Antti-Jussi Lakanen

NUORTEN PELIOHJELMOINTI

Tietotekniikan pro gradu -tutkielma

Aineenopettajankoulutuksen linja

6.5.2010

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Antti-Jussi Lakanen

Yhteystiedot: ajlakanen@gmail.com

Työn nimi: Nuorten peliohjelmointi

Title in English: Game programming course for teenagers

Työ: Pro gradu -tutkielma

Sivumäärä: 136+53

Linja: Aineenopettajankoulutus

Teettäjä: Jyväskylän yliopisto, tietotekniikan laitos

Avainsanat: Pro gradu, peliohjelmointi, ohjelmoinnin opettaminen, lyhytkurssi

Keywords: Pro gradu, master's thesis, game programming, teaching programming, intensive course

Tiivistelmä: Tässä pro gradu -tutkielmassa tutkitaan yläkouluikäisille (13–16-vuotiaat) tarkoitetun peliohjelmoinnin lyhytkurssin innostavuutta, ja vaikutuksia osallistujien suhtautumiseen luonnontieteiden opiskeluun. Osallistujien kurssikokemuksiin pyrittiin löytämään yhteyksiä taustatekijöistä. Lisäksi aineistosta etsittiin ryhmiä, jotka kuvaisivat tyypillisiä kurssille tulleita oppilaita. Tämän avulla on tarkoitus kehittää kurssin sisältöä, opetusmenetelmiä ja markkinointia. Tässä työssä esitellään myös pääpiirteet kurssin suunnittelusta ja toteutuksesta, ja lisäksi joitakin oppilaiden valmiista pelituotoksista ja pelisuunnitelmista.

Abstract: In this master's thesis we study an intensive game programming course for teenagers (from 13 to 16-year-olds), how inspiring the participants find the course and does attending the course have an effect on youngsters' interest in studying ICT, mathematics and science. The connections between course experiences and background information were sought for. In addition, from among the data we searched for groups that describe the

typical participants of this game programming course. This information will be used in developing the course content, teaching methods and marketing. In this study we also represent the essentials of the elaboration and implementation of the course. We also represent some of the participants' game blueprints and screenshots of the ready games.

Esipuhe

Ensimmäisenä haluan kiittää *Nuorten peliohjelmointikurssin* projektitiimiä: lehtori Vesa Lappalainen, Tero Jäntti, Tomi Karppinen, Janne Nikkanen ja tutkija Tuukka Puranen, teidän kanssanne on ollut hienoa tehdä töitä.

Kiitokset *Microsoftille* ja Jukka Wallasvaaralle kursseilla käytettyjen ohjelmistojen ja peliohjainten ystävällisestä tarjoamisesta.

Suurkiitos *Teknologiateollisuuden 100-vuotissäätiölle* merkittävästä huomionosoituksesta tätä projektia kohtaan, ja tulevien kurssien rahoittamisesta seuraavan kahden vuoden ajan.

Haluan kiittää myös *Agora Centeriä* ja professori Marja Kankaanrantaa sekä tutkija Tuula Nousiaista avusta tutkimussuunnitelmani tarkistamisessa, sekä kaikesta käytännön avusta tutkimuksen tekemisessä. Suurkiitos myös tutkielmani ohjaajalle, yliassistentti Leena Hiltuselle, joka jaksoi koko ajan kannustaa työn jatkamiseen, sekä lehtori Vesa Lappalaiselle, joka toimi tutkielmani toisena tarkastajana ja koko tämän kurssin idean isänä.

Jyväskylässä 6. toukokuuta 2010

Antti-Jussi Lakanen

Termiluettelo

Bugi	Ohjelmointivirhe, tietokoneohjelman lähdekoodissa oleva virhe.
Debuggeri	Tietokoneohjelma, jota käytetään muiden tietokoneohjelmien ohjelmointivirheiden (bugien) jäljittämiseen ja korjaamiseen.
IDE	Integrated Development Environment, integroitu ohjelmointiympäristö tai yleensä lyhyesti ohjelmointiympäristö, joka sisältää koodin tuottamisen apuvälineiden lisäksi kääntäjän ja debuggerin.
Ohjelmointikirjasto	Kokoelma ohjelmakoodia (esimerkiksi aliohjelmia, luokkia, ja niin edelleen), jota käytetään tietokoneohjelmia kehitettäessä. Esimerkiksi Nuorten peliohjelmointikurssi varten tehtyyn Jypeli-ohjelmointikirjastoon ohjelmoitua koodia voidaan ottaa käyttöön muissa ohjelmissa.

Sisältö

1 JOHDANTO 1

2 TUTKIMUSASETELMA 4

2.1 TUTKIMUKSEN KOHDE 4
2.2 TUTKIMUKSEN TAVOITE 5
2.3 KUVAILEVA TUTKIMUSASETELMA 5
2.4 AINEISTON KERÄYSTAVAT 6
2.5 TUTKIMUKSEN RAJAUS 6

3 TAUSTAA 8

3.1 LUONNONTIETEIDEN OPISKELU JA IT-ALAN KIINNOSTAVUUS TYÖNANTAJANA 8
3.2 LUONNONTIETEIDEN KYTKENNÄT OHJELMOINTIIN 11
3.3 ENNAKKOTIETO AUTTAA YLIOPISTO-OPINNOISSA 12
3.4 PELIT JA LEIRIT VARHAISEN REKRYTOINNIN MUOTONA IT-ALALLA 15

4 OHJELMOINNIN PARADIGMAT 18

4.1 MIKÄ ON OHJELMOINTIPARADIGMA 18
4.2 MIKSI OHJELMOINTIPARADIGMAN VALINTA ON TÄRKEÄÄ 19
4.3 DEKLARATIIVINEN OHJELMOINTIPARADIGMA 20
4.4 IMPERATIIVINEN OHJELMOINTIPARADIGMA 20
4.5 RAKENTEINEN OHJELMOINTI (IMPERATIIVINEN OHJELMOINTIPARADIGMA) 22
4.6 OLIO-OHJELMOINTI (IMPERATIIVINEN OHJELMOINTIPARADIGMA) 24

5 OHJELMOINNIN OPETTAMINEN 26

5.1 OPPIMINEN ETENEE ”ALHAALTA YLÖS” 26
5.2 KONSTRUKTIVISMI 28
5.3 PEDAGOGISIA MENETELMIÄ 29
5.3.1 Esittävä opetus 29
5.3.2 Tekemällä oppiminen 30
5.3.3 Kokemuksellinen oppiminen 32
5.3.4 Tutkiva oppiminen 33
5.3.5 Suunnittelun kautta oppiminen 35
5.3.6 Mallioppiminen 35
5.3.7 Pedagogiset menetelmät opetuksessa 36
5.4 ENSIOHJELMOINTI 36
5.5 ERILAISET LÄHESTYMISTAVAT OHJELMOINNIN OPETUKSESSA 37
5.6 ”OLIOT ENSIN”-LÄHESTYMISTAVASTA ”MUUTTUJAT ENSIN”-LÄHESTYMISTAPAAN 40
5.7 OHJELMOINNIN INTEGROINTI MUUHUN OPETUKSEEN 42
5.8 MUKAVUUSTASO ALKEISOHJELMOINNIN KURSSILLA ON MERKITYKSELLINEN 46
5.9 PELIN KEHITYS OHJELMOIMALLA JA PELINKEHITYSOHJELMISTOT 46
5.10 OPPILAAT SOVELLUSTEN TUOTTAJINA – OPPIMINEN OHJELMOINNIN AVULLA 48

6 PELIOHJELMOINTIKURSSIN SUUNNITTELU 50
6.1 ENNAKKOMARKKINOINTI 50
6.2 OSAAMISEN KEHITTYMINEN 50
6.3 TYÖKALUJEN VALINTA 52
6.3.1 Ohjelmointikieli: C# 52
6.3.2 Kehitysympäristö: Microsoft Visual Studio 54
6.3.3 Luokkakirjasto: XNA Game Studio 54
6.3.4 Fysiikka- ja matematiikkakirjastot: Physics2D.Net ja AdvanceMath 55
6.3.5 Jypeli – ”Kirjastojen kirjasto” 55
6.4 KURSSIN TEKNINEN VALMISTELU 56
6.5 AIKATAULU JA SISÄLTÖ 56
6.6 ENSIMMÄISEN HARJOITUSPELIN TEKEMINEN 58
6.7 OMAN PELIN TEKEMINEN 60
7 TUTKIMUKSEN TOTEUTTAMINEN 63
7.1 TUTKIMUSONGELMAT 63
7.2 TUTKIMUKSEEN VALITUT TUTKIMUSMENETELMÄT 65
7.3 ANALYSOINTIMENETELMÄT 66
7.3.1 Korrelaatioanalyysi 67
7.3.2 Tyypittely 68
7.3.3 Klusterianalyysi 68
8 TUTKIMUKSEN TULOKSET JA ANALYSOINTI 70
8.1 ALKUKYSELY (N=45) 70
8.2 ALKUKYSELYN KORRELAATIOANALYYSI 76
8.3 ALKUKYSELYN MONIVASTAUSMUUTTUJIEN ANALYSOINTI 80
8.4 LOPPUKYSELY (N=41) 91
8.5 LOPPUKYSELYN KORRELAATIOANALYYSI 100
8.6 LOPPUKYSELYN MONIVASTAUSMUUTTUJAN ANALYSOINTI 103
8.7 ALKUKYSELYN JA LOPPUKYSELYN KORRELAATIOANALYYSI 104
8.8 KLUSTERIANALYYSI 106
8.9 OPPILAIDEN TUOTOKSET 116
8.10 OHJAAJIEN ROOLI KURSSILLA 121
9 POHDINTA 122
9.1 PELIKESKEISEN OHJELMOINTIKURSSIN INNOSTAVUUS 122
9.2 OPPILASTYYPIT PELIOHJELMOINTIKURSSILLA 123
9.3 KEHITYSEHDOTUKSET 125
9.4 JATKOTUTKIMUSAIHEET 125
LÄHTEET 126
LIITTEET 137
LIITE 1. PELIKURSSIN MAINOS 137
LIITE 2. SAATEKIRJE OPETTAJILLE 138

LIITE 3. OPETTAJIEN SAATEKIRJEEN LIITEOSA 139
LIITE 4. JYVÄSKYLÄN YLIOPISTON TIEDOTE, 10.6.2009. 140
LIITE 5. ESIMERKKI PELISUUNNITELMAN MALLISTA. 142
LIITE 6. KURSSIEN ALKUKYSELYN KYSELYLOMAKE. 144
LIITE 7. ALKUKYSELYIDEN TULOKSET (N=45). 146
LIITE 8. KURSSIEN LOPPUKYSELYN KYSELYLOMAKE. 161
LIITE 9. LOPPUKYSELYIDEN TULOKSET (N=41). 163
LIITE 10. ESIMERKKI OPPILAAN TEKEMÄSTÄ PELISUUNNITELMASTA. 188
LIITE 11. ESIMERKKI OPPILAAN TEKEMÄSTÄ PELISUUNNITELMASTA. 189

1 Johdanto

Tietokone- ja videopeliteollisuus on kasvanut nopeasti miljardiluokan bisnekseksi. Lasten ja nuorten arkeen ovat jo vuosia kuuluneet monenlaiset video- ja tietokonepelit. Vielä 1980-luvun alussa jokaisessa kodissa ei ollut tietokonetta. Televisio useimmista talouksista kuitenkin löytyi (ITU 2009), ja niinpä pelikonsolit ja muut videopelit alkoivat tehdä tuloaan. Nykyään erilaisia pelaamiseen soveltuvia laitteita löytyy jo lähes jokaisesta suomalaisesta taloudesta (Kallio ym. 2009, 1).

Alkujaan pelit olivat hyvinkin yksinkertaisia, eikä niissä välttämättä ollut kunnollista juonta tai selkeää tavoitetta. Nykyään pelit ovat tietenkin monipuolistuneet, ja niitä voi pelata yksin, mutta monet pelit ovat siirtyneet verkkoon. Verkossa niitä voi pelata useampi pelaaja yhtä aikaa ja pelin tiimellyksessä luodaan myös sosiaalisia verkostoja. Koulut ja jopa päiväkodit ovat huomanneet verkkoyhteisöjen voiman, ja alkaneet hyödyntää toiminnassaan kasvatukseen ja opetukseen soveltuvia pelejä ja peleihin liittyviä verkkoympäristöjä (Kupiainen & Sintonen 2009). Pelejä ja pelaamista myös tutkitaan useiden alojen piirissä, sekä teknisillä että ihmistieteellisillä aloilla (Kallio ym. 2009, 1). Kallion ym. mukaan (2007, 119) digitaalinen pelaaminen on levittäytynyt laaja-alaisesti eri-ikäisten ja eri sukupuolta edustavien ihmisten keskuuteen. Kuitenkin aktiivisimpia nimenomaan peliharrastuksen suhteen ovat nuoret miehet ja pojat.

Tietokone- ja konsolipelien pelaaminen on hauskaa, mutta vielä hauskempaa on suunnitella ja tehdä itse pelejä, ja antaa muiden pelata. Tietokonepelien tekeminen ei kuitenkaan ole helppo tai yksinkertainen tehtävä. Esimerkiksi suomalaisen Remedy Entertainmentin kehittämän toimintapelin, *Max Payne*, ensimmäinen versio julkaistiin vuonna 2001. Pelin kehittäminen alkoi vuonna 1997 (Remedy 2010) ja ensimmäinen esittelyvideo pelin kehitysversiosta julkaistiin vuonna 1998 (emt.). Pelin kehitys julkaisupäivään mennessä oli vienyt siis noin neljä vuotta, henkilötyövuosista puhumattakaan. Pelitehtaissa saattaa työskennellä kymmeniä, jopa satoja ihmisiä kehittämässä pelin eri osa-alueita, kuten musiikkia, grafiikkaa, juonta, tietenkin ohjelmakoodia, ja niin edelleen. Tämä ei silti tarkoita, ettei pelejä voisi tehdä myös helposti ja nopeasti, kunhan ei nosta rimaa liian ylös. Pelejä voi oppia tekemään

suhteellisen pienellä panostuksella ja vaivalla, kunhan aiheeseen löytyy kiinnostusta ja halua.

Idea nuorten peliohjelmointikurssista syntyi Jyväskylän yliopiston ohjelmistotekniikan lehtorille, FT Vesa Lappalaiselle keväällä 2009. Hän huolestui yliopistojen IT- ja luonnontieteellisten oppiaineiden suosion vähenemisestä, sekä siitä, etteivät nuoret saa riittävästi oikeaa tietoa siitä, mitä näillä aloilla opiskelu todella on. Lappalainen esitti huolensa informaatioteknologian tiedekunnassa ja sittemmin Jyväskylän yliopisto rahoitti nuorten kesäkurssit. Huoli perustui siihen, että vaikka tietotekniikka, Internet, sähköinen media ja viestimet ovat arkipäivää niin kodeissa kuin työpaikoilla, ei IT-alan opiskelu ole yliopistojen hakijamäärien perusteella enää niin kiinnostavaa kuin 10 vuotta sitten (esim. Jyväskylän yliopisto 2009; Savela 2006, 40). Yliopistojen onkin alati muuttuvassa yhteiskunnassamme vakavasti harkittava niin markkinointikanavien ja -tapojen kuin opetusmenetelmien uudistamista. Nuorille suunnattu yliopistokurssi (luonnollisesti opetuksen sisältöä muokaten) nähtiin yhtenä pitkän tähtäimen vaihtoehtona kehittää ja parantaa markkinointia; kurssi toimi huomaamatta myös ikään kuin rekrytointikanavana – toki tuloksia tässä suhteessa voimme ehkä odottaa vasta vuosien päästä. Tutkimus osoitti kuitenkin, että mikäli pelien ja luonnontieteiden opiskelun väliset kytkökset pystytään heille osoittamaan tällaisella kurssilla, saattaa se lisätä heidän kiinnostustaan luonnontieteiden opiskelua kohtaan.

Tämä tutkimus osoitti myös, että nuoret pelaavat paljon, ja että yläkouluikäiset voivat sopivilla työkaluilla sekä ammattitaitoisessa ohjauksessa suunnitella ja toteuttaa pelattavan tietokonepelin yhden viikon aikana.

Kurssin suunnittelua varten kerättiin keväällä 2009 viisihenkinen suunnitteluryhmä. Ryhmä koostui kolmesta ohjelmoijasta sekä kahdesta ohjauksen suunnittelijasta. Toinen ohjauksen suunnittelijoista (ja tämän pro gradu -tutkielman tekijä) toimi kurssin vastaavana opettajana. Ohjelmoijien tehtävänä oli laatia käytettävissä olevia työkaluja hyväksikäyttäen ohjelmointikirjasto, jossa olisi valmiiksi ohjelmoituna esimerkiksi fysiikan ilmiöitä sekä matemaattisia malleja, joita oppilaat voisivat harjoituksissa ja peleissään hyödyntää.

Tutkimuksen tulosten ja ensimmäisten kurssien hyvän palautteen perusteella tehtiin päätös hakea vastaaville peliohjelmointikursseille seuraaville vuosille rahoitusta, jonka Teknologiateollisuuden 100-vuotissäätiö sittemmin myönsi.

Luvussa 2 esitellään tutkimuksen kohde, tutkimuksen tavoitteet sekä tutkimusasetelma. Lisäksi selvitetään aineiston keräystavat ja kerrotaan, mitä kerätystä aineistosta rajattiin tähän tutkimukseen. Luvussa 3 käydään läpi tutkimukseen liittyvää taustatietoa. Luvussa esitellään muun muassa tilastotietoja luonnontieteiden opiskelun suosiosta viime vuosina, sekä ennakkotiedon merkitystä yliopisto-opinnoissa.

Luvuissa 4 ja 5 käsitellään ohjelmointia ja pohditaan erityisesti ensiohjelmoinnin opettamisen problematiikkaa sen tiedon valossa, mitä oppimiskäsityksistä ja -menetelmistä tällä hetkellä tiedetään. Lisäksi luvussa 5 käydään läpi ohjelmointiaiheisia ja yleisesti tietoteknisiä sisältöjä perusopetuksen opetussuunnitelman perusteissa.

Luku 6 keskittyy kurssin valmistelujen ja sisältöjen esittelyyn. Luvussa esitellään kurssilla käytettäväksi valitut työkalut ja perustelut niille. Luvussa 7 selvitetään, kuinka tutkimus toteutettiin, esitellään tutkimusongelmat ja valittu tutkimusmenetelmä sekä aineiston analysointimenetelmät.

Luku 8 on varattu tulosten esittelylle ja niiden analysoinnille. Lisäksi lukuun on poimittu otteita oppilaiden työnäytteistä. Luvussa 9 esitetään yhteenveto tutkimuksesta ja sen tuloksista, esitellään ja pohditaan tutkimuksesta saatuja johtopäätöksiä sekä pohditaan, millaisia ideoita jatkotutkimuksesta työ poiki.

2 Tutkimusasetelma

Koska tämän tutkielman teoriaosuudessa viitataan useita kertoja tämän tutkimuksen kohteeseen ja tutkimusasetelmaan, on tässä vaiheessa syytä selvittää lyhyesti tutkimuksen kohde, tutkimuksen tavoite sekä kuvailla tutkimuksen asetelma ja tutkimusaineiston keräystavat. Tutkimuskysymykset ja -metodit esitellään myöhemmin teoriaosuuden jälkeen.

2.1 Tutkimuksen kohde

Tutkimuksen kohteena oli kesällä 2009 ensimmäisen kerran järjestetty Nuorten peliohjelmointikurssi (NPO). Kurssin järjestäjä oli Jyväskylän yliopiston tietotekniikan laitos. Tutkimuksen toteuttamiseen osallistui myös Agora Center, niin ikään Jyväskylän yliopiston yksikkö. Alun perin suunnitelmissa oli järjestää kurssi yhden kerran, ja kerätä tutkimusaineisto siitä, mutta koska kurssille ilmoittautui yli 50 nuorta, päätettiin kurssi järjestää kahteen otteeseen, ja ottaa kullekin kurssille noin 25 nuorta.

”Kutsut” kurssille lähetettiin postin välityksellä Jyvässeudun yläkoulujen matemaattisten aineiden lehtoreille, jotka sitten jakoivat ne edelleen oppilasryhmille. Projektiryhmän jäsen vieraili joissakin kouluissa mainosten jakamisen yhteydessä, mutta käytännössä mahdollisuutta sen tarkistamiseen, olivatko mainoskirjeet kaikissa tapauksissa todella menneet oppilaille saakka, ei ollut. Mainoskirjeitä tehtiin nelisen tuhatta, ja siis noin 50 heistä vastasi kutsuun, eli ilmoittautui kurssille, joten suhteessa mainosten määrään ilmoittautumisprosentti jäi reiluun yhteen. Mainokset menivät kouluille aivan toukokuun loppupuolella, lukuvuoden kiireisimpään aikaan, joten on todennäköistä, ettei mainos saavuttanut aivan kaikkia kohderyhmään kuuluneita oppilaita.

Kurssin tarkoituksena oli ohjelmoinnin alkeita opettamalla tutustuttaa nuoret peliohjelmoinnin maailmaan. Osallistujilta ei vaadittu aikaisempaa ohjelmointikokemusta. Kurssin mainoksessa mainittiin vain, että riittää, kun kurssille osallistuvalla "on avointa mieltä ja innostusta". Kurssin sisällöllisenä tavoitteena oli, että ohjelmoinnin alkeiden opettelun lisäksi kurssin aikana kukin oppilas suunnittelee ja toteuttaa oman kaksiulotteisen pelin, joka sisältää valmista tai itse tehtyä grafiikkaa, ääniä tai musiikkia.

Lukijalle huomautettakoon, että myöhemmin saatetaan ajoittain puhua ”kurssista” ja ”kursseista” synonyymeinä – koko ajan on kysymys tutkimuksen kohteena olevasta *kurssista, joka järjestettiin kaksi kertaa* mahdollisimman samalla sisällöllä ja toiminnoilla.

Yhden kurssin kesto oli viisi päivää. Kunakin päivänä opetusta annettiin viisi tuntia. Lisäksi päivään kuului säännölliset tauot sekä puolen tunnin ruokatauko, joten päivän pituus oli yhteensä noin 6–6,5 tuntia.

2.2 Tutkimuksen tavoite

Tämän tutkimuksen tavoitteena oli ensinnäkin kuvailla prosessia, mikä alkeisohjelmointikurssin järjestämiseen yläkouluikäisille liittyy. Toisaalta tulevaisuutta ajatellen yksi tärkeä tavoite oli antaa tietoa pelikeskeisen ohjelmointikurssin innostavuudesta sekä tietoa siitä, *ketä* kurssille osallistui. Tämän avulla oli tavoitteena selvittää, onko esimerkiksi iällä, sukupuolella tai aikaisemmalla ohjelmointikokemuksella vaikutusta kokonaistyytyväisyyteen tai siihen, kuinka hyödylliseksi oppilas kokee kurssin.

Tutkimuksen kohteena olevan kurssin tavoitteena oli myös kasvattaa nuorten kiinnostusta luonnontieteen alojen opiskelua kohtaan, sekä tuottaa tutkimustietoa innovatiivisista ohjelmoinnin opetusmenetelmistä (muun muassa tämä pro gradu -tutkielma).

2.3 Kuvaileva tutkimusasetelma

Tässä tutkimuksessa käytettiin kuvailevaa tutkimusasetelmaa. Kuvailevaa tutkimusasetelmaa käytetään, jos tarkoituksena on kuvailla, miten erilaiset ominaisuudet, mielipiteet jne. ovat jakautuneet tutkittavassa joukossa (Hirsjärvi ym. 2009, 155).

Kuvailevassa asetelmassa kuvailu- ja analyysimenetelminä käytetään tyypillisesti

- frekvenssijakaumia
- tunnuslukuja, kuten keskiarvo ja keskihajonta
- ristiintaulukointia ja korrelaatiokertoimia.

Kurssin aikana tehtiin tutkimuksellista seurantaa useassa vaiheessa. Oppilaat täyttivät kurssin aikana kyselylomakkeita, joka sisälsi niin avoimia kysymyksiä kuin asteikollisia

kysymyksiä. Kirjallisten vastausten lisäksi oppilaita havainnoitiin reaaliaikaisesti, ja näitä havaintoja kirjattiin ylös.

2.4 Aineiston keräystavat

Kurssin alussa tehtiin *alkukysely*, jossa kartoitettiin oppilaan kiinnostusta peleihin, sekä ohjelmointiin. Lisäksi alkukyselyssä selvitettiin oppilaalle mieluisimpia kouluaineita, sekä hänen kiinnostustaan jatkaa opintoja luonnontieteellisellä alalla. Edelleen oppilaalta kysyttiin, uskooko hän voivansa hyödyntää saamiaan oppeja jossakin kouluaineessa.

Kurssin aikana kunkin päivän päätteeksi tehtiin *välikysely*, joka koostui seitsemästä avoimesta kysymyksestä ja kahdesta asteikollisesta kysymyksestä. Välikyselyssä oppilaalta kysyttiin, mitä hän teki päivän aikana, mitä päällimmäisenä päivästä jäi mieleen, mikä hänelle päivässä oli ikävintä, helppoa ja vaikeaa, sekä kaksi kysymystä työskentelytapojen mielekkyydestä. Välikyselyn tavoitteena oli kartoittaa oppilaan senhetkisiä kokemuksia ja tuntemuksia niistä aiheista, jota sen päivän aikana oli opiskeltu.

Kurssin lopuksi tehtiin *loppukysely*, jossa selvitettiin, miten mielekkäänä oppilas koki kurssin ja vastasiko se odotuksia, kuinka vaikeita tehtävät olivat, ja millaisena hän koki kurssin ilmapiirin. Edelleen kysyttiin, miksi päätyi tekemään juuri sellaisen pelin, kuin teki. Lisäksi kysyttiin halukkuutta jatkaa opintoja myöhemmin luonnontieteellisellä alalla.

Kirjallisen aineiston lisäksi ohjaajat ja erilliset havainnoijat tekivät *havaintoja* oppilaiden käyttäytymisestä, reaktioista, toiminnasta ja sanallisista ilmaisuista. Havaintoja kirjattiin havaintovihkoihin, ja ne purettiin kunkin päivän päätteeksi.

2.5 Tutkimuksen rajaus

Tutkimusaineisto kerättiin yhteistyössä Jyväskylän yliopiston Agora Center -yksikön kanssa, ja aineiston keräämisen jälkeen päätettiin, että osa kurssilla kerätystä aineistosta jätetään huomiotta tässä pro gradu -tutkielmassa. Tämä yksinkertaisesti siitä syystä, että aineistoa kertyi hyvin paljon. Tässä työssä käsitellään kaikesta kerätystä aineistosta vain alku- ja loppukyselyt, sekä joitakin poimintoja laadullisesta aineistosta (lähinnä oppilaiden tuotoksista), kuitenkaan niitä tarkemmin analysoimatta.

Tämän tutkimuksen tekemisen suurin ongelma olikin siinä, että määrällistä aineistoa kertyi melko vähän (n=45). Toisaalta laadullista aineistoa kertyi erittäin paljon: jokaiselta oppilaalta kerättiin runsaasti vapaamuotoista kirjoitettua (eli laadullista) aineistoa, ja lisäksi havainnoijat tekivät subjektiivisia havaintoja koko kurssin ajan. Kaiken tämän aineiston käsittely ei pro gradu -tutkielman tekemiseen varatussa ajassa olisi ollut mielekästä. Tästä johtuen oli tehtävä valinta laadullisen ja määrällisen aineiston käsittelyn välillä, ja päädyin valitsemaan määrällisen aineiston käsittelyn. Laadullisen aineiston käsittely jää siis vielä jatkotutkimuksen varaan.

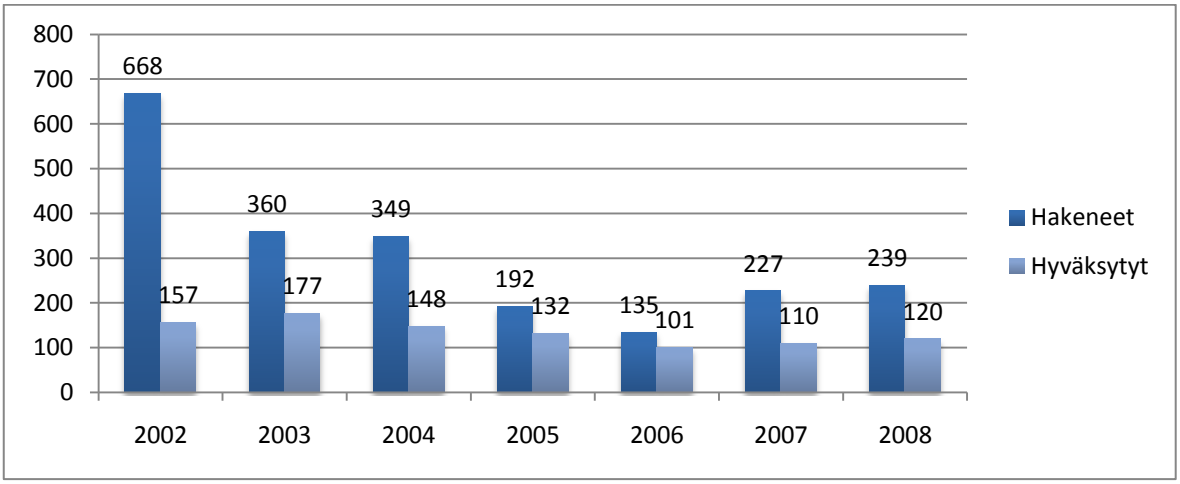
3 Taustaa

Tässä luvussa käydään läpi tutkimukseen liittyvää taustatietoa. Ensin kerrotaan IT- ja luonnontieteiden alojen opiskelijamäärien kehittymisestä 2000-luvulla. Sen jälkeen luodaan katsaus ohjelmoinnin ja luonnontieteiden välisiin kytkentöihin, ja viimeisenä tarkastellaan ennakkotietojen merkitystä suhteessa yliopisto-opinnoissa menestymiseen.

3.1 Luonnontieteiden opiskelu ja IT-alan kiinnostavuus työnantajana

Hieman ennen vuosituhannen vaihdetta koettiin valtava tietotekninen hype, jonka seurauksena muun muassa teknologiapörssi NASDAQ:n arvo seitsenkertaistui vuosina 1994–2000 ja vielä tuplasi arvonsa noin vuoden aikana vuosina 1999–2000 (Galbraith & Hale 2004, 2). Internet muuttui hetkessä tiedon välityskanavasta tiedon syntykanavaksi, missä kaikki halusivat olla mukana. Yritykset maksoivat valtavia rahoja siitä, että saivat omat tuotteensa ja palvelunsa sähköiseen muotoon kaikkien ihmeteltäviksi. Internetin ja langattomien palveluiden yleistyminen aiheutti valtavat odotukset niiden käytölle ja mahdollisuuksille, mikä osaltaan edesauttoi ylikuumenemista ja taloudellista yliarvostusta. Kuten hyvin tiedetään, hype kesti muutaman vuoden, jonka jälkeen ”IT-kupla” puhkesi ja monet yritykset kaatuivat.

IT-kupla ja valtavat odotusarvot sähköisiä viestimiä, palveluita ja Internetiä kohtaan näkyivät myös uusien opiskelijoiden innokkuudessa hakeutua IT-alalle opiskelemaan (Siljander 2009, 15–16). Pienen, jopa tappiota tuottavan IT-alan yrityksen markkina-arvo saattoi olla suurempi kuin suuremman liikevaihdon ja taseen omaava yritys, joka toimi ns. perinteisellä alalla. Raha kiinnosti luonnollisesti myös nuoria, jotka hakeutuivat teknillisten alojen koulutuksiin. Kuplan puhkeamisen jälkeen kiinnostus luonnontieteellisten ja IT-alan yliopistokoulutukseen on vähentynyt (mts. 15). Esimerkiksi Jyväskylän yliopiston tietotekniikan laitoksella (kuva 1) vuodet 2002–2008 ovat olleet paria viime vuotta lukuun ottamatta alamäkeä perusopiskelijaksi hakeneiden määrän suhteen.

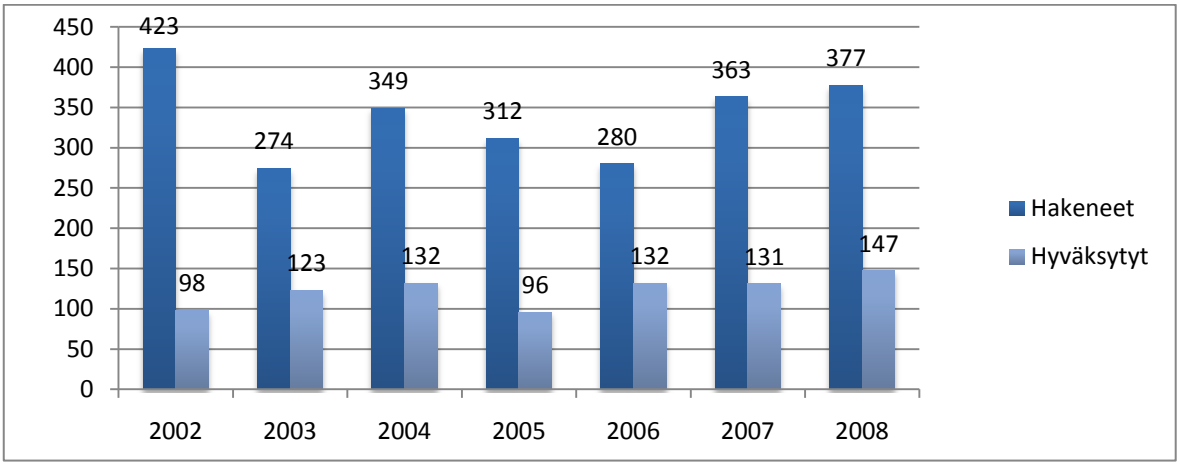


Kuva 1. Jyväskylän yliopiston tietotekniikan laitokselle perusopiskelijaksi hakeneet ja hyväksytyt vuosina 2002–2008 (Jyväskylän yliopisto 2009).

Vaikka vuosina 2007–2008 hakemusten määrä onkin hieman lisääntynyt (kuva 1), on hyväksyttyjen hakemusten määrä noussut huomattavasti vähemmän (Jyväskylän yliopisto 2009). Yksi hakijamäärän lisäystä selittävä tekijä on informaatioteknologian tiedekunnan merkittävät ponnistukset opiskelijarekrytoinnin alueella, ja myös tietotekniikan laitos on panostanut aiempaa enemmän opiskelijoiden rekrytointiin (Ihanainen 2010).

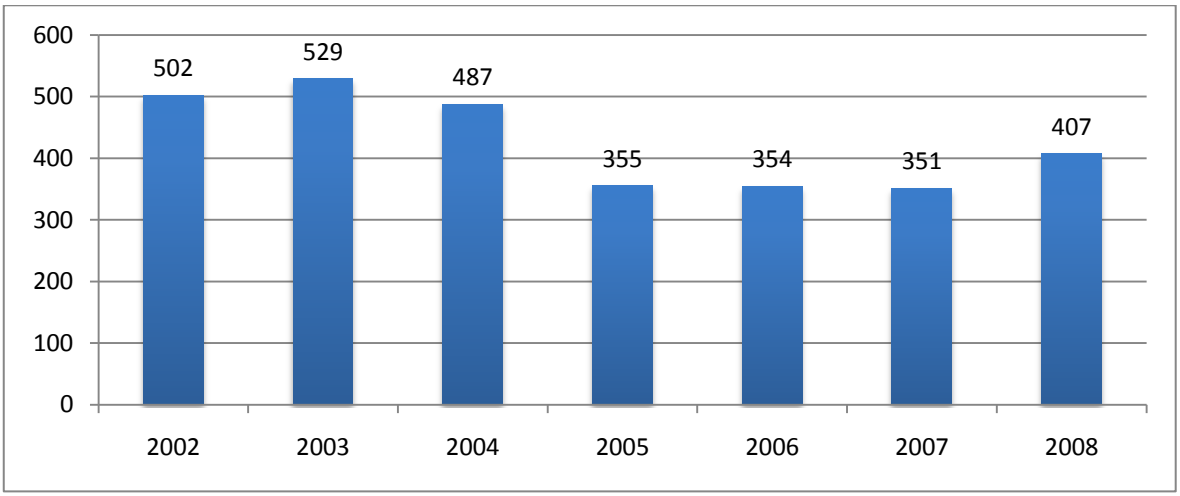
Vuonna 2005 Suomessa otettiin käyttöön yhteisvalintakoe, jossa yhdellä hakulomakkeella voi hakea useaan eri yliopistoon (Savela 2006). Vuodesta 2005 eteenpäin luvuissa ovat mukana vain ne, jotka ovat asettaneet Jyväskylän yliopiston haussa ensimmäiselle sijalle. Kaikkiaan tietotekniikan laitokselle haki vuonna 2005 580, vuonna 2006 463, 2007 466 ja vuonna 2008 453 hakijaa (Jyväskylän yliopisto 2009).

Vastaavasti Jyväskylän yliopiston tietojärjestelmätieteiden laitoksella tilanne ei ole muuttunut niin dramaattisesti (kuva 2), vaikkakin myös siellä laskua on tapahtunut 2000-luvun alusta (Jyväskylän yliopisto 2009).



Kuva 2. Jyväskylän yliopiston tietojärjestelmätieteen laitokselle perusopiskelijaksi hakeneet ja hyväksytyt vuosina 2002–2008 (Jyväskylän yliopisto 2009).

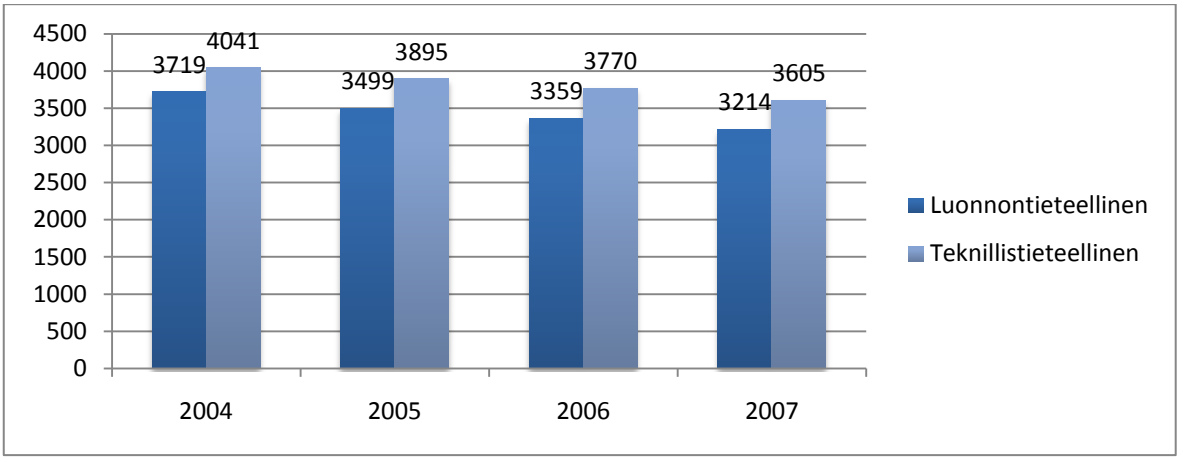
Vastaava tilanne on Jyväskylän yliopiston matemaattis-luonnontieteellisen tiedekunnalla: esimerkiksi vuonna 2007 (Alén ym. 2008, 37) uusien opiskelijoiden määrä koko tiedekunnassa oli noin 30 prosenttia pienempi kuin vuonna 2002 (kuva 3). Vuonna 2008 tilanne tosin kääntyi hienoiseen nousuun.



Kuva 3. Jyväskylän yliopiston matemaattis-luonnontieteellisen tiedekunnan uudet opiskelijat vuosina 2002–2008 (Alén ym. 2008, 37).

Jos tarkastellaan asiaa valtakunnallisella tasolla, ei tilanne näytä kovin paljon paremmalta (kuva 4). Suomen yliopistojen luonnontieteellisen ja teknillistieteellisen opintoalojen yhteenlaskettu uusien opiskelijoiden kokonaismäärä on pudonnut neljä vuotta peräkkäin vuoteen 2007 saakka: luonnontieteellisen alan uusien opiskelijoiden määrä on pudonnut

valtakunnallisesti 13,6 prosenttia ja teknillistieteellisen alan 10,8 prosenttia (Tilastokeskus 2008).



Kuva 4. Yliopistojen uudet opiskelijat opintoaloittain vuosina 2004–2007 (Tilastokeskus 2008).

Suosion lasku ei koskettele pelkästään opiskeluintoa. Tietotekniikkaa pidettiin vielä kymmenen vuotta sitten houkuttelevimpana alana toimihenkilöiden keskuudessa, ja ICT-alan työnantajia kiinnostavimpina (Hirvelä 2010). Nykyään tietotekniikan alan työnantajat kuuluvat keskikastiin samalla kun kaupan ala on noussut suosituimmaksi työnantajaksi (emt.). On mielenkiintoinen kysymys, mistä kiinnostuksen lasku johtuu, ja onko tilanteeseen lääkettä. Tähän palataan luvussa 3.4.

3.2 Luonnontieteiden kytkennät ohjelmointiin

Otetaan esimerkkinä kaksi luonnontieteen oppiainetta, matematiikka ja fysiikka, ja tarkastellaan hetki näiden aineiden kytkeytymistä ohjelmointiin.

Ensinnäkin ohjelmoinnin ja matematiikan välillä on aivan selvästi olemassa yhteys (Dijkstra 1964, 1). Aivan kuten matematiikka, ohjelmointi on mitä suurimmassa määrin ongelmanratkaisua. Annettuun ohjelmointiongelmaan pyritään löytämään ratkaisu käyttäen induktiota, logiikkaa ja muita matemaattisia työkaluja, mutta ainoaa oikeaa ratkaisua ei välttämättä ole olemassa – aivan kuten matematiikassa. Toisaalta on olemassa ongelmia, joiden todistus pelkästään matematiikan työkalujen avulla olisi ”tylsä ja monimutkainen”, mutta tietokoneen ja matemaattisen induktion avulla mielenkiintoinen ja yksinkertaisempi

(Hvoreczký 1990, 54). Matematiikasta pelinkehitykseen läheisesti liittyviä aiheita ovat logiikka, geometria ja algebra. Logiikka näkyy peleissä erilaisina ehto- ja valintarakenteina, algebra operaatioina ja laskutoimituksina sekä geometria liikkuvina objekteina. Tietokonepelien ohjelmoimisen näkökulmasta näitä matematiikan osa-alueita hyödynnetään jatkuvasti, ja mitä syvällisemmin oppilas tuntee näitä matematiikan alueita, sitä paremmin hän pääsee itse ”jyvälle” pelissä esiintyvistä ohjelmointiongelmista.

Ohjelmoinnin ja fysiikan yhteys tulee sitä tärkeämmäksi, mitä enemmän siirrytään teoreettisen fysiikan alueelta fysiikan sovelluksiin. Fysiikalla ja tietotekniikalla on yhdessä suuri paino huipputeknologiassa, ja fyysikoille asetetaan jatkuvasti uusia vaatimuksia esimerkiksi mikroprosessoripohjaisessa laitesuunnittelussa, tietoliikenteessä, ohjelmoinnissa ja tietokoneiden käytön hallinnassa. Esimerkiksi Oulun yliopistossa fysiikan opiskelijoiden on mahdollista valita aivan erityinen tietotekniikan suuntautumisvaihtoehto, missä syventymiskohteeksi voi valita esimerkiksi ohjelmistotekniikan tai laskennallisen fysiikan (Oulun yliopisto 2010).

Vaikka matematiikka auttaa ymmärtämään ohjelmointia (Hvoreczký 1990, 53), asia toimii myös käänteisesti: Ohjelmointi auttaa ymmärtämään matematiikkaa, kuten myös muita tieteenaloja. Näillä kaikilla on yhteys toisiinsa, ja ohjelmoinnin aloittelijoiden tulisi olla tietoinen näistä kytköksistä niin pian kuin mahdollista (Hvoreczký 1990, 53). Tavoitteena tulisi olla, että oppilas ymmärtää matematiikan olevan käyttökelpoinen työkalu niin arkipäivän kuin tekniikan sovelluksissa, eikä suinkaan vain kokoelma toisistaan irrallisia kaavoja ja sääntöjä.

