprojekteja voi olla vaikeampi löytää. Tärkeä syy on varmasti avoimien pelien vaatiman sisällön, grafiikan, äänien yms. tuottamiseen liittyvät ongelmat.

Avoimen pelin kehittäjän voi olla vaikea saada huomiota pelilleen itse tuottamansa grafiikan (programmer art) avulla. Samat ongelmat ilmenevät myös seuraavaksi käsiteltävien modien yhteydessä.

7 Modit

”Modi”-käsitteellä tarkoitetaan nykyään muutoksia pelin koodiin, ja modi on yleensä yhdistelmä uusia kenttiä, pelisääntöjä ja grafiikkaa [Clev01]. Modien koko voi vaihdella pienistä korjaustiedostoista valtaviin ”total conversion” -projekteihin, jotka korvaavat alkuperäisen pelin sisällön kokonaan uudella. Modeja on kehitetty peleihin jo NetHackin alkua ajoista. Avointen peliprojektien yhteydessä modit ja muut epäviralliset päivitykset ja lisäykset voidaan ottaa mukaan viralliseen versioon, mikä on luonnollinen tapa kehittää avoimia projekteja. Kaupallisten, suljetun lähdekoodin pelien kehittäjät eivät ole aikaisemmin olleet yhtä suojeita modeille. Doom-modien suuri suosio kuitenkin havahdutti id Softwaren, joka alkoi tukea modien kehitystä [Clev01]. Tätä pidetään yleisesti PC:n ”modikulttuurin” alkuna. Muita merkittäviä modikehitystä tukevia pelitaloja ovat Epic Games ja etenkin Valve Software.

Kaikkien modien lähdekoodi ei ole avointa. Avointa ohjelmistokehitystä käsittelevissä artikkeleissa modit on saatettu kuitenkin rinnastaa avoimiin ohjelmistoprojekteihin, koska useimmat modit kuitenkin käyttävät monia samankaltaisia käytäntöjä kuin varsinaiset avoimen lähdekoodin projektit. Tällaisia yhteisiä käytäntöjä ovat projektien ympärille rakentuneet yhteisöt, kommunikointikanavat, projektinhallinta ja versionhallintamenetelmät [Scac02] [Scac04]. Myös projekteihin osallistuvien yksilöiden motivaatiot ovat samanlaisia niin modiprojekteissa kuin avoimissa ohjelmistoprojekteissa [WeGa06]. Motivaatioista modien yhteydessä erityisesti korostuu mahdollisuus vakuuttaa mahdolliset työnantajat ja saada töitä pelialalta [Scac04] [WeGa06].

Voidaan myös tarkastella pelintekijöiden motivaatioita modien kehityksen tukemiseen. Pelit poikkeavat muista ohjelmistoista siten, että niiden kuluttajat saattavat kyllästyä niihin nopeasti. Modien avulla ”ulkoinen innovaatio pidentää alkuperäisen (sisäisesti kehitetyn) innovaation elinikää” [WeGa06]. Toisin sanoen, modattavat pelit pysyvät suosittuina pidempään [Clev01]. Ne myös vetoavat useampiin pelaajiin ja hämärtävät rajaa pelintekijän ja pelaajan välillä. Hyvät modit lisäävät pelin myyntiä, sillä lähes kaikki modit vaativat alkuperäisen pelin toimiakseen. Pelintekijät voivat lisäksi hyötyä myymällä peleistään uudelleenjulkaisuja ilmaiseksi kehitettyjen modien kera [Clev01].