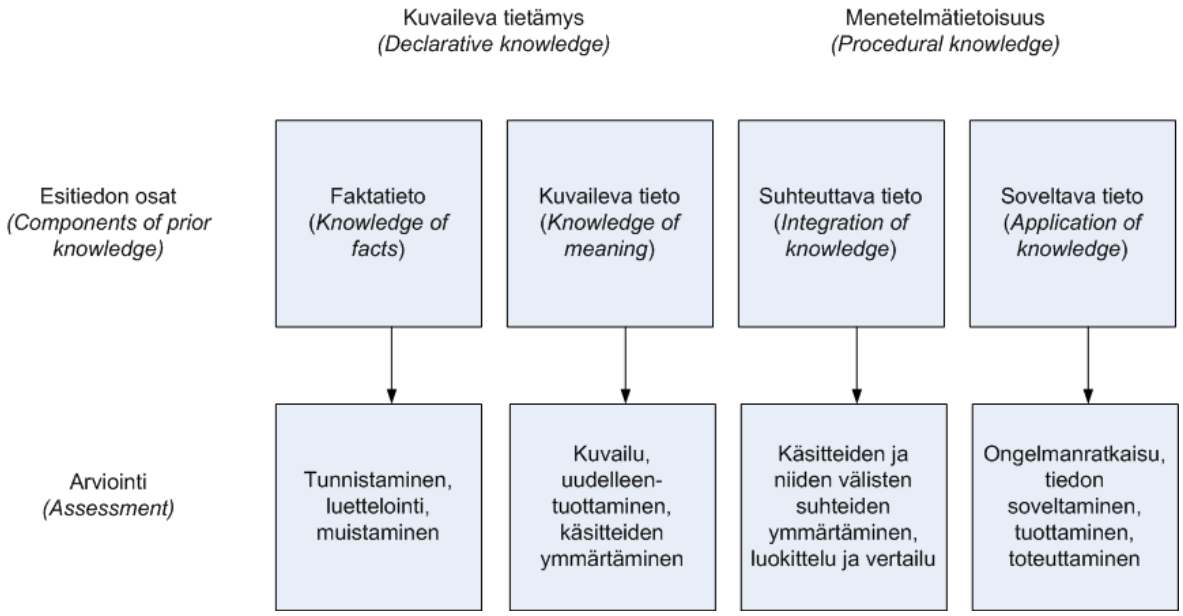
3.3 Ennakkotieto auttaa yliopisto-opinnoissa

Tutkimuksen kohteena oleva kurssi oli viikon (5 päivää) mittainen, eikä luonnollisesti voi tarjota aivan alkeiden ja joidenkin yksittäisten faktojen lisäksi kovinkaan syvällistä tietämystä ohjelmistotekniikan koko tietämysalueesta, eikä se suinkaan ollut kurssin tavoite. Ennemminkin tavoitteena oli antaa lisätietoa ohjelmoinnin mahdollisuuksista, ja tuoda se askeleen verran lähemmäs tämän päivän nuoren arkitodellisuutta pelien muodossa – ja erityisesti *innostaa* kokeilemaan ja tutkimaan itse lisää, ja hankkimaan sellaista

tietämystä, jonka avulla oppilas voi omassa tietämyksessään edetä sellaiselle tietämyksen tasolle, josta on hyötyä myöhemmissä opinnoissa.

Hailikarin tutkimuksessa (2009) tutkittiin lääketieteen opiskelijoita, ja heidän matematiikan, lääketieteen ja kemian oppiaineiden ennakkotietojen vaikutusta suhteessa menestymiseen yliopisto-opinnoissa. Ennakkotiedosta kehitettiin malli, jonka toimivuutta testattiin kolmella eri luonnontieteellisellä alalla. Hailikarin mukaan (mts. 53–54) yliopisto-opiskelijoiden ennakkotieto ja sen laatu ennustavat hyvin opintomenestystä. Ennakkotiedon taso on myös yhteydessä kurssin keskeyttämiseen ja suorittamisnopeuteen. Opiskelijat, jotka pystyivät ratkomaan suhteuttavan tai soveltavan tiedon (ks. kuva 5) tasoisia tehtäviä jo kurssin alussa, menestyivät paremmin kurssilla. Sen sijaan faktatiedolla tai kuvailevalla tiedolla ei ollut merkitystä, kuten ei myöskään sillä, millaista palautetta omasta suorituksesta opiskelija sai. Opiskelijan pystyvyysuskomukset (eli kuinka hyvin opiskelija itse uskoi pärjäävänsä kurssilla) tai aiempi opintomenestys ennustivat myös jossain määrin tulevaa opintomenestystä, mutta suurin merkitys oli (oikealla) ennakkotiedolla. Sama päti myös yksittäisten kurssien osalta. Ne opiskelijat, joilla oli syvällistä ennakkotietoa ja osaamista jo kurssin alussa, menestyivät paremmin ja saivat korkeampia arvosanoja kuin ne, joiden ennakkotiedon taso oli heikompi. Heikot ennakkotiedot johtivat todennäköisemmin kurssin keskeytykseen tai kurssin hitaampaan suorittamiseen uusintakokeiden avulla. Heikoin ennakkotiedoin opiskelevat myös arvioivat oman pystyvyytensä alhaisemmaksi kuin ne opiskelijat, joilla oli hyvä ennakkotiedon taso. Toisin sanoen pystyvyyskokemusten ja ennakkotiedon välillä oli vahva yhteys. (Hailikari 2009, 50–51.)

Kuvassa 5 on esitetty Hailikarin (mts. 28) ennakkotiedon malli, missä hän on erotellut ennakkotiedon eri tasot toisistaan.



Kuva 5. Esitiedon osat (tasot) jaetaan neljään kategoriaan: faktatietoon, kuvailevaan, suhteuttavaan ja soveltavaan tietoon (Hailikari 2009, 28).

Ennakkotiedon merkitys on siis suuri. Tietotekniikan opetusta tämän päivän perusopetuksessa käsitellään myöhemmissä luvuissa, mutta mainittakoon tässä vaiheessa, että kaikissa kouluissa ei ole olemassa pakollisia kursseja (Isomäki 2010), ja siten ensimmäinen vaihtoehto on, että tietotekniikan opiskelu jää oppiaineiden integroinnin varaan. Tällöin esimerkiksi ohjelmoinnin opettaminen on lähinnä opettajien oman mielenkiinnon ja harrastuneisuuden varassa. On muistettava, että tietotekniikka on suhteellisen nuori tieteenala, ja vielä nuorempi, mitä tulee tietotekniikan hyödyntämiseen opetuksessa. Kansainvälisen mittapuun mukaan suomalaiset opettajat eivät osaa tai halua hyödyntää viestintätekniikkaa osana opetusta – puhumattakaan, että alkaisivat opettaa tekniikan, sosiaalisen median tai verkkoyhteisöjen *avulla* (Kallionpää 2010). Opettajien kiinnostus tietotekniikan tekniikan ja pedagogisen käytön hallintaan, tai tietotekniikkaan liittyvän täydennyskoulutukseen osallistumiseen ei ole järin suurta, ja itse asiassa suuri osa rehtoreista on sitä mieltä, ”ettei tieto- ja viestintätekniikalla ja oppimistuloksilla ole mitään tekemistä keskenään” (Kankaanranta & Puhakka 2006, 73; Kallionpää 2010).

Yksi luontevimmista yhteyksistä integroida tietotekniikka muihin aineisiin on matematiikka. Matematiikan ja tietotekniikan integrointi on erittäin tärkeää, kun otetaan

huomioon, että Wilsonin ja Schrockin mukaan (2001, 186–187) matemaattisilla taidoilla on vieläpä merkitystä ohjelmoinnin alkeiskurssilla menestymiseen. On kuitenkin huomattava, että Suomessa matematiikan ja luonnontieteiden opettajat hyödyntävät tietotekniikkaa vähemmän kuin esimerkiksi äidinkielen ja vieraiden kielten tai yhteiskunnallisten aineiden opettajat (Kankaanranta & Puhakka 2006, 50). Eniten käytetty tietotekniikan hyödyntämistapa perusopetuksessa on tekstinkäsittely sähköpostin ohella.

Toinen, edellä mainitun nojalla todennäköisempi vaihtoehto on, että tietotekniikan opiskelu jää kokonaan tai lähes kokonaan oman harrastuneisuuden varaan. Tästä johtuen yliopistoon tullessaan opiskelijoiden tietoteknisissä ennakkotiedoissa voi olla huomattavan suuria eroja. Olisikin erittäin tärkeää, että tietotekniikan pintaa, niin käytännön alueella kuin sen luonnontieteellistä perustaa, raapaistaisiin jo perusopetuksessa. Näin erot ennakkotiedoissa esimerkiksi juuri tietotekniikan alan opiskelujen aloittamisen yhteydessä eivät olisi niin suuret.

Oppilaiden ennakkotietojen kartuttamiseen on monia tapoja. Seuraavassa luvussa esitellään kaksi hyväksi havaittua IT-alan varhaisen rekrytoinnin muotoa, joiden on todettu antavan lapsille ja nuorille sellaista ennakkotietoa, joilla on merkitystä myöhemmin opiskelupaikan valinnassa ja ammatinvalinnassa: pelit ja leirit.

3.4 Pelit ja leirit varhaisen rekrytoinnin muotona IT-alalla

Aiemmassa luvussa puhuttiin muun muassa IT-alan opiskelijamäärien muuttumisesta 2000-luvulla: ensin nousua, sitten pitkä lasku. Se, että nuoret eivät ole enää niin kiinnostuneita alasta, voi johtua monestakin syystä. Blumin ja Cortinan (2007, 20) mukaan syitä voivat olla muun muassa:

- se, että uutisissa esiintyy paljon IT-yritysten ulkoistamisilmoituksia
- alan kilpailuhenkisyys
- oppilaiden tietämättömyys alasta
- alan opiskelu tai työnteko alalla ei ole tarpeeksi kivaa
- ala on miesvaltainen
- opettajien kiinnostuksen puute alaa kohtaan.

Kysely tehtiin Yhdysvalloissa Carnegie Mellon -yliopistossa nimenomaan tietotekniikan opettajille. Viimeinen kohta on tietenkin äärimmäisen huolestuttava, mikäli se pitää paikkansa yleisemmin, ja vaatisi tarkan katsauksen nykyiseen aineenopettajien koulutukseen ja opetussuunnitelmiin. Arveluttava seikka on myös oppilaiden tietämättömyys alasta. Ensimmäisenä herää kysymys, miten voi olla mahdollista, että oppilaat eivät tiedä IT-alan opiskelu- ja työskentelymahdollisuuksista, vaikka nimenomaan IT-alalla pitäisi olla kykyä käyttää ja hyödyntää sähköistä mediaa kaikkine mahdollisuuksineen. On todettu, että yksittäisillä opettajilla voi olla jopa omia vanhempia merkittävämpi vaikutus jatko-opiskelupaikan valinnassa (Blum & Cortina 2007, 20), mutta opettajilla ei useinkaan ole kovin syvällistä kiinnostusta panostaa oppilaiden jatkosuunnitelmien ohjaustyöhön, vaan oppilaanohjaus sysätään kaikkinensa opinto-ohjaajan vastuulle (Nykänen 2010). Näihin kysymyksiin tuskin on yhtä oikeaa tai väärää vastausta, mutta oikean tiedon välittäminen nuorille riittävän aikaisin on yksi seikka, johon tässä pro gradu -työssä paneudutaan, ja ehdotetaan yhtä välinettä ratkaisuun nuorille suunnatun pelikeskeisen alkeisohjelmointikurssin muodossa.

Pro gradu -tutkielmassaan Siljander (2009, 20–24) käy läpi toimiviksi havaittuja varhaisia rekrytointitapoja ja -muotoja. Sanalla *varhainen* viitataan tässä nimenomaan lapsiin ja nuoriin, ja *rekrytoinnilla* siihen, kuinka heidän kiinnostusta IT-alaa kohtaan voitaisiin herättää ja tukea siten, että he päätyisivät myöhemmin alan opiskelijoiksi ja työntekijöiksi. Tutkijat ovat melko vakuuttuneita siitä, että *leirit* ja *kerhot* ovat hyviä keinoja tutustuttaa lapsia ja nuoria IT-alaan, ja tuoda alaa lähemmäksi heitä. Tällä tavalla voidaan madaltaa kynnystä omatoimiseen aiheeseen tutustumiseen, sekä kannustaa hakeutumaan alan kursseille ja myöhemmin alan opiskelijoiksi. Huomionarvoista on, ettei leirien ja kerhojen tarkoitus ole niinkään vaikuttaa heti tuloksia synnyttävästi, vaan herättää mielenkiinto sekä jättää positiivinen kokemus ja mielikuva aiheesta. Turnerin, Berntin & Pecoran (2002, 12) tutkimuksessa kuvattiin tilanne, missä naispuolinen henkilö oli lapsena käynyt IBM:n järjestämällä kesäleirillä (siis *yhdellä* leirillä), ja myöhemmin, opiskeltuaan ja oltuaan ensin töissä muulla alalla, muisti mukavan leirikokemuksen, ja hakeutui IT-alalle opiskelemaan.

Pelien tapauksessa asia ei välttämättä ole niin yksiselitteinen. Esimerkiksi viihdepelien pelaamisella on todettu olevan vaikutusta lasten kognitiivisten taitojen monipuoliseen kehittymiseen: pelaaminen kehittää mm. lapsen visuospatiaalisia kykyjä (kyky hahmottaa itsensä ja ympäristönsä välisiä etäisyyksiä ja ympäristön muotoja), tehostaa tarkkaavaisuuden jakamista usean kohteen kesken, kehittää informaation prosessoinnin taitoja ja strategista ajattelua ja niin edelleen (Salokoski & Mustonen 2007, 34). Toisaalta vaarana on monien pelien seksualisoituminen ja siihen liittyvät vaarat, väkivallan korostuminen, virtuaalimaailmaan uppoutuminen sekä kaikenlaiset riippuvuudet ja sen myötä sosiaalisen elämänpiirin suppeneminen (mts. 61, 78, 96 ja 102). Kuitenkin Siljanderin (2009, 23–24) mukaan ne lapset, jotka pelaavat videopelejä, hankkivat enemmän kokemusta tietokoneen käytöstä ja osallistuvat enemmän lukion atk-opetukseen kuin lapset, jotka eivät pelaa. Tällä tavalla he ensinnäkin kiinnostuvat lisää aiheesta, ja saavat tärkeää ennakkotietoa ajatellen tulevia jatko-opintoja. Van Eckin (2006) mukaan tietokonepelien pelaaminen voi muuttaa asenteita teknologiaa kohtaan, ja saada oppilaat näkemään IT-ala vähemmän vaikeana alana, ja siten pelaamisella voi olla vaikutusta myös ammatinvalintaan.

4 Ohjelmoinnin paradigmat

Tutkimuksen kohteena oli viiden päivän mittainen kurssi, jossa opeteltiin ohjelmoinnin alkeita, ja jokainen oppilas (iältään 12–16-vuotiaita) teki oman tietokonepelin käyttäen valittuja työkaluja, sekä Jyväskylän yliopiston tietotekniikan laitoksella erityisesti peliohjelmointia varten kehitettyä ohjelmointikirjastoa. Tässä luvussa esitellään lyhyesti ne ohjelmointiparadigmat, joita kurssilla (ja kehitetyssä ohjelmointikirjastossa) hyödynnettiin: niitä olivat deklaratiivinen sekä imperatiivinen paradigma, ja erityisesti rakenteinen ohjelmointi ja olio-ohjelmointi. Lisäksi tässä luvussa esitetään perustelut, miksi kyseiset paradigmat on valittu. Näiden paradigmojen soveltumisesta opetukseen puhutaan enemmän luvussa 5.

Paradigmojen taustoista esitellään sellainen keskeinen tieto, joka tutkimuksen kannalta on aiheellista esitellä. Tässä käsitellyt paradigmat ovat vain pieni murto-osa – joskin hyvin yleisesti käytössä olevia – kaikista olemassa olevista.

4.1 Mikä on ohjelmointiparadigma

Ohjelmointiparadigma on perustavanlaatuinen tapa tai malli, jolla tietokoneohjelmia ohjelmoidaan (Van Roy 2008). Se on filosofia ja lähestymistapa mieltää, jäsentää ja esittää ohjelmointiongelma tietyn dogmin mukaan. Paradigma määrittelee konseptitasolla muun muassa ohjelman elementtien (kuten funktiot, muuttujat, vakiot ja niin edelleen) abstraktiot, sekä ne vaiheet jotka ohjaavat laskentaa (jatkuvuus, vuon ohjaus, ja niin edelleen). Huomaa, että ohjelmointiparadigmalla ei tarkoiteta tapaa tiettyyn ohjelmistotuotannon ongelman ratkaisuun. Kullakin ohjelmointiongelmalla voi olla kussakin paradigmassa toteutettavissa oleva ratkaisu, tai sitten ei ole.

Ohjelmointiparadigmat voidaan jakaa päähaaroihin. Brookshear (2003) jakaa ohjelmoinnin neljään kategoriaan: (i) imperatiivinen, (ii) deklaratiivinen, (iii) funktionaalinen ja (iv) oliperustainen, Koskimies (1997) taas viiteen (i) imperatiivinen, (ii) funktionaalinen, (iii) logiikka- ja (iv) rajoite sekä (v) olioperustainen. Koska olioperustainen ohjelmointi voidaan katsoa olevan osa imperatiivista ohjelmointia (Van Roy 2008), funktionaalinen sekä logiikka- ja rajoiteohjelmointi osa deklaratiivista ohjelmointia, voidaan sanoa, että

ohjelmoinnin kaksi tärkeintä päähaaraa (Van Roy 2008) ovat *deklaratiivinen ohjelmointiparadigma*, sekä sen ”vastakohta”, *imperatiivinen ohjelmointiparadigma*. Seuraavassa esitellään lyhyesti nämä paradigmat. On vielä syytä huomata, että tämä jako on yksi esimerkki tavasta jakaa paradigmat, mutta joillain muilla perusteilla toisenlainen jako olisi varmasti myös mielekäs.

Edellä mainitut ohjelmointiparadigmat ovat sateenvarjotermejä ja niiden alle kuuluu useita paradigmoja, joista tässä käsitellään vain tämän tutkimuksen kannalta olennaiset. Edelleen, rajat eivät ole absoluuttisia, sillä monet ohjelmointikielet ovat omaksuneet ohjelmointityylejä usean eri ohjelmointiparadigman alueilta (Van Roy 2008). Esimerkiksi *Oz*-kielen kehittämisen tavoitteena on nimenomaan ollut yhdistellä eri paradigmoja siten, että ohjelmoijalla on mahdollisuus ottaa käyttöön se paradigma, joka käsiteltävänä olevaan ohjelmointiongelmaan parhaiten sopii (Van Roy ym. 2003, 22–23).

4.2 Miksi ohjelmointiparadigman valinta on tärkeää

Ohjelmointi on laaja tieteenala ja käytännölliset ohjelmointikielet ovat yleensä melko kompleksisia. Ohjelmointiongelmien ratkaiseminen vaatii oikeiden lähestymistapojen valintaa. Jos jätetään pois laskuista kaikkein yksinkertaisimmat ohjelmat (kuten *Hello World*), voidaan sanoa, että *jokainen* ohjelmointiongelma on yksilöllinen ja lähestymistavan valinnalla voi olla ratkaiseva merkitys – kaikki lähestymistavat eivät ole parhaita mahdollisia yksittäisen ongelman kohdalla (Van Roy 2009, 10). Ongelma voi toki olla sellainen, että se itsessään sisältää eri lähestymistapoja vaativia osioita. Erilaisiin ohjelmointiongelmiin on olemassa valmiiksi määriteltyjä lähestymistapoja, ja eri ohjelmointikielet tukevat kielestä riippuen joko yhtä tai useampaa lähestymistapaa (Van Roy 2009, 10). Näitä eri lähestymistapoja kutsutaan *paradigmoiksi*.

Tässä työssä tutkittavana olevan kurssin ohjelmointikieleksi valittiin C# (työkalujen valinnasta kerrotaan lisää luvussa 6). Kyseinen ohjelmointikieli tukee niin deklaratiivista kuin imperatiivista ohjelmointia, mitkä ovat paradigmojen kaksi päähaaraa, ja on ns. oliokieli (Schildt 2005, 11). Rakenteinen ohjelmointi on imperatiiviseen ohjelmointiin sisältyvä paradigma (Van Roy 2008), ja hyvin suosittu ja käytännönläheinen ohjelmointiongelmien lähestymistapa erityisesti alkeisohjelmoinnissa (tästä lisää luvussa

5). Imperatiiviseen ohjelmointiin sisältyvä olio-ohjelmoinnin paradigma laajentaa rakenteista ohjelmointia (Schildt 2005, 11) ja on tämän hetken ohjelmistotuotannon ja ohjelmistotekniikan tutkimuksen yksi eniten toistetusta avainsanoista (Koskimies 1997, 1) ja siten luonnollinen valinta myös alkeisohjelmoinnin kurssille – joskin pienemmässä roolissa korkean abstraktiotasonsa vuoksi. Deklaratiivinen paradigma oli tutkittavalla kurssilla käytännössä melko näkymättömässä roolissa. Koska kirjoitushetkellä C#:n uusin versio 3.0 tukee tätä paradigmaa (Horton 2010), ja koska ohjelmointikirjaston kirjoittamisessa sillä oli oma paikkansa, se esitellään myös tässä.

4.3 Deklaratiivinen ohjelmointiparadigma

Ohjelmointiparadigmaa, joka ilmaisee ohjelman *logiikan*, mutta ei sen vuonohjausta, kutsutaan deklaratiiviseksi ohjelmointiparadigmaksi (myöhemmin DOP) (Lloyd 1994). Toisin sanoen, ohjelmointikielten näkökulmasta DOP on sitä, että kuvataan *mitä* ohjelman pitäisi tehdä tai saavuttaa sen sijaan, että kuvattaisiin *miten* ohjelman suoritus etenee tavoitteeseen pääsemiseksi.

Brookshearin mukaan (2003, 230–232) deklaratiivisessa ohjelmoinnissa suurin vaikeus on ratkaisualgoritmin löytäminen haluttuun ohjelmointiongelmaan. Tästä johtuen varhaiset deklaratiiviset kielet olivat tiettyyn tarkoitukseen kehitettyjä, ja tarkoitettu sovellettaviksi vain tietyillä alueilla. DOP:a on käytetty esimerkiksi simuloinnissa (taloudelliset, fysikaaliset, poliittiset jne. systeemit) hypoteesien testaamiseen. Näissä yhteyksissä algoritmi on aina ajan kulumisen simulointi laskemalla uudestaan ja taas uudestaan parametrien arvot edellisellä kerralla lasketuista. Tällaisten simulaatioiden deklaratiivinen kieli edellyttää algoritmia, joka laskee toistuvan proseduurin aina uudestaan.

DOP:n mukaisia lähestymistapoja ovat esimerkiksi funktionaalinen ohjelmointi (esimerkkinä taulukkolaskentaohjelmat), rajoiteohjelmointi ja logiikkaohjelmointi (joka on itse asiassa rajoiteohjelmoinnin osajoukko) (Van Roy 2008).

4.4 Imperatiivinen ohjelmointiparadigma

Imperatiivista ohjelmointiparadigmaa (myöhemmin IOP) voidaan sanoa DOP:n vastakohdaksi (Glaser ym. 1984; Van Roy 2008). Käyttäen ohjelmointikielen komentoja se

kuvaa annetun ohjelmointiongelman ratkaisun komentojonona yksiselitteisesti vaihe vaiheelta riittävällä tarkkuudella. Komennot suoritetaan yksi kerrallaan järjestyksessä ensimmäisestä viimeiseen. IOP:aan kuuluvat osana myös vuonohjauslauseet, jotka voivat siirtää ohjelman suorituksen jatkumaan jostain muusta kohdasta, kuin järjestyksessä seuraavasta komennosta. Komennot muuttavat suorituksen tilaa muokkaamalla laitteiston muistiin tallennettuja arvoja, eli muuttujia.

Yksi kuvaava esimerkki IOP:n ja DOP:n eroista on seuraava kuvitteellinen ohjelmointiongelma: Olkoon meillä tehtävänä rakentaa talo. Edellä mainittuja ohjelmointiparadigmoja (paradigmakategorioita) soveltaen, käyttämällä sopivaa ohjelmointikieltä, voisimme kuvata ratkaisun ongelmaan seuraavasti (esim. Glaser ym. 1984).

Deklaratiivinen ohjelmointiparadigma	Imperatiivinen ohjelmointiparadigma
Talo on rakennelma, jossa on 1. lattia, jota perustukset tukevat. Valetaan betonista. 2. seinät, joita perustukset tukevat. Pystytetään suorassa kulmassa lattiaan nähden. 3. katto, joka asetetaan seinien päälle. (ja niin edelleen…)	1. Rakenna lattia 1.1. Aseta putket ja johdot paikoilleen 1.2. Vala betoni 1.3. Anna betonin kuivua 2. Rakenna seinät 2.1. Aseta seinä suoraan kulmaan lattiaan nähden. 2.2. Laita seinän sisään villaa.

Taulukko 1. Esimerkki deklaratiivisen ja imperatiivisen ohjelmointiparadigmojen eroista (Glaser ym. 1984).

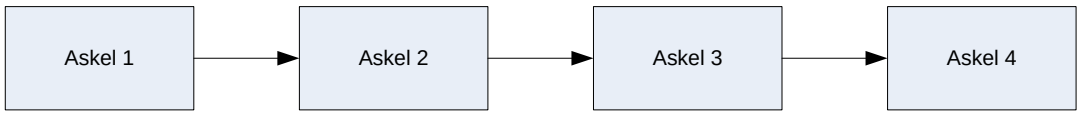
4.5 Rakenteinen ohjelmointi (imperatiivinen ohjelmointiparadigma)

Rakenteinen ohjelmointi (*structured programming*) voidaan nähdä imperatiivisen ohjelmointiparadigman osajoukkona (Van Roy 2008). Rakenteiset ohjelmistot ovat ennalta suunniteltuja, modulaarisia, niissä on sisäinen hierarkia ja niiden luettavuus on hyvä (Dijkstra 1976).

Rakenteisen ohjelmoinnin yksi tärkeimmistä ominaisuuksista verrattuna ei-rakenteiseen ohjelmointiin (joka on myös osa imperatiivista ohjelmointia) on, että ohjelmat nojautuvat olennaisesti vähemmän nk. *Go to* -lauseeseen. Ei-rakenteista ohjelmointia on kritisoitu mm. siitä, että *Go to* -lause tuottaa hyvin vaikeasti luettavaa koodia, ”spagettikoodia”, eikä sen ole katsottu olevan sopiva ohjelmointitapa sellaisissa ohjelmistoprojekteissa, joissa käytetään korkean tason ohjelmointikieltä (Cobern 1986, 9-10; Dijkstra 1968).

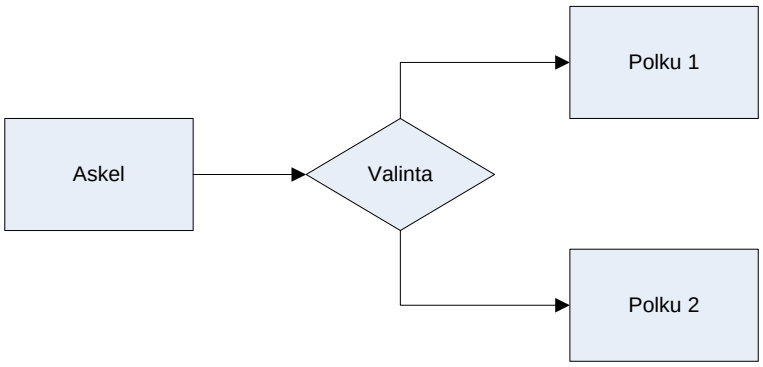
Matalan tason rakenteinen ohjelma kirjoitetaan käyttäen yksinkertaisia, hierarkkisia vuonohjausrakenteita. Näitä rakenteita ovat *peräkkäisyys* (järjestys, sekvenssi) *valinta* ja *toisto* (Dahl ym. 1972).

- *Peräkkäisyydellä* (kuva 6) tarkoitetaan sitä, että komennot suoritetaan yksi kerrallaan järjestyksessä ensimmäisestä viimeiseen (periytyy IOP:stä).



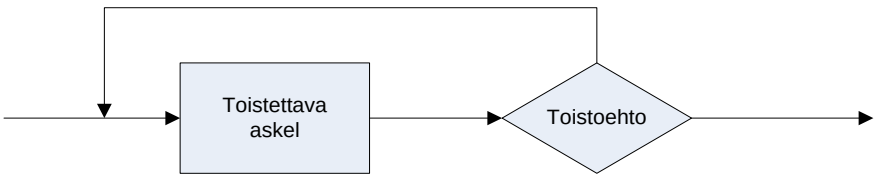
Kuva 6. Komennot suoritetaan peräkkäin yksikäsitteisessä järjestyksessä (Dahl ym. 1972, 18).

- *Valinta* (kuva 7) on tilanne, missä yksi useista valittavissa olevista lauseista suoritetaan sen perusteella, mikä on ohjelman sen hetkinen tila. Valintaa ilmaisevat ohjelmointikielissä yleensä lauseet `if`, `then`, `else`, `endif` sekä `switch` ja `case`.



Kuva 7. Syötteestä riippuen valintatilanteesta jatketaan joko polulle 1 tai polulle 2 (Dahl ym. 1972, 18).

- *Toistossa* (kuva 8) lause suoritetaan, kunnes ohjelma saavuttaa jonkin ennalta määritellyn tilan, tai kun toisto on suoritettu kaikkiin valitun kokoelman elementteihin. Toistoa ilmaisevat ohjelmointikielissä tyypillisesti lauseet `while`, `repeat`, `for` (`each`) sekä `do` ja `until`.



Kuva 8. Toistettavaa lausetta suoritetaan niin kauan, kunnes toistoehto ei ole enää voimassa, ja ohjelman suoritusta jatketaan (Dahl ym. 1972, 19).

Rakenteista ohjelmointia voi toteuttaa melkein millä tahansa ohjelmointikielellä, mutta eräitä tunnettuja rakenteisen ohjelmointiparadigman kieliä ovat ALGOL, Pascal, PL/I ja Ada (Van Roy 2008). Myös kurssilla käytettäväksi valittu C# tukee rakenteista ohjelmointia.

4.6 Olio-ohjelmointi (imperatiivinen ohjelmointiparadigma)

Olio-ohjelmointi on ohjelmoinnin lähestymistapa, jossa ohjelmointiongelmien ratkaisut jäsennetään useiden *olioiden* yhteistoimintana (Khor 1995). Koska ohjelmointiongelmat ovat usein laajoja ja käsittävät monia eri osa-alueita, on olio-ohjelmointi yksi keino ongelman paloitteluun ja jakamiseen pienempiin, helpommin hallittaviin osiin (Garrido 2003, 39–40). Näitä palasia kutsutaan siis olioiksi. Seuraavassa määritellään lyhyesti olio-ohjelmoinnin keskeisimmät käsitteet (mukaillen Lahdelma & Lappalainen 1999, 5–9).

- **Oliolla** (*object*) tarkoitetaan joko todellista tai käsittellistä kokonaisuutta, jolla on identiteetti. Olio on ohjelmiston perusyksikkö, joka sisältää joukon loogisesti yhteenkuuluvaa tietoa ja toiminnallisuutta. Olio voi esittää mitä tahansa reaalimaailman asiaa, kuten pöytää, autoa tai moottoria. Toisaalta olio voi olla jokin abstrakti kokonaisuus, kuten avioliitto, lista tai liike. Oliot voivat esimerkiksi kommunikoida keskenään lähettämällä toisilleen viestejä, niillä voi olla ominaisuuksia ja toimintoja.

- **Luokka** (*class*, usein myös *olioluokka*) määrittelee tietyn oliojoukon yhteiset piirteet. Se on määritys, josta voidaan luoda uusia olioita. Olio on luokan ilmentymä, instanssi (*instance*). Luokka on ”piparkakkumuotti”, jolla voidaan tehdä monia samanmuotoisia pipareita – kuitenkin jokainen yksittäinen pipari on oma yksilönsä ja jokaisella piparilla on yksilöllisiä piirteitä. Luokassa määritellään ominaisuudet ja toiminnot, jotka jokaisella siitä luodulla oliolla ovat.

- **Metodi** (*method*), jäsenfunktio (*member function*) tai operaatio (*operation*) tarkoittavat luokassa määriteltyä aliohjelmaa, joka käsittelee olion tietoa.

- **Attribuutti** (*attribute*), jäsenmuuttuja (*member variable*) tai kenttä (*field*) tarkoittavat luokassa oliolle määriteltyä muuttujaa, joka tallentaa jonkun olioon liittyvän tietoalkion.

- **Rakentaja** (*constructor*) on metodi, jota kutsutaan oliota luotaessa. Rakentaja alustaa olion jäsenmuuttujien alkuarvot.

- **Hajottaja** tai **purkaja** (*destructor*) on metodi, jota kutsutaan olion tuhoamiseksi.

- **Perintä** (*inheritance*) mahdollistaa uusien luokkamääritysten tekemisen aikaisemmin tehtyjen määritysten pohjalta. Perinnässä aiemmin tehtyjä luokkia käytetään uusien luokkien pohjana. Periyttäminen mahdollistaa myös sellaisten luokkien käytön, joista periytetään useita luokkia, joita kaikkia voidaan käyttää samannimisillä funktioilla ja samoilla parametreilla kuin perusluokkaa, josta nämä ominaisuudet perittiin. Tällaisia luokkia sanotaan abstrakteiksi luokiksi. Periyttäminen ja abstraktien perusluokkien käyttö ovat oliopohjaisten kielien vahvimpia mutta myös vaikeimpia ominaisuuksia.

- **Moniperintä** tai (*multiple inheritance*) viittaa joidenkin oliokielien ominaisuuteen, jossa luokka voi periä useamman kuin yhden luokan ominaisuudet ja toiminnot. Normaalissa perinnässä luokka voi periä vain yhden luokan ominaisuudet ja toiminnot. Eräs moniperintää tukeva kieli on C++. Esimerkiksi Javassa ja C#:ssa moniperintä on mahdollista vain tietyissä erikoistapauksissa.

- **Kapselointi** (*encapsulation*) -termiä käytetään kahdessa merkityksessä. Ensimmäisen merkityksen mukaan kapseloinnin ideana on datan ja käyttäytymisen kokoaminen yhteen yksikköön, olioon. Toinen merkitys lisää tähän vielä tiedonpiilotuksen: olion sisäisiin muuttujiin ei (pääsääntöisesti) päästä suoraan käsiksi olion ulkopuolelta, jolloin ohjelmointivirheiden määrä vähenee. Olion käyttäjän ei tarvitse tietää enää käyttövaiheessa sitä, miten olio toimii sisäisesti: riittää kun tietää, miten oliota itseään käytetään ja miten olio käyttäytyy. Olion käyttäjät voivat käyttää oliota sen julkisen rajapinnan kautta (*public interface*). Olion luokkakuvauksessa voi olla myös yksityisiä muuttujia tai funktiota (*private interface*), jotka ovat vain olion itsensä käytettävissä.

Tässä luvussa esiteltiin kurssilla käytettyjä ohjelmointiparadigmoja. Seuraavassa luvussa (alaluvusta 5.4 eteenpäin) käsitellään kurssille valittuja opetusmenetelmiä, ja mitä niissä oli otettava huomioon paradigmojen valinnan jälkeen ja miten valitut paradigmat soveltuvat opetukseen.

5 Ohjelmoinnin opettaminen

Tässä luvussa esitellään ensin mitä ohjelmointiin liittyvistä oppimiskäsityksistä ja pedagogisista menetelmistä tiedetään. Luvussa pohditaan myös ensiohjelmointiin liittyvää problematiikkaa, sekä erilaisia ensiohjelmointikurssin lähestymistapoja. Näistä lähestymistavoista valitaan yksi ja valinta perustellaan. Lisäksi tässä luvussa käydään läpi perusopetuksen opetussuunnitelman perusteisiin kirjoitettuja sisältöjä ohjelmoinnin ja yleisesti tieto- ja viestintätekniikan näkökulmasta, ja pohditaan tietotekniikan integraation tilaa perusopetuksessa. Edelleen luvussa otetaan kantaa mukavuustason merkitykseen alkeisohjelmoinnin kurssilla, sekä pohditaan erilaisia olemassa olevia pelinkehitysmenetelmiä.

5.1 Oppiminen etenee ”alhaalta ylös”

Elämme aikaa, jolloin tiedon sisällöt muuttuvat nopeasti. Ohjelmointikielet tulevat ja menevät. Esimerkiksi Jyväskylän yliopistossa uusille opiskelijoille ensimmäisenä tarjottavalla ohjelmointikurssilla (aikaisemmin ”Johdatus ohjelmointiin”, nykyään ”Ohjelmointi 1”) on opetettu monia eri ohjelmointikieliä: Fortran, Pascal, C, C++, Java, ja uusimpana tulokkaana C#. Koulutuksen on tarjottava jotain muuta kuin opettajalta oppilaalle tapahtuvaa faktojen siirtämistä – faktojen päivittäminen käy paljon näppärämmin tietokoneen tai kirjojen avulla. Opettajan tehtävä sen sijaan on edistää erilaisten toimintavalmiuksien osaamisen oppimista (Rauste-Von Wright 1997, 8), kuten sopeutumista muutokseen ja oppimaan oppimisen taitoja. Nämä ovat jatkuvasti muuttuvassa yhteiskunnassamme elämänhallinnan välttämättömiä perusedellytyksiä.

Vaikka Suomessa keskustellaan aktiivisesti koulutuksesta, on keskustelu oppimisesta ja sen ehdoista vähäistä. Kuitenkin meistä jokaisella on jonkinlainen käsitys siitä, mitä oppiminen tarkoittaa, ja mitä tapahtuu silloin kun opitaan jotakin. Tyypillistä arkitodellisuuden oppimiskäsityksiä koskevien tutkimusten tuloksille on ollut, että valtaosa ihmisistä pitää oppimista faktojen mieleen painamisena (Rauste-Von Wright 1997, 13). Hieman yllättävää on, että otettaessa lähtökohdaksi yliopiston opettajien käsitykset siitä, minkälaista osaamista eri arvosanojen opiskelun tulisi tuoda mukanaan (ja yliopistojen opetussuunnitelmat, kurssivaatimukset jne.), ei tilanne poikkea paljonkaan (mts. 13–14).

- **Perusopintotasolla** odotukset koskevat pääasiassa faktojen hallintaa,
- samoin **aineopintotasolla**, ja
- vasta **syventävien opintojen tasolla** oppilailta odotetaan tutkimuksen ja tutkimusmenetelmien ymmärtämistä ja hallintaa.

Opettajien käsitykset osaamisen eri tasoista kussakin opintojen vaiheessa vastaavat hyvin ns. *Bloomin taksonomiaa*, jota käytetään yleisesti opetusalalla kuvaamaan oppimisen eri tasoja, ja jäsentämään, millaiseen tiedon omaksumisen tasoon pyritään (Huitt 2009).

Tiedolliset tavoitteet jaetaan Bloomin taksonomiassa kuuteen eri tasoon (Huitt 2009):

1. mieleenpalauttaminen; kyky muistaa asioita siinä muodossa kuin ne on esitetty
2. ymmärtäminen; kyky ymmärtää ja tulkita oppimaansa
3. soveltaminen; kyky käyttää tietoa oikeassa tilanteessa
4. analysointi: kyky pilkkoa ongelma pienempiin osiin ja ymmärtää niiden suhteet
5. syntetisointi; kyky luoda jotain uutta olemassa olevan tiedon pohjalta
6. arviointi; kyky arvioida ajatusten ja ratkaisujen arvoa; sisältää kaikki edellä listatut tasot sekä arviointikriteerit

Yliopiston odotuksissa kuvastuu siis odotusten kasvaminen ”faktoista kohti ajattelemista, osista kohti kokonaisuutta”, tiiliskivi tiiliskiven päälle, josta syntyy ensin seinä ja lopulta talo – jako alemman ja ylemmän tason osaamisen välillä. Oppiminen etenee ”alhaalta ylös”, yksittäisten reaktioiden oppimisesta kohti ajattelua, ja tämä on merkittävä erottelija oppimisen tutkimuksen paradigmojen välillä.

Aina oppiminen ei kuitenkaan tapahdu siten, että se etenisi esimerkiksi Bloomin taksonomian mukaisesti vaihe vaiheelta eteenpäin alkaen vaiheesta yksi ja päätyen kenties kuuteen. Analysoikaamme hetki jonkin yksinkertaisen taidon, vaikkapa lukemisen tai laskemisen, oppimista. On helppo todeta, että ne eivät tyypillisesti etene siten, että yksinkertaisten taitokomponenttien harjoittelusta edetään vähitellen kohti korkeamman tason prosesseja ja taitoja. Lukemaan opetellessaan lapsi joutuu ratkaisemaan ongelmia, tekemään päätelmiä ja käyttää hyväkseen sellaista tekstiin sisältymätöntä informaatiota *jo* pyrkiessään ymmärtämään yksinkertaisiakin tekstejä (Rauste-Von Wright 1997, 14).

Toisin sanoen ajattelu ei ole seurausta siitä, että johdanto-opinnot ovat edeltäneet syventäviä opintoja, vaan se on *olennainen tekijä* taitojen ja tietojen oppimisessa. Tämä pätee myös ohjelmoinnissa, ja myös tässä työssä tutkittavassa kurssissa. Vaikka ohjelmoinnin opetuksessa ohjelmointiparadigmaksi valittaisiinkin olio-ohjelmointi, ei opetuksen tarvitse lähteä liikkeelle olioista, vaikka se lopulta onkin kyseisen paradigman keskiössä. Tästä lisää luvuissa 5.4–5.6.

Tutkijat ja opettajat ovat vuosien saatossa kehittäneet lukuisia erilaisia ja muunneltuja oppimiskäsityksiä sekä pedagogisia lähestymistapoja opetuksessa. Seuraavissa alaluvuissa on kuvattu lyhyesti tämän tutkimuksen kannalta keskeinen teoriaperusta oppimisesta, ja lisäksi erilaisia pedagogisia lähestymistapoja.

5.2 Konstruktivismi

Oppimiskäsitykset voidaan jakaa karkeasti empiristiseen ja konstruktivistiseen (Tynjälä 1999, 28–29). Konstruktivistisen oppimisen juuret pohjautuvat konstruktivistiseen tietoteoriaan, missä yksittäiset oppijat konstruoivat mielikuvamalleja (*mental models*) ymmärtääkseen ympäröivää todellisuutta (Tynjälä 1999, 9–10). Konstruktivistisen teorian mukaan oppija käyttää aikaisempia tietojaan ja kokemuksiaan uuden tiedon ja uusien käsitysten muodostamisessa oppimisprosessin aikana (Tynjälä 1999, 9–10). Oppija muokkaa tietoa aktiivisesti ja omaehtoisesti, toisin kuin behavioristisessa (empiirisessä) oppimiskäsityksessä, missä ihminen nähtiin ympäristönsä ärsykkeisiin passiivisesti reagoivana olentona, jota saattoi opettaa manipuloimalla ympäristöä ja kutsumalla toistuvasti esiin ”oikeita” (haluttuja) reaktioita (Tynjälä 1999, 29; Ahonen 1989, 8). Konstruktivistinen oppimiskäsitys sen sijaan johtaa joustavan ja oppijan valmiuksia painottavan opetuksen korostamiseen (Rauste-Von Wright ym. 2003, 162). Oppimistilanteessa tulisi siis olla joustavuutta työtavoissa, tilaa kysymyksille, tiedon kritiikille, tiedollisille ristiriidoille ja yhdessä oppimiselle. Konstruktivistista tutkimusta harjoitetaan kansainvälisissä oppimisen tutkimuksen huippuyksiköissä ja se on tällä hetkellä tärkein ihmisen oppimiseen kohdistuva paradigma (Rauste-Von Wright 1997, 8).