8 Yhteenveto

Avoin ohjelmistokehitys on pelialallakin hyvä ratkaisu haasteisiin, joita perinteiset kaupallisen, suljetun lähdekoodin menetelmät nostavat esiin. Avoimet peliprojektit voivat olla pidempikestoisempia kuin tiukkoihin deadlineihin perustuvat kaupalliset peliprojektit. Pitkäkestoisissa projekteissa peleihin voi syntyä monimutkaisia vaikutussuhteita ja sääntöjä, jotka saavat aikaan ”syvempää” pelattavuutta. Avoimet projektit voivat pitää hengissä pelejä, joiden ylläpito ei ole enää kaupallisesti kannattavaa, ja etenkin modit voivat myös pidentää pelien hyllyikää ja generoida lisätuloja pelifirmoille. Kaupalliset pelikehittäjät voivat myös tukea avoimia kirjastoja ja ohjelmistoja kehitettäviä projekteja ja käyttää näiden tuotoksia omissa peleissään. Tulevaisuudessa ainakin id Software aikoo jatkaa peliensä lähdekoodien julkaisua ja myös muilla avoimen pelikehityksen aloilla tulevaisuudennäkymät ovat hyvät. Monet pelijulkaisijat ovat tajunneet myös vanhojen pelien arvon niiden nettimyynnin kautta, joten kenties vanhojen pelien lähdekoodeista pidetään jatkossa enemmän huolta, jotta pelejä voitaisiin pelata myös tulevaisuudessa. Tällaisen suuntauksen myötä mahdollisuudet useampien pelien lähdekoodien avaamiseen paranevat.

Lähteet

- Alep08 Aleph one -projektin kotisivu. <http://marathon.sourceforge.net/>. [5.9.2008]
- Bres00 Bresnick, J., On the train of life with Nethack's papa, joulukuu 2000. <http://www.linux.com/articles/5501>. [5.9.2008]
- Clev01 Cleveland, C., The Past, Present, and Future of PC Mod Development. *Game Developer*, 8,2(2001), sivut 46–49.
- DWJG02 Dempsey, B. J., Weiss, D., Jones, P. ja Greenberg, J., Who is an open source software developer? *Communications of the ACM*, 45,2(2002), sivut 67–72.
- FIFE08 Fife-pelimoottorin kotisivu. <http://www.fifengine.de/>. [5.9.2008]
- Gebh08 Gebhardt, N., Irrlicht engine -kotisivu. <http://irrlicht.sourceforge.net/>. [5.9.2008]
- Mace01 Macedonia, M., Will Linux be computer games' dark horse OS? *Computer*, 34,12(2001), sivut 161–162.
- MAME08 Mame-projektin kotisivu. <http://mamedev.org/>. [5.9.2008]
- OGRE08 Ogre-pelimoottorin kotisivu. <http://ogre3d.org/>. [5.9.2008]
- Raym99 Raymond, E. S., The Case of the Quake Cheats, joulukuu 1999. <http://www.catb.org/~esr/writings/quake-cheats.html>. [5.9.2008]
- Raym00a Raymond, E. S., The Cathedral and the Bazaar, 2000. <http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/>. [5.9.2008]

- Raym00b Raymond, E. S., The Magic Cauldron, 2000. <http://www.catb.org/~esr/writings/cathedral-bazaar/magic-cauldron/>. [5.9.2008]
- Raym03 Raymond, E. S., Guidebook for NetHack 3.4, 2003. <http://www.nethack.org/v343/Guidebook.html>. [5.9.2008]
- Scac02 Scacchi, W., Understanding the requirements for developing open source software systems. *Software, IEE Proceedings*, 149,1(2002), sivut 24–39.
- Scac04 Scacchi, W., Free and open source development practices in the game community. *Software, IEEE*, 21,1(2004), sivut 59–66.
- Scum08 Scummvm-projektin kotisivu. <http://scummvm.sourceforge.net/>. [5.9.2008]
- SiVL06 Simmons, M. M., Vercellone-Smith, P. ja Laplante, P. A., Understanding Open Source Software through Software Archaeology: The Case of Nethack. *Software Engineering Workshop, 2006. SEW '06. 30th Annual IEEE/NASA*, huhtikuu 2006, sivut 47–58.
- WeGa06 West, J. W. ja Gallagher, S., Challenges of open innovation: the paradox of firm investment in open-source software. *R&D Management*, 36,3(2006), sivut 319–331.
- Wich97 Wichman, G. R., A Brief History of "Rogue", 1997. <http://www.wichman.org/roguehistory.html>. [5.9.2008]
- WaNC02 Warren, R., Nafees, O. ja Champaign, J., Topics in Software Evolution and Design, 2002. http://www.cs.uwaterloo.ca/~omnafees/cs846/group_asst_1/NafeesWarreenChampaign-nethack-report.pdf. [5.9.2008]