Tietokoneohjelmien näkökulmasta näiden kahden oppimiskäsityksen ero on merkittävä. Behavioristisen käsityksen mukaisia tietokoneohjelmia ovat harjaannuttamisohjelmat eli

ns. drilliohjelmat, joissa ohjelma esittää kysymyksiä ja oppilas valitsee vastausvaihtoehdon (Ahonen 1989, 8). Ohjelma antaa sitten yksiselitteisen oikein/väärin-palautteen, tai ohjaa lisätehtävien pariin. Laaja-alaisempaa ja arvioivaa tietoa on vaikea käsitellä ”oikeiden” vastausten puitteissa. Tutkittavan kurssin kannalta tällainen ajatusmalli olisi kaikkea muuta kuin mielekäs: kaikkien pitäisi käytännössä päätyä sen saman pelin tekemiseen, jonka joku muu on katsonut ”oikeaksi”. Muut vaihtoehdot olisivat vääriä. Kuitenkin kantava ajatus kurssin suunnittelussa oli nimenomaan, että oppilas saa vapaasti ”luoda” ja ideoida, tehdä (tietyissä raameissa) sellaisen pelin kuin itse haluaa. Oppilaalle pyrittiin antamaan tilaa ja joustavuutta niin työtapojen kuin etenemistahdin suhteen. Tässä luonnollisesti auttoi se, että kurssilla oli useita ohjaajia, joiden avulla esimerkiksi eri tahtiin etenevät oppilaat voitiin hyvin ottaa huomioon ja esimerkiksi muita nopeammille (tai enemmän esitietoja omanneille) oppilaille voitiin tarjota jatkuvasti mielekästä tekemistä.

5.3 Pedagogisia menetelmiä

Seuraavissa alaluvuissa esitellään lyhyesti ne pedagogiset menetelmät, jotka valittiin käytettäväksi tutkimuksen kohteena olevalla opintojaksolla. Alla esitetään myös perusteluja sille, miksi juuri ne menetelmät on valittu.

5.3.1 Esittävä opetus

Esittävä opetus (tunnetaan myös nimellä ohjattu opetus) on yksipuolista tiedon jakamista kuuntelijajoukolle (Vuorinen 2001, 79). Esimerkiksi luento kuluu esittävään opetukseen, joka on usein käytetty opetustapa silloin, kun on paljon ihmisiä läsnä. Työtapa soveltuu asioiden kuvaamiseen, tietojen jakamiseen sekä erilaisten näkemysten ja kannanottojen esittämiseen. Tietopuolista informaatiota voidaan elävöittää esimerkiksi AV-välineillä. Työtapa ei aseta suuria vaatimuksia opetustiloille. Luennointi on yliopisto-opinnoissa keskeinen opetusmenetelmä, ja siten esittävällä opetuksella on yliopistossa merkittävä rooli – yksittäisenä opetusmenetelmänä joskus jopa liiankin merkittävä (Tampereen yliopisto 1995). Esittävän opetuksen vahvuudet tulevat esiin tiedollisiin tavoitteisiin pyrittäessä.

Ohjelmoinnin alkeiden opettamisen näkökulmasta esittävällä opetuksella on oma tärkeä roolinsa: erityisesti työvälineiden käytön opettamiseen esittävä opetus on usein mielekäs vaihtoehto, varsinkin silloin, kun työvälineet eivät ole itse tarkoitus.

5.3.2 Tekemällä oppiminen

Tekemällä oppiminen eli toiminnasta oppiminen (*learning by doing, learning by action, action learning*) on ehkä ensimmäinen ihmisten välinen oppimismenetelmä ja se on yhä pienen lapsen perusoppimismenetelmä. Menetelmän ajatuksena on: "Ota mallia ja tee perässä tai tee kokeilemalla yrityksen ja erehdyksen kautta". (Vuorinen 2001, 179.)

Oppiminen tapahtuu tekemällä ja osallistumalla. Kirjoittajasta riippuen oppimiseen sisällytetään erilaisia aktiiviseen toimintaan perustuvia opetusmenetelmiä aina käsitöistä draamaan. Kysymyksessä ei ole tarkoin määritelty oppimismenetelmä, vaan se on tarkoitettu monenlaisille lähestymistavoille, joita yhdistävänä ajatuksena on: "Sellainen toiminta, jossa on runsaasti yhteyksiä opiskeltavaan aiheeseen, tuottaa parhaan oppimistuloksen". (Vuorinen 2001, 179.)

Yhdysvaltalainen John Dewey (1859–1952) mainitaan usein yhtenä huomattavimmista 1900-luvun filosofeista ja opetuksen ja koulumaailman kehittäjistä. Työnteon avulla oppiminen oli hänen johtoajatuksensa: tiedon saavuttaminen edellyttää toimintaa, ja tieto on pohjimmiltaan yksi kokemuksen muoto (Dewey 1957, 6). Deweyn mukaan (1966, 186) harjoitteiden tuomat kokemukset siirtyvät yksittäisistä tilanteista arkitodellisuuteen, kun oppiminen tapahtuu tekemisen kautta. Kun opeteltava asia on oppijan välittömässä kokemuspiirissä, on Deweyn mukaan aihetta ympäröivän teoreettisen aineiston oppimisen halu myös luonnollista ja itsestään selvää. Teoksessaan *Koulu ja yhteiskunta* (1957, 18) hän vertaa tekemällä oppimista ja tiedon jakamista kotitilan pitoon:

Ei suurikaan määrä havaintotunteja, jotka on järjestetty havaintotunteina *tietojen jakamistarkoituksessa, korvaa edes vähäistä osaa sitä tutustumista talon eläimiin ja puutarhan kasveihin, joka saavutetaan elämällä niiden parissa ja huolehtimalla niistä. Koulussa tapahtuva aistien harjoitus, joka suoritetaan vain aistinten harjoituksen vuoksi, ei koskaan voi kilpailla sen aistielämän täydellisyyden ja vireyden kanssa, joka herää joka päivä tapahtuvassa läheisessä kosketuksessa ja kiintymyksessä kotoisiin ammatteihin.*

Lapsella ei ole paljon taipumusta abstraktiseen selvittelemiseen. Lapsella on tarve saada jotakin aikaan: ensiksi leikissä, liikkeissä, eleissä, ja muuttuessaan selväpiirteisemmäksi,

aineiden muovaamisessa selviin muotoihin ja pysyviin ruumiillistumiin. Dewey kutsuu tätä *luomisen viettymykseksi*. (Dewey 1957, 48–50.)

Tarkastellaan asiaa vielä ohjelmoinnin näkökulmasta. Tekemällä oppimisessa oppilaat kirjoittavat pieniä ohjelmia, soveltaen aikaisempaa tietoa, joko opittavasta ohjelmointikielestä tai muista kielistä. Kirjoittaessaan pieniä ohjelmia oppilaiden taidot karttuvat, ja pikku hiljaa tehtävien vaikeustaso nousee, ja he voivat siirtyä ohjelmoimaan yhä kompleksisempia ohjelmia. Tekemällä oppimisen pedagogisena ajatuksena on, että harjoitellessaan kirjoittamaan yksinkertaisia ohjelmia, oppilaat oppivat samalla kielen perussyntaksin ja -semantiikan (Nicholson ym. 1997, 84–86). Pelien kehittäminen antaa näin ollen hyvät mahdollisuudet oppia ohjelmointia oivaltamisen ja kokeilemisen kautta ympäristössä, jossa muutosten tekeminen ja toimintoihin tutustuminen on turvallista ja helppoa. Ohjelmointikirjastoon tehdyt valmiit pienet ohjelmat sekä esimerkkipelit antavat oppilaalle hyvät lähtökohdat kokeilla tehdä omia muutoksia ohjelmakoodiin ja nähdä tekemiensä muutosten vaikutus lopullisessa ohjelmassa.

Lähestymistapa saa Nicholsonilta ym. (1997, 84–86) myös kritiikkiä: ongelmina nähdään mm. laadullisesti heikompi ohjelmakoodin laatu. Kun opiskelijoille annetaan yksinkertaisia harjoitustehtäviä, jää vuorovaikutus monimutkaisten ohjelmien kanssa vähäiseksi, eivätkä he näin ollen opi oikeaa ohjelmointitekniikkaa.

Toisaalta oppilaille voidaan antaa valmis, monimutkainen ohjelmakoodi, jota he voivat tutkia ja muokata, ja sitten katsoa, kuinka tehdyt muokkaukset vaikuttavat lopullisessa ohjelmassa. Tällöin avoimeksi saattaa jäädä se, miten ohjelma rakennetaan alusta alkaen, sillä oppilas ei näe ohjelmakoodin syntyvaiheita.

Hakkaraisen ym. (2001, 205) mukaan tekemällä oppiminen johtaa vain harvoin olennaisiin muutoksiin oppimistuloksissa tai oppilaiden ajattelun ja asiantuntijuuden kehittymisessä. Ajatus perustuu Bereiterin (2002) väitteeseen siitä, että tekemällä oppimista korostavien pedagogisten käytäntöjen taustalla on kolme virhepäätelmää (Hakkarainen ym. 2001, 205–206): Ensinnäkin oletetaan, että kun lapsilla on hauskaa, niin he oppivat. Oppimisen kannalta olennaista on kuitenkin oppilaiden rohkaiseminen heidän omaan älylliseen ponnisteluunsa (mts. 205). Menestyäkseen elämässä ja pysyäkseen perässä alati muuttuvan

maailman asettamissa vaatimuksissa ihmisen on oltava valmis oppimaan myös asioita, jotka ovat tylsiä ja vaikeita; asioita, jotka uhkaavat omaa egoa; asioita, joiden takia joudumme olemaan tekemisissä sellaisten ihmisten tai tahojen kanssa keistä emme välttämättä pidä (Bereiter 2002, 227).

Toiseksi oletetaan, että lapset ovat kiinnostuneita vain konkreettisista ja tutuista asioista (Bereiter 2002, 300–301). Tämä näkyy erityisesti siinä, että ala-asteen opetussuunnitelma ja oppimiskäytännöt kohdistuvat ennen kaikkea lasten lähiympäristöön, vaikka heitä kiinnostavat usein käsitteellisen ymmärryksen kannalta tärkeät asiat ja ilmiöt, kuten ihmisen kehityshistoria, tähdet ja avaruus ja niin edelleen, jotka ovat hyvin kaukaisia sekä ajassa että paikassa.

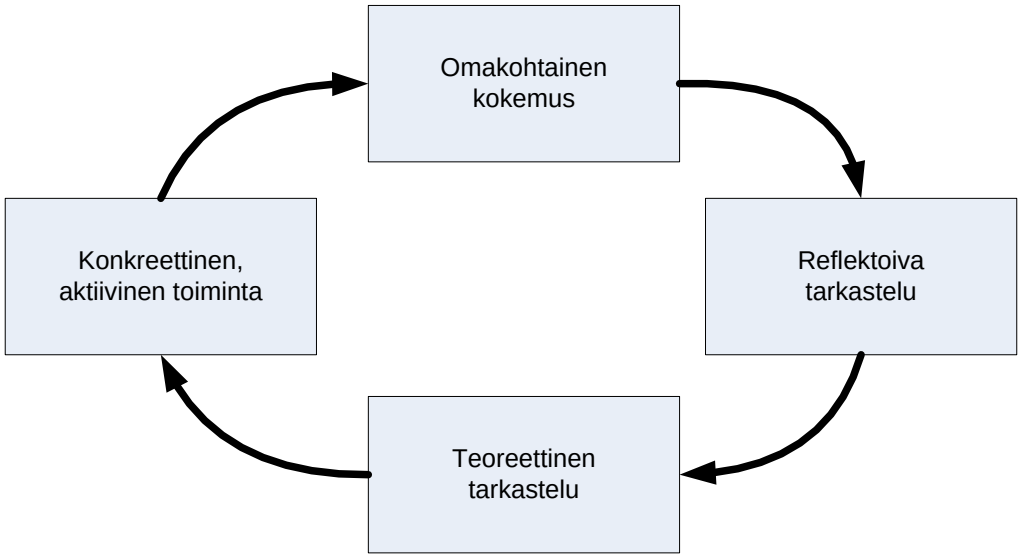
Kolmanneksi uskotaan, että konkreettisten ja välittömästi havaittavien asioiden kanssa työskentely johtaa käsitteellisen ymmärryksen syvenemiseen, mutta tämän uskomuksen, kuten kahden aikaisemmankaan, tueksi ei ole esitetty tieteellisiä perusteita.

Hakkarainen ym. (mts. 206) kiteyttää asian seuraavasti:

(…) vaikka tekemällä oppiminen on sinänsä pedagogisesti arvokas käytäntö, niin yleensä opitaan ainoastaan niitä asioita, joihin toiminta kohdistuu. (…) [Tekemällä oppiminen] ei itsestään johda oppilaiden ymmärryksen syvenemiseen tai yhteisön tiedontason kehittymiseen. Tekemällä oppiminen tuottaa parhaita tuloksia silloin, kun siihen tietoisesti liitetään opiskelijoiden käsitteellisen ymmärryksen kehittymistä tukevia oppimistoimintoja, kuten opiskeltavien ilmiöiden selittämistä.

5.3.3 Kokemuksellinen oppiminen

Kokemuksellinen oppiminen on tekemällä oppimisesta ”jalostettu” oppimisnäkemys, jonka ydinajatus on, että oppiminen etenee konkreettisia kokemuksia ja toimintaa reflektoiden kohti ilmiöiden teoreettista ymmärtämistä sekä parempia toimintamalleja (Rauste-Von Wright 2003, 150). Teorian mukaan oppiminen etenee syklisesti, ja voi käynnistyä mistä vaiheesta tahansa. Vaiheiden kokoelmaa sanotaan oppimisen kiertokuluksi (kuva 9).



Kuva 9. Kokemuksellisen oppimisen kiertokulku (Kolb 1984, 18; Rauste-Von Wright 2003, 156).

5.3.4 Tutkiva oppiminen

Hakkarainen ym. (2001, 202–206) luovat mallin *tutkivan oppimisen* prosessista ja sen keskeisistä osatekijöistä. Tutkivassa oppimisessa olennaista on tiedon käsittely toiminnan kohteena. Tällä viitataan tietoiseen ja tavoitteelliseen toimintaan, jossa oppimisyhteisö pyrkii ymmärtämään ja selittämään tutkimuksen kohteena olevia ilmiöitä luomalla niitä edustavia käsitteellisiä luomuksia (mts. 201).

Tutkivan oppimisen keskeisiä osatekijöitä ovat (mts. 203–204):

1. opetuksen ankkuroiminen opiskelijoiden aikaisempiin kokemuksiin: tämä mm. auttaa opiskelijoita sitoutumaan ja motivoitumaan, ja soveltamaan oppimaansa tietoa myöhemmin

2. ongelmalähtöinen oppiminen: luonteenomaista on, että opiskelija käsittelee uutta tietoa ongelmallisena asiana, joka täytyy selittää

3. selittämiseen tähtäävä oppiminen: tavoitteena on rohkaista opiskelijoita ulkoistamaan omia intuitiivisia käsityksiään, ja omien työskentelyteorioiden

(hypoteesien, selitysten, tulkintojen ja mallien) luominen tutkimuksen kohteena olevista ilmiöistä

4. kriittinen arviointi: oppimisyhteisön työskentelyteorioista nostetaan esiin epäselvyyksiä tai puutteellisuuksia, ja asetetaan uuden, syventävän tiedon hankintaan liittyviä tavoitteita.

5. uuden tiedon hankkiminen: opiskelijat testaavat teorioitaan etsimällä tietoa tiedonlähteistä.

6. asiantuntijuuden jakaminen: kaikki tutkivan oppimisen osaprosessit voidaan jakaa oppimisyhteisön jäsenten kesken.

Tutkivan oppimisen pedagogiikkaa luonnehtivat seuraavat tekijät (mts. 206):

1. Työskentely kohdistuu ennemmin käsitteellisten ongelmien ratkaisemiseen kuin sisältöalueiden käsittelyyn.

2. Opiskelijoiden omien ideoiden ja tulkintojen tuottaminen, kehitteleminen ja sitoutuminen syvenevään tutkimusprosessiin on tärkeämpää kuin tiedon kertaluonteinen tuottaminen.

3. Käsitteellisten luomusten etsiminen, kehittely ja jakaminen on tärkeämpää kuin vastausten löytäminen ja tiedon kopioiminen.

4. Tiedon kehittelyn tulisi olla julkista.

5. Välittömän suoriutumisen korostamisen sijaan tulisi olla mahdollisuus osallistua palautteen antamiseen ja vastaanottamiseen, vuorovaikutukseen ja pohdintaan.

6. Ohjattuun tutkimusprosessiin osallistuminen ennemmin kuin luottaminen opiskelijan itseilmaisuun ja kykyyn toteuttaa ohjaamattomasti luovaa toimintaa.

7. Opiskelijan roolin sijaan tulisi hiljalleen, asteittain, kyetä omaksumaan asiantuntijan tai tutkijan rooli.

On tietenkin muistettava, että tutkiva oppiminen on malli oppimisprosessista, eikä sen soveltaminen mielivaltaisesti välttämättä paranna oppimistuloksia (mts. 206–207). Opettajan tulisi tuntea malli, menetelmät, ja hallita niiden soveltaminen, ennen kuin hän alkaa hyödyntää niitä omassa opetuksessaan – muutoin on vaarana, että opiskelijat tekevät opetuksen toteutuksesta perusteettomia johtopäätöksiä, kuten esimerkiksi Helsingin yliopiston maatalous-metsätieteellisessä tiedekunnassa syyslukukaudella 1999 käynnistyneellä oppimisen ja opetuksen laadun arviointihankkeessa (Heikkilä & Repo-Kaarento 2000): Opiskelija voi kokea tutkivan oppimisen ”huonosti suunnitelluiksi ryhmätöiksi luentojen lisäksi”, tai että ”opettaja käyttää ’tutkivaa oppimista’ vähentääkseen omaa työmääräänsä”, ja edelleen ”jaettu asiantuntijuus on usein pelkkää monisteiden jakelua ja tiedon kopiointia toisilta” (lainaukset teoksesta Heikkilä & Repo-Kaarento 2000).

5.3.5 Suunnittelun kautta oppiminen

Suunnittelun kautta oppimista (*learning through design*) tapahtuu samalla, kun suunnitellaan jotain tuotetta, kuten peliä, robottia, vaatetta, laitetta jne. (Kalantzis ym. 2005, 99). Suunnitteluympäristö voidaan räätälöidä jokaiselle henkilökohtaisesti merkitykselliseksi oppimisympäristöksi.

Kun oppilaille annetaan oppimistilanteessa ”valta” tutkia ja suunnitella itse (pelkän tiedon siirtämisen sijaan), voi se toimia luokan ”katalysaattorina” – siitä voi tulla suorastaan oppimisen keskipiste (Lehrer ym. 1994, 248).

5.3.6 Mallioppiminen

Mallioppimisella tai sosiaalisella oppimisella (*social learning theory*) tarkoitetaan oppimisen muotoa, jossa yksilö oppii tarkkailemalla, jäljittelemällä ja mallintamalla muiden toimintaa (Bandura 1977). Oppilaille voidaan esittää esimerkiksi mallisuoritus jonkin ohjelmointiongelman ratkaisusta, jonka perusteella oppilaat voivat ratkaista samankaltaisia ongelmia. Tässä ongelmana voi olla (tekemällä oppimisen tapaan, ks. luku 5.3.2), etteivät oppilaat ymmärrä, miksi alkuperäisen ongelman ratkaisu on mielekäs (Nicholson & Fraser 1997).

5.3.7 Pedagogiset menetelmät opetuksessa

On tavallista, että edellä esitellyt pedagogiset menetelmät ja lähestymistavat sekoittuvat ohjelmointikurssin käytännön opetuksessa (King 1992). Ne voivat myös muokkaantua ryhmän mukaan. Osa näkökulmista soveltuu paremmin niille, kenellä on jo hieman ennakkotietoja ohjelmoinnista; esimerkiksi tutkivan oppimisen pedagogiikkaa (ks. luku 5.3.4) ja työskentelyn kohdistamista puhtaasti käsitteellisten ongelmien ratkaisuun voi olla vaikea toteuttaa käytännössä, mikäli oppilaalla ei ole aavistustakaan siitä, miten ohjelmakoodia kirjoitetaan käytännössä.

Roine (2007, 24) ehdottaa perusopetukseen lähestymistapojen hybridiä, missä käsitelähtöisen lähestymistavan lisäksi käytettäisiin sekä mallista oppimista että tekemällä oppimista. Näin työskentelyyn saataisiin riittävästi motivaatiota (konkretia), mutta ohjelmakoodi pysyisi myös syntaktisesti oikeana mallikoodin avulla. Liiallinen abstraktiotason nostaminen voi olla huono asia (ks. luku 5.6), vaikka Kölling (1999) suositteleekin ensimmäiseksi käsitteiden opiskelua olio-ohjelmoinnissa.

5.4 Ensiohjelmointi

Sillä, miten opimme jonkin asian *ensimmäisen kerran*, tuntuu olevan erityinen merkitys. Varmasti paras esimerkki tästä asiasta on oma äidinkielemme. Monet voivat oppia käyttämään myös vieraita kieliä varsin hyvin, mutta lähes koskaan eivät täydellisesti. Niinpä äidinkielen rooli jää jollain tavalla erityiseksi. Joillakin saattaa olla kaksi tai kolmekin äidinkieltä, jolloin tämä pätee heidän kohdallaan toki noihin kaikkiin äidinkieliin.

Analogia ensimmäisen ohjelmointikielen ja äidinkielen välillä ei ole täydellinen, koska äidinkieli opitaan hyvin erilaisten vaiheiden kautta, eri tavalla ja vieläpä eri iässä kuin ensimmäinen ohjelmointikieli. Oppiessamme äidinkieltä meillä ei ole minkäänlaista käsitystä kielen rakenteesta tai kielioppisäännöistä. Ymmärryksemme kielestä vain pikkuhiljaa karttuu kokeilemalla, yrittämällä ja erehtymällä – samalla kuitenkin uskomattomalla tavalla luovasti käyttäen jo oppimaamme. Wittgenstein (1981, 135) käyttää pohdinnoissaan *kielipeli*-käsitettä liittyen kielen oppimiseen: Me opimme pelaamaan peliä, tuntematta pelin sääntöjä, tietämättä edes että pelissä ylipäätään on

sääntöjä. Tämä ei varmasti ole se tapa, millä aikuiset oppivat uuden vieraan kielen tai ohjelmointikielen. Ennemminkin he oppivat (tai oikeastaan heidät *laitetaan opiskelemaan*, ks. luku 5.1 ja Bloomin taksonomia) ensin kielen käsitteitä ja rakenteita, ja taitavaan käyttämiseen on kuljettava pitkä matka. On kuitenkin totta, että ensimmäinen opittu ohjelmointikieli antaa hyvät puitteet oppia myöhemmin myös muita uusia ohjelmointikieliä (Böszörményi 1998, 141), ja siksi on hyvin tärkeää, että ohjelmoinnin opiskelun alkutaipaleelle on valittu juuri oikeat oppimistyylit, pedagogiset menetelmät sekä ohjelmoinnin opetuksen lähestymistapa.

5.5 Erilaiset lähestymistavat ohjelmoinnin opetuksessa

Association for Computing Machinery eli ACM on 1947 perustettu maailmanlaajuinen tieteellinen yhteisö, jonka tarkoituksena on edistää tietotekniikan tutkimusta ja opetusta. Sen jäseninä on opettajia, tutkijoita, opiskelijoita ja tietotekniikka-alan ammattilaisia. Jäseniä on yli 100 maasta, yhteensä noin 95 000. Näistä kaksi kolmasosaa on ammatillisia jäseniä, kolmasosa opiskelijoita. (ACM 2010.)

IEEE Computer Society, lyhyemmin IEEE-CS, on IEEE:n (Institute of Electrical and Electronics Engineers) jäsenjärjestö. IEEE-CS perustettiin vuonna 1971, ja se on noin sadalla tuhannella jäsenellään (vuonna 2004) yksi suurimmista kansainvälisistä tekniikan alan järjestöistä. Järjestön toiminnan piiriin kuuluu julkaisutoiminta, tieteellisten konferenssien järjestäminen, koulutuksen edistäminen sekä monien alan keskeisten standardien määrittely (IEEE-CS 2010). Tunnettuja IEEE:n määrittelemiä standardeja ovat muiden muassa IEEE 754 (liukulukujen toteuttaminen tietokoneissa) ja IEEE 1394 (*Firewire* tai *i.Link*). IEEE Suomen osastoon kuuluu noin tuhat jäsentä.

Vuonna 2001 ACM ja IEEE-CS laativat yhteistyössä opetussuunnitelman tietotekniikan opintojen suositukseksi Yhdysvalloissa. Suositus on nimeltään Computing Curricula 2001. (Jatkossa ACM/IEEE-CS-työryhmään viitataan tästä eteenpäin yksinkertaisuuden vuoksi vain lyhenteellä ACM.) Oppaassa annetaan suositukset tietotekniikan (*Computer Science*) opetussuunnitelmille, oppiaineen sisällön perustietämykselle ja sen toteuttamiselle alemmissa ja ylemmissä korkeakoulututkinnoissa (ACM 2001, 1–4, 14–17). ACM on julkaissut opetussuunnitelmasuosituksia muillekin koulutusohjelmille, kuten

tietokonetekniikka (*Computer Engineering*), tietojärjestelmätiede (*Information Systems*), informaatioteknologia (*Information Technology*) ja ohjelmistotekniikka (*Software Engineering*).

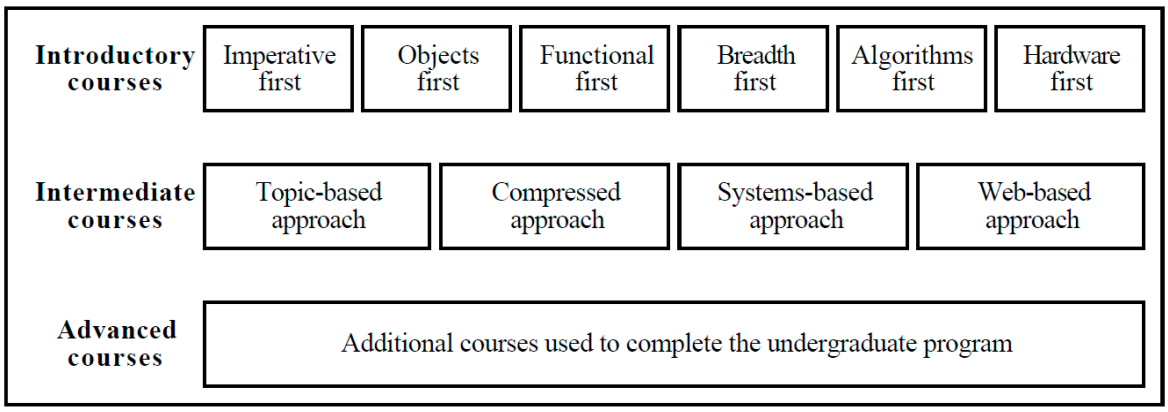
Opetussuunnitelmasuosituksessa yksittäiset opintojaksot jaotellaan vaatimustason mukaan kolmeen kategoriaan (ACM 2001, 22–35): (i) ydinaineskursseihin (tai alkeiskursseihin, *introductory courses*), (ii) täydentävään tietämykseen (tai keskitason kursseihin, *intermediate courses*) ja (iii) erityistietämykseen (tai syventäviin kursseihin, *advanced courses*). Ohjelmointikurssit kuuluvat luonnollisesti tähän jaotteluun mukaan.

Ydinaines kattaa tiedot ja taidot joiden hallitseminen on välttämätöntä uusien tietojen omaksumisen kannalta. Ydinaines on tietoa, jonka hallitseminen on välttämätöntä jatkon kannalta ja jonka ymmärtäminen mahdollistaa syventävän ja laajentavan tiedon hankkimisen. Ydinaineksessa on harvemmin mukana yksittäisiä faktoja – enemmänkin teorioita, malleja ja periaatteita. Tavoite on, että kaikki opiskelijat hallitsevat ydinaineksen.

Täydentävä tietämys kattaa teorioiden, mallien ja periaatteiden yksityiskohtia ja laajennuksia, jotka toisinaan voivat olla tarpeellisia, mutta aika- ja oppimisresurssin takia tätä tietämystä ei painoteta eikä sitä opeteta ydinaineksen oppimisen kustannuksella. Erityistietämys on tietoa, joka toimii ydinaineksen ja täydentävän tietämyksen yksityiskohtina. Sillä tuskin koskaan on käyttöarvoa perusasioiden omaksumisen kannalta ja tämä tietämys on oppijan itsensä harrastuneisuuden ja erikoistumisen varassa.

Erityistietämykseen ei mainintaa enemmän käytetä aikaa eikä sen omaksumista ja oppimista vaadita tutkinnossa.

Kuvassa 10 on ACM:n esittelemät lähestymistavat. Koska tässä tutkimuksessa on nimenomaan kysymys alkeisohjelmoinnista ja sen opettamisesta, ei keskitason tai syventävien kurssien lähestymistapoihin puututa.



Kuva 10. Toteutusstrategiat kurssien tason mukaan ryhmiteltynä (ACM 2001, 18).

Vastaavaa kolmijakoa (ydinaines – *must know*, täydentävä tietämys – *should know*, erityistietämys – *nice to know*) on käytetty esimerkiksi ns. *ydinainesanalyysin* soveltamiseen yliopistojen opetussuunnitelmien laatimisessa. Ydinainesanalyysi on Oulun yliopiston (ks. Karjalainen 2003, 74) kehittämä työväline korkeakoulututkinnon suunniteluun ja kehittämiseen, jonka tehtäväksi on määritelty seuraavaa:

- - auttaa opettajaa hahmottamaan opettamansa aiheen tietojen ja taitojen väliset hierarkiat ja yhteydet sekä suhteuttamaan nämä opiskelijan oppimisaikaan, tutkintovaatimukseen ja opetussuunnitelmaan. Ydinainesanalyysin tehtyään opettaja pystyy hahmottamaan kurssinsa työmäärän oikein suhteessa kurssiin varattuun aikaan.

ACM:n mukaan (2001, 28–34) ydinaineksen opettamiseen hyväksi havaittuja lähestymistapoja ovat

- ”imperatiivisuus ensin” (imperative-first),
- ”oliot ensin” (objects-first),
- ”laajuus ensin” tai ”leveys ensin” (breadth-first),
- ”funktionaalisuus ensin” (functional-first),
- ”algoritmit ensin” (algorithms-first) sekä
- ”laitteet ensin” tai ”rauta ensin” (hardware-first).

Koska Java-kieli on saanut vahvan jalansijan niin opetuksessa kuin ohjelmistoteollisuudessa (ks. luku 4.2; Van Roy 2009, 10), ja olio-ohjelmointi on

(erityisesti opetuksessa) tämän hetken eniten käytetyin ohjelmointiparadigma (ks. luku 4.2), on ”oliot ensin” -lähestymistapa monelle opettajalle ohjaava konseptuaalinen määrittely oman kurssin sisällöllisessä ja pedagogisessa suunnittelussa.

Seuraavassa luvussa tarkastellaan joitakin oliot ensin -lähestymistapaan liittyviä opetuksellisia haasteita ja esitellään ACM:n lähestymistavoista hieman poikkeava lähestymistapa alkeisohjelmointikurssin opetukseen.

5.6 ”Oliot ensin”-lähestymistavasta ”muuttujat ensin”-lähestymistapaan

Luvussa 4 käsiteltiin erilaisia ohjelmoinnin paradigmoja, ja kerrottiin olio-ohjelmoinnilla olevan tällä hetkellä vahva jalansija maailmalla niin ohjelmistotuotantoteollisuuden kuin ohjelmistotekniikan tutkimuksessa (ks. luku 4.2). Olio-ohjelmoinnin suosioon on monia syitä: muun muassa se, että olio-ohjelmointi soveltuu hyvin graafisten käyttöliittymien toteuttamiseen. Olio-ohjelmointikielille on olemassa valmiita käyttöliittymäkirjastoja (esimerkiksi Swing, SWT tai MFC), joiden avulla käyttöliittymien ohjelmointi on mahdollista.

Olio-ohjelmoinnissa ohjelmat koostuvat olioista ja niiden välisestä yhteistyöstä ja kommunikaatiosta (Khor 1995). Niinpä olioparadigman käsitteiden, kuten luokkien, metodien ja attribuuttien, perinnän, kapseloinnin, jne. (käsitteet esiteltiin lyhyesti luvussa 4.6), on katsottu usein olevan myös asia, mistä opettaminen alkaa. Tätä kutsutaan ns. *oliot ensin* (*objects-first*) -lähestymistavaksi. Vaikka olio-ohjelmointi kiistatta on tärkeä kokonaisuus ohjelmistoteollisuudessa ja jotkut suosittelevat oliokäsitteiden ottamista opetukseen varhaisessa vaiheessa (esim. Meyer 1993, 586–587), sisältää se ohjelmoinnin alkuopetuksen kannalta ongelmia.

Muun muassa Böszörményi (1998) sekä Sajaniemi ja Hu (2006) ovat sitä mieltä, ettei oliot ensin -lähestymistapa ole didaktisesti hyvä alkeisohjelmointitasoisten kurssien lähestymistavaksi. Böszörményi (1998, 142) perustelee tätä siten, että esimerkiksi modularisoinnin käsite on perustavanlaatuisempi, kuin mitä olio-ohjelmointi antaa ymmärtää. Toisaalta Böszörményi huomauttaa, ettei yliopistojen tulisi olla ”ehdotonta viisautta jakava laitos, vaan opettaa oppilaat ennemmin ajattelemaan erilaisten

mahdollisuuksien olemassaoloa”, mutta kuitenkin monet yliopistot toimivat juuri jakaen olio-ohjelmointia jonkinlaisena ehdottomana viisautena käsitellessään ohjelmoinnin alkeita (mts. 142). Edelleen hän muistuttaa ohjelmoinnin kehityksen historiankulusta: tietokoneohjelmointi alun perin nivoutui tiukasti peruskäsitteiden (*basic consepts*), kuten vakio, muuttuja, algoritmi ja funktio, ympärille, eikä kehityksen kulku suinkaan alkanut abstrakteista ja monimutkaisista luokan ja olion käsitteistä (toki kehitys niihin aikanaan johti). Ensimmäinen olio-ohjelmointikieli **Simula-67** ilmestyi 60-luvun puolenvälin jälkeen (mts. 142) – lähes kaksi vuosikymmentä ensimmäisten tietokoneohjelmien jälkeen. On siis syytä pohtia, onko didaktisesti mielekästä aloittaa opetusta olionäkökulmasta, ja voidaanko opiskelijoilta odottaa näiden käsitteiden omaksumista muutamassa viikossa.

Niin ikään Sajaniemi ja Hu (2006) esittävät artikkelissaan *Teaching Programming: Going beyond "Objects First"* oliot ensin -lähestymistavan kaltaisen asettelun opetuksen suunnittelussa olevan haasteellinen ensiohjelmoinnin kannalta. Abstraktiotason nostaminen korkealle johdantokurssin aluksi voi vähentää mielenkiintoa ohjelmointia kohtaan ja lisätä kurssin keskeyttäjien määrää. Henkilön, joka on aloittamassa ohjelmoinnin opiskelua, kannalta on huomattavan useita haasteita. Ensiohjelmoinnissa asiasisällön oppimiselle voi asettaa haasteita sinänsä kehälliset asiat, kuten valitun ohjelmointikielen syntaksi ja notaatiot. Toisaalta syntaksin ja notaatioiden hallintaa usein myös painotetaan osaamisen testeissä, kuten tenteissä ja harjoitustehtävissä, mikä taas siirtää huomiota pois varsinaisesta aiheesta. (Sajaniemi & Hu 2006.)

Vaihtoehtoiseksi lähestymistavaksi Sajaniemi ja Hu (2006) ehdottavat ns. *muuttujakonseptia*, missä edetään ”konkretian kautta abstraktioon”. Muuttujakonseptissa ennen olioajatteluun siirtymistä esitellään *muuttujien* (*variables*) roolit ja vastuut. Lähtökohtana muuttujien vastuun esittelyssä on niille osoitetut tehtävät, esimerkiksi *luvun pitäminen asioista*. Oliomielessä muuttujakonseptissa käsitellään esimerkiksi attribuutteja, paikallisen muuttujan käsite sekä metodiparametrit. Näin siirtymä muuttuja-ajattelusta olioajatteluun – sitten kun sen aika on – tapahtuu mahdollisimman luonnollisesti ilman liiallista abstraktiotason nostamista opiskeluiden alkuvaiheessa. Järjestys vastaa myös Böszörményin (1998, 142–143) kuvausta ohjelmointikielen historian huomioimisesta suunniteltaessa asioiden opetusjärjestystä.

Hyvänä esimerkkinä abstraktiotason nostamisesta liian varhaisessa vaiheessa on 1970-luvun huonomaineinen koulutuskokeilu (Lyyra 2007), jossa joukko-oppi tuotiin uutena metodina matematiikan perusteiden opetukseen. Uusi matematiikka miellettiin kokonaisuudessaan liian teoreettiseksi, symboleja ja nimityksiä ylikorostavaksi sekä käytännön elämälle vieraaksi. Oppisisällöt olivat tuntikehykseen verrattuna liian laajat, eikä uusi jaottelu johtanut odotettuihin oppimistuloksiin (Lyyra 2007). Joukko-opin käytöstä matematiikan alkeiden opetuksessa luovuttiin virallisesti vuonna 1983.

Muuttujakonsepti ei siis hylkää vallalla olevaa olioajattelua, vaan ennemminkin pyrkii ottamaan huomioon rakenteisen ohjelmoinnin konseptin yksinkertaisuuden, ja hyödyntämään sitä *ennen* olioajatteluun siirtymistä, jonka tulisi tapahtua vähitellen.

5.7 Ohjelmoinnin integrointi muuhun opetukseen

Käsite *integrointi* ja sen synonyymitermit *integraatio*, *eheyttäminen* ja *kokonaistuminen* (Hirsjärvi 1983, 65; Lahdes 1997, 211) ovat olleet käytössä useilla aloilla, ja sanalla voidaan tilanteesta ja alasta riippuen tarkoittaa hieman eri asioita. Integraatio on käsitteenä käytössä koulutuksen lisäksi myös esimerkiksi psykologiassa ja kasvatustieteessä. Integraatiolla ei tässä raportissa tarkoiteta erityisopetuksen ja yleisopetuksen yhteensulauttamista, josta esimerkiksi Moberg (1998, 155) puhuu. Hirsjärven (1983, 65) mukaan integroinnilla (vastakohta differentioituminen) tarkoitetaan esimerkiksi oppiainesten opetussisältöjen sulauttamista yhdeksi kokonaisuudeksi. Kysymyksessä voi olla esimerkiksi esteettisen, humanistisen ja luonnontieteellisen oppiaineksen integrointi tiettyä opetusmenetelmää käyttäen tarkoituksenmukaiseksi kokonaisuudeksi. Malisen (1985, 146) mukaan integrointi on opetuksen eheyttämistä, jossa opetuksen kautta muodostetaan opetettavasta aineksesta oppilaalle mielekkäitä opetuskokonaisuuksia. Erikssonin (1986, 65) mukaan integraatio merkitsee opetustoiminnan suuntaamista siten, että oppilaan oppilastoiminta muodostaisi ajatuksellisesti ja emotionaalisesti loogisen kokonaisuuden suhteessa ainekseen. Hirsjärven, Malisen ja Erikssonin määritelmät ovat melko lähellä toisiaan, ja jatkossa integroinnilla viitataan nimenomaan Hirsjärven määritelmään.

Opetushallitus edellyttää tiettyjen kokonaisuuksien sulauttamista perusopetuksen kaikkiin oppiaineisiin, joista yksi on tieto- ja viestintätekniikkaan läheisesti liittyvä Viestintä ja mediataito -aihekokonaisuus (Opetushallitus 2004, 38–43).

Tietotekniikka on osa lähes jokaisen ihmisen arkea. On monia työpaikkoja, joita ei olisi olemassa ilman tietotekniikkaa, mutta myös ”tavallisissa” töissä tietokoneet ja tietotekniikan hyödyntämisen taito ovat yksilölle välttämättömyys. Toimisto-ohjelmien, sähköpostin ja sähköisen tiedonhaun taitoja pidetään yrityksissä itsestään selvinä. Toisaalta myös jatko-opinnoissa tietotekniikan perussovellusten ja sähköisen viestinnän hallinta on opiskelun ehdoton edellytys.

Suomalaisten koulujen käytössä oleva tietokoneiden määrä on keskimäärin hyvä: yli 70 prosentilla kouluista on alle 10 oppilasta tietokonetta kohden (Kankaanranta & Puhakka 2006, 27). Tästä huolimatta tutkijoiden mukaan tietoteknisten perustaitojen, kuten tekstinkäsittelyn, taulukkolaskennan, esitysgrafiikan, kuvankäsittelyn jne., hallinnassa on vakavia puutteita (Siukonen 2008). Peruskoulun opetussuunnitelman perusteiden mukaan (Opetushallitus 2004, 38–43) viestintäteknisten välineiden monipuolisen käytön oppiminen pitäisi tapahtua peruskoulussa. Tietotekniikka ei kuitenkaan ole pakollisten opetettavien aineiden listalla, ja niinpä tietotekniikan oppisisällöt, kuten edellä mainittu viestintäteknisten välineiden käytön opiskelu, kuuluvat jokaiseen oppiaineeseen niin sanotun Viestintä ja mediataito -aihekokonaisuuden kautta.

Syitä tietotekniikan integroinnille

Vahvan perusteen oppiaineiden integroimiselle antaa Opetushallituksen *Perusopetuksen opetussuunnitelman perusteet* (POPS), jossa integroinnista on kerrottu (Opetushallitus 2004, 38) seuraavasti kohdassa 7.1:

Opetus voi olla ainejakoista tai eheytettyä. Opetuksen eheyttämisen tavoitteena on ohjata tarkastelemaan ilmiöitä eri tiedonalojen näkökulmista rakentaen kokonaisuuksia ja korostaen yleisiä kasvatuksellisia ja koulutuksellisia päämääriä.

Tukea oppiaineiden integroimiseen tarjoaa myös luvussa 5.3.2 esitelty Deweyn näkemys tekemällä oppimisesta, jonka mukaan tiedot ja taidot siirtyvät yksittäisistä harjoittelutilanteista arjen käytännön tilanteisiin, kun oppiminen tapahtuu tekemisen kautta.

Viestintä ja mediataito sekä *Ihminen ja teknologia* ovat kaksi seitsemästä aihekokonaisuudesta, joiden avulla oppiaineiden integraatio tulisi tapahtua. Näiden aihekokonaisuuksien tavoitteina on (Opetushallitus 2004, 38–43) muun muassa, että

- oppilas oppii ymmärtämään teknologiaa, sen kehittämistä ja vaikutuksia eri elämänalueilla, yhteiskunnan eri sektoreilla ja ympäristössä
- käyttämään teknologiaa vastuullisesti
- kehittämään tiedonhallintataitojaan (…)
- käyttämään viestinnän ja median välineitä tiedonhankinnassa, -välittämisessä sekä erilaisissa vuorovaikutustilanteissa

Edelleen aihekokonaisuuksia kuvaillaan seuraavasti (Opetushallitus 2004, 38–43):

Ihminen ja teknologia -aihekokonaisuuden päämääränä on auttaa oppilasta ymmärtämään ihmisen suhdetta teknologiaan ja auttaa näkemään teknologian merkitys arkielämässämme. Perusopetuksen tulee tarjota perustietoa teknologiasta, sen kehittämisestä (…)

Opetuksessa tulee kehittää välineiden, laitteiden ja koneiden toimintaperiaatteiden ymmärtämistä ja opettaa niiden käyttöä.

Opetushallitus ja tietotekniikan opettajakunta ovat yhtä mieltä siitä, että ohjelmoinnin pitäisi olla (ainakin jollain tasolla) osa perusopetusta. Matemaattisten aineiden opettajien liitto (MAOL ry) suosittelee, että tietotekniikan yhtenä keskeisenä valinnaiskurssina tulisi olla ohjelmointikurssi, kuvankäsittely-, kotisivusto- ja multimediakurssin ohella (MAOL ry 2006, 8). Ohjelmointikurssin sisällöksi ehdotetaan muuttujalähtöistä, rakenteiseen ohjelmointiin perustuvaa lähestymistapaa. Opetushallitus linjaa perusopetuksen tieto- ja viestintätekniikan opetuskäytön kehittämissuunnitelmassa (2005, 42), että peruskoulun päättävän oppilaan pitäisi tunnistaa ja osata tietokoneohjelmien ja ohjelmoinnin periaatteita. Tällä hetkellä tilanne on kuitenkin se, että vain harvalla oppilaalla on

mahdollisuus tutustua ohjelmointiin, koska valinnaista tietotekniikkaa on vain vähän tarjolla (MAOL ry 2006, 2–3), ja pakollisilla tietotekniikan kursseilla (niissä kouluissa missä sellainen on saatavilla) keskitytään perustietojärjestelmien opiskeluun ja toimisto-ohjelmien käyttöön (Kankaanranta & Puhakka 2006, 53). Niissä kouluissa, joissa järjestetään pakollinen tietotekniikan kurssi 7. vuosikurssin oppilaille, on kurssin laajuus tavallisesti 0,5 tai 1 vuosiviikkotuntia. Joissakin kouluissa oppilas voi valita tietotekniikan valinnaisaineeksi 8. tai 9. luokalle. Tietotekniikalle ei perusopetuksen tuntijaossa ole kuitenkaan määritelty pakollisia tunteja lainkaan (MAOL ry 2006, 2–3), joten on mahdollista, ettei oppilaalla ole lukujärjestyksessään ainuttakaan tietotekniikka (tai ATK) -nimellä kulkevaa oppituntia.

Edellä mainittujen seikkojen nojalla on mielekästä sisällyttää ohjelmointi näihin kahteen aihekokonaisuuteen. Ohjelmoinnin opetus tulee siis aloittaa yläkoulussa, ja peruskoulun päättäessään oppilaalla tulisi olla perustietämys teknologiasta siten, että hän ymmärtää laitteiden ja koneiden toimintaperiaatteita, mihin ohjelmointi osana kuuluu. Edellä mainittuja tavoitteita ja sisältöjä tulisi siis POPS:n mukaan toteuttaa kaikissa opetettavissa aineissa.

Siirtyessään peruskoulusta jatko-opiskeluiden pariin opiskelijat, joilla ei ole ohjelmoinnista aikaisempaa kokemusta, joutuvat ensimmäisellä ohjelmointikurssilla tutustumaan samanaikaisesti niin ohjelmoinnin peruskäsitteisiin ja -rakenteisiin, kehitysympäristöön jossa lähdekoodia kirjoitetaan ja työstetään, sekä tietenkin käytössä olevan ohjelmointikielen syntaksiin ja semantiikkaan. Tämä saattaa johtaa oppilaan kognitiiviseen ylikuormittumiseen, eikä opiskelija välttämättä pysty omaksumaan kaikkea tietoa kerralla, ja tilanne saattaa oppimisen kannalta kääntyä itse asiassa negatiiviseen suuntaan (DuHadway ym. 2002, 7–8). Koska ohjelmointi on ongelmanratkaisua, eikä suinkaan pelkkää uuden kielen opettelua, olisi tärkeää, että ohjelmointia käsiteltäisiin jo perusopetuksessa – ainakin raapaistaisiin pintaa. Edelleen opettajia tulisi kouluttaa opettamaan oppilaille kieliä, työkaluja ja välineitä, jotka sopivat erityisesti perusopetukseen. Pelien hieman ”leikillinen” lähestymistapa ohjelmointiin voisi tässä kohtaa olla hyvä keino madaltamaan perinteisesti ohjelmoinnin oppimisessa olevaa korkeaa kynnystä (Van Eck 2006).

5.8 Mukavuustaso alkeisohjelmoinnin kurssilla on merkityksellinen

Lapset ja nuoret – ja toki aikuisetkin – leikkivät ja pelaavat, koska se on heille ajanvietettä ja hauskaa toimintaa. Lapsille pelit ja leikit muodostavat parhaimmillaan sellaisen ilmapiirin, jossa on helppo toimia, olla oma itsensä ja esittää mielipiteitään ja ajatuksiaan. Peliteeman ohjelmointikurssille tuomisen tavoitteena oli luoda mukava, välitön ja ”leppoisa” ilmapiiri kurssille, jossa on helppo kysyä kysymyksiä, antaa vertaistukea, esittää omia näkemyksiään ja arvioitaan, sekä edesauttaa nuorten pystyvyyskokemusten muodostumista ohjelmoinnin alueella. Miellyttävällä kokemuksella ohjelmointikurssista on todettu olevan positiivinen korrelaatio ohjelmointikurssilla menestymiseen (Wilson & Shrock 2001, 184–187). Miellyttävällä kokemuksella tarkoitetaan tässä seuraavia asioita:

- Mahdollisuus esittää kysymyksiä luokassa tai laboratoriotilanteessa (harjoitustehtävien tekemisen yhteydessä)
- Työskentelyinnokkuus tehtävien tekemisen aikana
- Mielikuva kurssin vaikeustasosta
- Käsitys kurssin käsitteiden hallitsemisesta verrattuna luokkatovereiden käsityksiin
- Käsitys harjoitustehtävien vaikeustasosta ja kyvystä saattaa harjoitustehtävät loppuun

Wilson ja Schrock huomauttavat (mts. 187), että vaikka heidän tutkimuksensa ei suoraan sitä sanokaan, voidaan oikeutetusti olettaa, että mitä korkeampi kurssin mukavuustaso on, sitä paremmin opiskelijat menestyvät ohjelmoinnissa.

5.9 Pelin kehitys ohjelmoimalla ja pelinkehitysohjelmistot

Pelin kehittäminen ohjelmoimalla, eli kirjoittamalla ohjelmakoodia, ei varmastikaan ole helpoin tapa tehdä omaa peliä. Pelejä voi tehdä monella muullakin tavalla, jotka eivät vaadi lainkaan kurkistusta ”pintaa syvemmälle”. Näissä ideana on, että pelinkehityksessä keskitytään itse peliin toteutustekniikan sijaan. Yksi tällainen tapa on ns. visuaalinen ohjelmointi, missä ohjelmia luodaan manipuloimalla ja järjestelemällä graafisia elementtejä (esim. Apple Inc 2010; Microsoft 2010). Otetaan esimerkkinä Kodu Game Lab, joka on Microsoftin kehittämä visuaalinen ohjelmointiympäristö (Microsoft 2010).

Sen sijaan, että käyttäjä näppäilisi itse ohjelmakoodia, hän voi rakentaa ohjelmia valitsemalla hiirellä tai peliohjaimella ruudulta visuaalisia elementtejä, ja asettamalla niitä sitten elementeille varattuun paikkaan tiettyyn järjestykseen. Elementtien järjestelemisen avulla käyttäjä voi saada esimerkiksi pelihahmon tekemään asioita, kun hän painaa vaikkapa peliohjaimen nappia. Oman pelin voi tehdä valmiiksi jopa päivässä, ja ”jotakin pelattavaa” saa aikaiseksi jo aivan hetkessä. Toinen esimerkki visuaalisesta ohjelmointikielestä on Alice. Alice on verkosta ladattava ilmainen ohjelmisto, jossa on sovitettu yhteen olio-ohjelmointikieli sekä ohjelmointiympäristö (Carnegie Mellon University 2010). Ympäristön avulla käyttäjä voi luoda ”oman 3D-maailmansa”, ja tehdä sisäänrakennetun korkean tason ohjelmointikielen, Alicen (siis samanniminen kuin ohjelma itse), avulla erilaisia asioita luomassaan maailmassa. Maailman luomisen yhteydessä käyttäjä oppii Alice-kieltä, joka perustuu olio-ohjelmoinnin paradigmaan. Toisin kuin ohjelmoinnissa tavallisesti, Alicessa ohjelmakoodirivit eivät ole keskeisellä paikalla käyttöliittymässä – itse asiassa niitä ei näy oletusarvoisesti ruudulla lainkaan. Käyttäjälle ohjelmakoodi ei siis näyttäydy keskeisenä elementtinä.

Aliceen on valmiiksi kirjoitettu ohjeita, joiden avulla voi harjoitella, kuinka maailmassa tehdään erilaisia asioita, esimerkiksi laitetaan taitoluistelija tekemään piruettia tai puu kaatumaan. Maailman luomisen ja muokkaamisen ohessa käyttäjälle opetetaan olio-ohjelmointiparadigman perusteita ja reaalimaailman ilmiöiden yhteyttä oliokielten toimintaperiaatteisiin.

Visuaalisten ympäristöjen ehdottomana etuna on, että ne antavat onnistumisen elämyksiä verrattain hyvin nopeasti, ja pelien tekeminen voi tuntua niiden avulla hyvin helpolta. Visuaalisten ohjelmointikielten (tai visuaalisen ohjelmoinnin) heikkoutena on se, ettei niitä voi oikeastaan käyttää mihinkään muuhun, kuin tarkasti rajattuun tarkoitukseen – juuri tietynlaisten pelien tai ohjelmien tekemiseen. Esimerkiksi Kodu ei myöskään ole sisällyttänyt ohjelmointikieleensä yleisiä ohjelmoinnin perusperiaatteita, kuten symbolisia muuttujia, silmukkarakenteita, aliohjelmia ja niin edelleen (Microsoft 2010), joten jos ohjelmointiharrastuksen aloittaa Kodulla, voi ”oikeaan” kieleen, esimerkiksi Javaan tai C#:n, siirtyminen tuottaa aika paljon harmaita hiuksia.

Koska tässä työssä tutkittavalla kurssilla tavoitteena oli muutakin, kuin että oppilaat saavat pelin tehtyä, päädyttiin nimenomaan siihen lähestymistapaan, että oppilaat kehittävät pelinsä itse ohjelmoimalla – kuitenkin käyttäen valmista ohjelmointikirjastoa. Kun kysymyksessä on pelien kehittämisen lisäksi ohjelmoinnin alkeiden opettelu, on mietittävä tarkasti, mitä käsitteitä ja kuinka paljon ohjelmoinnin teoriaa voidaan opettaa ilman, että oppilaat kokevat kurssin liian vaikeana ja työläänä. Täytyy muistaa, että kurssia markkinoitaessa painotettiin kurssin olevan kaikille avoin, myös niille, jotka eivät ole aikaisemmin ohjelmoineet lainkaan.

5.10 Oppilaat sovellusten tuottajina – oppiminen ohjelmoinnin avulla

Multimedia, WWW ja tietokonepelit ovat vakiinnuttaneet asemansa niin opiskelussa kuin vapaa-aikana. Näitä välineitä voidaan hyödyntää oppimisessa. Yksi tapa tieto- ja viestintätekniikan hyödyntämiseen opetuskäytössä on antaa oppilaiden suunnitella ohjelmasovelluksia, jonka kautta oppilaat oppivat myös käsillä olevaa sisältöä (Kafai 1995, 117).

Harelin (1991) sekä Kafain (1995) mukaan suunnittelemalla ja toteuttamalla jokin ohjelmasovellus tietystä aiheesta, voidaan tietyissä tilanteissa saavuttaa parempia oppimistuloksia, kuin ”perinteisellä luokkaopetuksella”. Kafai ym. (1997) tarkastelevat tutkimuksessaan 26:a peruskoulun 5–6 luokilla, iältään 10–12 vuotta olevaa oppilasta, jotka suunnittelivat ja toteuttivat Logo-ohjelmointiympäristössä interaktiivisen multimediasovelluksen nuoremmille oppilaille. Sovelluksen aiheena oli tähtitiede. Oppilailla oli yhteensä 46 tuntia aikaa projektia varten, josta 23 tuntia käytettiin ohjelmointiin. Oppilaiden tavoissa luoda sovellusta oli yllättävän suuria eroja – toiset käyttivät huomattavasti enemmän aikaa sisällön tuottamiseen ja siihen, kuinka asiat voitaisiin parhaalla mahdollisella tavalla esittää kohdeyleisölle.

Toisaalta suunnittelutehtävä oli joillekin niin vaikea, että sovelluksen sisältö oli lopulta köykäinen, ja oppilas pyrki kasvattamaan näyttöjen määrää tekemällä esimerkiksi drilliharjoituksia, kuten kyselyjä ja tietovisoja, jotka koostuvat vain irrallisista faktoista ilman sen kummempaa syvempää sisältöä (Kafai ym. 1997, 123–125). Kafai ym. ehdottavatkin, että ratkaisu voisi olla se, että oppilaille tarjotaan työskentelyvaiheessa

enemmän tukea työskentelyyn, jotta aihesisällön oppiminen olisi jokaisen oppilaan kohdalla mahdollisimman tehokasta.

Palataan vielä hetkeksi matematiikan ja ohjelmoinnin väliseen yhteyteen. Kafain ym. (1998, 180) mukaan tietokonepelien suunnittelu ja tuottaminen tarjoaa niin oppilaille kuin opettajillekin väylän matemaattisten käsitteiden hahmottamiseen, sekä osallistua jatkuvaan itsearvioon ja reflektioon matemaattisten käsitteiden oppimisessa.

6 Peliohjelmointikurssin suunnittelu

Tässä luvussa esitellään lyhyesti kurssia varten tehdyt toimenpiteet, valmistelut ja suunnitelmat. Kursseja pidettiin kaksi kappaletta, jotka olivat suunnitelmien ja valmisteluiden osalta likimain samanlaisia. Mikäli eroja oli, ne on erikseen mainittu.

6.1 Ennakkomarkkinointi

Kurssin mainostaminen aloitettiin huhtikuussa 2009. Pääasiallisena markkinointiväylänä päätettiin käyttää Jyvässeudun yläkoulujen matemaattisten aineiden lehtoreita. Näin tavoitettiin iso osa kohderyhmään kuuluneista nuorista Jyväskylän alueelta. Toisaalta oppilaat saivat viestin heille tutuilta henkilöiltä, millä pyrittiin madaltamaan kynnystä ottaa asiasta selvää ja ilmoittautua kurssille. Käytännössä kouluille vietiin tai postitettiin painettuja mainoslehtisiä jaettavaksi oppilaille (liite 1), opettajalle saatekirje (liite 2), missä annettiin ohjeet kurssin mainostamisesta oppilaille, sekä saatekirjeen liiteosa (liite 3), missä opettajille selvitettiin kurssin tarkoitus ja tavoitteet tarkemmin.

Markkinoinnissa pyrittiin tietoisesti välttämään sanan *ohjelmointi* liikaa painottamista, koska se saattaa vähentää kiinnostusta, erityisesti tyttöjen kohdalla (esim. Rasku & Kuukkanen 2004, 19). Sen sijaan puhuttiin pelin tekemisestä, ja korostettiin ennemminkin, ettei aikaisempi ohjelmointikokemus ole tarpeen, ja että vaikka pelinkehitys vaatiikin normaalisti paljon ohjelmakoodirivejä, on Jypeli-kirjaston avulla yksinkertaisten pelien ohjelmointi helpompaa ja nopeampaa. Kohdemarkkinoinnin seurauksena kurssille saatiin vajaa viisikymmentä ilmoittautumista. Mainoksia jaettiin kouluille neljä tuhatta kappaletta, joten *ilmoittautumisia-per-jaettu mainos* -suhdeluku jäi noin yhteen prosenttiin.

Kesäkuun alussa Jyväskylän yliopisto julkaisi kurssista tiedotteen (liite 4), minkä seurauksena useat tietotekniikkalehdet julkaisivat jutun internet-sivuillaan. Lisäksi Radio Keski-Suomi teki kaksi juttua kurssista ja myös Keskisuomalainen julkaisi jutun painetussa lehdessä.

6.2 Osaamisen kehittyminen

Oppilailta ei vaadittu esitietoina ohjelmointikokemusta, ja tietokoneen käyttötaidosta riitti perusteet. Kurssin tavoitteena oli, että jakson päätteeksi jokainen oppilas saa mukaansa

tuotteensa, valmiin tietokonepelin. Halutessaan hänellä oli mahdollisuus jatkaa pelin tekemistä ja ohjelmointiharrastusta kotitietokoneella, sillä jokaiselle oppilaalle annettiin ilmainen latauskoodi samoihin ohjelmistoihin, kun mitä kurssilla käytettiin.

Oppilaan osaamisen kehittymiselle asetettiin seuraavia tavoitteita:

1. Kurssin aikana oppilaan osaaminen kehittyy mm. seuraavilla perusopetuksen opetussuunnitelmaan kuuluvilla osa-alueilla (ja muilla alueilla):
   - ongelmanratkaisu
   - syy-seuraussuhteiden hahmottaminen ja kappaleiden väliset riippuvuudet
   - fysiikan ilmiöiden kuten maan vetovoima ja kitka, voimista aiheutuvat liike- ja tasapainoilmiöt, liikkeen ja voimien vaikutus kappaleeseen
   - yhtälöt ja niiden ratkaiseminen
   - suureet, kuten aika, matka, nopeus, kiihtyvyys ja voima
   - geometriset käsitteet: suora, skaalaukset, muodot
   - asteet ja radiaanit
   - vektorit ja niiden esittäminen lukuparina tai napakoordinaatistossa
   - todennäköisyys ja satunnaisuus
   - lukuparin käsite.

2. Kurssille osallistumisen jälkeen oppilas
   - tietää pelisuunnittelun vaiheita ja kyetään suunnittelemaan pelejä
   - osaa käyttää ohjelmien kehitysympäristöä ohjelmien luomiseen ja ohjelmien kääntämiseen
   - hahmottaa algoritmisen ajattelun perusideaa ja tiedetään, mikä algoritmi on
   - tuntee C#-ohjelmointikielen perussyntaksia sekä hieman olioista, ohjausrakenteista ja aliohjelmista
   - osaa lisätä omia kuvia ja ääniä peliin
   - osaa asentaa kotikoneelle tarvittavat ohjelmat, jotta voi jatkaa ohjelmointia kurssin jälkeen

- osaa etsiä itsenäisesti apua ohjeista ja netistä.

3. Kurssille osallistumisen tulisi tukea lisäksi seuraavia asioita:
   - mukava kokemus
   - positiivinen kuva tietotekniikasta ja ohjelmoinnista
   - kiinnostuksen herättäminen tietotekniikkaa ja sen opiskelua kohtaan
   - innostaa jatkamaan pelien tekemistä kurssin jälkeenkin.

Edelleen tavoitteena on kasvattaa nuorten kiinnostusta luonnontieteen alojen koulutusohjelmia kohtaan sekä tarjota ajankohtaista tietoa Jyväskylän yliopiston tarjoamista koulutusmahdollisuuksista luonnontieteellisillä aloilla.

Tavoitteiden toteutumista analysoidaan tämän työn pohdintaosuudessa (luku 9).

6.3 Työkalujen valinta

Työkalun valinnassa päädyttiin Microsoft Visual Studio 2008 -ohjelmointiympäristöön, jossa ohjelmointikieleksi valittiin C#. Kyseiselle ohjelmointikielelle oli kehitetty peliohjelmointiin keskittyvä XNA Game Studio 3.1 -ohjelmointikirjasto, jota edelleen kehittämällä saatiin keväällä 2009 valmiiksi **Jypeli -ohjelmointikirjasto**. XNA:n lisäksi Jypeliin liitettiin fysiikkakirjasto Physics2D.Net sekä matematiikkakirjasto AdvanceMath. Seuraavaksi käydään lyhyesti läpi perustelut valituille työkaluille. Nimessä Jypeli "Jy" viittaa Jyväskylän yliopistoon.

Lähtökohtaisesti työkalujen valinnassa kriteereinä olivat ilmaisuus, uudelleenkäytettävyys, kytkennät ”oikeisiin” ohjelmistoprojekteihin sekä helppokäyttöisyys.

6.3.1 Ohjelmointikieli: C#

C# on (lausutaan englanniksi *c sharp*) on vuonna 2000 julkaistu Microsoftin kehittämä ohjelmointikieli. C# on syntaksiltaan hyvin ”javamainen”, ja kielten onkin sanottu jopa kopioineen toinen toisiaan niin ominaisuuksiensa kuin syntaksin suhteen (Kreft & Langer 2003). Kielen kehitti Anders Hejlsberg, jonka tavoitteena oli yhdistää C#:ssa Javan

helppokäyttöisyys ja C++:n tehokkuus (T10 Media 2006), joskin tehokkuusväitteen todentaminen voi olla haasteellista (Thomsen 2009, 15). ECMA-standardiin kirjatun lausunnon mukaan (ECMA International 2006) kielen suunnittelutavoitteita olivat yksinkertaisuus, modernius, yleiskäyttöisyys ja olioperustaisuus. Kieli sisältää muun muassa vahvan tyypityksen, taulukon rajojen tarkistuksen ja automaattisen roskienkeruun, joita voidaan pitää modernin ohjelmointikielen perusvaatimuksina.

Kielen laaja ominaisuuksien kirjo ei kuitenkaan ole tärkein tekijä valittaessa ohjelmointikieltä alkeisohjelmoinnin kursseille, vaan tärkeämpiä ovat mm. kielen soveltuvuus didaktisesta perspektiivistä, oppimateriaalin saatavuus sekä kustannukset (Roine 2007, 28–35). Alkeisohjelmointiin sopivan kielen etsimisestä ja löytämisestä on tehty paljon tutkimusta, ja Roineen (2007, 78) mukaan perusopetukseen soveltuvia ohjelmointikieliä ovat Alice ja Visual Basic .Net. Roineen mukaan C# ei päässyt listalle, koska silloinen olemassa oleva oppimateriaali oli vähäistä. Kolmessa vuodessa tilanne on kuitenkin muuttunut sikäli, että kurssikirjaksi sopivia oppikirjoja on jonkin verran saatavilla – lähinnä englanniksi, mutta myös suomeksi. Käytännössä oman oppimateriaalin kirjoittaminen oli kuitenkin välttämättömyys, ja niinpä tätä kurssia varten kehitettiin satoja sivuja oppimateriaalia sekä lukuisia esimerkkipelejä ja tutoriaaleja, jotka osaltaan madaltavat oppimisen kynnystä. Visual Basic .Net ja C# ovat myös syntaktisesti ja semanttisesti melko lähellä toisiaan, joten myös Roineen selvitys vahvistaa C#:n valintaa ensimmäiseksi kieleksi .Net-perheen jäsenenä.

Ehkäpä edellä mainittujen vaatimusten lisäksi vielä tärkeämpi kriteeri C#:n valinnalle verrattuna esimerkiksi Aliceen, Kielixiin tai muihin erityisesti alkeisohjelmointiin keskittyneisiin ohjelmointikieliin oli se, että C# on yksinkertaisuudestaan huolimatta täysiverinen sovellusohjelmointikieli, joten kurssin ulkopuolisten ”todellisten” ohjelmointiongelmien ratkaiseminen mielekkäästi on mahdollista, toisin kuin Alicella tai Kielixillä. C# elää kirjoitushetkellä versionumeroa 3.0, joten kieli on myös kehittynyt pitkälle ensimmäisestä julkaisusta. C#:lla on hyvin laaja käyttäjäjoukko, joten vastauksia käytännön ohjelmointiongelmiin löytää helposti Internetistä. Microsoftin Developer Networkissa saatavilla olevat C#-dokumentaatiot ja referenssit ovat myös erittäin kattavat.

C# on saanut mm. ISO-standardin vuonna 2003 (ISO 27230:2003). Kirjoitushetkellä (maaliskuu 2010) uusin ja kurssilla käytössä ollut versio oli 3.0, joskin 4.0-version julkaisun oli tarkoitus tapahtua aivan lähiaikoina.

6.3.2 Kehitysympäristö: Microsoft Visual Studio

Kehitysympäristöksi valittiin Microsoft Visual Studio (myöhemmin VS), jonka uusin ja kurssilla käytetty versio oli 2008. Ohjelmointikielen valinnan jälkeen kehitysympäristön valinta oli selvä, sillä koska työskentely tapahtui Jyväskylän yliopiston tietokoneilla Windows-ympäristössä, ei esimerkiksi Mono-ympäristö (Linux, Mac OS X) tullut kysymykseen. Borland on julkaissut oman maksuttoman kehitysympäristönsä Windowsille, C# Builderin, jo vuonna 2003, mutta ympäristö ei ole ollut tuettuna vuoden 2006 jälkeen. Näin ollen se ei myöskään ollut yhteensopiva C# 3.0 -version kanssa, eikä soveltunut kurssin kehitysympäristöksi.

VS:sta oli kirjoitushetkellä saatavilla kaksi erilaista jakelupakettia: ilmainen Express Edition -versio, sekä kaupallinen Pro-versio. Microsoft sponsoroi kurssia antamalla Pro-version veloituksetta kaikkien kurssilaisten sekä kurssin opettajien ja ohjaajien käyttöön.

6.3.3 Luokkakirjasto: XNA Game Studio

XNA Framework (**X**NA's **N**ot **A**cronymed, myöhemmin lyhyesti XNA) on Microsoftin kehittämä peliohjelmointiin tarkoitettu ohjelmointikirjasto. XNA-ohjelmia voidaan ohjelmoida .Net-yhteensopivilla kielillä, kuten C# – tosin käytännössä, ainakin kirjoitushetkellä, ainoa tuettu ohjelmointikieli on C#. XNA-ohjelmien tekemistä varten on olemassa laajennus Visual Studioon, XNA Game Studio, joka on maksuton ja vapaasti ladattavissa Microsoftin internet-sivuilta. Kirjoitushetkellä ja näillä kursseilla käytössä oleva versio oli 3.1.

Microsoftin mukaan XNA:lla on pyritty houkuttelemaan nimenomaan aloittelevia ohjelmoijia tekemällä yksinkertaisten 2D-pelien ohjelmoinnista helppoa (Microsoft 2004).

XNA:lla tehdyt pelit toimivat ainakin Windows-tietokoneissa ja XBox 360 -pelikonsolissa, mutta Microsoftin annettua XNA Frameworkin vapaampaan käyttöön, on muillekin käyttöjärjestelmille kehitetty vastaavanlaisia kirjastoja. Niinpä XNA:lla

kirjoitettujen pelien ajaminen esimerkiksi OS X- tai Linux-käyttöjärjestelmissä Mono-ympäristössä on mahdollista tietyin edellytyksin.

Koska vastaavia kirjastoja C#:lle ei kurssin järjestämisen aikaan ollut olemassa, oli XNA luonnollinen valinta aloittaa Jypeli-ohjelmointikirjaston kehittäminen.

6.3.4 Fysiikka- ja matematiikkakirjastot: Physics2D.Net ja AdvanceMath

XNA ei itsessään sisällä fysiikkaa, joten jonkin fysiikkakirjaston liittäminen XNA:han katsottiin mielekkääksi. Tavoitteena oli löytää helppokäyttöinen vapaan lähdekoodin fysiikkakirjasto, mikä sopisi kaksiulotteisten pelien tekemiseen. C#/XNA-ympäristöön sopivia fysiikkakirjastoja löydettiinkin useita, esimerkiksi

- Farseer
- JigLibX
- Bullet
- Newton
- Oops!
- Bepu physics
- Jello Physics

sekä sittemmin valittu Physics2D.Net. Tämän kirjaston valintaperusteena käytettiin sen soveltuvuutta erityisesti 2D-peleihin, ja se oli suunniteltu helppokäyttöiseksi ja oli toteutukseltaan riittävän kompakti. Lisäksi sen fysiikkamallinnus oli riittävän yksinkertaista, jolloin pelit toimisivat myös hieman heikommilla tehoilla varustetuissa tietokoneissa.

Physics2D.NET oli riippuvainen AdvanceMath-matematiikkakirjastosta, joten se liitettiin myös mukaan Jypeli-kirjastoon. AdvanceMath on erityisesti C#:lle kehitetty ja optimoitu matematiikkakirjasto.

6.3.5 Jypeli – ”Kirjastojen kirjasto”

Vaikka XNA oli sinänsä erityisesti peliohjelmointia varten kehitetty ohjelmointikirjasto, se sisälsi tämänmuotoisen kurssin kannalta muutamia vakavia puutteita, joten päädyttiin

ratkaisuun, että yhdistetään valitut kirjastot. Jypeli-ohjelmointikirjasto pohjautui siis XNA-kirjastoon, sekä valittuihin fysiikka- ja matematiikkakirjastoihin, jonka seurauksena pelilogiikan kirjoittaminen saatiin yksinkertaistettua ja ensimmäisen pelin tekemiseen vaaditut koodirivit olivat vielä mielekkyyden rajoissa.

6.4 Kurssin tekninen valmistelu

Kurssin tekninen valmistelu käsitti **Jypeli-ohjelmointikirjaston** kehittämistyön. Tarkoituksena oli kirjoittaa omia olioluokkia yhdistellen XNA:ta, valittua fysiikka- ja matematiikkakirjastoa, tehdä esimerkkipelejä sekä kirjoittaa ohjeita Jypeli-kirjaston käyttöä varten ja muuta oppimateriaalia kurssille.

Helposti omaksuttavan kirjaston avulla ohjelmoinnin kokonaisuuksien ymmärtäminen on helpompaa. Tämä saavutettiin muun muassa tekemällä luokkien nimistä helposti ymmärrettäviä, tarvittaessa käärimällä (*wrap*) fysiikkakirjaston ja matematiikkakirjaston luokkia uusiksi luokiksi sekä kirjoittamalla dokumentaatiot suomen kielellä. Jypeli-kirjaston avulla oppilaan ei tarvitse välttämättä ymmärtää ohjelmistojen yksityiskohtia, jotka eivät ole pelin suunnittelun ja tekemisen kannalta keskeisiä. Toisaalta ohjelmointikirjastosta ei myöskään poistettu sellaisia osia, jotka katsottiin olevan ”yleisesti hyväksyttyjä”, ja rakenteiseen ohjelmoinnin kannalta yleisesti kriittisen tärkeitä, kuten tyypitys, symboliset muuttujat, aliohjelmat ja niin edelleen.

Niinpä Jypeliä kehitettiin keväästä 2009 lähtien. Ohjelmointikirjaston kehittämiseen osallistui vuonna 2009 noin neljän kuukauden ajan kolme ohjelmoijaa.

6.5 Aikataulu ja sisältö

Kurssin ohjelma koostui luennoista, harjoitustehtävien tekemisestä, oman pelin suunnittelusta ja toteuttamisesta, oman pelin esittelemisestä sekä palautteen antamisesta.

Kunkin luennon pituus oli 30–45 minuuttia. Luentojen tarkoitus oli johdatella algoritmiseen ajatteluun sekä käsitellä ohjelmoinnin perusrakenteita ja ohjelmointikielen syntaksia. Luentoja oli enintään kaksi kappaletta päivässä, yhteensä viikon aikana noin kymmenen tuntia.

Harjoitustehtäviä tehtiin ennen oman pelin suunnittelun aloittamista. Harjoitustehtävien tavoitteena oli saada tuntumaa C#-kielen syntaksiin, ja tutustua ohjelmointikielen perusrakenteisiin. Harjoitustehtävät laadittiin siten, että oppilaiden tehtäviä varten tuottama koodi palveli myös hyvänä lähtöpisteenä myöhemmin tehtävää peliä varten. Harjoitustehtävien tekemistä varten varattu aika oli noin kuusi tuntia.

Oman pelin suunnittelu ja toteutus oli luonnollisesti kurssin odotetuin vaihe. Tähän vaiheeseen ajoitettiin noin 15 tuntia. Tarkoituksena oli, että jokainen oppilas kirjoittaa valitsemastaan peliaiheesta tarinan ja juonen (verbaalinen suunnittelu) ja piirtää kuvan, miltä valmis peli näyttäisi (visuaalinen suunnittelu). Mikäli aikaa jäi edellisten tehtävien jälkeen, oppilaat saivat suunnitella ja piirtää oman pelihahmon, joka myöhemmin siirrettiin valmiiseen peliin.

Kurssin viimeiset kaksi tuntia varattiin sille, että jokainen esitteli oman pelituotoksensa muille. Yleisöllä oli mahdollisuus esittää kysymyksiä oppilaan tekemästä pelistä. Oppilaat saivat myös esitellä pelin ohjelmakoodista sellaisia osioita, joissa he mielestään olivat parhaiten onnistuneet.

Kurssin ajankäyttösuunnitelma oli seuraavanlainen.

- Maanantai (5 tuntia)
    1. Työkaluihin tutustuminen, järjestelyt (1 tunti)
    2. Pelisuunnittelu ja algoritminen ajattelu (1 tunti)
    3. Alkeisohjelmoinnin harjoituksia (1 tunti)
    4. Ensimmäisen pelin tekeminen, osa 1 (2 tuntia)
- Tiistai (5 tuntia)
    1. Aliohjelmat, muuttujat (1 tunti)
    2. For- ja if-lauseet (1 tunti)
    3. Ensimmäisen pelin tekeminen, osa 2 (2 tuntia)
    4. Oman pelin suunnittelu, osa 1 (1 tunti)
- Keskiviikko (5 tuntia)
    1. Ohjelmointiympäristön tehokas käyttö (0,5 tuntia)

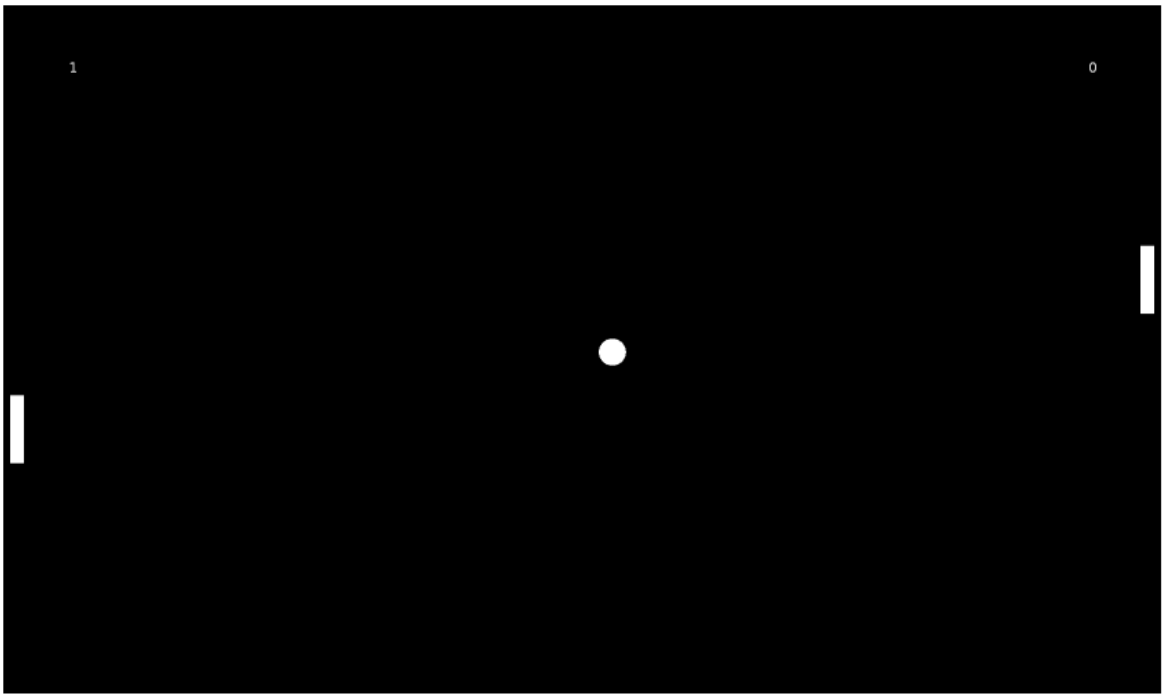
2. Vektoreiden käyttö omassa ohjelmassa (0,5 tuntia)
3. Törmäysten käsittely (0,5 tuntia)
4. Oman pelin suunnittelu, osa 2 (1 tunti)
5. Oman pelin tekeminen (2,5 tuntia)

- Torstai (5 tuntia)
    1. XBox-ohjaintapahtumien lisääminen peliin, kertausta (0,5 tuntia)
    2. Oman pelin tekeminen (4,5 tuntia)
- Perjantai (5 tuntia)
    1. Oman pelin tekeminen ja pelin testaus (3 tuntia)
    2. Pelien esittely (2 tuntia)
    3. Palautteen antaminen

6.6 Ensimmäisen harjoituspelin tekeminen

Erityisen suuri strateginen painoarvo kurssilla asetettiin ensimmäisten harjoitusten tekemiselle. Kuten aiemmin todettiin, kun jotain tehdään ensimmäisen kerran, sillä on erityisen suuri merkitys niin mielikuvien muodostumisessa kuin oppimismielessäkin.

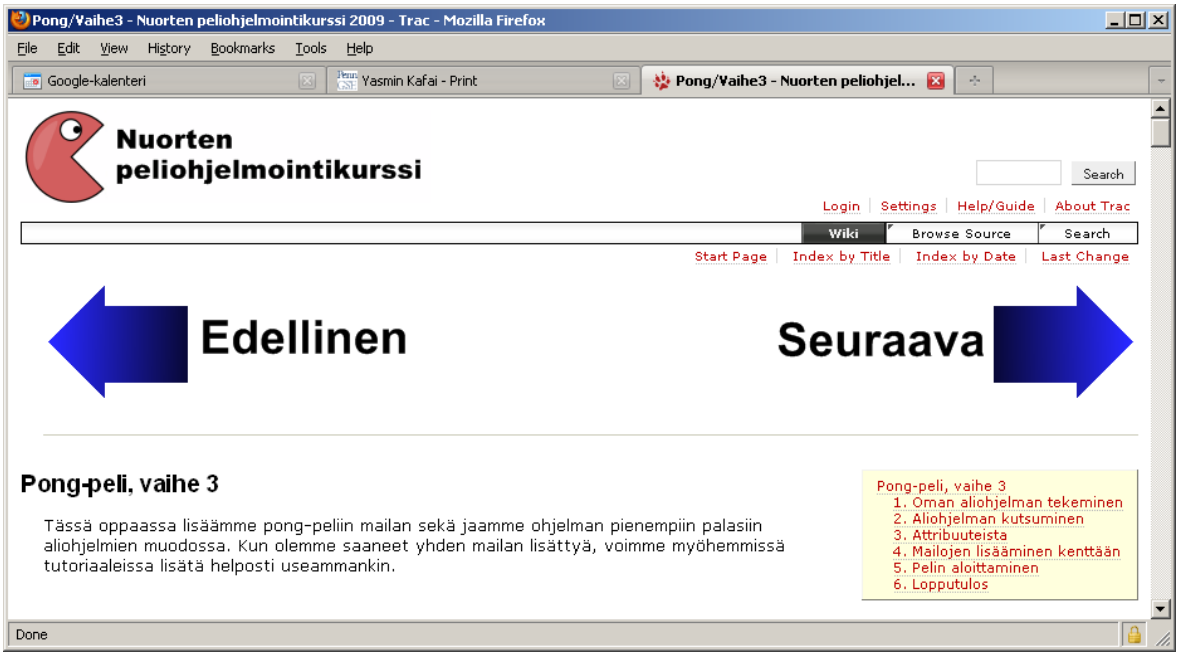
Kahden ensimmäisen päivän aikana oppilaat tekivät tutun pelihallipelin, *Pong*, missä kaksi pelaajaa käyttää mailoja muuttaakseen pallon suuntaa (kuva 11). Pelaajat pyrkivät saamaan pallon toisen pelaajan selän taakse, jolloin toinen pelaaja saa pisteen.



Kuva 11. Pong-pelissä pyritään saamaan pallo pelaajan ”selän” taakse, mistä saa pisteitä.

Ensimmäisen pelin tarkoituksena oli antaa oppilaille karkea käsitys siitä, millaisia pelejä Jypelillä on mahdollista tehdä, tutustua pelin tekemisen yhteydessä käytettäviin työkaluihin, sekä tärkeimpänä päästä heti ”asiaan”, eli tekemään ohjatusti oikea peli.

Pong-peliä varten oppilaille tehtiin yksityiskohtainen WWW-tutoriaali (kuva 12), jossa koodiesimerkkien lisäksi kuvien ja kerronnan kautta opetettiin pelin tekeminen vaihe vaiheelta. Oppilaat lukivat pelin tekemisen ohjeet tutoriaalista, ja saivat tarvittaessa ohjaajilta tukea tekemiseen. Pelin tekemisen aikana tulivat käytännön kautta tutuksi mm. muuttujat, aliohjelmat ja parametrien välitys.



Kuva 12. Oppilaat tekivät ensimmäisen pelinsä WWW-tutoriaalin ohjeiden mukaan.

6.7 Oman pelin tekeminen

Oman pelin suunnittelu ja tekeminen oli luonnollisesti kurssin tärkein ja kiinnostavin osa – sitä vartenhan oppilaat olivat kurssille tulleet. Pelin suunnittelussa ja toteutuksessa noudatettiin väljästi mukaillen tähän kurssiin sopivaa Raskun ja Kuukkasen (2004, 12) tulkintaa Scott Palmerin (1996, 15) pelinkehitysprosessimallista, jota seuraamalla voidaan Palmerin mukaan varmistaa peliprojektin onnistuminen.

Askeleet pelinkehitysprosessissa olivat

1. **Pelin suunnittelu**. Pelin tekemisen ensimmäinen tehtävä on suunnittelu, missä päätetään pelin idea, juoni sekä pelin tyyppi (esimerkiksi tasohyppely, seikkailu, urheilu, jne.). Tässä vaiheessa ei paneuduta vielä tekniseen toteutukseen lainkaan, vaan enemmänkin siihen, mitä erilaisia toimijoita peliin liittyy, mitkä ovat niiden vuorovaikutussuhteet, millaisessa miljöössä peli etenee, miten pelissä voi ”hävitä” tai voittaa”, ja niin edelleen.
2. **Pelin mallinnus.** Mallinnusvaiheessa peliä pilkotaan pienempiin osiin. Pelin toimijat ja toiminnot nimetään hyvien nimeämiskäytäntöjen mukaisesti, ja

objektien toiminta ja tarkoitus määritellään. Mikäli mallinnuksessa esiintyy ristiriitoja ja epäjohdonmukaisuuksia, ne pyritään karsimaan heti pois, sillä ne saattavat kostautua myöhemmin suurina muutostöinä pelin peruslogiikkaan.

3. **Pelin rakentaminen**. Pelin toteuttaminen aloitetaan. Vaiheessa 2 määritellyt toimijat lisätään peliin ja niiden vuorovaikutussuhteet toteutetaan. Kehitys etenee mahdollisimman pienin askelin siten, että, pelistä on kokoajan toimiva prototyyppi käytössä. Tässä vaiheessa ei vielä keskitytä ulkoasun hiomiseen, vaan keskitytään toimintoihin ja sisältöön.
4. **Musiikki, ääniefektit ja grafiikka**. Kun pelin rakentaminen alkaa olla valmis, aloitetaan viimeistely, eli visuaalisen ulkoasun hiominen sekä musiikin ja ääniefektien lisäys. Näitäkin ilman peli toimii, mutta nämä antavat pelille ”viimeisen silauksen”, eli tekevät siitä persoonallisen ja houkuttelevan näköisen. Koska musiikin ja grafiikan tekemiseen tulee käytettyä herkästi liian paljon aikaa, on hyvä määritellä itselle sopiva taso etukäteen.
5. **Pelin testaus ja levitys**. Pelin rakentamisen jälkeen peli ei vielä ole valmis, vaan se vaatii testausta, että kaikki pelin mahdolliset skenaariot toimivat, myös sellaiset syötteet ja toimintatavat, joita pelin tekijä ei toivo käyttäjän tekevän. Testaaja voi olla myös jokin muu kuin pelin tekijä. Jos ja kun virheitä löytyy, ne korjataan ja testaamista jatketaan, kunnes voidaan sanoa pelin olevan toimiva. Mitä paremmin ja huolellisemmin testaus tehdään, sitä todennäköisempää on, että pelikokemus on parempi.

Vaiheita 1 ja 2 varten jokainen oppilas teki omasta pelistään suunnitelman, sekä toteutuksen suunnitelman ja pelin visuaalisen hahmotelman. Tämän tavoitteena oli varmistaa, että oppilaat ovat ymmärtäneet Jypelin mahdollisuudet ja rajoitteet, ja että pelien toteuttaminen käytettävissä olevassa ajassa on mahdollista. Suunnitelma tuli hyväksyttää ohjaajalla, joka varmisti, että suunniteltu harjoitustyö voidaan toteuttaa käytettävissä olevassa ajassa.

Pelisuunnitteluun varattiin aikaa noin kaksi tuntia. Ennen suunnitelman tekemistä oppilaille annettiin malli siitä, millainen suunnitelman rungon tulisi olla, ja mitä asioita ainakin sen tulisi sisältää. Suunnitelmaan sisältyi vähintään tekijöiden tietojen lisäksi pelin

taustatarina, kuvaus pelin teemasta, pelin idea ja tavoitteet (miten pelissä edetään ja miten kehittyy), sekä hahmotelma pelistä kynällä ja paperilla. Lisäksi suunnitelman tuli sisältää toteutuksen suunnitelma, siten että pelin toteuttamiseen liittyvät aktiviteetit jaetaan vaiheisiin, ja arvioidaan niihin tarvittava työmäärä. Vaiheiden kuvausten tuli olla mahdollisimman tarkkoja, jotta niihin liittyvää työmäärä olisi helpompi arvioida. Liitteessä 5 on malli suunnitelmasta – oppilaiden itse tekemiä suunnitelmia esitellään hieman myöhemmin.

7 Tutkimuksen toteuttaminen

Tässä työssä tutkittiin peliohjelmointikurssille osallistuneiden oppilaiden taustoja, kurssin innostavuutta luonnontieteiden opiskelua kohtaan, sekä pyrittiin löytämään kurssille osallistuneista oppilaista sellaisia ryhmiä, joiden perusteella voitaisiin esittää millainen oli ”tyypillinen” kurssille osallistunut oppilas (tai ”tyypilliset” oppilaat). Tutkimuksen kohteena oli 13–16-vuotiaille suunnattu alkeisohjelmointikurssi, jossa tavoitteena oli tehdä viikon aikana oma tietokonepeli. Kurssin sisältöä ja rakennetta on kuvattu tarkemmin luvussa 6.

Tässä luvussa on kuvattu aluksi tutkimusongelmat. Sen jälkeen on kerrottu tutkimukseen valituista tutkimusmenetelmistä, ja eritelty, miten tutkimusmenetelmiä sovellettiin kurssien aikana. Viimeisenä esitellään tutkimuksessa käytettyjä tilastollisia analyysejä.

7.1 Tutkimusongelmat

Tässä alaluvussa esitellään tutkimusongelmat teemoittain ryhmiteltynä. Ensimmäiseksi teemaksi valittiin pelikeskeisen ohjelmointikurssin innostavuus. Toiseksi teemaksi valittiin pelikurssin oppilastyypit. Tutkimusongelmat sijoitettiin sitten näiden teemojen alle. Tutkimuskysymykset on numeroitu juoksevasti siten, että pääkysymykset ja alakysymykset juoksevat samassa numeroinnissa.

Pelikeskeisen ohjelmointikurssin innostavuus

Kurssin innostavuutta tarkasteltiin monella tasolla.

Korkeakoulujen perinteinen lähestymistapa ohjelmointiin on tiukan aksiomaattinen: ensin on opeteltava tarkasti aihetta käsittelevä teoreettinen tausta, jotta voidaan tuottaa itse mitään "omaa". Toisin sanoen, pitää tuntea tarkasti kaikki mahdolliset rajat, jotta voi edes harkita niiden rikkomista. Suomalaisissa kouluissa esimerkiksi kielten opiskelussa ilmiö on tuttu. Oppilaita on vaikea saada keskustelemaan keskenään, elleivät he mielestään omaa täydellistä kieliopin ja ääntämisen hallintaa. Keskustelu jää vähäiseksi ja lopulta kielen käyttämisen taitokin hiipuu.

NPO-kurssin lähestymistapa ohjelmointiin oli hyvin erilainen kuin korkeakouluissa yleensä. NPO-kurssin lähtökohta on itse asiassa täysin päinvastainen. Tarkoituksena oli, että oppimista tapahtuisi itse kokeilemalla ja tutkimalla (tutkiva oppiminen, ks. luku 5.3.4) sekä itse tekemällä koodia (tekemällä oppiminen, ks. luku 5.3.2) esimerkkikoodin ja -pelien avulla (mallioppiminen, ks. luku 5.3.6). Lähestymistapa ohjelmointiaiheeseen oli nimenomaan pelin ja pelaamisen näkökulmasta, ja toisaalta tuominen näennäisen monimutkaiset ja vaikeat asiat kontekstissa, joka on nuorelle ominainen.

Muuttamalla oppilaat pelien käyttäjistä pelien tuottajiksi, toivotaan saavutettavan monia keskenään toisiinsa linkittyviä etuja niin teknologioiden oppimiseen kuin tieteen oppimisen tiimoilta. Ensinnäkin, oppilaille annettiin mahdollisuus ilmaista heille itselleen merkityksellisiä sisältöjä teknologian ilmaisukeinoilla. Toisaalta, ohjelmoinnin oppiminen voi tukea muidenkin kurssilla käsiteltyjen asioiden, kuten matematiikan ja fysiikan käsitteiden sekä luonnontieteisiin liittyvien ilmiöiden oppimista. Kurssin edetessä oli selvää, että oppilaiden toteuttaessaan omia pelejään, oli viittauksia ja yhteyksiä luonnontieteisiin otettava esille ja käytäntöön, sekä antaa jatkuvasti vihiä siitä, että luonnontieteet ovat läheisessä yhteydessä ohjelmointiin – myös peliohjelmointiin.

Tässä tutkimuksessa nouseekin aiheelliseksi kysyä:

1. Kuinka innostavan kokemuksen tietokonepelien suunnitteluun ja ohjelmointiin keskittyvä kurssi antaa nuorille?
2. Lisääkö yksittäinen peliohjelmointikurssi nuorten kiinnostusta sisällyttää matemaattis-luonnontieteellisiä tai tietoteknisiä oppiaineita opiskeluunsa?

Alakysymykset:

3. Kokevatko oppilaat pelikeskeisen ohjelmoinnin johdantokurssin kokonaisuutena mielekkäänä?
4. Mitkä taustatekijät vaikuttavat kurssille osallistuneiden kokonaistyytyväisyyteen?

Oppilastyypit peliohjelmointikurssilla

Ohjelmointi-sana sisältää monien mielessä jokseenkin kaksijakoisen latauksen. Ensinnäkin ohjelmointi on käsitteenä erittäin laaja, eikä nuorilla ole välttämättä käsitystä siitä, mitä ohjelmointi tarkalleen ottaen tarkoittaa. Tästä syystä kurssin markkinoinnissa painotettiin, ettei aikaisempaa ohjelmointikokemusta tarvitse olla kurssille tullessaan (toisin sanoen, ei tarvitse tietää *mitä ohjelmointi on*), vaan kurssille voi tulla ”takki auki”, kunhan ottaa mukaan oppimisen halun ja innokkaan mielen.

Näin ollen nousikin aiheelliseksi kysyä:

5. Millaisia nuoria peliohjelmointikurssille osallistui?

Alakysymykset

6. Onko heillä ennakkoasenteita luonnontieteiden opiskelua kohtaan ja muuttuiko se kurssin aikana?
7. Löytyykö osallistujien joukosta ryhmiä, joista kukin kuvastaisi eräänlaisia ”tyyppioppilaita”?

7.2 Tutkimukseen valitut tutkimusmenetelmät

Tutkimusmenetelmät jaetaan tavallisesti määrällisiin ja laadullisiin. Tässä luvussa esitellään lyhyesti nämä molemmat menetelmäkategoriat. Koska tämän tutkimuksen pääpaino oli määrällisessä tutkimuksessa, se käydään hieman tarkemmin läpi. Määrällisen tutkimuksen tutkimusstrategioista esitellään survey-tutkimus (kyselytutkimus), koska se oli tässä tutkimuksessa suurimmassa roolissa.

Tässä tutkimuksessa yhdistettiin siis sekä määrällisen että laadullisen tutkimuksen menetelmiä. Tutkimusmenetelmien yhdistämistä kutsutaan triangulaatioksi tai metodien yhdistämiseksi ja sitä käytetään tutkimuksen validiuden parantamiseksi (Hirsjärvi ym. 2009, 233). Vaikka pääpaino oli määrällisessä tutkimuksessa, niin myös laadullista aineistoa saatiin – tosin suuri osa aineistosta rajattiin pois käsittelystä (ks. luku 2.5). Kirjallinen, määrällisellä menetelmällä käsiteltävä havaintoaineisto oli kuitenkin lopulta melko pieni (alkukyselyn n=45, loppukyselyn n=41), eikä välttämättä anna luotettavasti

yleistettäviä tuloksia, ja näin ollen tutkimustapa oli lähinnä laadullinen – joskin aineisto kerättiin pääasiassa määrällisen tutkimusmenetelmän keinoilla.

Tutkimus toteutettiin osin survey-tutkimuksena, missä oppilaiden palautteita ja mielipiteitä kerättiin kyselylomakkeilla. Kyselylomake soveltui tähän tutkimukseen, koska haluttiin selvittää melko laajan joukon mielipiteitä ja tutkia niitä määrällisillä analysointimenetelmillä. Tutkimuksen laajentaminen suuremmallekin joukolle olisi samanlaisella menetelmällä ollut mahdollista. Kaikki asteikolliset kysymykset analysoitiin määrällisesti. Asteikkona käytettiin Likertin viisiportaista asteikkoa. Kyselylomakkeissa esitettiin muutamia avoimia kysymyksiä, joista osa analysoitiin määrällisesti ja osa laadullisesti. Kuten edellä mainittiin, laadullisen analyysin tavoitteena oli antaa tutkimukselle parempi validiteetti.

On huomattava, ettei tutkimuksen tarkoituksena ollut tuottaa peliohjelmointikurssista tieteellistä mallia, vaan antaa kokonaisvaltainen kuva siitä, mitä järjestelyjä, toimenpiteitä ja valmisteluja kurssi vaatii, sekä kuinka mielekkäänä ja innostavana oppilas voi kokea alkeisohjelmointikurssin, missä on selkeästi asetettu tavoite – tehdä oma tietokonepeli, eli tavallaan valmis tuote, jonka voi sitten antaa muille pelattavaksi ja tutkittavaksi.

7.3 Analysointimenetelmät

Tässä tutkimuksessa tilastollisiksi merkitsevyysrajoiksi valittiin yleisesti tilastollisessa tutkimuksessa käytössä olevat rajat, jotka ovat mm. Heikkilän (1998, 185) mukaan

- 0,05 (5 %)
- 0,01 (1 %) ja
- 0,001 (0,1 %).

Näistä rajoista saadaan Heikkilän (1998, 186) mukaan seuraavat merkitsevyystasot

- tilastollisesti erittäin merkitsevä, jos $p \leq 0{,}001$ (symboli $***$)
- tilastollisesti merkitsevä, jos $0{,}001 < p \leq 0{,}01$ (symboli $**$)
- tilastollisesti melkein merkitsevä, jos $0{,}01 < p \leq 0{,}05$ (symboli $*$)

Käytännössä usein tulosta ilmoitettaessa ilmaistaan vain esimerkiksi ”tilastollisesti merkitsevä 5 % merkitsevyystasolla”. Mitä pienempi merkitsevyystaso on, sitä merkitsevämpi on myös tulos. Kaikki tässä raportissa tehdyt tilastolliset analyysit tehtiin SPSS 15.0 for Windows -ohjelmalla. SPSS-ohjelman tulosteissa tähtisymbolit ilmaisevat tilastollista merkitsevyyttä juuri kuten yllä on mainittu.

7.3.1 Korrelaatioanalyysi

Korrelatiivinen tutkimus soveltuu tilanteisiin, missä halutaan tietää jotain muuttujien välisten yhteyksien, eli korrelaatioiden, voimakkuudesta. Korrelatiiviset menetelmät soveltuvat käyttöön, kun muuttujat ovat luonteeltaan moniselitteisiä. Korrelatiivisten menetelmien ansiosta useampia muuttujia ja niiden korrelaatioita voidaan tutkia samanaikaisesti. Korrelatiivisten menetelmien etu on siinä, että ne antavat myös yhteyden voimakkuuden, eikä pelkästään tietoa siitä, onko asioiden välillä yhteyttä vai ei. (Anttila 1996, 178–179.)

Muuttujien välillä olevaa mahdollisen yhteyden voimakkuutta voidaan mitata Pearsonin korrelaatiokertoimella. Korrelaatiokerroin voi saada arvoja väliltä -1…1. Kertoimeen liittyy selitysaste, joka ilmoittaa, kuinka paljon korrelaatio selittää testattavien muuttujien vaihtelusta. Esimerkiksi korrelaatiokerroin 0,5 ilmoittaa toiseen potenssiin korotettuna selityskertoimen 0,25 – toisin sanoen korrelaatiokerroin 0,5 selittää 25 % muuttujien vaihteluista. Korrelaatiota käytetään melko usein ja etenkin silloin, kun analysoitavana on iso määrä muuttujia. Pearsonin korrelaatiokerroin mittaa muuttujien välillä olevaa lineaarista riippuvuutta. Toisin sanoen, jos muuttujien välillä oleva yhteys ei ole lineaarinen, niin Pearsonin korrelaatiokerroin ei anna oikeaa kuvaa muuttujien välisen yhteyden voimakkuudesta. (Holopainen ym. 2004, 173; Heikkilä 1998, 88.)

Korrelaatioanalyysillä haluttiin tässä tutkimuksessa selvittää, onko tiettyjen kysymysten vastauksilla merkitsevää yhteyttä muiden kysymysten vastausten kanssa. Tällä siis saatiin selville, korreloiko esimerkiksi vastaajan aikaisempi ohjelmointikokemus muiden kysymysten vastauksien kanssa, ja niin edelleen. Korrelaatioanalyysin avulla haettiin vastauksia siihen, mitkä taustatekijät vaikuttivat kurssille osallistuneiden kokonaistyytyväisyyteen (tutkimuskysymys 3) ja siihen, mitkä tekijät vaikuttavat nuoren

kiinnostuksen kasvuun (tai laskuun) IT- tai luonnontieteellisten alojen opiskelua kohtaan (tutkimuskysymys 2).

7.3.2 Tyypittely

Tyypittelyllä tarkoitetaan aineiston ryhmittelyä tyypeiksi. Aineistosta muodostetaan ryhmiä, jotka sisältävät samankaltaisia (esimerkiksi) tarinoita. Tyyppien tehtävä on tiivistää ja tyypillistää. Yhteen tyyppiin voidaan sisällyttää sellaista, mitä välttämättä ei ole yksittäisessä vastauksessa. Toisin sanoen, tyypit kuvaavat laajasti mutta tehokkaasti aineistoa. (Kemppainen 2010.)

Tyypittelyn tavoitteena ei ole vastaajien vaan heidän tarjoamansa informaatioaineksen tyypittely. Esimerkiksi haastatteluaineistosta voidaan etsiä tietyn tyyppisiä vastauksia, joilla on tiettyjä yhdistäviä elementtejä ja siten edustavat jotakin tyyppiä.

Kemppaisen (2010) mukaan tyyppejä ovat esimerkiksi

1. Autenttinen tyyppi, eli yhden vastauksen sisältävä tyyppi. Käytetään esimerkkinä laajemmasta aineiston osasta.

2. Yhdistetty tyyppi, eli mahdollisimman yleinen tyyppi. Tässä tyypissä on mukana niitä asioita, jotka esiintyvät mahdollisimman suuressa osassa vastauksista.

3. Mahdollisimman laaja tyyppi. Jotkut tyyppiin mukaan otetut piirteet esiintyvät vain yhdessä vastauksessa. Tämän tyypin on kuitenkin oltava sisäisesti looginen.

On huomattava, että myös epätyypillisiä tapauksia voidaan ja usein pitääkin analysoida. Esimerkiksi fenomenografiassa ollaan kiinnostuneita ristiriidoista ja erilaisuuksista nostamalla juuri epätyypillisiä asioita tutkimuksessa esiin (Rissanen 2006).

7.3.3 Klusterianalyysi

Alku- ja loppukyselyn vastauksille tehtiin klusterianalyysi. Klusterianalyysin avulla selvitettiin, löytyykö vastaajien joukosta sellaisia ryhmiä, joilla olisi yhtäläisyyksiä esimerkiksi sukupuolen, iän tai aiemman ohjelmointikokemuksen suhteen. Lisäksi klusteroinnin avulla pyrittiin löytämään oppilaista yhdistettyjä tyyppejä (ks. edellinen

alaluku), jotka edustaisivat mahdollisimman hyvin koko aineistoa. Tällä tavalla saataisiin selville, millaisia kurssille osallistuneet oppilaat tyypillisesti olivat.

8 Tutkimuksen tulokset ja analysointi

Kurssi 1 pidettiin heinäkuussa 2009, ja kurssilla aloitti 22 oppilasta. Kurssi 2 pidettiin elokuussa, ja kurssilla aloitti 23 oppilasta. Ensimmäisellä kurssilla ei ollut keskeyttäneitä, toisella kurssilla keskeytti kolme ennen kurssin loppua, ja lisäksi yksi sairastui niin, ettei voinut osallistua enää ensimmäisen päivän jälkeen. Alkukyselyissä on kyselyn yleisluontoisuuden vuoksi mukana myös keskeyttäneiden vastaukset, mutta tulosten analyysistä heidän vastauksensa on poistettu. Luonnollisesti loppukyselystäkään ei heidän vastauksiaan löydy.

8.1 Alkukysely (n=45)

Tässä luvussa esitellään alkukyselyyn vastanneiden oppilaiden vastausten koosteet. Kurssin alkukysely (liite 6) oli kummallakin kurssilla samanlainen. Alkukyselyn tavoitteena oli saada taustatietoa oppilaista, jonka avulla heitä voidaan myöhemmin ryhmitellä.

Kaaviot on tilan säästämisen vuoksi ja luettavuuden helpottamiseksi liitetty tutkielman loppuun, ja ne löytyvät liitteestä 7. Asteikollisissa kysymyksissä käytettiin ”normaalia” Likert-asteikkoa 1–5 (täysin eri mieltä, …, täysin samaa mieltä).

Ikä ja sukupuoli

Kurssin aloitti yhteensä 45 oppilasta, joista 7 oli tyttöjä ja 38 poikia. Keski-ikä oli lähes sama kummallakin kurssilla: 13,8 vuotta. Yhteensä kurssilaisten enemmistö (64,3 %, n=27) oli 13- tai 14-vuotiaita (ks. liite 7, kaavio 1-a). Huomautettakoon, että ensimmäisellä kurssilla tyypillisin ikä oli 14 vuotta, kun toisella eniten oli 13-vuotiaita. Kaksi oppilasta ei ilmoittanut ikäänsä ja yksi oppilas ilmoitti iäkseen 13,5 mikä tulkittiin 13:ksi.

Ristiintaulukoinnilla nähdään (taulukko 2), että tyttöjen ja poikien ikäjakaumissa ei ole suurta eroa – joskin tyttöjen määrä kurssilla oli kovin pieni (n=7).

Ikä vuosina * Sukupuoli Crosstabulation

Count

		Sukupuoli		
		Nainen	Mies	Total
Ikä vuosina	12	1	3	4
	13	2	14	16
	14	2	10	12
	15	1	5	6
	16	1	4	5
Total		7	36	43

Taulukko 2. Ikä ja sukupuoli ristiintaulukoituna.

Pelaaminen ja pelityypit

Kolmas ja neljäs kysymys käsittelivät pelaamista ja pelityyppejä. Vastausten perusteella kursseille valikoitui erittäin aktiivisia pelaajia: Yhtä lukuun ottamatta kaikki pelaavat tietokone- tai konsolipelejä vähintään muutamana päivänä viikossa, ja suurin osa (53,3 %, n=24) *joka päivä* (ks. liite 7, kaavio 2-a & 2-b).

Ristiintaulukoinnilla nähdään, että 14–15-vuotiaat olivat aktiivisimpia pelaajia, sillä noin 67 % heistä ilmoitti pelaavansa joka päivä (ks. liite 7, kaavio 2-c). 12-vuotiaat (25 %, n=4) pelasivat hieman muita vähemmän.

Neljännessä kysymyksessä oppilaat valitsivat pelityyppejä, joita he pelaavat. Mahdollisuus oli valita useita pelityyppejä. Taulukossa 3 on esitetty eri vastausvaihtoehdot frekvensseineen. Suosituimmat pelit on taulukossa lihavoituna.

		Responses		Percent of Cases
		N	Percent	
Minkä tyyppisiä pelejä pelaat?(a)	Auto- ja ajopelit	12	6,9%	26,7%
	Pulmapelit (Lemmings, Tetris, Portal jne.	16	9,1%	35,6%
	Räiskintäpelit (HalfLife, Counter Strike, Halo jne.)	26	**14,9%**	57,8%
	Roolipelit (RuneScape, EverQuest, World of Warcraft jne.)	23	13,1%	51,1%
	Seikkailupelit (RuneScape, EverQuest, World of Warcraft jne.)	26	**14,9%**	57,8%
	Strategiapelit (Command and Conquer, StarCraft jne.)	22	12,6%	48,9%
	Taistelu- tai tappelupelit (Mortal Kombat jne.)	10	5,7%	22,2%
	Taitopelit (biljardi, shakki, tammi, jne.)	8	4,6%	17,8%
	Tasohyppelypelit (Mario Bros jne.)	25	14,3%	55,6%
	Urheilupelit (NHL jne.)	7	4,0%	15,6%
Total		175	100,0%	388,9%

Taulukko 3. Oppilaiden pelaamat pelityypit.

Yksi oppilas antoi ”muu”-kenttään vapaan vastauksen ”simulaattori”. Kurssien välillä ei ollut mainittavaa eroa sen suhteen, millaisia pelejä oppilaat pelaavat.

Aikaisempi pelisuunnittelu- ja ohjelmointikokemus

Kysymys 4 oli muotoiltu siten, että oletko suunnitellut pelejä ”tietokoneella tai ilman tietokonetta”. Kysymyksessä ei siis rajoituttu pelkästään tietokone- tai konsolipeleihin, vaan kaikenlaisiin peleihin. Niinpä yhteensä 49 % (n=22) kertoi suunnitelleensa pelejä aikaisemmin, ja 51 % (n=23) ei ollut suunnitellut (ks. liite 7, kaavio 4-a). Ensimmäisen kurssin oppilaista pelisuunnittelusta löytyi hieman enemmän kokemusta (55 %, n= 12) kuin jälkimmäiseltä kurssilta (44 %, n=10) (ks. liite 7, kaavio 4-b).

Vaikka kurssin mainoksessa painotettiin, ettei aikaisempi ohjelmointikokemus ole tarpeen, oli silti hieman yllätys, että niinkin moni vastasi kysymyksessä 5 omaavansa paljon ohjelmointikokemusta (11 %, n=5) tai ainakin kokeillut joskus ohjelmointia (42 %, n=19)

(ks. liite 7, kaavio 5-a). Yleisin vastaus kuitenkin oli, että aikaisempaa ohjelmointikokemusta ei ollut (47 %, n=21).

Kurssien välillä oli hieman eroja ohjelmointikokemuksessa. Jälkimmäisellä kurssilla 57 % (n=13) ei ollut ohjelmoinut aikaisemmin, kun ensimmäisellä kurssilla täysin vailla kokemusta oli vain 36 % (n=8) kurssin oppilasmäärästä (ks. liite 7, kaavio 5-b).

Kokemus tietoteknisten laitteiden käyttötaidosta

Seitsemäs kysymys koski omaa kokemusta tietoteknisistä käyttötaidoista. Pääsääntöisesti oppilaat kokivat omat tietokoneen käyttötaitonsa kohtalaisen hyväksi, keskiarvolla 3,7. Yleisin vastaus oli 4 (37 %, n=16). Melko moni antoi kuitenkin vastauksen 1-3 (yhteensä 18 oppilasta), joten monella oli myös epävarmuutta omista käyttötaidoistaan (ks. liite 7, kaavio 6-a). Kurssien välillä ei ollut mainittavia eroja (ks. liite 7, kaavio 6-b).

Yksi ensimmäisen kurssin sekä yksi toisen kurssin oppilaista ei vastannut kysymykseen.

Ristiintaulukoinnilla nähdään, että oppilaat, joilla oli edes hieman ohjelmointikokemusta, kokivat itsensä myös jonkin verran parempina tietokoneen käyttäjinä. Aikaisempaa ohjelmointikokemusta vähän (vastaavasti paljon) omaavat kokivat tietokoneen käyttötaitonsa keskiarvolla 4,0 (vastaavasti 4,1), kun taas ne, joilla ei aikaisempaa ohjelmointikokemusta ole, vastasivat keskimäärin 3,3 (ks. liite 7, kaavio 7).

Lukeminen

Oppilaista 64 % (n=29) ilmoitti harrastavansa lukemista vähintään muutaman kerran viikossa. Kaksikymmentä prosenttia (n=9) kertoi lukevansa harvemmin kuin kerran kuussa (ks. liite 7, kaavio 8-a). Kurssien välillä ei vastauksissa ollut merkittävää eroa (ks. liite 7, kaavio 8-b).

Mieluisimmat kouluaineet ja kouluaineiden hyödyntäminen

Kysymyksessä 9 vajaa 60 prosenttia oppilaista (n=26) ilmoitti tietotekniikan mieluisimpien aineiden joukkoon (ks. liite 7, kaavio 9-a). Yli 50 prosentin kannatuksen saivat myös

matematiikka (n=24) ja englannin kieli (n=23). Heti perässä seurasivat kemia (n=22) ja fysiikka (n=20).

Suurimmat erot kurssien välillä muodostuivat englannin kielessä, sillä ensimmäisen kurssin oppilaista 15 (68 %) ilmoitti englannin mieluisimpien aineiden joukkoon, mutta jälkimmäisellä kurssilla vain 8 oppilasta (35 %). Muuten erot olivat pieniä (ks. liite 7, kaavio 9-b).

Seuraavaksi kysyttiin, mitä oppiaineita osallistujat kuvittelivat voivansa hyödyntää kurssilla. Suurimman kannatuksen (64 %, n=29) sai matematiikka, perässä tietotekniikka (47 %, n=21), englanti (38 %, n=17), fysiikka (13 %, n=6), kuvaamataito (4 %, n=2) ja äidinkieli (4 %, n=2). Yksittäisiä ääniä saivat myös musiikki ja kemia (ks. liite 7, kaavio 10-a). Kaksi oppilasta ei vastannut mitään, ja yksi vastaus hylättiin, koska se ei sisältänyt oppiainetta (vastaukset annettiin avoimena).

Oppilaat eivät olleet niin varmoja kurssin oppiaineita vahvistavasta vaikutuksesta. Eniten ääniä sai tietotekniikka (64 %, n=29), mutta kauaksi jäivät matematiikka (33 %, n=15), fysiikka (11 %, n=5) ja musiikki (4 %, n=2). Yksittäisiä ääniä saivat englanti, kuvaamataito ja kemia (ks. liite 7, kaavio 10-b). Kysymyksenasettelu saattoi olla oppilaille lievästi konstikas, sillä ensimmäisen kurssin vastauksissa oli jopa yhdeksän tyhjää. Toisen kurssin vastauksissa oli yksi tyhjä vastaus ja yksi vastauksista hylättiin, koska vastaus ei sisältänyt mitään oppiainetta.

Kurssille tulemisen syitä ja kaverit kurssilla

Suurin tekijä kurssille tulemiseen näyttää olleen se (kysymys 12), että oppilaat olivat kiinnostuneita peleistä (69 %, n=31) (ks. liite 7, kaavio 11-a). Lähes yhtä moni (60 %, n=27) ilmoittaa kurssille tulemisen syyksi sen, että haluaa oppia ohjelmoimaan. Melko monella (31 %, n=14) on myös toive siitä, että ohjelmointi voisi olla tulevaisuuden ammatti. Näiden lisäksi 5 oppilasta (11 %) tuli kurssille, koska kaverikin tuli. Kolme oppilasta (7 %) ilmoitti olevansa kiinnostunut luonnontieteistä, ja niin ikään kolme ilmoitti kurssille tulemisen syyksi sen, että halusi mielekästä tekemistä kesälle.

Kaksi oppilasta (4 %) sanoi tulleensa vanhempien pakottamana. Oppilaat kuitenkin suorittivat kurssin loppuun saakka, ja molemmat näistä oppilaista antoivat myöhemmin loppukyselyssä maksimipisteet kysymyksiin ”Olen tyytyväinen kurssiin kokonaisuutena” ja ”Minulle on hyötyä siitä, että osaan suunnitella ja ohjelmoida pelejä”. He olivat myös samaa mieltä loppukyselyn kysymyksen ”Tunnilla oli mukava ilmapiiri” kanssa. Kurssien välillä ei ollut merkittävää eroa (ks. liite 7, kaavio 11-b).

Kysymyksessä 13 kysyttiin, onko kurssilla mukana kavereita. Tämän kysymyksen tarkoituksena oli se, että tietoa oppilaiden keskinäisistä kaverisuhteista olisi käytetty parinmuodostuksessa harjoitustehtävien tekemisen aikana. Ajatuksesta kuitenkin luovuttiin alkukyselyn tekemisen jälkeen, koska vaikutti siltä, ettei ryhmien pakonomaisella muodostamisella olisi hedelmällisiä seurauksia.

Vajaalla puolella (48 %, n=10) ensimmäisen kurssin oppilaista oli kavereita mukana kurssilla (ks. liite 7, kaavio 12-a). Toisella kurssilla heitä oli vähemmän, 30 % (n=7). Keskimäärin siis 39 % (n=17) oppilaista oli kaveri mukana kurssilla (ks. liite 7, kaavio 12-b).

Kiinnostus luonnontieteiden opiskelua kohtaan

Vastausten perusteella 15 oppilasta (35 %) oli täysin samaa tai lähes samaa mieltä siitä, että on kiinnostunut luonnontieteiden opiskelusta (ks. liite 7, kaavio 13-b). Eri mieltä tai lähes eri mieltä oli 16 oppilasta (37 %). Vajaa kolmannes (28 %, n=12) ei ollut samaa eikä eri mieltä.

Ensimmäisen kurssin keskiarvo oli 2,9, ja toisen kurssin 3,0 (ks. liite 7, kaavio 13-c). Yhteensä luonnontieteiden opiskelu kiinnosti oppilaita keskiarvolla 2,9 asteikolla 1–5 (täysin eri mieltä, …, täysin samaa mieltä). Yleisin vastaus oli 3, joskin jälkimmäisen kurssin tyypillisin vastaus oli 2 (jonkin verran eri mieltä) (ks. liite 7, kaavio 13-a).

Kurssiodotukset

Suurin osa oppilaista (89 %, n=40) odotti saavansa kurssilla tietoa pelien tekemisestä (ks. liite 7, kaavio 14-a & 14-b). Lähes yhtä moni (84 %, n=38) oletti, että kurssilla saa tietoa

ohjelmoinnista. Perässä seurasivat vastaukset ”tietoa pelien suunnittelusta” (69 %, n=31) ja ”tietoa ohjelmoijan ammatista” (38 %, n=17). Muutama (9 %, n=4) odotti myös saavansa kavereita kurssilla ja yksi oppilas tietoa yliopistosta.

Muuta-kohtaan tuli yksi vastaus:

- *Jo koneille olisi asennettu jotain kivaa, miin nitä olisi kiva pelata, kun on vapaata, kute CoD tai WC3*

8.2 Alkukyselyn korrelaatioanalyysi

Alkukyselyn tavoitteena oli selvittää kurssille tulleiden oppilaiden taustoja ja ennakkotietoja. Tässä luvussa selvitetään korrelaatioita tarkastelemalla, onko kyselyyn annettujen vastausten välillä tilastollisesti merkitsevää yhteyttä.

Taulukossa 4 on esitetty korrelaatiokaavio siitä, miten kysymyksen 6 vastaukset korreloivat muiden kysymysten vastauksien kanssa. Tällä siis saatiin selville, että korreloiko vastaajien aikaisempi ohjelmointikokemus muiden kysymysten vastauksien kanssa. Taulukossa on merkitty kahdella tähdellä ne kysymykset, joissa korrelaatio on tilastollisesti merkitsevä (vähintään 1 % merkitsevyystaso) ja yhdellä tähdellä ne kysymykset, joissa korrelaatio on tilastollisesti melkein merkitsevä (vähintään 5 % merkitsevyystaso). Monivastausmuuttujien osalta ei voida laskea tilastollista merkitsevyyttä, mutta ne on käsitelty frekvenssitaulukkojen sekä ristiintaulukoinnin avulla.

		Sukupuoli	Ikä vuosina	Pelaan tietokone-/konsolipelejä	Olen suunnitellut pelejä (tietokoneella tai ilman tietokonetta	Koen olevani taitava tietokoneiden ja teknisten laitteiden käyttäjä	Harrastan lukemista	Aion hakeutua luonnontieteiden opiskelun pariin tulevaisuudessa
Minulla on aikaisempaa ohjelmointikokemusta	Pearson Correlation	,412(**)	,119	,025	,451(**)	,350(*)	,061	,403(**)
	Sig. (2-tailed)	,005	,446	,871	,002	,022	,691	,007
	N	45	43	45	45	43	45	43

Taulukko 4. Kysymyksen 6 vastausten korrelaatiot muiden kysymysten vastauksiin alkukyselyssä.

Taulukosta nähdään, että tilastollisesti merkitsevä positiivinen (*Pearson Correlation*-rivin luku on suurempaa kuin nolla) korrelaatio löytyy sukupuolen kohdalta, sekä kysymyksistä 5 ja 14. Näyttää siis siltä, että aikaisempi ohjelmointikokemus vaikutti merkitsevästi oppilaiden aikaisempaan pelien suunnittelukokemukseen sekä siihen, aikooko tulevaisuudessa hakeutua luonnontieteelliselle alalle opiskelemaan (tätä asiaa kysyttiin myös loppukyselyssä, josta lisää tuonnempana). Edelleen sukupuolella ja aikaisemmalla ohjelmointikokemuksella oli myös merkittävä yhteys. Tarkemmin sanottuna yhdelläkään kurssin aloittaneista tytöistä (n=7) ei ollut aikaisempaa ohjelmointikokemusta, sen sijaan pojilla sitä jonkin verran oli.

Tilastollisesti melkein merkitsevä korrelaatio löytyy kysymyksestä 7, joka on myös positiivinen. Toisin sanoen, mitä enemmän oppilaalla oli aikaisempaa ohjelmointikokemusta, sitä paremmaksi tietokoneen käyttäjäksi hän itsensä koki.

Kysymyksessä viisi kysyttiin, onko oppilas suunnitellut pelejä ennen kurssille tuloaan. Pelin suunnitteluksi katsottiin tietokoneella tai ilman tietokonetta tapahtunut suunnittelu.

Tarkastellaan seuraavaksi kysymyksen 5 korrelaatioita muihin kysymyksiin nähden (taulukko 5).

		Sukupuoli	Ikä vuosina	Pelaan tietokone-/konsolipelejä	Minulla on aikaisempaa ohjelmointikokemusta	Koen olevani taitava tietokoneiden ja teknisten laitteiden käyttäjä	Harrastan lukemista	Aion hakeutua luonnontieteiden opiskelun pariin tulevaisuudessa
Olen suunnitellut pelejä (tietokoneella tai ilman tietokonetta	Pearson Correlation	,052	,077	,197	,451(**)	,318(*)	,013	,057
	Sig. (2-tailed)	,735	,622	,196	,002	,038	,932	,715
	N	45	43	45	45	43	45	43

Taulukko 5. Kysymyksen 5 vastausten korrelaatiot muiden kysymysten vastauksiin alkukyselyssä.

Tilastollisesti melkein merkitsevä yhteys löytyy kysymyksen 7 kanssa, eli missä oppilas arvioi omaa tietoteknistä osaamistaan. Korrelaatio on positiivinen, eli mikäli käyttäjä oli suunnitellut pelejä ennakkoon, hän koki myös olevansa parempi tietokoneiden ja teknisten laitteiden käyttäjä.

Aikaisemman ohjelmointikokemuksen ja pelien suunnittelun tilastollisesti merkittävä yhteys löytyi jo edellisestä taulukosta.

Kysymyksen 14 (”Aion hakeutua luonnontieteiden opiskelun pariin…”) kohdalla huomataan vielä, että melkein merkitsevä tilastollinen korrelaatio löytyy sukupuolen kohdalta (taulukko 6).

		Sukupuoli	Ikä vuosina	Pelaan tietokone-/konsolipelejä	Olen suunnitellut pelejä (tietokoneella tai ilman tietokonetta	Minulla on aikaisempaa ohjelmointikokemusta	Koen olevani taitava tietokoneiden ja teknisten laitteiden käyttäjä	Harrastan lukemista
Aion hakeutua luonnontieteiden opiskelun pariin tulevaisuudessa…	Pearson Correlation	,345(*)	,286	,326(*)	,057	,403(**)	,255	-,078
	Sig. (2-tailed)	,023	,070	,033	,715	,007	,099	,617
	N	43	41	43	43	43	43	43

Taulukko 6. Kysymyksen 14 vastausten korrelaatiot muiden kysymysten vastauksiin alkukyselyssä.

Katsotaan vielä ristiintaulukoinnin avulla, miten tyttöjen ja poikien vastaukset suhteutuivat toisiinsa (taulukko 7).

			Aion hakeutua luonnontieteiden opiskelun pariin tulevaisuudessa (esim. ammattikoulun luonnontieteellinen linja, tai lukiossa pitkä matematiikka/fysiikka/kemia/tietotekniikka)					Total
			Täysin eri mieltä	Jokseenkin eri mieltä	Ei samaa eikä eri mieltä	Jokseenkin samaa mieltä	Täysin samaa mieltä	
Sukupuoli	Tyttö	Count	3	2	1	1	0	7
		% within Sukupuoli	42,9%	28,6%	14,3%	14,3%	,0%	100,0%
	Poika	Count	3	8	11	10	4	36
		% within Sukupuoli	8,3%	22,2%	30,6%	27,8%	11,1%	100,0%
Total		Count	6	10	12	11	4	43
		% within Sukupuoli	14,0%	23,3%	27,9%	25,6%	9,3%	100,0%

Taulukko 7. Kysymys 14 ja vastaajan sukupuoli ristiintaulukoituna.

Tytöistä kukaan ei täysin varmasti aikonut hakeutua luonnontieteelliselle alalle, ja jokseenkin samaa mieltä oli vain yksi. Pojilla vastausten jakauma lähentelee normaalijakaumaa.

8.3 Alkukyselyn monivastausmuuttujien analysointi

Tässä luvussa tarkastellaan vielä ristiintaulukoinnin avulla niitä alkukyselyn kysymyksiä, joissa vastausvaihtoehtoja oli useampia kuitenkin siten, että analysoitavaksi valittiin se aineisto, jossa tulokset olivat tavalla tai toisella mielenkiintoisia. Taustamuuttujiksi valittiin sukupuoli, ikä ja aiempi ohjelmointikokemus. Oppilaiden vastausten frekvenssitaulukot ja yhteenveto on esitetty edellisessä luvussa. Oppilaalla oli mahdollisuus valita vastausvaihtoehdoista joko ei yhtään, yksi tai useampia.

Kysymys 4: Minkä tyyppisiä pelejä pelaat?

Sukupuolen perusteella oppilaat näyttävät jakautuneen melko tasaisesti eri pelityyppien kesken (taulukko 8). Ainoastaan räiskintäpeleissä on selvä ero: tytöistä vain yksi ilmoitti pelaavansa räiskintäpelejä, kun pojista näin kertoi 25 oppilasta eli pojista kaksi kolmesta. Suosituimmat pelityypit on lihavoitu taulukossa.

			Pelityypit										Yht
			Auto- ja ajo	Pulma	**Räis- kintä**	**Rooli**	Seik- kailu	Strate- gia	Taiste- lu	Taito	Taso- hypp.	Urheilu	
Sukupuoli	Tyttö	Count	1	2	1	**4**	3	2	2	1	5	1	7
		% within suku- puoli	14,3%	28,6%	14,3%	**57,1%**	42,9%	28,6%	28,6%	14,3%	71,4%	14,3%	
	Poika	Count	11	14	**25**	19	23	20	8	7	20	6	38
		% within suku- puoli	28,9%	36,8%	**65,8%**	50,0%	60,5%	52,6%	21,1%	18,4%	52,6%	15,8%	
Total		Count	12	16	26	23	26	22	10	8	25	7	45

Taulukko 8. Kysymys 4 ja vastaajan sukupuoli ristiintaulukoituna.

Myöskään vastaajan iällä ei näytä olevan suurta merkitystä. Eri pelityyppejä pelattiin melko tasaisesti kaikissa ikäluokissa (taulukko 9).

			Pelityypit										
			Auto- ja ajo	Pulma	Räis- kintä	Rooli	Seik- kailu	Strate- gia	Taiste- lu	Taito	Taso- hypp.	Urheilu	Yht
Ikä vuosina	12	Count	1	1	2	1	0	1	0	0	3	2	4
		% within ikä	25,0%	25,0%	50,0%	25,0%	,0%	25,0%	,0%	,0%	75,0%	50,0%	
	13	Count	4	6	9	10	11	8	4	2	9	2	16
		% within ikä	25,0%	37,5%	56,3%	62,5%	68,8%	50,0%	25,0%	12,5%	56,3%	12,5%	
	14	Count	6	2	6	4	7	6	2	2	6	2	12
		% within ikä	50,0%	16,7%	50,0%	33,3%	58,3%	50,0%	16,7%	16,7%	50,0%	16,7%	
	15	Count	0	4	4	4	4	4	2	1	3	0	6
		% within ikä	,0%	66,7%	66,7%	66,7%	66,7%	66,7%	33,3%	16,7%	50,0%	,0%	
	16	Count	1	2	3	3	3	2	2	2	3	1	5
		% within ikä	20,0%	40,0%	60,0%	60,0%	60,0%	40,0%	40,0%	40,0%	60,0%	20,0%	
Total		Count	12	15	24	22	25	21	10	7	24	7	43

Taulukko 9. Kysymys 4 ja vastaajan ikä (kysymys 2) ristiintaulukoituna.

Vastaajan aikaisempi ohjelmointikokemus näkyi jonkin verran pelatuimmissa pelityypeissä (taulukko 10). Ne oppilaat, joilla oli eniten ohjelmointikokemusta, pelasivat eniten strategia- ja räiskintäpelejä. Oppilaat, joilla ei ollut aikaisempaa ohjelmointikokemusta, pelasivat eniten tasohyppely-, seikkailu- ja roolipelejä.

			Pelityypit										
			Auto- ja ajo	Pulma	**Räis- kintä**	Rooli	Seik- kailu	**Strate- gia**	Taiste- lu	Taito	**Taso- hypp.**	Urheilu	Yht
Aik. ohjelm.kokemusta	Ei	Count	4	9	9	12	11	9	4	4	**13**	6	21
		% within k6	19,0%	42,9%	42,9%	57,1%	52,4%	42,9%	19,0%	19,0%	**61,9%**	28,6%	
	Vähän	Count	6	6	13	10	12	9	4	2	10	1	19
		% within k6	31,6%	31,6%	68,4%	52,6%	63,2%	47,4%	21,1%	10,5%	52,6%	5,3%	
	Paljon	Count	2	1	**4**	1	3	**4**	2	2	2	0	5
		% within k6	40,0%	20,0%	**80,0%**	20,0%	60,0%	**80,0%**	40,0%	40,0%	40,0%	,0%	
Total		Count	12	16	26	23	26	22	10	8	25	7	45

Taulukko 10. Kysymys 4 ja vastaajan aikaisempi ohjelmointikokemus (kysymys 6) ristiintaulukoituna.

Kysymys 9: Minulle mieluisimpia kouluaineita ovat

Sukupuolien väliset erot suhteessa mieluisimpiin kouluaineisiin on nähtävissä taulukossa 11. Tyttöjen keskuudessa suosituimmat aineet olivat kuvaamataito, musiikki ja liikunta, kun pojilla ne olivat tietotekniikka, matematiikka ja kemia. Kummankin sukupuolen suosituin aine on taulukossa lihavoituna.

			Englanti	Fysiikka	Historia, yko	Kemia	Kotitalous	**Kuvaamat**	Käsityöt	Liikunta
Sukupuoli	Tyttö	Count	3	0	2	1	1	**6**	1	4
		% within sukupuoli	42,9%	,0%	28,6%	14,3%	14,3%	**85,7%**	14,3%	57,1%
	Poika	Count	20	20	5	21	6	9	7	13
		% within sukupuoli	52,6%	52,6%	13,2%	55,3%	15,8%	23,7%	18,4%	34,2%
Total		Count	23	20	7	22	7	15	8	17

			Maantieto	Matemat.	Musiikki	Ruotsi	**Atk**	Uskonto, et	Äidinkieli	Yht
Sukupuoli	Tyttö	Count	0	1	4	2	2	0	2	7
		% within sukupuoli	,0%	14,3%	57,1%	28,6%	28,6%	,0%	28,6%	
	Poika	Count	2	23	15	3	24	2	2	38
		% within sukupuoli	5,3%	60,5%	39,5%	7,9%	**63,2%**	5,3%	5,3%	
Total		Count	2	24	19	5	**26**	2	4	45

Taulukko 11. Kysymys 9 ja vastaajan sukupuoli ristiintaulukoituna.

Ristiintaulukoidaan seuraavaksi ikä kysymyksen 9 kanssa (taulukko 12). Luettavuuden parantamiseksi ikä-muuttujan arvot on nyt laitettu sarakkeisiin ja aineet riveille. Kunkin ikäryhmän suosituin aine on taulukossa lihavoituna.

			Ikä vuosina					
			12	13	14	15	16	Total
Mieluisat aineet	**Englanti**	Count	**4**	8	3	3	3	21
		% within ikä	**100,0%**	50,0%	25,0%	50,0%	60,0%	
	Fysiikka	Count	1	7	**7**	2	2	19
		% within ikä	25,0%	43,8%	**58,3%**	33,3%	40,0%	
	Historia, yo	Count	0	3	2	1	0	6
		% within ikä	,0%	18,8%	16,7%	16,7%	,0%	
	Kemia	Count	1	**9**	6	3	2	21
		% within ikä	25,0%	**56,3%**	50,0%	50,0%	40,0%	
	Kotitalous	Count	0	4	3	0	0	7
		% within ikä	,0%	25,0%	25,0%	,0%	,0%	
	Kuvaama-taito	Count	1	5	3	4	2	15
		% within ikä	25,0%	31,3%	25,0%	66,7%	40,0%	
	Käsityöt	Count	1	2	3	1	1	8
		% within ikä	25,0%	12,5%	25,0%	16,7%	20,0%	
	Liikunta	Count	3	6	6	0	2	17
		% within ikä	75,0%	37,5%	50,0%	,0%	40,0%	
	Maantieto	Count	0	0	1	0	1	2
		% within ikä	,0%	,0%	8,3%	,0%	20,0%	
	Matema-tiikka	Count	3	**9**	6	3	2	23
		% within ikä	75,0%	**56,3%**	50,0%	50,0%	40,0%	
	Musiikki	Count	1	7	5	1	3	17
		% within ikä	25,0%	43,8%	41,7%	16,7%	60,0%	
	Ruotsi	Count	0	1	1	0	2	4
		% within ikä	,0%	6,3%	8,3%	,0%	40,0%	
	Atk	Count	1	8	**7**	**5**	**4**	25
		% within ikä	25,0%	50,0%	**58,3%**	**83,3%**	**80,0%**	
	Uskonto, et	Count	0	2	0	0	0	2
		% within ikä	,0%	12,5%	,0%	,0%	,0%	
	Äidinkieli	Count	0	2	1	1	0	4
		% within ikä	,0%	12,5%	8,3%	16,7%	,0%	
Total		Count	4	16	12	6	5	43

Taulukko 12. Kysymys 9 ja vastaajan ikä ristiintaulukoituna.

Seuraavaksi ristiintaulukoidaan oppilaan aikaisempi ohjelmointikokemus kysymyksen 9 kanssa (taulukko 13). Ohjelmointikokemuksen taso on laitettu sarakkeisiin ja oppiaineet riveille. Suosituimmat oppiaineet on lihavoitu kustakin ohjelmointikokemuksen ilmoittavasta sarakkeesta. Huomionarvoista on, että ne oppilaat, joilla ei aikaisempaa ohjelmointikokemusta ollut, useimmin ilmoittivat liikunnan lempiaineekseen. Taas ne, joilla oli ohjelmointikokemusta, ilmoittivat useammin matematiikan, fysiikan ja kemian mieluisaksi kouluaineeksi.

			Minulla on aikaisempaa ohjelmointikokemusta			
			Ei	Kyllä, mutta vain hyvin vähän	Kyllä, melko paljon tai paljon	Total
Mieluisat aineet	Englanti	Count	10	10	3	23
		% within k6	47,6%	52,6%	60,0%	
	Fysiikka	Count	5	11	4	20
		% within k6	23,8%	57,9%	80,0%	
	Historia, yo	Count	4	3	0	7
		% within k6	19,0%	15,8%	,0%	
	Kemia	Count	6	12	4	22
		% within k6	28,6%	63,2%	80,0%	
	Kotitalous	Count	4	2	1	7
		% within k6	19,0%	10,5%	20,0%	
	Kuvaama-taito	Count	9	4	2	15
		% within k6	42,9%	21,1%	40,0%	
	Käsityöt	Count	3	4	1	8
		% within k6	14,3%	21,1%	20,0%	
	Liikunta	Count	**11**	6	0	17
		% within k6	**52,4%**	31,6%	,0%	
	Maantieto	Count	0	2	0	2
		% within k6	,0%	10,5%	,0%	
	Matema-tiikka	Count	8	11	**5**	24
		% within k6	38,1%	57,9%	**100,0%**	
	Musiikki	Count	9	9	1	19
		% within k6	42,9%	47,4%	20,0%	
	Ruotsi	Count	3	1	1	5
		% within k6	14,3%	5,3%	20,0%	
	Atk	Count	6	**16**	4	26
		% within k6	28,6%	**84,2%**	80,0%	
	Uskonto, et	Count	1	0	1	2
		% within k6	4,8%	,0%	20,0%	
	Äidinkieli	Count	3	1	0	4
		% within k6	14,3%	5,3%	,0%	
Total		Count	21	19	5	45

Taulukko 13. Kysymys 9 ja vastaajan aikaisempi ohjelmointikokemus ristiintaulukoituna.

Kysymys 10: Voin hyödyntää näitä kouluaineita tällä kurssilla

Pojat olivat tyttöjä useammin sitä mieltä, että voivat hyödyntää matematiikkaa tällä kurssilla. Tytöt uskoivat useimmin, että englannin kielen taito on heille eduksi tällä kurssilla. Eniten vastauksia saaneet on lihavoitu sukupuolittain taulukossa 14.

			Sukupuoli		
			Tyttö	Poika	Total
Voin hyödyntää näitä kouluaineita	**Englanti**	Count	**4**	13	17
		% within sukupuoli	**66,7%**	36,1%	
	Fysiikka	Count	1	5	6
		% within sukupuoli	16,7%	13,9%	
	Kemia	Count	1	0	1
		% within sukupuoli	16,7%	,0%	
	Kuvaama-taito	Count	0	2	2
		% within sukupuoli	,0%	5,6%	
	Matema-tiikka	Count	2	**27**	29
		% within sukupuoli	33,3%	**75,0%**	
	Musiikki	Count	1	0	1
		% within sukupuoli	16,7%	,0%	
	Atk	Count	**4**	18	22
		% within sukupuoli	**66,7%**	50,0%	
	Äidinkieli	Count	1	1	2
		% within sukupuoli	16,7%	2,8%	
Total		Count	6	36	42

Taulukko 14. Kysymys 10 ja vastaajan sukupuoli ristiintaulukoituna.

Myös iällä oli jonkin verran vaikutusta suhteessa siihen, mitä oppiaineita vastaaja koki voivansa hyödyntää kurssilla. Vastaajamäärät ovat kuitenkin niin pieniä, ettei tämän ristiintaulukoinnin perusteella voi tehdä kovin pitkälle meneviä johtopäätöksiä. Eniten vastauksia saaneet on lihavoitu ikäryhmittäin taulukossa 15.

			Ikä vuosina					
			12	13	14	15	16	Total
Voin hyödyntää näitä kouluaineita	**Englanti**	Count	**3**	4	3	**3**	**3**	16
		% within ikä	**75,0%**	28,6%	27,3%	**50,0%**	**60,0%**	
	Fysiikka	Count	0	3	1	0	2	6
		% within ikä	,0%	21,4%	9,1%	,0%	40,0%	
	Kemia	Count	0	1	0	0	0	1
		% within ikä	,0%	7,1%	,0%	,0%	,0%	
	Kuvaama-taito	Count	0	0	0	1	1	2
		% within ikä	,0%	,0%	,0%	16,7%	20,0%	
	Matema-tiikka	Count	1	**12**	**8**	**3**	**3**	27
		% within ikä	25,0%	**85,7%**	**72,7%**	**50,0%**	**60,0%**	
	Musiikki	Count	0	0	1	0	0	1
		% within ikä	,0%	,0%	9,1%	,0%	,0%	
	Atk	Count	2	7	7	2	**3**	21
		% within ikä	50,0%	50,0%	63,6%	33,3%	**60,0%**	
	Äidinkieli	Count	0	2	0	0	0	2
		% within ikä	,0%	14,3%	,0%	,0%	,0%	
Total		Count	4	14	11	6	5	40

Taulukko 15. Kysymys 10 ja vastaajan ikä ristiintaulukoituna.

Katsotaan vielä vastaajan aikaisempien ohjelmointitaitojen vaikutus siihen, mitä oppiaineita vastaaja uskoo voivansa hyödyntää kurssilla (taulukko 16). Mitä enemmän vastaajalla oli ohjelmointikokemusta, sitä vahvemmin hän uskoi siihen, että hän voi hyödyntää matemaattista osaamistaan kurssilla. Matematiikka sai kaikissa ryhmissä eniten ääniä, joskin jaetulle ykkössijalle nousi tietotekniikka niiden oppilaiden kohdalla, keillä ei aikaisempaa ohjelmointikokemusta ollut lainkaan.

Vaikka kurssi ei varsinaisesti tietotekniikan kurssi ollutkaan, oli tietotekniikan taitojen hyödyntäminen tällaisella kurssilla luonnollisesti itsestään selvää, ja sikäli kysymys siltä osin oli hieman kompakysymys.

			Minulla on aikaisempaa ohjelmointikokemusta			
			Ei	Kyllä, mutta vain hyvin vähän	Kyllä, melko paljon tai paljon	Total
Voin hyödyntää näitä kouluaineita	Englanti	Count	9	8	0	17
		% within ak6	45,0%	47,1%	,0%	
	Fysiikka	Count	3	3	0	6
		% within ak6	15,0%	17,6%	,0%	
	Kemia	Count	1	0	0	1
		% within ak6	5,0%	,0%	,0%	
	Kuvaama-taito	Count	0	2	0	2
		% within ak6	,0%	11,8%	,0%	
	Matema-tiikka	Count	**11**	**13**	**5**	29
		% within ak6	**55,0%**	**76,5%**	**100,0%**	
	Musiikki	Count	1	0	0	1
		% within ak6	5,0%	,0%	,0%	
	Atk	Count	**11**	8	3	22
		% within ak6	**55,0%**	47,1%	60,0%	
	Äidinkieli	Count	2	0	0	2
		% within ak6	10,0%	,0%	,0%	
Total		Count	20	17	5	42

Taulukko 16. Kysymys 10 ja vastaajan aikaisempi ohjelmointikokemus ristiintaulukoituna.

Kysymys 12: Tulin kurssille, koska…

Kysymyksessä 12 tiedusteltiin oppilaiden motivaatiota tulla kurssille. Sukupuolella ei näytä olevan kovin suurta merkitystä tämän suhteen. Iällä sen sijaan on jonkin verran merkitystä (taulukko 17). Nuorimmista oppilaista kaikki vastasivat tulleensa kurssille, koska pelit kiinnostavat heitä. Kuitenkin mitä vanhempi oppilas oli, sitä useammin hän vastasi, että hän haluaa oppia ohjelmoimaan, tai että uskoo ohjelmoinnin olevan tulevaisuudessa jopa ammatti. Kurssin viidestä 16-vuotiaasta oppilaasta jopa 3 kertoi unelmoivansa ohjelmoijan ammatista.

			Ikä vuosina					
			12	13	14	15	16	Total
Tulin kurssille, koska…	Mielekästä tekemistä kesälle	Count	0	1	1	0	0	2
		% within ak2	,0%	6,3%	8,3%	,0%	,0%	
	Pelit kiinnostavat minua	Count	**4**	**11**	7	**5**	3	30
		% within ak2	**100,0%**	**68,8%**	58,3%	**83,3%**	60,0%	
	Kaverini tuli myös kurssille mukaan	Count	0	2	1	0	1	4
		% within ak2	,0%	12,5%	8,3%	,0%	20,0%	
	Olen kiinnostunut luonnontieteistä	Count	0	2	0	0	1	3
		% within ak2	,0%	12,5%	,0%	,0%	20,0%	
	Haluan oppia ohjelmoimaan	Count	1	9	**9**	4	2	25
		% within ak2	25,0%	56,3%	**75,0%**	66,7%	40,0%	
	Uskon, että ohjelmointi voi olla tulevaisuudessa jopa ammattini	Count	1	5	3	2	**3**	14
		% within ak2	25,0%	31,3%	25,0%	33,3%	**60,0%**	
	Vanhemmat pakottivat	Count	1	0	0	1	0	2
		% within ak2	25,0%	,0%	,0%	16,7%	,0%	
Total		Count	4	16	12	6	5	43

Taulukko 17. Kysymys 12 ja vastaajan ikä ristiintaulukoituna.

Edelleen ne, joilla oli hieman enemmän aikaisempaa kokemusta ohjelmoinnista, vastasivat useammin haluavansa oppia ohjelmoimaan, tai jopa unelmoivat ohjelmoijan ammatista (taulukko 18).

			Minulla on aikaisempaa ohjelmointikokemusta			
			Ei	Kyllä, mutta vain hyvin vähän	Kyllä, melko paljon tai paljon	Total
Tulin kurssille, koska…	Mielekästä tekemistä kesälle	Count	0	1	2	3
		% within ak6	,0%	5,3%	40,0%	
	Pelit kiinnostavat minua	Count	**16**	**11**	**4**	31
		% within ak6	**76,2%**	**57,9%**	**80,0%**	
	Kaverini tuli myös kurssille mukaan	Count	3	1	1	5
		% within ak6	14,3%	5,3%	20,0%	
	Olen kiinnostunut luonnontieteistä	Count	2	1	0	3
		% within ak6	9,5%	5,3%	,0%	
	Haluan oppia ohjelmoimaan	Count	12	**11**	**4**	27
		% within ak6	57,1%	**57,9%**	**80,0%**	
	Uskon, että ohjelmointi voi olla tulevaisuudessa jopa ammattini	Count	3	8	3	14
		% within ak6	14,3%	42,1%	60,0%	
	Vanhemmat pakottivat	Count	1	1	0	2
		% within ak6	4,8%	5,3%	,0%	
Total		Count	21	19	5	45

Taulukko 18. Kysymys 12 ja aikaisempi ohjelmointikokemus ristiintaulukoituna.

Kysymys 15: Odotan saavani kurssilta…

Jälleen iällä tai sukupuolella ei ollut kovinkaan paljon merkitystä kurssiodotusten suhteen, mutta aikaisempi ohjelmointikokemus lisäsi jonkin verran myös oppilaan odotuksia saada kurssilla tietoa pelisuunnittelusta, ohjelmoinnista yleisesti ja jopa ohjelmoijan ammatista (taulukko 19).

			Minulla on aikaisempaa ohjelmointikokemusta			
			Ei	Kyllä, mutta vain hyvin vähän	Kyllä, melko paljon tai paljon	Total
Odotan saavani	**Tietoa ohjelmoinnista**	Count	17	**16**	**5**	38
		% within ak6	81,0%	**84,2%**	**100,0%**	
	Tietoa pelien tekemisestä	Count	**21**	14	**5**	40
		% within ak6	**100,0%**	73,7%	**100,0%**	
	Tietoa pelien suunnittelusta	Count	16	10	**5**	31
		% within ak6	76,2%	52,6%	**100,0%**	
	Kavereita	Count	3	0	1	4
		% within ak6	14,3%	,0%	20,0%	
	Tietoa yliopistosta	Count	0	1	0	1
		% within ak6	,0%	5,3%	,0%	
	Tietoa ohjelmoijan ammatista	Count	7	7	3	17
		% within ak6	33,3%	36,8%	60,0%	
Total		Count	21	19	5	45

Taulukko 19. Kysymys 15 ja aikaisempi ohjelmointikokemus ristiintaulukoituna.

8.4 Loppukysely (n=41)

Tässä luvussa esitellään loppukyselyyn vastanneiden oppilaiden vastausten koosteet. Loppukyselyyn (liite 8) osallistui 41 oppilasta. Kolme toisella kurssilla ollutta oppilasta keskeytti kurssin, ja yksi oppilas sairastui, joten toisen kurssin alkuperäinen osallistujamäärä (23) pieneni neljällä ja siten loppukyselyyn osallistuneiden oppilaiden kokonaismääräksi saatiin 41.

Kaaviot on tilan säästämisen vuoksi ja luettavuuden helpottamiseksi liitetty tutkielman loppuun, ja ne löytyvät liitteestä 9. Asteikollisissa kysymyksissä käytettiin ”normaalia” Likert-asteikkoa 1-5 (täysin eri mieltä, …, täysin samaa mieltä).

Tyytyväisyys kurssiin, ohjaukseen ja opetukseen

Kokonaistyytyväisyys kurssiin oli erittäin korkea (kysymys 2). Ensimmäiset kurssilaiset olivat tyytyväisiä kurssiin keskimäärin pistein 4,77, ja toisen kurssin oppilaat pistein 4,63

(ks. liite 9, kaavio 1-a). Vain yksi antoi kysymykseen alle 4 pistettä (ks. liite 9, kaaviot 1-a & 1-b).

Suurin osa oppilaista (78 %, n=32) piti annettuja ohjeita selkeinä (kysymys 3). Vain yksi oppilas ei pitänyt annettuja ohjeita lainkaan selvinä (ks. liite 9, kaavio 2-a). Keskiarvoksi oppilaiden vastauksista muodostui 4,0. Kurssien välillä ei ollut juurikaan eroa niin vastausjakauman kuin keskiarvon suhteen (ks. liite 9, kaavio 2-b). Suurin osa oppilaista (78 %, n=31) piti myös oppituntien aloituksia hyvinä (kysymys 4, ks. liite 9, kaavio 3-a), eikä kurssien välillä ollut sanottavaa eroa vastausjakauman tai vastausten keskiarvon suhteen (ks. liite 9, kaavio 3-b).

Viidennessä kysymyksessä tiedusteltiin tehtävien vaikeustasoa, ja vastaukset jakautuivat hyvin ”normaalisti” (ks. liite 9, kaavio 4-b). Vajaa kolmannes (n=12) piti tehtäviä melko vaikeina tai vaikeina, ja vastaavasti noin kolmannes (n=14) piti tehtäviä ennemminkin helppoina. Kurssien välillä oli jonkin verran eroa: Kysymyksen keskiarvoksi ensimmäisellä kurssilla saatiin 3,1, kun toisella kurssilla arvo oli hieman alhaisempi 2,7 (ks. liite 9, kaavio 4-a).

Oppilaiden vastausten perusteella he ovat todella viihtyneet oppitunneilla (kysymys 6), sillä 73 % (n=30) oli täysin samaa mieltä väittämästä (ks. liite 9, kaavio 5-a). Vain kolme oppilasta (7 %) ei osannut sanoa mielipidettään, eikä yksikään oppilas ollut eri mieltä väittämästä. Keskiarvoksi saatiin kursseittain 4,8 ja 4,5, yhteensä 4,7 (ks. liite 9, kaavio 5-b).

Pelisuunnittelun osaamisesta on hyötyä

Oppilaat olivat melko vakuuttuneita pelisuunnittelun hyödyllisyydestä heille (kysymys 7): 80 % (n=31) vastasi, että ovat joko jokseenkin tai täysin samaa mieltä. Viisi (13 %) ei osannut sanoa mielipidettään ja kolme oppilasta oli eri mieltä (8 %) (ks. liite 9, kaaviot 6-a & 6-b).

Mitä oppilaat saivat kurssilta

Kurssilaiset olivat lähes yksimielisiä siitä, että kurssilta saatiin (kysymys 8) ainakin tietoa ohjelmoinnista, tietoa pelien tekemisestä (kumpaankin kysymykseen 95 %, n=39) sekä tietoa pelisuunnittelusta (88 %, n=36). Lähes puolet (n=19) kertoi saaneensa tietoa yliopistosta ja reilu kolmannes uusia kavereita. Muutamat oppilaat vastasivat saaneensa tietoa ohjelmoijan ammatista ja luonnontieteiden opiskelusta (ks. liite 9, kaaviot 7-a & 7-b).

Huomautettakoon, että kummallakin kurssilla järjestettiin noin tunnin mittainen tutustumiskierros Jyväskylän yliopiston kampusalueelle. Kierroksen aikana kerrottiin opiskelumahdollisuuksista IT-tiedekunnassa ja matemaattis-luonnontieteellisessä tiedekunnassa, sekä tutustuttiin tietokonemuseoon.

Millaisia pelejä oppilaat tekivät

Suurin osa oppilaista (48 %, n=20) kertoi tehneensä tasohyppelypelin (kysymys 9, ks. liite 9, kaaviot 8-a & 8-b). Syy tähän on varmasti siinä, että tasohyppelypeliä varten oli kirjoitettu eniten valmista koodia, toisin sanoen, oppilaiden ei tarvinnut tehdä paljonkaan itse saadakseen aikaiseksi omannäköinen toimiva tasohyppelypeli. Koska tasohyppelypelin pohja oli jo valmiiksi ohjelmoitu, oli tyypillinen tuotos sellainen, että oppilas oli korvannut valmiit grafiikat itse tekemillään, ja muuttanut kenties pelissä olevia äänitehosteita. Lisäksi hän oli itse rakentanut kentän, jota pitkin hahmo etenee pelissä.

Lisäksi tehtiin melko paljon taito- ja tarkkuuspelejä (10 kpl), kuten labyrinttipeli, skeittauspeli, tornipuolustuspeli, pallojen ampumispeli jne. On huomattava, että näihin peleihin ei ollut paljonkaan valmista koodia tarjolla, ja niinpä ne olivat peleistä omintakeisimpia ja persoonallisimpia.

Tankkipeli tai muu taistelupeli syntyi viiden oppilaan toimesta (12 %). Autopelin teki kolme oppilasta (8 %) ja avaruuspelin kaksi oppilasta (5 %).

Oppilaiden tuotoksista on kerrottu lisää luvussa 8.8.

Kysymys oli avoin ja annetut vastaukset ryhmiteltiin. Vastausten ryhmittely on liitteessä 9, taulukossa 1. Yksi vastauksista hylättiin ei-informatiivisena.

Oppilaista moni kertoi tehneensä valitsemansa pelin siitä syystä, että pitää itse sellaisten pelien pelaamisesta:

- *Tuntui hyvältä idealta*
- *Sen kun tietäis*
- *Se vaikutti kivalta*
- *koska olin tehnyt sellaisen pelin aiemmin*
- *En tiedä. Varmaan siks kun ei osannut näitä asioita.*
- *Tasohyppelyt ovat kivoja.*
- *Pidän niistä.*

Muutamat halusivat pelin tekemiseen haastetta.

- *Se on sopivan vaikeaa tehdä ja hauska*
- *Se kiinnosti ja sattui olemaan sopivan vaikea.*
- *Se tuntui sopivan haastavalta*

Moni kertoi valinneensa pelin kuitenkin sattumanvaraisesti, tai ei osannut kertoa syytä valitsemalleen pelille.

- *heitin idean hatusta*
- *ainut hyvä idea minkä keksin.*
- *En keksinyt muuta*
- *koska jumala sanoi näin*

Mikä kurssilla oli parasta

Avoimeen kysymykseen saatiin monenlaisia vastauksia, jotka ryhmiteltiin niissä esiintyneiden yhtäläisyyksien perusteella yhteensä seitsemään ryhmään. Lisäksi kaksi vastausta ei sopinut mihinkään muista ryhmistä, ja ne katsottiin sisällöllisesti merkityksettömiksi. Jotkin vastaukset liitettiin kahteen ryhmään, mikäli vastauksen sisällön kannalta se katsottiin mielekkääksi. Vastaukset löytyvät liitteestä 9, taulukosta 2.

Vastaukset ryhmiteltiin seuraavasti:

1. Oman pelin tekeminen (16 vastausta)
2. Ohjelmointi (8 vastausta)
3. Kurssilla saatu oppi (4 vastausta)
4. Haasteet ja haastavuus (6 vastausta)
5. Kaverit ja ohjaajat (3 vastausta)
6. Pelisuunnittelu (2 vastausta)
7. Pelien esittely (1 vastaus)
8. Grafiikan tekeminen (1 vastaus)

Kysymyksenasettelu saattoi olla hieman liian väljä, sillä vastauksena ”*oman pelin tekeminen*” on melko geneerinen ja ilmeinen, eikä kovin informatiivinen, sillä onhan kyseessä kurssi, jossa tavoitteena oli tehdä oma peli. Edelleen vastaus ”ohjelmointi” on hieman väljä, mutta kertoo kuitenkin, että oppilas on selkeästi nauttinut siitä, että on oppinut asioita ohjelmoinnista.

Mikä kurssilla oli huonointa

Tämänkin avoimen kysymyksen (kysymys 12) vastaukset ryhmiteltiin niissä esiintyneiden yhtäläisyyksien perusteella. Ryhmiä löydettiin vastausten samankaltaisuuksia tutkimalla yhteensä kahdeksan kappaletta, joista yhteen sisällytettiin merkityksettömät tai sisällöttömät vastaukset. Vastaukset löytyvät liitteestä 9, taulukosta 3.

Vastaukset ryhmiteltiin seuraavasti:

1. Kurssi oli liian lyhyt tai kurssin rytmitys epäonnistui (9 vastausta)
2. Oppilas ei mielestään oppinut riittävän hyvin opetettuja asioita tai ei ymmärtänyt annettuja ohjeita (6 vastausta)
3. Luennot (4 vastausta)
4. Opetus, ohjaus tai ohjeistus oli puutteellista (4 vastausta)
5. Ei juuri mikään (4 vastausta)
6. Kyselylomakkeita oli liian paljon (2 vastausta)
7. Tekniset ongelmat (1 vastaus)

Tähän kysymykseen saatiin hieman informatiivisempia vastauksia, ja antaa jatkoa varten hyödyllistä tietoa. Eniten oppilaat moittivat kurssin lyhyttä kestoa, tai että kurssi oli muuten rytmitykseltään (oppituntien kesto, taukojen pituus, jne.) epäonnistunut.

Yhteensä viisi oppilasta ei vastannut kysymykseen.

Kehittäisin kurssia seuraavasti

Kysymys 13 esitettiin edellisten kysymysten tavoin avoimena, ja vastaukset ryhmiteltiin etsimällä niistä samankaltaisuuksia ja yhtäläisyyksiä. Vastaukset löytyvät liitteestä 9, taulukosta 4.

Vastaukset ryhmiteltiin seuraavasti:

1. Kurssin pitäisi olla pidempi, monipuolisempi tai sen pitäisi sisältää enemmän opetusta (10 vastausta)
2. Annettujen ohjeiden pitäisi olla selkeämmät (6 vastausta)
3. Aikataulua tai rytmitystä tulisi kehittää (3 vastausta)
4. Ei kehitettävää (3 vastausta)
5. Ohjelmointia harrastaneille pitäisi järjestää oma, vaativampi kurssi (2 vastausta)
6. Kirjaston pitäisi olla monipuolisempi (2 vastausta)
7. Tiettyjä pelityyppejä lisää (1 vastaus)
8. Enemmän pelaamista (1 vastaus)

Lisäksi kuusi oppilasta ei osannut sanoa, joten ne vastaukset hylättiin ei-informatiivisina. Seitsemän oppilasta ei vastannut kysymykseen lainkaan.

Kurssi vastasi odotuksiani, ja jos ei vastannut, niin miksi

Yhteensä 33 oppilasta (81 %) oli lähes samaa tai täysin samaa mieltä siitä, että kurssi vastasi heidän odotuksiaan (kysymys 14, ks. liite 9, kaaviot 9-a & 9-b). Oppilaista 12 % (n=5) ei osannut sanoa, ja kolmen oppilaan kohdalla (7 %) kurssi ei ollut vastannut odotuksia.

Ensimmäisen kurssin keskiarvoksi muodostui 4,2, toisen 4,1 ja yhteensä 4,1. Yleisesti ottaen kurssi siis vastasi hyvin pitkälti oppilaiden ennakko-odotuksia.

Kysymyksessä 15 tiedusteltiin syitä sille, miksi kurssi ei vastannut odotuksia. Tähän antoi vastauksensa myös muutama sellainen oppilas, kuka oli ilmaissut edellisessä kysymyksessä kurssin vastanneen odotuksia.

- *Luulin koodaamista vaikeammaksi kuin se oli. Oli mukavampaa kuin luulin.*
- *luulin, että minua ylenkatsottaisiin, koska olin nuorinta ikäluokkaa.*
- *en ihan tiennyt, mitä odottaa… mut yllätyin kuitenkin positiivisesti*
- *kurssilla tehtiin vain yksi peli*

Toisen kurssin oppilaista kukaan ei vastannut kysymykseen 15.

Kiinnostus hakeutua opiskelemaan luonnontieteelliselle alalle

Oppilaista 22 (49 %) ilmoittaa olevansa jonkin verran tai täysin samaa mieltä (ks. liite 9, kaavio 10-a). Kantaansa ei osaa sanoa 15 oppilasta (37 %) ja 6 (15 %) on jonkin verran tai täysin eri mieltä.

Jos verrataan tuloksia alkukyselyn vastaavaan kysymykseen, nähdään, että 25 oppilaan kiinnostus on noussut kurssin mittaan. Alkukyselyn kaikkien oppilaiden keskiarvo oli 2,9, kun kurssin päättyessä keskiarvo oli 3,6. Alkukyselyssä arvoja 1, 2 tai 3 antaneista oppilaista loppukyselyssä arvon 4 tai 5 antoi yhteensä kuusi oppilasta. Toisin sanoen 15 %

kurssin läpikäyneistä oppilaista oli alussa sitä mieltä, ettei luonnontieteiden opiskelu kiinnosta (tai eivät osanneet ilmaista mielipidettään), mutta kurssin jälkeen heidän mielipiteensä on muuttunut.

Kahden oppilaan kiinnostus luonnontieteiden opiskelua kohtaan vähentyi yhdellä pykälällä kurssin aikana.

Aion valita kouluni mahdollisesti tarjoamia valinnaisia ATK-kursseja (vain 2. kurssi)

Kysymys esitettiin vain jälkimmäisellä kurssilla, eikä vastauksia siksi oteta tutkimuksen tulosten analyysiin mukaan. Kysymyksen tulokset esitetään kuitenkin tässä mahdollisen yleisen kiinnostavuuden takia.

Enemmistö vastanneista (n=11) joko ei osannut sanoa mielipidettään tai ei aikonut ottaa valinnaisia ATK-kursseja opintoihinsa. Kahdeksan oppilasta kuitenkin oli melko varmoja tai varmoja, että aikoo ottaa valinnaisia kursseja (ks. liite 9, kaavio 11).

Mikä kurssilla oli helpointa ja mikä vaikeinta

Kysymykset esitettiin avoimena, ja vastauksista löydettiin samankaltaisuuksia ja yhtäläisyyksiä, joiden perusteella vastaukset ryhmiteltiin. Joitakin vastauksia saatettiin sijoittaa jälleen kahteen ryhmään.

Vastaukset ryhmiteltiin seuraavasti (mikä oli helpointa, kysymys 17, ks. liite 9, taulukko 5):

1. Pelin suunnittelu (11 vastausta)
2. Ohjelmointi, sen osa-alueet ja orientaatio ohjelmointiajatteluun (6 vastausta)
3. Luennot tai luennolla esiintyneet yksittäiset asiat (4 vastausta)
4. Pelin toteuttamiseen liittyvät oheistoiminnot (4 vastausta)
5. Harjoitustehtävät ja niiden tekeminen (4 vastausta)
6. Valmiin koodin hyväksikäyttäminen ja sen muuntelu (3 vastausta)
7. Pelien pelaaminen (3 vastausta)
8. Yleinen työskentely kurssilla (2 vastausta)

9. Lähes kaikki (2 vastausta)

Kaksi vastausta hylättiin ei-informatiivisina. Yksi oppilas ei vastannut kysymykseen lainkaan.

Kysymykseen kurssin vaikeimmista asioista vastaukset ryhmiteltiin seuraavasti (ks. liite 9, taulukko 6):

1. Ohjelmointi yleisesti (14 vastausta)
2. Pelin suunnittelu ja tuottaminen (5 vastausta)
3. Työskentely kurssilla ja aikataulussa pysyminen (4 vastausta)
4. Ohjelmointikielen kieliopin ja rakenteen omaksuminen (4 vastausta)
5. Virheiden etsiminen (4 vastausta)
6. Ei juuri mikään (3 vastausta)
7. Yksittäiset harjoitustehtävät (2 vastausta)
8. Ohjeiden seuraaminen (2 vastausta)

Kaksi vastausta hylättiin ei-informatiivisina. Yksi oppilas ei vastannut kysymykseen lainkaan.

Mielestäni parhaiten kurssilla onnistuin…

Kysymys esitettiin jälleen avoimena, ja vastauksista löydettiin samankaltaisuuksia ja yhtäläisyyksiä, joiden perusteella vastaukset ryhmiteltiin yhteensä yhdeksään ryhmään, joista yksi sisälsi hylätyt tai ei-informatiiviset vastaukset. Joitakin vastauksia saatettiin sijoittaa kahteen ryhmään (ks. liite 9, taulukko 7).

Vastaukset ryhmiteltiin seuraavasti:

1. Pelin toteutus kokonaisuutena (17 vastausta)
2. Omaan peliin liittyvät erikoisuudet toteutuksessa (7 vastausta)
3. Aktiivisuus kurssilla (4 vastausta)
4. Ohjelmoinnin oppiminen (4 vastausta)
5. Pelin suunnittelu tai testaustoimenpiteet (3 vastausta)

6. Yksittäiset harjoitustehtävät (2 vastausta)
7. Pelaaminen (1 vastaus)
8. Kavereiden saaminen kurssilla (1 vastaus)

Yksi vastaus hylättiin ei-informatiivisena. Lisäksi yksi oppilas ei vastannut kysymykseen lainkaan.

8.5 Loppukyselyn korrelaatioanalyysi

Tässä luvussa selvitetään korrelaatioita tarkastelemalla, onko kyselyyn annettujen vastausten välillä tilastollisesti merkitsevää yhteyttä.

Taulukossa 20 on esitetty korrelaatiokaavio siitä, miten kysymyksen 2 vastaukset korreloivat muiden kysymysten vastauksien kanssa. Tällä siis saatiin selville, että korreloiko vastaajien kokonaistyytyväisyys kurssiin muiden loppukyselyssä esitettyjen kysymysten vastauksien kanssa. Taulukossa on merkitty kahdella tähdellä ne kysymykset, joissa korrelaatio on tilastollisesti merkitsevä (vähintään 1 % merkitsevyystaso) ja yhdellä tähdellä ne kysymykset, joissa korrelaatio on tilastollisesti melkein merkitsevä (vähintään 5 % merkitsevyystaso). Monivastausmuuttujien osalta ei voida laskea tilastollista merkitsevyyttä, joten ne on käsitelty frekvenssitaulukkojen sekä ristiintaulukoinnin avulla erikseen luvussa 8.6.

		k3	k4	k5	k6	k7	k14	k16	k17 (vain 2. kurssi)
Olen tyytyväinen kurssiin kokonaisuutena	Pearson Correlation	,645(**)	,429(**)	,232	,061	,412(**)	,662(**)	,108	-,318
	Sig. (2-tailed)	,000	,006	,145	,706	,008	,000	,501	,185
	N	41	40	41	41	40	41	41	19

Taulukko 20. Kysymyksen 2 vastausten korrelaatiot muiden kysymysten vastauksiin loppukyselyssä.

Taulukosta nähdään, että tilastollisesti merkitsevä positiivinen (*Pearson Correlation*-rivin luku on suurempaa kuin nolla) korrelaatio löytyy neljän kysymyksen kohdalta:

kysymyksen 3 kohdalta (”Annetut ohjeet olivat selkeitä”), kysymyksen 4 kohdalta (”Oppituntien aloitukset olivat hyviä”), kysymyksen 7 kohdalta (”Minulle on hyötyä siitä, että osaan suunnitella ja ohjelmoida pelejä”) sekä kysymyksen 14 kohdalta (”Kurssi vastasi odotuksiani”). Niinpä oppilaiden kokonaistyytyväisyys vaikutti merkittävästi myös siihen, kuinka selkeinä he kokivat annetut ohjeet, hyvinä oppituntien aloitukset, sekä kuinka hyödyllinen kurssi heidän mielestään oli ja myös kuinka hyvin se vastasi heidän odotuksiaan. Kysymysten 3 ja 14 kohdalla korrelaatio on vieläpä melko voimakas (molemmissa noin 0,65).

Kysymyksessä 3 siis tiedusteltiin, olivatko annetut ohjeet selkeitä. Tarkastellaan seuraavaksi tämän kysymyksen korrelaatioita muihin kysymyksiin nähden (taulukko 21), pois lukien jo käsitelty kysymys 2.

		k4	k5	k6	k7	k14	k16	k17 (vain 2. kurssi)
Annetut ohjeet olivat selkeitä	Pearson Correlation	,606(**)	,114	,048	,380(*)	,755(**)	,180	-,135
	Sig. (2-tailed)	,000	,476	,764	,016	,000	,260	,581
	N	40	41	41	40	41	41	19

Taulukko 21. Kysymyksen 3 vastausten korrelaatiot muiden kysymysten vastauksiin loppukyselyssä.

Tilastollisesti merkitsevä yhteys löytyy kysymysten 4 (”Oppituntien aloitukset olivat hyviä”), ja 14 (”Kurssi vastasi odotuksiani”) kanssa. Korrelaatio on positiivinen, joten mitä selkeämpinä oppilas koki ohjeet, sitä korkeammin pistein hän arvioi oppitunnin aloitukset sekä kurssin vastaavuuden suhteessa odotuksiin.

Tilastollisesti melkein merkitsevä yhteys löytyy kysymyksen 7 kanssa (”Minulle on hyötyä siitä, että osaan suunnitella ja ohjelmoida pelejä”).

Tarkastellaan vielä kysymyksen 4 (”Oppituntien aloitukset olivat hyviä”) korrelaatioita muihin kysymyksiin nähden (taulukko 22), pois lukien jo käsitellyt kysymykset 2 ja 3.

		k5	k6	k7	k14	k16	k17 (vain 2. kurssi)
Oppituntien aloitukset olivat hyviä	Pearson Correlation	,119	,387(*)	,252	,522(**)	,111	,067
	Sig. (2-tailed)	,463	,014	,122	,001	,495	,786
	N	40	40	39	40	40	19

Taulukko 22. Kysymyksen 4 vastausten korrelaatiot muiden kysymysten vastauksiin loppukyselyssä.

Tilastollisesti merkittävä vastaavuus löytyy kysymyksen 14 kanssa (”Kurssi vastasi odotuksiani”), ja melkein merkitsevä yhteys kysymyksen 6 kanssa (”Tunnilla oli mukava ilmapiiri”). Edelleen korrelaatiot ovat positiivisia, joten mitä parempina oppilas koki oppituntien aloitukset, sitä todennäköisemmin kurssi vastasi myös hänen odotuksiaan, ja hän myös koki oppituntien ilmapiirin hyvänä.

Edelleen tarkastellaan kysymyksen 7 (”Minulle on hyötyä siitä, että osaan suunnitella ja ohjelmoida pelejä”) korrelaatioita muihin kysymyksiin nähden (taulukko 23), pois lukien jo käsitellyt kysymykset 2, 3 ja 4.

		k5	k6	k14	k16	k17 (vain 2. kurssi)
Minulle on hyötyä siitä, että osaan suunnitella ja ohjelmoida pelejä	Pearson Correlation	,131	,058	,378(*)	,292	-,192
	Sig. (2-tailed)	,421	,723	,016	,068	,446
	N	40	40	40	40	18

Taulukko 23. Kysymyksen 7 vastausten korrelaatiot muiden kysymysten vastauksiin loppukyselyssä.

Oppilaan kokema hyöty pelien suunnittelusta ja ohjelmoinnista korreloi tilastollisesti melkein merkitsevästi kysymyksen 14 (”Kurssi vastasi odotuksiani”) kanssa. Voidaan siis

sanoa (varauksin), että mitä hyödyllisempänä oppilas koki kurssilla saadun opin, sitä todennäköisemmin kurssi myös vastasi oppilaan odotuksia (ja päinvastoin).

8.6 Loppukyselyn monivastausmuuttujan analysointi

Tässä luvussa tarkastellaan vielä ristiintaulukoinnin avulla loppukyselyn kysymystä 8, jossa vastausvaihtoehtoja oli useampia kuitenkin siten, että analysoitavaksi valittiin se aineisto, jossa tulokset olivat tavalla tai toisella mielenkiintoisia. Taustamuuttujiksi valittiin sukupuoli, ikä ja aiempi ohjelmointikokemus. Oppilaiden vastausten yhteenveto on esitetty edellisessä luvussa. Oppilaalla oli mahdollisuus valita vastausvaihtoehdoista joko ei yhtään, yksi tai useampia.

Kysymys 8: Sain tältä kurssilta

Sukupuolella ei näytä olevan juuri muuta tekemistä kurssilta saatujen asioiden suhteen, kuin että pojista kahdeksan ilmoitti saaneensa tietoa ohjelmoijan ammatista, tytöistä ei yksikään. Eniten vastauksia saaneet kohdat on sukupuolittain lihavoituna taulukossa 24.

			Sukupuoli		
			Tyttö	Poika	Total
Sain tältä kurssilta	**Tietoa ohjelmoinnista**	Count	**6**	**33**	39
		% within sukupuoli	**100,0%**	**94,3%**	
	Tietoa pelien tekemisestä	Count	**6**	**33**	39
		% within sukupuoli	**100,0%**	**94,3%**	
	Tietoa pelien suunnittelusta	Count	**6**	30	36
		% within sukupuoli	**100,0%**	85,7%	
	Uusia kavereita	Count	4	11	15
		% within sukupuoli	66,7%	31,4%	
	Tietoa yliopistosta	Count	2	17	19
		% within sukupuoli	33,3%	48,6%	
	Tietoa luonnontieteiden opiskelusta	Count	1	3	4
		% within sukupuoli	16,7%	8,6%	
	Tietoa ohjelmoijan ammatista	Count	0	8	8
		% within sukupuoli	,0%	22,9%	
Total		Count	6	35	41

Taulukko 24. Kysymys 8 ja sukupuoli ristiintaulukoituna.

Iällä ja aikaisemmalla ohjelmointikokemuksella ei näyttänyt olevan mainittavaa merkitystä suhteessa kysymyksen 8 vastauksiin.

8.7 Alkukyselyn ja loppukyselyn korrelaatioanalyysi

Tässä luvussa selvitetään korrelaatioita tarkastelemalla, onko alku- ja loppukyselyiden vastausten välillä tilastollisesti merkitseviä yhteyksiä.

Taulukossa 25 on esitetty, miten vastaajan sukupuoli korreloi loppukyselyn kysymysten vastausten kanssa.

		k2	k3	k4	k5	k6	k7	k14	k16
Sukupuoli	Pearson Correlation	,028	,251	,096	-,441(**)	-,006	,140	,203	,452(**)
	Sig. (2-tailed)	,861	,114	,556	,004	,973	,387	,202	,003
	N	41	41	40	41	41	40	41	41

Taulukko 25. Sukupuolen korrelaatiot muiden kysymysten vastauksiin loppukyselyssä.

Taulukosta nähdään, että kysymyksen 5 (”Tehtävät olivat vaikeita”) kohdalla on tilastollisesti merkitsevä negatiivinen korrelaatio ja kysymyksen 16 (”Aion hakeutua luonnontieteiden pariin […] ”) kohdalla tilastollisesti merkitsevä positiivinen korrelaatio. Vastauksissa tytöt koodattiin 0:ksi ja pojat 1:ksi, joten voidaan päätellä tyttöjen mielestä tehtävät olivat vaikeampia. Vastaavasti pojat aikovat kurssin päätöskyselynkin jälkeen todennäköisemmin hakeutua luonnontieteellisille aloille opiskelemaan.

Ikä vaikutti melkein merkitsevästi siihen, kuinka miellyttävänä oppilas koki oppituntien ilmapiirin. Korrelaatio oli negatiivinen, eli mitä nuorempi oppilas, sitä epämiellyttävämpänä hän koki oppituntien ilmapiirin (taulukko 26).

		k2	k3	k4	k5	k6	k7	k14	k16
Ikä vuosina	Pearson Correlation	-,088	-,003	-,081	-,179	-,343(*)	-,029	,080	,267
	Sig. (2-tailed)	,596	,984	,628	,275	,032	,862	,629	,101
	N	39	39	38	39	39	38	39	39

Taulukko 26. Oppilaan iän korrelaatiot muiden kysymysten vastauksiin loppukyselyssä.

Aikaisempi ohjelmointikokemus korreloi myös joidenkin loppukyselyn vastausten kanssa (taulukko 27).

		k2	k3	k4	k5	k6	k7	k14	k16
Minulla on aikaisempaa ohjelmointikokemusta	Pearson Correlation	,012	,174	-,034	-,462(**)	,033	,184	,180	,489(**)
	Sig. (2-tailed)	,942	,277	,833	,002	,837	,256	,259	,001
	N	41	41	40	41	41	40	41	41

Taulukko 27. Aikaisemman ohjelmointikokemuksen korrelaatiot muiden kysymysten vastauksiin loppukyselyssä.

Kysymysten 5 ja 16 kanssa löydetään tilastollisesti merkitsevä korrelaatio edellisen taulukon tapaan. Toisin sanoen, mitä enemmän aikaisempaa ohjelmointikokemusta oli, sitä helpompia tehtävät oppilaalle olivat, ja sitä todennäköisemmin hän kurssin jälkeenkin on halukas hakeutumaan luonnontieteelliselle alalle opiskelemaan.

Vastaavat korrelaatiot löydettiin myös alkukyselyn kysymyksen ”Koen olevani taitava tietokoneiden ja teknisten laitteiden käyttäjä” sekä ”Aion hakeutua luonnontieteiden pariin opiskelemaan” ja loppukyselyn kysymysten 5 ja 16 kanssa. Näiden korrelaatiot ovat tilastollisesti melkein merkitseviä, paitsi alku- ja loppukyselyn sama kysymys luonnontieteiden pariin hakeutumisesta saa tilastollisesti erittäin merkitsevän (p=0.000) positiivisen korrelaation 0,744.

Alku- ja loppukyselyn korrelaatioanalyysi kertoi kokonaisuudessaan, että sukupuoli, aikaisempi ohjelmointikokemus ja tietotekniikan käyttötaito vaikuttivat ainakin jonkin

verran siihen, kuinka vaikeina oppilas koki tehtävät. Nuoremmat oppilaat eivät olleet aivan niin tyytyväisiä oppituntien ilmapiiriin.

8.8 Klusterianalyysi

Alku- ja loppukyselyille tehtiin hierarkkinen klusterianalyysi. Analyysin avulla oli tarkoitus selvittää löytyikö vastaajien joukosta tiettyjä ryhmiä, jotka olisivat vastanneet suunnilleen samalla tavalla esitettyihin alku- ja loppukyselyn kysymyksiin. Mahdollisesti löytyneiden ryhmien pohjalta voitaisiin sitten tehdä päätelmiä tyypillisestä (tai tyypillisistä) oppilastapauksista. Klusterianalyysi tehtiin Wardin menetelmällä.

Kaikkiaan kurssilla aloitti 45 oppilasta, 4 keskeytti kurssin, ja 6 oppilaan kohdalla puuttui jokin klusterianalyysissä käsiteltävistä vastauksista. Kaiken kaikkiaan siis 10 vastaajaa 45:stä olisi jouduttu poistamaan analyysistä. Aineiston pienuuden johdosta päädyttiin siihen, että yksittäiset puuttuvat vastaukset koodattiin uudelleen siten, että niihin sijoitettiin kysymyksen tyyppiarvo (moodi). Näin klusterianalyysiin kelpaavien vastaajien määrä saatiin nostettua 41:een.

Klusterianalyysia ei tehty monivastausmuuttujille (kysymykset, joissa on useampi vastausvaihtoehto), vaan ne on analysoitu erikseen luvuissa 8.3 ja 8.6.

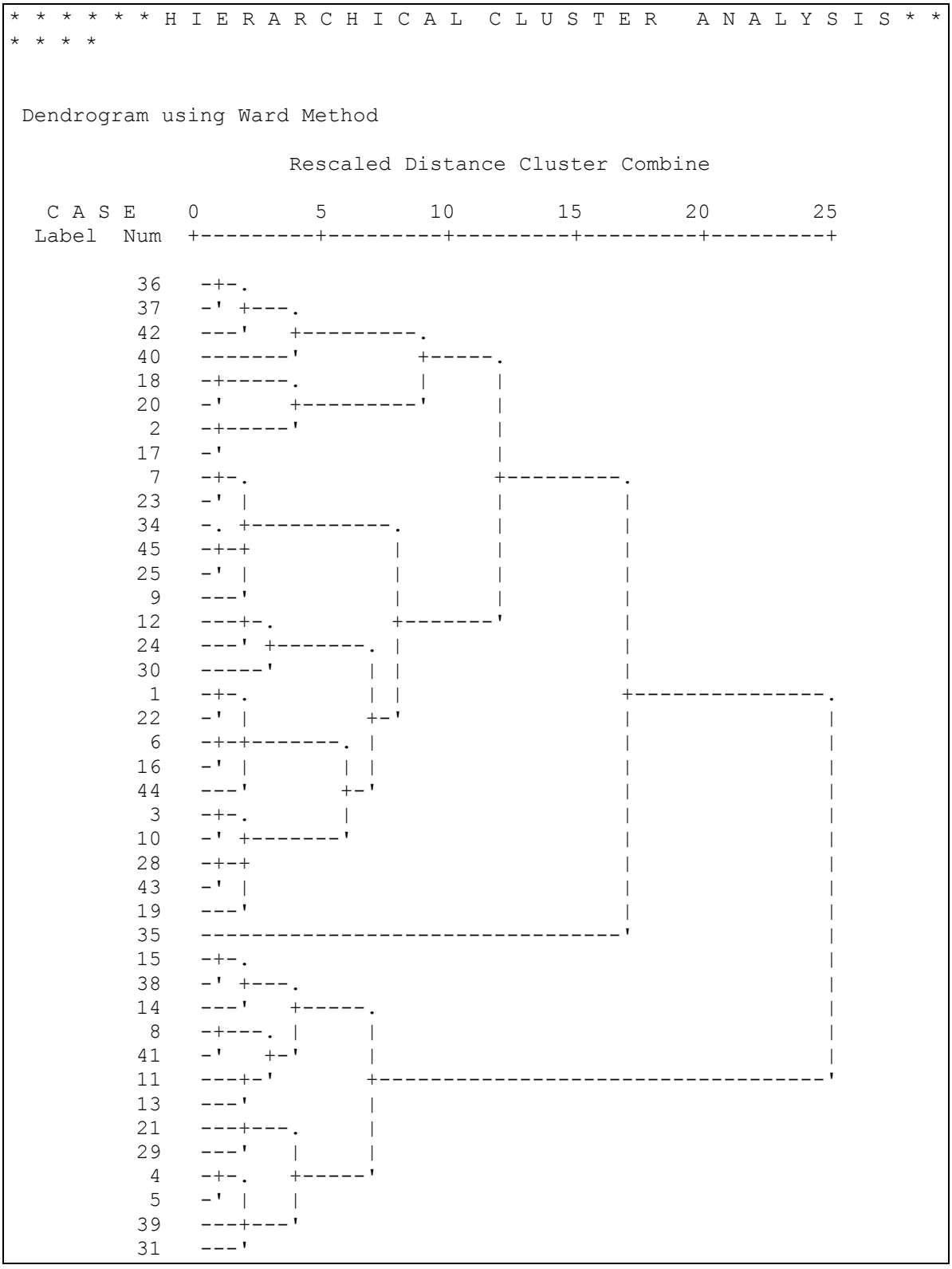
Klustereiden määrää voidaan arvioida kasautumataulukon (*agglomeration schedule*, taulukko 28) avulla. Kasautumataulukossa Coefficients-sarake mittaa vastausten keskiarvojen euklidisen etäisyyden neliötä toisistaan. Mitä suurempi kahden perättäisen rivin erotus kyseisessä sarakkeessa on, sitä varmemmin klustereita (suurempia ryhmiä) muodostuu. Erotus alkaa tässä tapauksessa olla merkittävä vaiheiden (*stage*) 37 ja 38 välillä, ja hyvin merkittävä vaiheiden 39 ja 40 välillä.

Agglomeration Schedule

Stage	Cluster Combined		Coefficients	Stage Cluster First Appears		Next Stage
	Cluster 1	Cluster 2		Cluster 2	Cluster 1	
1	36	37	1,500	0	0	25
2	18	20	3,000	0	0	32
Vaiheet 3-29 on poistettu, sillä ne eivät tuo lisäinformaatiota taulukon tulkinnassa.						
30	4	21	184,590	23	21	35
31	36	40	201,090	25	0	37
32	2	18	217,590	11	2	37
33	1	3	240,590	22	26	34
34	1	12	265,703	33	28	36
35	4	8	293,962	30	29	40
36	1	7	323,194	34	19	38
37	2	36	356,444	32	31	38
38	1	2	402,171	36	37	39
39	1	35	467,151	38	0	40
40	1	4	566,927	39	35	0

Taulukko 28. Klusterianalyysin kasautumataulukko eli *agglomeration schedule*.

Tarkastellaan vielä hierarkkisen klusterianalyysin antamaa dendogrammia (kuva 13). Dendogrammi on puumainen hierarkiakuvaaja, jolla saadaan esiin ryhmien hierarkkinen rakenne. Mitä pidempi viiva vaakasuunnassa on, sen varmemmin kyseessä todella on klusteri eli ryhmä. Dendogrammi on hyvä apuväline klustereiden määrän arvioimiseen. Tässä tapauksessa näyttäisi, että klustereita olisi kahdesta viiteen.



Kuva 13. Klusterianalyysin dendogrammi.

Seuraavaksi kokeiltiin, minkä kokoisia klustereita erilaisilla klusterimäärillä muodostuisi, eli tehtiin ns. *quick cluster analysis*. Kuudella klusterilla ryhmistä tulee vielä liian pieniä (taulukko 29)

Number of Cases in each Cluster

Cluster	1	17,000
	2	7,000
	3	1,000
	4	4,000
	5	8,000
	6	4,000
Valid		41,000
Missing		4,000

Taulukko 29. Tapausten sijoittuminen klustereihin kuudella klusterilla.

Niin ikään viidellä klusterilla saadaan vielä kaksi hyvin pientä klusteria (taulukko 30), joten viisikään ei ole mielekäs vaihtoehto klusterien määräksi.

Number of Cases in each Cluster

Cluster	1	11,000
	2	1,000
	3	17,000
	4	9,000
	5	3,000
Valid		41,000
Missing		4,000

Taulukko 30. Tapausten sijoittuminen klustereihin viidellä klusterilla.

Vähennetään vielä klusterien määrää (taulukko 31). Näin saadaan kaksi hieman isompaa klusteria (22 tapausta ja 11 tapausta), yksi vähän pienempi (7 tapausta) ja yksi yhden tapauksen klusteri.

Number of Cases in each Cluster

Cluster	1	22,000
	2	7,000
	3	1,000
	4	11,000
Valid		41,000
Missing		4,000

Taulukko 31. Tapausten sijoittuminen klustereihin neljällä klusterilla.

Toisin sanoen yhden oppilaan vastaukset poikkeavat niin merkittävästi muiden oppilaiden vastauksista, että hänestä muodostuu tilastollisesti oma ryhmänsä.

Niin sanotulla K-means-klusterianalyysillä selvitettiin kunkin klusterin keskiarvot – nämä on esitetty taulukossa 32. Tutkittaessa klustereihin sijoittuneiden tapausten keskiarvoja huomataan, että ”yksikseen” jääneen oppilaan vastauksissa on joitakin epäjohdonmukaisuuksia, joten kyseisen oppilaan vastaukset eivät välttämättä ole sitä mitä pitäisi. Tämä voi johtua siitä, että oppilas ei ole ymmärtänyt kysymyksiä, tai ei jostain syystä ole halunnut tai voinut vastata kysymyksiin totuudenmukaisesti. Tämä tuli ilmi sanallisessa palautteessa, mitä kerättiin kurssin aikana kunkin päivän päätteeksi. Näin ollen on perusteltua jättää klusteri numero 3 huomiotta analyysissä. Seuraavassa taulukossa se kuitenkin esitetään täydellisyyden vuoksi.

Final Cluster Centers

	Cluster			
	1	2	3	4
Sukupuoli	1	1	1	1
Ikä vuosina	14	15	13	13
Pelaan tietokone-/konsolipelejä	5	5	4	4
Minulla on aikaisempaa ohjelmointikokemusta	1	1	1	0
Koen olevani taitava tietokoneiden ja teknisten laitteiden käyttäjä	4	4	4	3
Harrastan lukemista	4	3	5	4
Aion hakeutua luonnontieteiden opiskelun pariin tulevaisuudessa (esim. ammattikoulun luonnontieteellinen linja, tai lukiossa pitkä matematiikka/fysiikka/kemia/tietotekniikka)	3	4	3	2
Olen tyytyväinen kurssiin kokonaisuutena	5	4	2	5
Annetut ohjeet olivat selkeitä	4	3	1	4
Oppituntien aloitukset olivat hyviä	5	3	1	4
Tehtävät olivat vaikeita	2	3	1	4
Tunnilla oli mukava ilmapiiri	5	4	5	5
Minulle on hyötyä siitä, että osaan suunnitella ja ohjelmoida pelejä	4	4	1	4
Kurssi vastasi odotuksiani	5	3	1	4
Aion hakeutua luonnontieteiden opiskelun pariin	4	4	1	3

Taulukko 32. Annettujen vastausten keskiarvot (*cluster centers*) klustereittain.

Tällä tavalla oppilaat jaoteltaisiin mielekkäästi kolmeen ryhmään, eli klustereihin 1, 2 ja 4. Oppilaat ovat jakautuneet ryhmiin jokseenkin tasaisesti, ja vastausten keskiarvot eroavat riittävästi toisistaan eri kysymyksissä. Näyttäisi siis siltä, että kolme klusteria (neljä, josta poistettu yksi) on tässä tapauksessa oikea määrä.

Edellisessä taulukossa on siis kunkin klusterin tyyppiarvot. Tulkitaan seuraavaksi kuhunkin ryhmään saatuja tyyppiarvoja siten, kuin ne olisivat tyypillisiä oppilaita. Huomaa, että nämä tyypit ovat yleistyksiä ja ryhmien (klustereiden) keskiarvoja, joten ne eivät kuvaa koko oppilasjoukkoa. Ne kuitenkin antavat yleiskuvan siitä, ketä kursseilla oli.

- Tyyppi 1 (22 tapausta, klusteri 1): Tyypillisin oppilas on 14-vuotias poika, joka pelaa päivittäin pelejä. Hänellä on jonkin verran aikaisempaa ohjelmointikokemusta ja kokee itsensä muutenkin melko taitavana tietotekniikan käyttäjänä. Ennen kurssia hän ei osaa sanoa, kiinnostaako häntä hakeutua myöhemmin luonnontieteelliselle alalle opiskelemaan, mutta kurssin päätteeksi hänen kiinnostuksensa on jonkin verran kasvanut. Hän on erittäin tyytyväinen kurssiin kokonaisuutena, niin ilmapiiriin, ohjeisiin kuin oppitunteihin, eikä pidä kurssilla tehtyjä harjoituksia erityisen vaikeina. Kurssi vastaa hänen odotuksiaan kaikin puolin ja hän kokee kurssilla saadun opin hyödylliseksi.
- Tyyppi 2 (11 tapausta, klusteri 4): Toiseksi tyypillisin tapaus on 13-vuotias poika, joka pelaa hieman harvemmin, hänellä ei ole aikaisempaa ohjelmointikokemusta, ja on muutenkin epävarmempi tietotekniikan käyttötaidoistaan. Ennen kurssia hän on sitä mieltä, että melko varmasti *ei* aio hakeutua luonnontieteelliselle alalle opiskelemaan, mutta kurssin jälkeen kiinnostus on hieman lisääntynyt. Hän on kurssiin erittäin tyytyväinen, mutta pitää tehtäviä melko vaikeina. Kurssi vastaa melko hyvin odotuksia ja oppilas kokee kurssin muutenkin hyödyllisenä.
- Tyyppi 3 (7 tapausta, klusteri 2): Kolmas tapaus oppilas on 15-vuotias poika, joka jälleen pelaa päivittäin. Hän on jo ennen kurssia asennoitunut suuntaamaan opintojaan luonnontieteellisille aloille. Hänellä on ensimmäisen tapaan myös jonkin verran ohjelmointikokemusta ja hän on taitava tietokoneiden käyttäjä. Hän ei pidä kurssia niin onnistuneena ohjeiden ja oppituntien osalta, eikä osaa varmasti sanoa vastasiko kurssi hänen odotuksiaan. Hän kuitenkin kokee kurssin hyödyllisenä.

Seuraavaksi tarkastellaan eri sukupuolien ja eri-ikäisten oppilaiden sijoittumista eri klustereihin. Lisäksi tarkastellaan oliko kurssien välillä eroa suhteessa klustereihin sijoittumiseen – tämä voisi antaa myös vihiä siitä, että kurssien sisältö oli jossain määrin erilainen.

Taulukossa 33 on ristiintaulukoitu eri sukupuolten vastaajat eri klustereihin. Ristiintaulukoinnille tehtiin myös khiin neliötesti (taulukko 34), millä selvitettiin, oliko vastaajien eri klustereihin sijoittuminen tilastollisesti merkitsevää sukupuolen kannalta. Klusterit on numeroitu kuten taulukossa 29.

Sukupuoli * Klusteri Crosstabulation

Count

		Klusteri			
		1	2	4	Total
Sukupuoli	Poika	22	5	7	34
	Tyttö	0	2	4	6
Total		22	7	11	40

Taulukko 33. Sukupuolet ja klusterit ristiintaulukoituna.

Tyttöjä oli varsin vähän mukana kursseilla, mutta kuitenkin neljä kuudesta tytöistä sijoittui klusteri 4:ään. Vastaavasti suuri enemmistö pojista sijoittui klusteri 1:een. Tilastollista merkitsevyyttä ilmaiseva p-arvo on 0,012, joten sukupuolen voidaan sanoa olevan tilastollisesti melkein merkitsevä oppilaiden sijoittumisessa klustereihin.

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)
Pearson Chi-Square	8,831(a)	2	,012
Likelihood Ratio	11,020	2	,004
Linear-by-Linear Association	7,415	1	,006
N of Valid Cases	40		

Taulukko 34. Khiin neliötesti sukupuolen tilastolliselle merkitsevyydelle suhteessa klustereihin sijoittumiseen.

Iän suhteen oppilaat ovat jakautuneet kohtalaisen tasaisesti eri klustereihin (taulukko 35), eikä ikä ei ole tilastollisesti merkitsevä tekijä klustereihin jakautumisessa (taulukko 36).

Ikä vuosina * Klusteri Crosstabulation

Count

		Klusteri			
		1	2	4	Total
Ikä vuosina	16	2	2	1	5
	15	3	2	1	6
	14	7	3	3	13
	13	9	0	3	12
	12	1	0	3	4
Total		22	7	11	40

Taulukko 35. Ikävuodet ja klusterit ristiintaulukoituna.

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)
Pearson Chi-Square	10,859(a)	8	,210
Likelihood Ratio	12,165	8	,144
Linear-by-Linear Association	,706	1	,401
N of Valid Cases	40		

Taulukko 36. Khiin neliötesti iän tilastolliselle merkitsevyydelle suhteessa klustereihin sijoittumiseen.

Eri kurssin käyneet oppilaat ovat jakaantuneet hyvin tasaisesti klustereihin (taulukko 37), joten kurssinumero ei ole tilastollisesti merkitsevä tekijä klustereihin jakautumisessa (taulukko 38).

Kurssinumero * Klusteri Crosstabulation

Count

		Klusteri			
		1	2	4	Total
Kurssinumero	Kurssi 2	11	2	5	18
	Kurssi 1	11	5	6	22
Total		22	7	11	40

Taulukko 37. Kurssinumerot ja klusterit ristiintaulukoituna.

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)
Pearson Chi-Square	,986(a)	2	,611
Likelihood Ratio	1,019	2	,601
Linear-by-Linear Association	,060	1	,807
N of Valid Cases	40		

Taulukko 38. Khiin neliötesti kurssinumeron (1 tai 2) tilastolliselle merkitsevyydelle suhteessa klustereihin sijoittumiseen.

Tarkastellaan vielä aikaisemman ohjelmointikokemuksen ja klustereihin sijoittumisen välistä yhteyttä (taulukko 39).

Minulla on aikaisempaa ohjelmointikokemusta * Klusteri Crosstabulation

Count

		Klusteri			
		1	2	4	Total
Minulla on aikaisempaa ohjelmointikokemusta	Kyllä, melko paljon tai paljon	5	0	0	5
	Kyllä, mutta vain hyvin vähän	11	4	2	17
	Ei	6	3	9	18
Total		22	7	11	40

Taulukko 39. Aikaisempi ohjelmointikokemus ja klusterit ristiintaulukoituna.

Enemmistö niistä, joilla ei ole aikaisempaa ohjelmointikokemusta, sijoittui klusteri 3:een, kun taas muista oppilaista ylivoimainen enemmistö klusteri 1:een. Näyttäisi siis siltä, että ohjelmointikokemuksella ja klustereihin sijoittumisella on yhteys keskenään. Tarkistetaan asia vielä khiin neliötestillä (taulukko 40).

Chi-Square Tests

	Value	df	Asymp. Sig. (2-sided)
Pearson Chi-Square	11,123(a)	4	,025
Likelihood Ratio	12,985	4	,011
Linear-by-Linear Association	9,096	1	,003
N of Valid Cases	40		

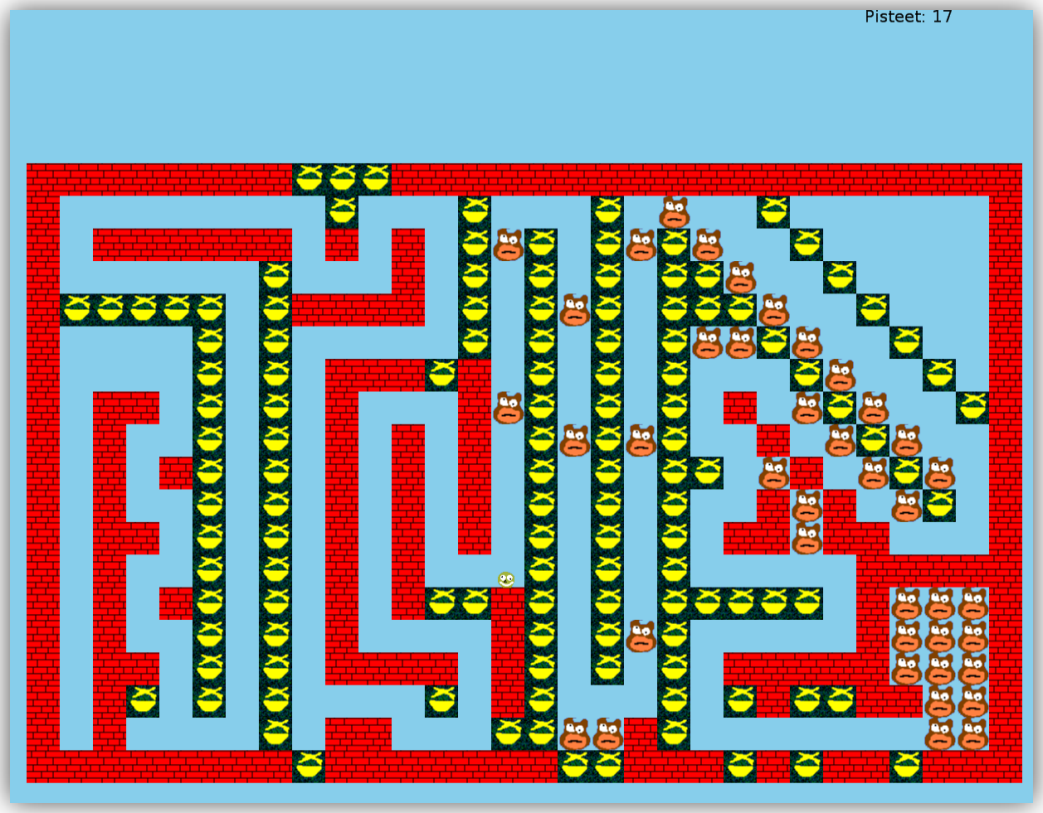
Taulukko 40. Khiin neliötesti aikaisemman ohjelmointikokemuksen tilastolliselle merkitsevyydelle suhteessa klustereihin sijoittumiseen.

Testi antaa p-arvoksi 0,025 eli tulos on tilastollisesti melkein merkitsevä.

Kokonaisuudessaan klusterianalyysin perusteella voidaan sanoa, että aineistosta löytyi ainakin kolme selkeää klusteria. Aikaisempi ohjelmointikokemus vaikutti tilastollisesti melkein merkitsevästi klustereihin jakautumiseen, kuten myös eri sukupuolten jakautuminen klustereihin. Ikä tai kurssinumero (kurssin ajankohta) ei vaikuttanut klustereihin sijoittumiseen.

8.9 Oppilaiden tuotokset

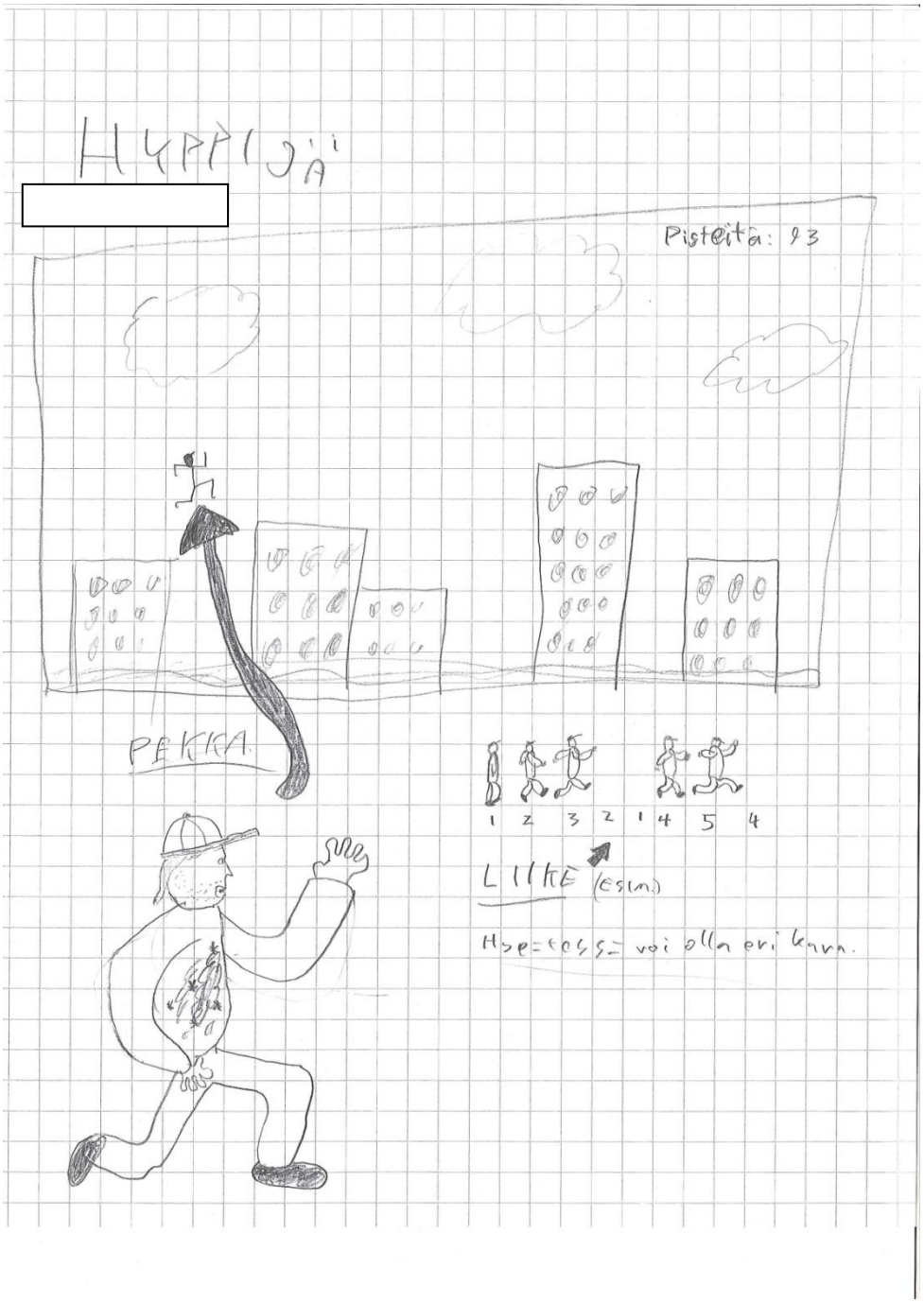
Viimeisenä päivänä oppilaille annettiin mahdollisuus esitellä oma pelinsä ja kertoa hieman sen toteutuksesta. Pelien esittelyn jälkeen oppilaat äänestivät mielestään parhaan pelin, ja se, kuka sai eniten pisteitä, palkittiin pienellä palkinnolla. Ensimmäisen kurssin parhaan pelin palkinnon sai yksinkertaisuudessaan omaperäinen taitopeli, jossa liikuteltiin pelaajaa labyrintissa (kuva 14). Pelin suunnitelma on liitteessä 10 (oppilaat muuttivat suunnitelmaa kurssin varrella, ja peli onkin hyvin erilainen kuin mitä suunnitelma antaa ymmärtää). Pelaajan oli vältettävä koskemasta keltaisia ruutuja, sillä niihin osuessaan pelaaja palautui takaisin alkupisteeseen. Apinan päitä keräämällä sai pisteitä, ja jos sai kaikki apinat kerättyä, pääsi pelin läpi.



Kuva 14. Ensimmäisen kurssin äänestyksen voittaja.

Kakkoseksi äänestettiin vauhdikas tasohyppelypeli, jossa pelaaja hyppi talon katolta toiselle (kuva 15). Pelin tavoitteena oli pelastaa kaupunki tulvalta sammuttamalla eräs tärkeä kytkin (eli pääsemällä maaliin). Pelaajan on pysyttävä jatkuvasti liikkuvan kameran liikkeessä mukana, tai muuten peli päättyy. Katoilta ei myöskään saa tipahtaa. Kirjoitettu suunnitelma on liitteessä 11. Suunnitelmakuva on esitetty kuvassa 16.

Kuva 15. Ensimmäisen kurssin äänestyksen kakkonen.

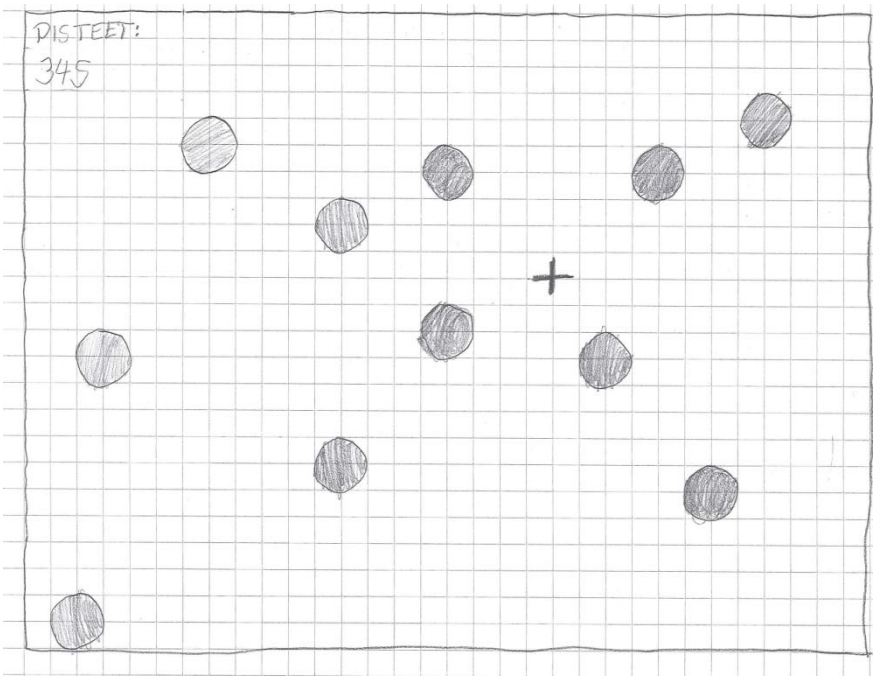


Kuva 16. Hahmotelma pelistä, joka äänestettiin ensimmäisen kurssin kakkoseksi. Suunnitelmassa näkyy mm. pistelaskuri, animaation eri vaiheet sekä pelihahmo yksityiskohtineen.

Kolmanneksi parhaaksi peliksi äänestettiin peli, missä liikkuvia palloja piti ”metsästää” ampumalla niitä hiiren kursorin avulla (kuva 17).

Kuva 17. Ensimmäisen kurssin äänestyksen kolmonen.

Mikäli tehtävässä onnistui alle 30 sekunnin aikana, pääsi mahdollisesti parhaiden pelaajien listalle, johon voi sitten laittaa oman nimensä. Pelin suunnitelmakuva on esitetty kuvassa 18.



Kuva 18. Hahmotelma pelistä, joka äänestettiin ensimmäisen kurssin kolmoseksi. Suunnitelmassa näkyy mm. pistelaskuri, tähtäin sekä pallot jotka ovat jakautuneet satunnaisesti ympäri näyttöruutua.

8.10 Ohjaajien rooli kurssilla

Yksi keskeisistä työskentelyyn liittyvistä erikoispiirteistä kurssilla oli jatkuva intensiivinen ohjaus. Kurssilla oli vastaavan opettajan lisäksi neljä ohjaajaa, jotka antoivat oppilaille henkilökohtaista ohjausta harjoitustehtävien tekemisen aikana, sekä oman pelin suunnittelun ja toteuttamisen aikana. Ohjaajat toimivat osaltaan myös opettajina näissä ohjaustilanteissa, ja heillä oli varsin keskeinen rooli pelien toteuttamisen avustamisessa.

Joissain tapauksissa oppilaan lähestymistapa ongelmaan saattoi olla niin monimutkainen, että ohjaaja tyytyi antamaan vastauksen oppilaan kysymykseen sellaisenaan. Tämä saattoi vaikuttaa siihen, etteivät oppilaat kovin omatoimisesti yrittäneet etsiä ratkaisua wiki-sivustolta, johon ohjaajat olivat kirjoittaneet runsaasti ohjemateriaalia.

9 Pohdinta

Tässä työssä tutkittiin 13–16-vuotiaille suunnatun peliohjelmointikurssin innostavuutta, sekä kurssilla esiintyneitä oppilastyyppejä. Tutkimusta varten kerättiin teoriatietoa eri ohjelmointiparadigmoista, ohjelmoinnin opettamisesta ja oppimisesta, alkeisohjelmoinnin opettamiseen soveltuvista pedagogisista menetelmistä sekä erilaisista tutkimusmenetelmistä. Suomessa ei ollut aiemmin järjestetty nuorille suunnattuja peliohjelmointikursseja, joten teoriaa peliohjelmoinnin opettamisesta nuorille kerättiin yleisesti ohjelmoinnin opetuksen tutkimustiedosta.

Tutkimuksen empiirisessä osuudessa hyödynnettiin teoriaosuudessa saatua tietoa, ja toteutettiin yläkouluikäisille (13–16-vuotiaat) suunnattu alkeisohjelmointikurssi. Tutkimusaineistoa kerättiin kurssilla eri tavoin, ja keräämisen jälkeen tutkimusta rajattiin siten, että tutkimusmenetelmäksi valittiin survey-tutkimus. Tässä työssä analysoitiin vain kurssin aluksi ja lopuksi tehdyt kyselyt oppilaille, muun aineiston analysointi jätettiin jatkotutkimuksen varaan.

Tutkimukseen valittiin kaksi teemaa: pelikeskeisen ohjelmointikurssin innostavuus ja pelikurssin oppilastyypit. Teemoihin asetettiin yhteensä kolme päätutkimuskysymystä, ja neljä alakysymystä.

9.1 Pelikeskeisen ohjelmointikurssin innostavuus

Tässä työssä tutkitun viisipäiväisen lyhytkurssin lähestymistapa ohjelmointiin oli hyvin motivaatiopainotteinen. Oppilaille annettiin mahdollisuus suunnitella ja tehdä oma peli, ja itse asiassa jokaiselle oppilaalle annettiin lupaus siitä, että jokainen pystyy tekemään pelin, vaikka ei olisi koskaan ennen ohjelmoinut. Tutkimuksessa selvisi, että kurssi onnistui erinomaisesti niin oppituntien kuin harjoitustehtävien osalta. Kokonaisuutena kurssi koettiin erittäin mielekkääksi ja innostavaksi (tutkimuskysymys 1).

Toisaalta mitä enemmän oppilaalla oli ennakkotietoa ohjelmoinnista tai mitä parempana tietotekniikan käyttäjänä hän itsensä koki, sitä helpompia tehtävät olivat hänelle. Aikaisempi ohjelmointikokemus tai muutkaan taustatekijät eivät kuitenkaan vaikuttaneet kokonaistyytyväisyyteen (keskiarvot kursseittain 4,8 ja 4,6), joten voidaan sanoa, että

yleisesti ottaen oppilaat olivat hyvin tyytyväisiä kurssiin taustoista riippumatta (tutkimuskysymykset 3 ja 4).

Oppilaiden kiinnostuksessa valita matemaattis-luonnontieteellisiä oppiaineita jatko-opintoihinsa tapahtui lisäystä (tutkimuskysymys 2). Jopa 25 oppilaan kiinnostus nousi kurssin mittaan, ja kuusi oppilasta, jotka aikaisemmin eivät osoittaneet kiinnostusta luonnontieteiden opiskeluun, olivat kiinnostuneet asiasta kurssin päätökseen mennessä (kiinnostuksen aste noussut arvosta 1, 2 tai 3 arvoon 4 tai 5, asteikko 1–5). Kahden oppilaan kiinnostus laski yhden pykälän kurssin aikana. Tämän tutkimuksen nojalla siis näyttää siltä, että kurssi lisää oppilaiden kiinnostusta luonnontieteiden opiskelua kohtaan, mutta tämän asian vahvistaminen vaatisi luonnollisesti seurantatutkimusta siitä, päätyvätkö he todella opiskelijoiksi näille aloille.

Näyttää siltä, että "rehellisesti ohjelmakoodia kirjoittamalla" voi saada aikaan toimivia pelejä tietyin edellytyksin, mutta ennen kaikkea tällainen kurssi voi tuoda onnistumisen elämyksiä oppilaille, joilla ei ole aikaisempaa kokemusta ohjelmoinnista. En toki väitä, etteivätkö visuaaliset pelistudiot antavat myös onnistumisen elämyksiä. Täytyy muistaa, että erittäinkin yksinkertaisen pelin valmiiksi saaminen itse koodaamalla vaatii jo muutamien ohjelmoinnin käsitteen (vähintään) pintapuolista ymmärtämistä, ja esimerkiksi ensimmäisen ”Pong”-pelin saaminen valmiiksi sai aikaan luokassa riemunkiljahduksia, ja tuntui monesta nuoresta itsensä ylittämiseltä.

Jotta koodia itse kirjoittamalla saadaan lyhytkurssilla jotain oikeaa ja toimivaa aikaiseksi, edellyttää että kurssilla on jonkin verran aikaa opettaa ohjelmoinnin peruskäsitteitä. Yksi tapa toteuttaa tämä on antaa oppilaiden tehdä itse ja kokeilla *ohjatusti*, sitten käydä pelin tekoon vaikkapa esimerkkikoodin avulla ja sitä muokaten (ks. luvut 4 ja 5). Jotta pelejä saadaan valmiiksi viikossa, vaatii se lisäksi asiantuntevaa ja jatkuvasti saatavilla olevaa ohjausta.

9.2 Oppilastyypit peliohjelmointikurssilla

Tutkimuksessa oppilaiden vastauksista löydettiin kolme ryhmää, joiden perusteella tutkittiin ryhmien vastausten keskiarvoja. Näiden vastausten keskiarvojen avulla saatiin

muodostettua kurssille osallistuneita tyypillisiä oppilaita (tutkimuskysymykset 5 ja 7), jotka kuvasivat yleisesti sitä joukkoa, joka kurssilla oli (ks. tarkemmat kuvaukset luvussa 8.8).

- Tyypillisin oppilas on 14-vuotias poika, joka pelaa päivittäin pelejä. Hänellä on jonkin verran aikaisempaa ohjelmointikokemusta ja kokee itsensä muutenkin melko taitavana tietotekniikan käyttäjänä. Hänen kiinnostuksensa luonnontieteellisten alojen opiskelua kohtaan kasvaa jonkin verran kurssilla. Hän on erittäin tyytyväinen kurssiin kokonaisuutena, niin ilmapiiriin, ohjeisiin kuin oppitunteihin, eikä pidä kurssilla tehtyjä harjoituksia erityisen vaikeina. Kurssi vastaa hänen odotuksiaan kaikin puolin ja hän kokee kurssilla saadun opin hyödylliseksi.
- Toiseksi tyypillisin tapaus on 13-vuotias poika, joka pelaa hieman harvemmin, hänellä ei ole aikaisempaa ohjelmointikokemusta, ja on muutenkin epävarmempi tietotekniikan käyttötaidoistaan. Ennen kurssia hän on sitä mieltä, että melko varmasti *ei* aio hakeutua luonnontieteelliselle alalle opiskelemaan, mutta kurssin jälkeen kiinnostus on hieman lisääntynyt. Hän on kurssiin erittäin tyytyväinen, mutta pitää tehtäviä melko vaikeina. Kurssi vastaa melko hyvin odotuksia ja oppilas kokee kurssin muutenkin hyödyllisenä.
- Kolmas tapaus oppilas on 15-vuotias poika, joka jälleen pelaa päivittäin. Hän on jo ennen kurssia asennoitunut suuntaamaan opintojaan luonnontieteellisille aloille. Hänellä on ensimmäisen tapaan myös jonkin verran ohjelmointikokemusta ja hän on taitava tietokoneiden käyttäjä. Hän ei pidä kurssia niin onnistuneena ohjeiden ja oppituntien osalta, eikä osaa varmasti sanoa vastasiko kurssi hänen odotuksiaan. Hän kuitenkin kokee kurssin hyödyllisenä.

Ryhmittelyn perusteella voidaan tehdä johtopäätös, että kokemus tietotekniikasta ja ikä vaikuttavat siihen, kuinka vaikeana oppilas kurssin kokee. Koska kurssi oli suunnattu yläkouluikäisille, on tietenkin mietittävä, miten esitystapaa ja sisältöä voisi kehittää siihen suuntaan, etteivät nuorimmat osallistujat tunne oloaan kurssilla epämiellyttäväksi. Vastausten perusteella nuorimmatkin osallistujat olivat kuitenkin tyytyväisiä kurssiin kokonaisuutena, joskin he kokivat tehtävät muita vaikeampina.

Kaikissa ryhmissä kiinnostus luonnontieteitä kohtaan joko nousi tai pysyi ennallaan (tutkimuskysymys 6). Ehkä merkittävintä on huomata ensimmäiseksi mainittu ryhmä, joka oli suurin, ja jossa kiinnostus luonnontieteiden opiskelua kohtaan nousi arvosta 3 (ei samaa eikä eri mieltä) arvoon 4 (jokseenkin samaa mieltä).

Suurin osa kurssilaisista oli poikia, kuten odottaa saattoi (ks. luku 1). Tyttöjen osalta huomattiin, että heitä on vaikea saada kiinnostumaan ohjelmoinnista – heitä tuli vain vähän mukaan kursseille ja yksi niistä muutamasta lopetti kesken. Jäljelle jääneet kuitenkin saivat oman pelinsä valmiiksi ja kuten tässä tutkimuksessa kävi ilmi, he olivat myös kaiken kaikkiaan tyytyväisiä kurssiin.

9.3 Kehitysehdotukset

Yksi kurssilla esiintynyt ongelma oli se, etteivät kurssilaiset ryhmäytyneet kovinkaan helposti. Jälkimmäisellä kurssilla pariutumista ja ryhmäyttämistä yritettiin painottaa enemmän, mutta pääsääntöisesti kurssilaiset työskentelivät koko ajan yksin. Kun kurssin aikataulu oli hyvin intensiivinen, ei ”kahvipöytäkeskusteluille” tai muulle sosiaaliselle toiminnalle jäänyt kovin paljon aikaa. Tämä ei varsinaisesti ollut kurssin vika vaan ennemminkin ominaisuus – kurssi oli toteutettu siten, että aikataulu on tiukka ja illaksi jokainen lähti omaan kotiinsa. Kesällä 2010 on tavoitteena järjestää peliohjelmointikurssi ensimmäisen kerran leirityyppisesti, jolloin kurssilaiset myös yöpyvät leirillä, ja ruudun äärellä istumisen lisäksi on mahdollista tehdä myös aivan jotain muuta, kuten viettää aikaa muiden kurssilaisten kanssa. Tällä tavalla voitaisiin ehkä saada osallistujille kurssista vielä parempi ja muistettavampi kokemus.

9.4 Jatkotutkimusaiheet

Kuten aiemmin todettiin, yksi jatkotutkimuksen aihe olisi se, kasvattaako tämän kaltainen kurssi todellisuudessa luonnontieteiden opiskelijoiden määrää. On tietenkin selvää, että kurssi, jolle tullaan vapaaehtoisesti, ei ole satunnaisotanta yläkouluikäisistä nuorista, vaan tietyssä mielessä hyvinkin vinoutunut otos, eikä tämän tutkimuksen tulos ole sen takia yleistettävissä.

Lähteet

ACM 2001. *Computing curricula 2001 Computer Science.* Final Report of the ACM/IEEE-CS Joint Task Force. Satavilla WWW-osoitteessa http://www.acm.org/education/education/education/curric_vols/cc2001.pdf. Viitattu 12.2.2010.

ACM 2010. *Association for Computing Machinery: ACM at a glance*. Saatavilla WWW-osoitteessa http://www.acm.org/membership/acm-at-a-glance. Viitattu 12.2.2010.

Ahonen, S. 1989. Tietokone ja tiedonkäsitys. Teoksessa: *Historian ja yhteiskuntaopin opettajien vuosikirja; 26*, toim. Hänninen, K. Sivut 7-12. Helsinki: HYOL Ry.

Alén, R., Julin, R., Karjalainen, J., Korhonen, M., Laapio, M., Matilainen, R., Nyblom, J. 2007. *Matemaattis-luonnontieteellinen tiedekunta: Vuosikatsaus 2008*. Saatavilla WWW-osoitteessa https://www.jyu.fi/science/tiedekunta/vuosikatsaukset/vk08. Viitattu 12.4.2010. Jyväskylän yliopiston matemaattis-luonnontieteellinen tiedekunta.

Anttila, P. 1996. *Tutkimisen taito ja tiedonhankinta – taito-, taide- ja muotoilualojen tutkimuksen työvälineet*. Helsinki: Akatiimi Oy.

Apple Inc 2010. *Quartz Composer Basic Concepts*. Saatavilla WWW-osoitteessa http://developer.apple.com/mac/library/documentation/GraphicsImaging/Conceptual/QuartzComposerUserGuide/qc_concepts/qc_concepts.html#//apple_ref/doc/uid/TP40005381-CH212-SW9. Viitattu 21.4.2010.

Bandura, A. 1977. *Social learning theory*. Englewood Cliffs, NJ, Yhdysvallat: Prentice Hall.

Bereiter, C. 2002. *Education and mind in the knowledge age*. Mahwah, New Jersey: Lawrence Erlbaum Associates.

Blum, L., Cortina, T. 2007. *CS4HS: An Outreach Program for Highschool CS Teachers*. Proceedings of the 38th SIGCSE technical symposium on 93 Computer science education. March 07-11. Sivut 19-23. New York: ACM Press.

Brookshear, J.G. 2003. *Tietotekniikka*. IT Press. Helsinki: Edita Prima.

Böszörményi, L. 1998. Why Java is not my favorite first-course language. *Software – Concepts & Tools*, vol. 19. Sivut 141-145. Heidelberg: Springer Berlin.

Carnegie Mellon University 2010. *What is Alice?* Saatavilla WWW-osoitteessa http://www.alice.org/index.php?page=what_is_alice/what_is_alice. Viitattu 24.4.2010.

Cobern, W. 1986. *Programming Language Choice: A Positive albeit Ambiguous Case for BASIC Programming in Secondary Science Teaching*. Sherman: Austin College.

Dahl, O.-J., Dijkstra, E. W., Hoare, C. A. R. 1972. *Structured Programming*. London: Academic Press.

Dewey, J. 1957. *Koulu ja yhteiskunta*. Alkuteos: The School and Society, 1915 (2. painos). Suom. Kalevi Kajava. Helsinki: Otava.

Dewey, J. 1966. *Democracy and education: an introduction to the philosophy of education*. Uudistettu painos. Alkuteos vuodelta 1916. New York: Free Press.

Dijkstra, E.W. 1964. On the interplay between mathematics and programming. *Lecture Notes in Computer Science*, vol. 69, 1979. Sivut 35-46. Saatavilla WWW-osoitteessa http://userweb.cs.utexas.edu/users/EWD/ewd06xx/EWD641.PDF. Viitattu 12.4.2010. Heidelberg: Springer Berlin.

Dijkstra, E. W. 1968. Go to statement considered harmful. *Communications of the ACM*, vol. 11, No. 3, March 1968. Sivut 147-148.

Dijkstra, E. W. 1976. *A discipline of programming*. Englewood Cliffs: Prentice-Hall.

DuHadway, L., Clyde, S., Recker, M., Cooley, D. 2002. A Concept-first Approach for an Introductory Computer Science Course. *Journal of computing sciences in colleges*, vol. 18, issue 2. Sivut 6-16.

ECMA International. 2006. *C# Language Specifications*. Geneve: Ecma International.

Eriksson, K. 1986. *Hoito-opin didaktiikka*. Helsinki: Sairaanhoitajien koulutussäätiö.

Galbraith, J., Hale, T. 2004. *Income Distribution and the Information Technology Bubble*. Austin: The University of Texas.

Garrido, J. M. 2003. *Object-Oriented Programming: From Problem Solving to Java*. Hingham, Massachusetts: Charles River Media.

Glaser H., Hankin C., Till D. 1984. *Principles of Functional Programming*. New Jersey: Prentice-Hall.

Hakkarainen, K., Lonka, K., Lipponen, L. 2001. *Tutkiva oppiminen. Älykkään toiminnan rajat ja niiden kehittyminen*. 1.-4. painos. Helsinki:WSOY.

Hailikari, T. 2009. *Assessing University Students' Prior Knowledge. Implications for Theory and Practice.* Väitöskirja. Helsingin yliopisto.

Harel, I. 1991. *Children Designers*. Norwood, New Jersey: Ablex.

Heikkilä, T. 1998. *Tilastollinen tutkimus*. Helsinki: Edita.

Heikkilä, H., Repo-Kaarento, S. 2000. *Oppimisen ja opetuksen laadun kartoitus ja arviointi. Väliraportti 8.5.2000.* Osa Maatalous-metsätieteellisen tiedekunnan opetuksen kehittämishanketta (Juonto) 1998–2002. Saatavilla WWW-osoitteessa http://www.mm.helsinki.fi/juonto/valirap.html#5. Viitattu 25.2.2010.

Hirsjärvi, S. (toim.) 1983. *Kasvatustieteen käsitteistö*. Helsinki: Otava.

Hirsjärvi, S., Remes, P., Sajavaara, P. 2009. *Tutki ja kirjoita*. 15., uudistettu painos. Hämeenlinna: Tammi.

Hirvelä, S. 2010. Kaupan ala nousi suosituimmaksi työnantajaksi. *Kauppalehti* 15.4.2010. Saatavilla WWW-osoitteessa http://www.kauppalehti.fi/5/i/talous/uutiset/pika/tulostus.jsp?news_list=uuti_uutiset&oid=20100401/12713076861550&news_xml=online_uuti_uutiset&type=pika&request_ahaa_info=true. Viitattu 15.4.2010.

Holopainen, M., Tenhunen, L., Vuorinen, P. 2004. *Tutkimusaineiston analysointi ja SPSS*. Hamina: TRAL ry ja Yrityssanoma Oy.

Horton, A. 2010. The *Evolution Of LINQ And Its Impact On The Design Of C#*. Saatavilla WWW-osoitteessa http://msdn.microsoft.com/en-us/magazine/cc163400.aspx. Viitattu 19.4.2010.

Huitt, W. 2009. *Bloom et al.'s taxonomy of the cognitive domain*. Educational Psychology Interactive. Saatavilla WWW-osoitteessa http://www.edpsycinteractive.org/topics/cogsys/bloom.html. Viitattu 22.4.2010. Valdosta, GA: Valdosta State University.

Hvorecký, J. 1990. On a Connection Between Programming and Mathematics. *ACM SIGCSE Bulletin*, vol. 22, nro 4. Sivut 53–64. New York: ACM.

IEEE-CS 2010. *The IEEE Computer Society*. Saatavilla WWW-osoitteessa http://www.computer.org. Viitattu 19.2.2010.

Ihanainen, E. 2010. Henkilökohtainen tiedoksianto. 11.3.2010.

ISO 23270:2003. *C# Language Specifications*. Saatavilla WWW-osoitteessa http://www.iso.org/iso/catalogue_detail.htm?csnumber=36768. Viitattu 24.4.2010.

Isomäki, H. 2010. *Tietojenkäsittelytieteen Seura ry:n kannanotto tietotekniikan opetuksesta*. Kirjelmä perusopetuksen yleisten tavoitteiden ja tuntijaon työryhmälle. Saatavilla WWW-osoitteessa http://www.oph.fi/download/121237_TKTSry-kannanotto-tuntijako1.pdf. 26.2.2010. Viitattu 14.4.2010.

ITU 2009. *World Telecommunication Indicators 2008*. Saatavilla WWW-osoitteessa http://earthtrends.wri.org/text/population-health/variable-558.html. Geneva: ITU (International Telecommunication Union).

Jyväskylän yliopisto 2009. *Opiskelijatilastoja vuosilta 2002–2009*. Jyväskylän yliopisto.

Kafai, Y. 1995. *Minds in Play: Computer Game Design as a Context for Children's Learning*. Hilisdale, New Jersey: Lawrence Erlbaum AssOciates.

Kafai, Y., Ching, C., Marshall, S. 1997. Children as Designers of Educational Multimedia Software. *Computers Educ*., vol. 29, No. 2/3. Sivut 117-126. Los Angeles: Elsevier Science Ltd.

Kafai, Y., Franke, M., Ching, C., Shih, J. 1998. Game Design as an Interactive Learning Environment for Fostering Students' and Teachers' Mathematical Inquiry. *International Journal of Computers for Mathematical Learning*, vol. 3, nro 2. Sivut 149–184. Hollanti: Springer.

Kalantzis, M., Cope, B. 2005. *Learning by design*. Melbourne, Australia: Victorian Schools Innovation Commission.

Kallio, K. P., Mäyrä, F., Kaipainen, K. 2007. *Gaming Nation? Piloting the International Study of Games Cultures in Finland*. Saatavilla WWW-osoitteessa http://tampub.uta.fi/tup/978-951-44-7141-4.pdf. Viitattu 12.4.2010. Tampereen yliopisto.

Kallio, K. P., Mäyrä, F., Kaipainen, K. 2009. *Pelikulttuurin monet kasvot*. Teoksessa J. Suominen et al (toim.) Pelitutkimuksen vuosikirja 2009. Sivut 1–15. Tampereen yliopisto.

Kallionpää, K. (toim.) 2010. *Suomen koulut rutkasti jäljessä uuden tekniikan käytössä*. Helsingin Sanomat 27.3.2010. Saatavilla WWW-muodossa http://www.hs.fi/tulosta/1135254998892. 27.3.2010. Viitattu 12.4.2010.

Kankaanranta, M., Puhakka, E. (toim.) 2006. *Kohti innovatiivista tietotekniikan opetuskäyttöä. Kansainvälisen SITES 2006 -tutkimuksen tuloksia*. Jyväskylän yliopisto koulutuksen tutkimuslaitos, Jyväskylä.

Karjalainen, A. (toim.) 2003. *Akateeminen opetussuunnitelmatyö*. Saatavilla WWW-osoitteessa http://www.oulu.fi/tutkintorakenne/tyokalut/akatops305.pdf. 1.12.2010. Viitattu 12.4.2010. Oulun yliopisto.

Kemppainen, L. 2010. *Kvalitatiivisen aineiston analyysi- ja tulkintatavat*. Saatavilla WWW-osoitteessa http://vanha.edu.utu.fi/rokl/staff/laukem/OL3/OL3_02d.pdf. Viitattu 1.2.2010.

King, K. N. 1992. *The evolution of the programming languages course.* Technical symposium of computer science education. Sivut 213-219. New York: ACM.

Khor, K. 1995. *Introduction: What is Object-Oriented Programming? IBM Smalltalk Tutorial*. Saatavilla WWW-osoitteessa http://www.inf.ufsc.br/poo/smalltalk/ibm/tutorial/oop.html. Viitattu 14.4.2010.

Kolb, D. 1984. *Experiential learning: experience as the source of learning and development*. New Jersey: Prentice-Hall.

Koskimies, K. 1997. *Pieni oliokirja*. Jyväskylä: Gummerus.

Kreft, K., Langer, A. 2003. *In Between the Curly Braces: After Java and C# - what is next?* Saatavilla WWW-osoitteessa http://www.artima.com/weblogs/viewpost.jsp?thread=6543. Viitattu 12.3.2010.

Kupiainen, R., Sintonen, S. 2009. *Medialukutaidot, osallisuus, mediakasvatus*. Helsinki: Palmenia Helsinki University Press.

Kölling, M. 1999. The problem of teaching object-oriented programming, Part I: Languages. *Journal of object-oriented programming*, vol 11, issue 8. Sivut 8–15.

Lahdelma, R., Lappalainen, V. 1999. *Olio-ohjelmointi ja C++*. Toim. Timo Tiihonen. Luentomoniste nro 2. Jyväskylä: Jyväskylän yliopisto, tietotekniikan laitos.

Lahdes, E. 1997. *Peruskoulun uusi didaktiikka*. Helsinki: Otava.

Lehrer, R., Erickson J., Connell, T. 1994. Learning by Designing Hypermedia Documents. *Computers in the Schools*, vol. 10, No 1 & 2, January 1995. Sivut 227–254.

Lloyd, J. W. 1994. *Practical Advantages of Declarative Programming*. Department of Computer Science, University of Bristol. U.K.

Lyyra, M. 2007. *Joukko-opin tie*. TV-haastattelu, asiantuntijana Prof. Tuomas Sorvali. YLE TV1, 8.10.2007. Viitattu 9.10.2007.

Malinen, P. 1985. *Opetussuunnitelmat nykyajan koulutuksessa*. Helsinki: Otava.

MAOL ry 2006. *Mitä peruskoulun päättävän oppilaan tulisi tietää tietotekniikasta*.

Meyer, B. 1993. *Towards an Object-Oriented Curriculum*. TOOLS 11 (Technology of Object-Oriented Languages and Systems). Sivut 585–594. Prentice Hall.

Microsoft 2004. *Microsoft: Next Generation of Games Starts With XNA*. Saatavilla WWW-osoitteessa https://www.microsoft.com/presspass/press/2004/mar04/03-24xnalaunchpr.mspx. Viitattu 21.4.2010

Microsoft 2010. *What is Kodu?* Saatavilla WWW-osoitteessa http://fuse.microsoft.com/kodu.html. Viitattu 21.4.2010.

Moberg, S. 1998. *Erityisopetuksen ja yleisopetuksen integraatio opettajien silmin*. Teoksessa T. Ladonlahti, A. Naukkarinen & S. Vehmas (toim.) Poikkeava vai erityinen? Erityispedagogiikan monet ulottuvuudet. Jyväskylä: Atena.

Nicholson, A. E., Fraser, K. M. 1997. Methodologies for teaching new programming languages: A case study teaching LISP. *ACM international conference proceeding series*, vol. 2. Sivut 84–90. New York: ACM Press.

Nykänen, S. 2010. *Nuorten koulutukseen ja työhön ohjaamisessa tarvitaan tiiviimpää yhteistyötä*. Väitöskirja. Jyväskylän yliopisto.

Opetushallitus 2004. *Perusopetuksen opetussuunnitelman perusteet*. Saatavilla WWW-osoitteessa http://www.oph.fi/ops/perusopetus/pops_web.pdf, 16.1.2004. Viitattu 12.4.2010.

Opetushallitus 2005. *Perusopetuksen tieto-ja viestintätekniikan opetuskäytön sekä oppilaiden tieto- ja viestintätekniikan perustaitojen kehittämissuunnitelma. Työryhmän raportti 21.4.2005*. Saatavilla WWW-osoitteessa http://www.oph.fi/instancedata/prime_product_julkaisu/oph/embeds/47215_tietojaviesti.pdf. Viitattu 21.4.2010.

Oulun yliopisto 2010. *Opetus ja FM-tutkinto, fysiikan laitos.* Saatavilla WWW-osoitteessa https://wiki.oulu.fi/display/fysiikka/Opetus. Viitattu 12.4.2010.

Palmer, S. 1996. *Opeta itsellesi peliohjelmointi*. Espoo: Suomen Atk-kustannus Oy.

Rasku, J., Kuukkanen, P. 2004. *Pelinkehitystyöpaja peruskoulun yläluokkien tietotekniikan opetuksessa*. Pro seminaari -työ. Tampereen yliopisto.

Rauste-Von Wright, M. 1997. *Opettaja tienhaarassa – konstruktivismia käytännössä*. Juva: WSOY.

Rauste-Von Wright, M., von Wright, J., Soini, T. 2003. *Oppiminen ja koulutus*. 9. uudistettu painos. Porvoo: WSOY.

Remedy. 2010. *Remedy Company History*. Saatavilla WWW-osoitteessa http://www.remedygames.com/company/history. Viitattu 18.3.2010.

Rissanen, R. 2006. *Fenomenografia*. Luku 5.1. kokonaisuudesta Anita Saaranen-Kauppinen & Anna Puusniekka. 2006. KvaliMOTV - Menetelmäopetuksen tietovaranto. Saatavilla WWW-osoitteessa http://www.fsd.uta.fi/menetelmaopetus. Viitattu 21.4.2010. Tampere: Yhteiskuntatieteellinen tietoarkisto.

Roine, E. 2007. *Perusopetukseen soveltuvaa ohjelmointikieltä etsimässä*. Pro gradu -työ. Jyväskylän yliopisto.

Sajaniemi, J., Hu, C. 2006. *Teaching Programming: Going beyond "Objects First"*. P. Romero, J. Good, S. Bryant, E. A. Chaparro (eds.) Proceedings of the 18th Annual Workshop of the Psychology of Programming Interest Group (PPIG2006), Brighton, U.K., September 2006, University of Sussex, 255-265.

Salokoski, T., Mustonen, A. 2007. *Median vaikutukset lapsiin ja nuoriin – katsaus tutkimuksiin sekä kansainvälisiin mediakasvatuksen ja -säätelyn käytäntöihin*. Mediakasvatusseuran julkaisuja 2/2007. Helsinki: Mediakasvatusseura ry.

Savela, J. (toim.) 2006. *Jyväskylän yliopiston informaatioteknologian tiedekunta: Toimintakertomus 2005*. Jyväskylän yliopisto.

Schildt, H. 2005. *C# 2. 0 : The Complete Reference.*

Siljander, M. 2009. *Asenteet ja rekrytointi - IT-alan opiskelijarekrytointi Jyväskylän yliopiston Informaatioteknologian tiedekunnassa*. Pro gradu -tutkielma. Jyväskylän yliopisto.

Siukonen, T. (toim.) 2008. *Harva nuori hallitsee tietokoneen hyötykäytön*. Helsingin sanomat 10.12.2008. Saatavilla WWW-muodossa http://www.hs.fi/tulosta/1135241856804. Viitattu 1.4.2010.

Tampereen yliopisto 1995. *Monologista dialogiin - Tampereen yliopiston opetuksen arviointiprojektin loppuraportti: Opiskelijoiden käsitykset opetuksesta*. Saatavilla WWW-osoitteessa http://www.uta.fi/opiskelu/opetuksen_tuki/arviointi/seiska.htm. Viitattu 19.2.2010.

Thomsen, B. 2009. *Java vs. .NET*. Saatavilla WWW-osoitteessa http://ia331421.us.archive.org/0/items/CVscJava/JavaVs.NET.pdf. Viitattu 12.3.2010.

Tilastokeskus 2008. *Yliopistojen uudet opiskelijat ja opiskelijat yhteensä opintoaloittain*. Tiedot koostettu yksittäisistä tilastoista vuosilta 2004–2007. Helsinki: Tilastokeskus.

Turner, S., Bernt, P., Pecora, N. 2002. *Why Women Choose Information Technology Careers: Educational, Social and Familial Influences.* Annual Meeting of the American Educationa Research Assosiation. New Orleans, LA, USA, April 1-5.

Tynjälä, P. 1999. *Oppiminen tiedon rakentamisena. Konstruktivistisen oppimiskäsityksen perusteita.* Helsinki: Tammi.

T10 Media 2006. *Java And C# Compared*. Saatavilla WWW-osoitteessa http://www.csharphelp.com/2006/02/java-and-c-sharp-compared/. Viitattu 12.3.2010.

Van Eck R. 2006. Using Games to Promote Girls Positive Attitudes Toward Technology. *Innovate Journal of Online Education*, vol. 2, issue 3. Saatavilla www-muodossa http://www.innovateonline.info. Viitattu 14.4.2010.

Van Roy, P. 2008. *The principal programming paradigms*. Saatavilla WWW-osoitteessa http://www.info.ucl.ac.be/~pvr/paradigmsDIAGRAMeng108.pdf. Viitattu 14.4.2010.

Van Roy, P. 2009. *Programming Paradigms for Dummies: What Every Programmer Should Know*. Delatour France: IRCAM.

Van Roy, P., Brand, P., Duchier, D., Haridi, S., Henz, M., Schulte, C. 2003. "Logic programming in the context of multiparadigm programming: The Oz experience", Theory and Practice of Logic Programming. *Lecture Notes in Computer Science*, vol. 3389. Sivut 715-763. Charleroi, Belgia.

Vuorinen, I. 2001. *Tuhat tapaa opettaa: Menetelmäopas opettajille, kouluttajille ja ryhmän ohjaajille*. Tampere: Resurssi.

Wilson, B., Shrock, S. 2001. Contributing to Success in an Introductory Computer Science Course: A Study of Twelve Factors. *ACM SIGCSE Bulletin*, vol. 33, issue 1. Sivut 184–188. New York: ACM.

Wittgenstein, L. 1981. *Filosofisia tutkimuksia*. Alkuteos: Philosophische Untersuchungen, 1953. Suom. Heikki Nyman. Taskutieto 155 (3. painos 2001). Porvoo, Helsinki, Juva: WSOY.

Liitteet

Liite 1. Pelikurssin mainos.

Liite 2. Saatekirje opettajille

Arvoisa opettaja,

Jyväskylän yliopiston Tietotekniikan laitos järjestää nuorille suunnatun kesäkurssin peliohjelmoinnista. Ohjelmoinnin ja tietotekniikan käytön lisäksi kurssin sisällössä sivutaan myös matematiikkaa ja fysiikan ilmiöitä. Ohjelmointikieli ja -työkalut ovat englanniksi, joten kielitaitoakin pääsee kehittämään.

Pyytäisimme, että kerrotte lyhyesti kurssista (seuraavien kappaleiden teksti, joka on lainausmerkeissä) ja sitten jakaisitte ohessa tulleet mainoslehtiset kaikille oppilaille. Voimme tarvittaessa toimittaa mainoslehtisiä lisää.

"Jyväskylän yliopiston Tietotekniikan laitos järjestää kesäkurssin pelien tekemisestä ja ohjelmoinnista. Kurssi on tarkoitettu kaikille yläkouluikäisille (13-16-vuotiaille tytöille ja pojille). Aiempaa kokemusta ei tarvitse olla ohjelmoinnista.

Kursseja järjestetään 1-2 riippuen kiinnostuneiden määrästä. Jos kurssi kiinnostaa, niin täytä netissä alustava ilmoittautumislomake. Valitse, mitkä ajankohdat sopisivat heinäkuussa tai elokuun alussa. Nettisivuilta löytyy myös enemmän tietoa kurssista."

Ystävällisin terveisin,

Projektitiimi

Vesa Lappalainen, Tuukka Puranen, Tero Jäntti, Tomi Karppinen & Janne Nikkanen

Liite 3. Opettajien saatekirjeen liiteosa

Kurssilla sivutaan muun muassa seuraavia perusopetuksen opetussuunnitelmaan kuuluvia aiheita sekä muutamia muita teemoja:

1. Fysiikka
    - syy-seuraussuhteet ja kappaleiden väliset riippuvuudet
    - fysiikan käsitteitä, kuten Maan vetovoima, kitka sekä voimista aiheutuvat liike- ja tasapainoilmiöt, erilaiset liikkeet ja voimien vaikutukset liikkeeseen
    - kappaleen massa ja sen vaikutukset
    - tasainen ja tasaisesti kiihtyvä liike
    - suureita, kuten aika, matka, nopeus, kiihtyvyys ja voima
2. Matematiikka
    - funktion käsite
    - lukuparin esittäminen koordinaatistossa
    - yksinkertaisten funktioiden tulkitseminen ja niiden kuvaajien piirtäminen koordinaatistoon
    - geometriasta suorat, suurennokset, pienennökset ja mittakaava sekä erilaiset muodot, esim. kolmiot, nelikulmiot, monikulmiot ja ympyrä
    - yhtälöt ja niiden ratkaiseminen
    - todennäköisyys ja satunnaisuus
    - loogisia elementtejä ja totuusarvoja
    - tapausten yhtäläisyyksiä ja säännönmukaisuuksia
    - ongelmanratkaisua
    - vektorit ja niiden esittäminen lukuparina tai napakoordinaatistossa
    - asteet ja radiaanit

Lisätietoja kurssista saat:

- Nettisivuilta: www.jyu.fi/it/nuortenkurssi
- Sähköpostitse: pelik2009.group@korppi.jyu.fi

Liite 4. Jyväskylän yliopiston tiedote, 10.6.2009.

Nyt tehtailemaan pelejä Agoraan

Oletko joskus halunnut tehdä oman tietokonepelin? Tänä kesänä siihen on mahdollisuus. Jyväskylän yliopiston tietotekniikan laitos järjestää heinä- tai elokuussa ensimmäistä kertaa nuorille suunnatun, täysin ilmaisen peliohjelmointikurssin. Kurssi on suunnattu 13-16-vuotiaille tytöille ja pojille ja sen kesto on viisi päivää. Minkäänlaista aikaisempaa kokemusta ohjelmoinnista ei tarvita. Iän jälkeen ainoana pääsyvaatimuksena on kiinnostus ja avoin mieli!

Jyväskylän Agoran nykyaikaisissa tiloissa järjestettävällä pelikurssilla opit mielenkiintoisessa ympäristössä tietoteknisten taitojen lisäksi matematiikan, fysiikan ja englannin kielen taitoja. Tarkoitus on herättää kiinnostusta luonnontieteellisistä ja tietoteknisistä aihepiireistä. Vaikka kurssin pääpaino onkin pelisuunnittelussa ja ohjelmoinnissa, niiden lisäksi kurssilla opitaan pelisuunnittelua, kuvankäsittelyä tekstuurien ja grafiikan tekemisen muodossa sekä äänten tekemistä ja editointia. Normaalisti ohjelmointikieliä käytettäessä jo pienintäkin asiaa varten on kirjoitettava monta riviä vaativaa ohjelmakoodia, varsinkin peliohjelmoinnissa. Tietotekniikan laitoksen kehitysryhmä on kuitenkin luonut valmiin kirjaston oppilaiden käytettäväksi, jolloin jo muutamalla rivillä saadaan jo hyvin paljon aikaan.

Kurssi on jo markkinointivaiheessa otettu hyvin vastaan Jyväskylän alueen yläkouluissa. Monella tietotekniikan aineopettajalla on heti tullut mieleen yksi tai useampi oppilas, joita varmasti kurssi kiinnostaisi ja kenen taidot on huomattu jo koulussa. Tietotekniikan laitos haluaa paitsi tarjota mahdollisuutta harrastuksensa kehittämiseen, myös toimia ponnahduslautana mahdollisen opiskelupaikan valinnassa sekä toimia omien kiinnostusten alueiden kartoittajana.

Useita varmasti mietityttää, millaisia pelejä kurssilla oikeastaan sitten tehdään. Osaanko varmasti? Ei hätää. Kurssin sisältö on suunniteltu niin, että kaikilla on mahdollisuus ottaa siihen osaa ja ymmärtää käsiteltävät aiheet. Jos osaat pelata koordinaatistossa laivan upotusta, selviät myös tällä kurssilla!

Kaikki kurssilla toteutettavat pelit ovat kaksiulotteisia. Niihin kuuluvat esimerkiksi tankki-, auto-, lento- ja biljardipeli. Osaamisen mukaan voi tietysti lähteä toteuttamaan lähes mitä vain, kurssilta saat varmasti ammattitaitoista apua kysymyksiisi!

Kurssin päiväohjelma koostuu muutamasta opetustunnista sekä sopivin väliajoin pidetyistä tauoista. Myös ruokailu hoituu paikan päällä kätevästi. Agoran oma ravintola Piato tarjoilee oppilaille herkullista ja monipuolista lounasruokaa hyvin edulliseen hintaan.

Lisätietoa kurssista löydät kurssin nettisivuilta osoitteesta www.jyu.fi/it/nuortenkurssi. Ilmoittautuminen on jo käynnissä! Kurssin tarkka ajankohta päätetään ilmoittautumisen päätyttyä.

Kurssin opetuksesta vastaavat:

Professori Pekka Neittaanmäki, pn@mit.jyu.fi, 040-5507005

Antti-Jussi Lakanen, anlakane@jyu.fi

Vesa Lappalainen, vesal@mit.jyu.fi

https://www.jyu.fi/ajankohtaista/arkisto/2009/06/tiedote-2009-09-11-00-51-40-691842

Liite 5. Esimerkki pelisuunnitelman mallista.

Tietoja pelistä

Pelin nimi: Avaruuslaskeutuja

Pelialusta: Windows ja Xbox 360

Pelaajien lukumäärä: 1 pelaaja

Pelin tarina

Ihmiset kartoittavat asutettavia planeettoja. Avaruuslaskeutujan tehtävänä on laskeutua turvallisesti planeetalle, jotta sitä voidaan tutkia. Aluksessa on herkkiä osia, jotka eivät kestä nopeaa vauhtia. Polttoainekin uhkaa nopeasti loppua. Tarvitaan siis tarkkaa aluksen ohjaajaa.

Pelin idea ja tavoitteet

Pelissä laskeudutaan avaruusaluksella planeetan pinnalle. Planetan pinta on rosoinen, eikä hyviä laskeutumispaikkoja välttämättä aina ole.

Tavoitteena on laskeutua mahdollisimman vähän polttoainetta käyttäen ja hyvässä asennossa planeetan pinnalle. Jos vauhti on liian nopea tai alus on huonossa kulmassa, niin alus räjähtää.

Pelissä arvotaan aina uusi planeetan pinta, johon laskeudutaan.

Toteutuksen suunnitelma

Keskiviikko:

1. Tekisin ensin kentän, jossa on littana planeetan pinta ja alus.
2. Lisään alukseen ohjauksen. Alusta voi pyörittää oikealle ja vasemmalle sekä ylöspäin. Aluksen sijainti on kentän yläreunassa.
3. Lisätään polttoainemittari. Kun polttoaine loppuu, niin aluksen moottoria ei voi käyttää.

Torstai:

4. Lisätään aluksen nopeuden ja kulman näyttävät mittarit.

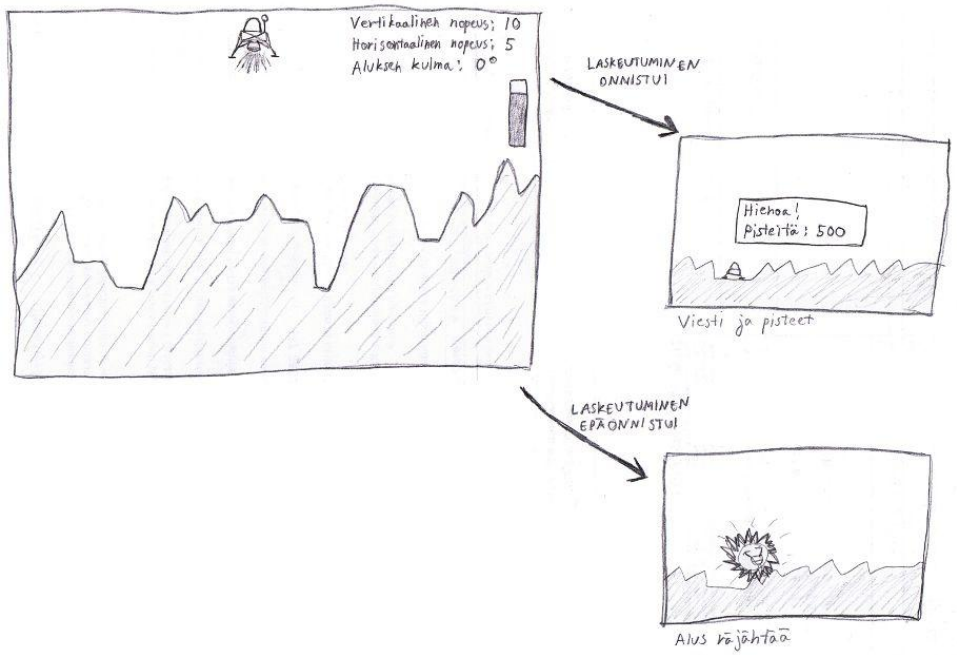
5. Lisätään pistelaskuri, joka laskee pisteet. Pisteet näytetään myös näytöllä.
6. Aluksen pitää räjähtää, jos törmätään liian suurella nopeudella planeettaan.

Perjantai:

7. Lisätään satunnaisesti luotava maasto.
8. Viimeistellään peli.

Jos on aikaa:

- Värinätehosteet ohjaimeen.
- Useampi pelaaja peliin. Tehdään kilpailumoodi, kumpi laskeutuu nopeammin.



Kuva 19. Pelisuunnitelman visuaalisen hahmotelman malli.

Liite 6. Kurssien alkukyselyn kyselylomake.

KURSSIN ALKUKYSELY

1. Nimi

_____ (etunimi ja sukunimi)

Huomaa, että vaikka kyselyyn laitetaan nimi, henkilöllisyytesi **ei tule esille** tuloksia raportoitaessa. Voit siis huoletta vastata rehellisesti kyselyyn.

2. Ikä

____ vuotta

3. Pelaan tietokone-/konsolipelejä

☐ Joka päivä ☐ Muutamana päivänä viikossa ☐ 1-4 kertaa kuukaudessa ☐ Harvemmin ☐ En koskaan

4. Minkä tyyppisiä pelejä pelaat? Voit valita useita.
(jos vastasit edelliseen kysymykseen *En koskaan*, voit ohittaa tämän kysymyksen)

☐ Auto- ja ajopelit

☐ Pulmapelit (Lemmings, Tetris, Portal jne.)

☐ Räiskintäpelit (Half-Life, Counter Strike, Halo jne.)

☐ Roolipelit (RuneScape, EverQuest, World of Warcraft jne.)

☐ Seikkailupelit (RuneScape, EverQuest, World of Warcraft jne.)

☐ Strategiapelit (Command and Conquer, StarCraft jne.)

☐ Taistelu- tai tappelupelit (Mortal Kombat jne.)

☐ Taitopelit (biljardi, shakki, tammi, jne.)

☐ Tasohyppelypelit (Mario Bros jne.)

☐ Urheilupelit (NHL jne.)

Muita, mitä?:

5. Olen suunnitellut pelejä (tietokoneella tai ilman tietokonetta)

☐ Kyllä ☐ En

6. Minulla on aikaisempaa ohjelmointikokemusta

☐ Kyllä, melko paljon tai paljon ☐ Kyllä, mutta vain hyvin vähän ☐ Ei lainkaan

7. Koen olevani taitava tietokoneiden ja teknisten laitteiden käyttäjä

Täysin eri mieltä 1 2 3 4 5 Täysin samaa mieltä

8. Harrastan lukemista

☐ Joka päivä ☐ Muutamia kertoja viikossa ☐ 1-4 kertaa kuukaudessa ☐ Harvemmin ☐ En koskaan

9. Minulle mieluisimpia kouluaineita ovat
(voit valita useita)

☐ Biologia

☐ Englannin kieli

☐ Fysiikka

☐ Historia, yhteiskuntaoppi

☐ Kemia

☐ Kotitalous

☐ Kuvaamataito

☐ Käsityöt

☐ Liikunta

☐ Maantieto

☐ Matematiikka

☐ Musiikki

☐ Ruotsin kieli

☐ Tietotekniikka

☐ Uskonto, elämänkatsomustieto

☐ Äidinkieli

Muu, mikä?:

10. **Uskon, että tällä kurssilla voin hyödyntää tietojani seuraavista kouluaineista**
(luettele ne aineet, joista uskot olevan hyötyä)

_____

_____

11. **Uskon, että tämä kurssi vahvistaa osaamistani seuraavissa aineissa**
(luettele ne aineet, joita tämä kurssi mielestäsi hyödyttää)

_____

_____

_____

12. **Tulin tänne kurssille, koska**

☐ Mielekästä tekemistä kesälle

☐ En halunnut vain lomailla kesällä

☐ Pelit kiinnostavat minua

☐ Kaverini tuli myös kurssille mukaan

☐ Olen kiinnostunut luonnontieteistä

☐ Haluan oppia ohjelmoimaan

☐ Uskon että ohjelmointi voi olla tulevaisuudessa jopa ammattini

☐ Vanhemmat pakottivat

Muu syy, mikä?:

13. **Tällä kurssilla on mukana kavereitani**

☐ Kyllä, (kirjoita nimet)

_____

_____

☐ Ei

14. **Aion hakeutua luonnontieteiden opiskelun pariin tulevaisuudessa**
(esim. ammattikoulun luonnontieteellinen linja, tai lukiossa pitkä matematiikka/fysiikka/kemia/tietotekniikka)

Täysin eri mieltä 1 2 3 4 5 Täysin samaa mieltä

15. **Odotan saavani tältä kurssilta**

☐ Tietoa ohjelmoinnista

☐ Tietoa pelien tekemisestä

☐ Tietoa pelien suunnittelusta

☐ Kavereita

☐ Tietoa yliopistosta

☐ Tietoa luonnontieteiden opiskelusta

☐ Tietoa ohjelmoijan ammatista

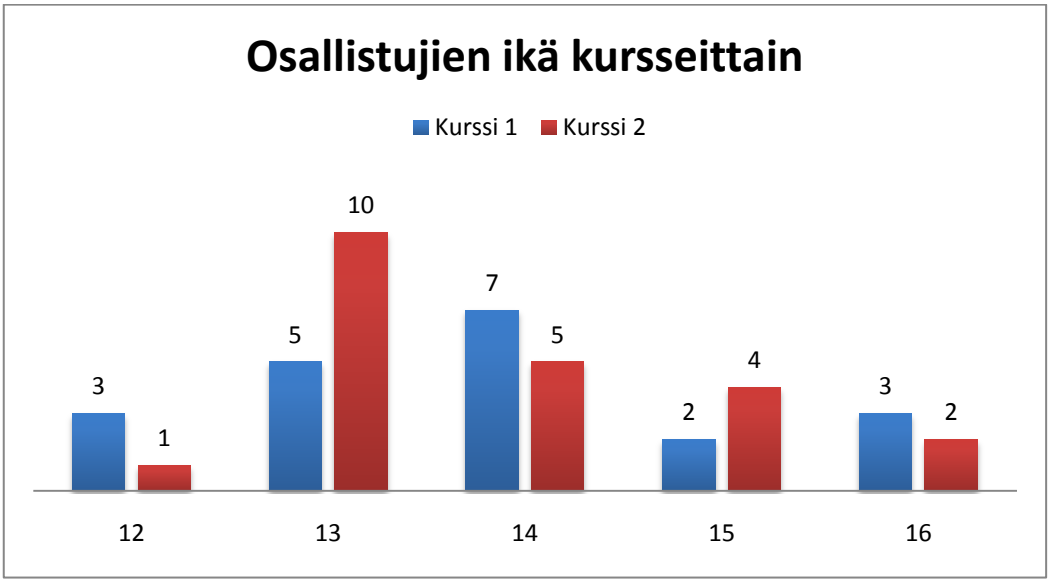
Muuta, mitä?:

Kiitos vaivannäöstäsi! Palauta valmis lomake opettajalle.

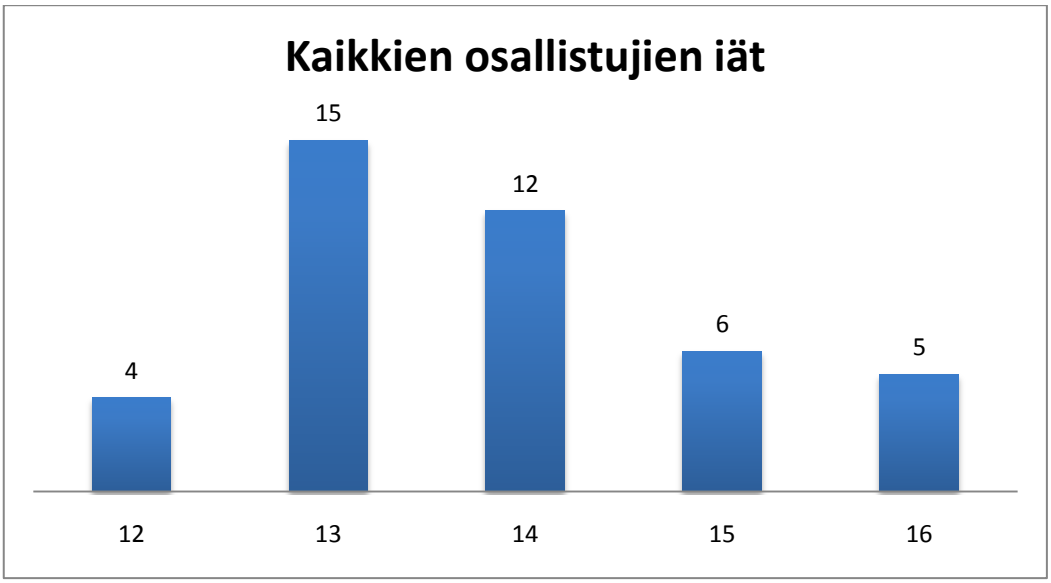
Liite 7. Alkukyselyiden tulokset (n=45).

Kaavioparvi 1. *Osallistujan ikä (kysymys 2, n=43)*

Kaavio 1-a

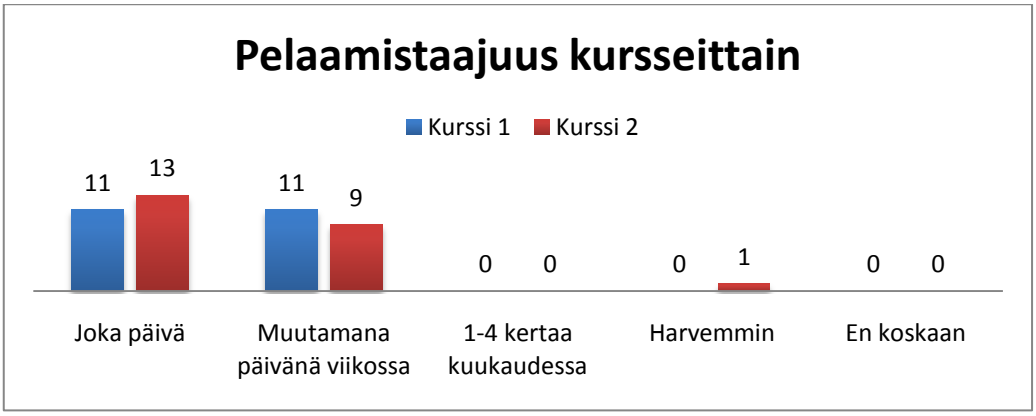


Kaavio 1-b

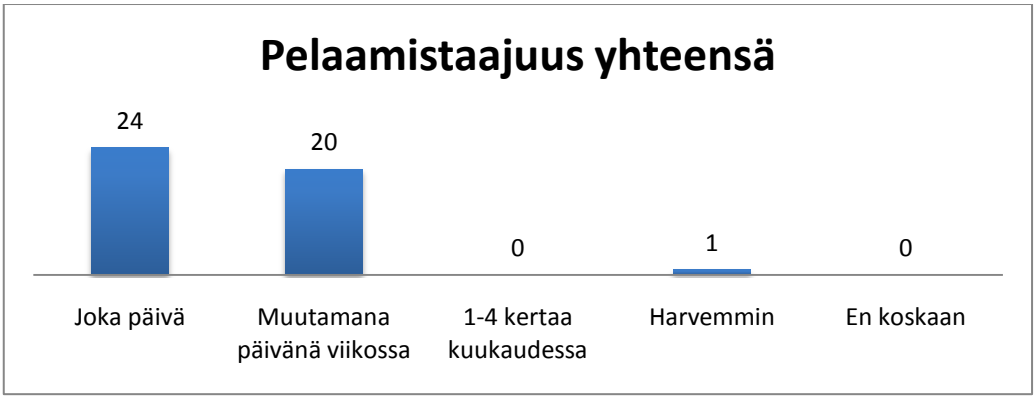


Kaavioparvi 2. *Pelaamistaajuus (kysymys 3, n=45)*

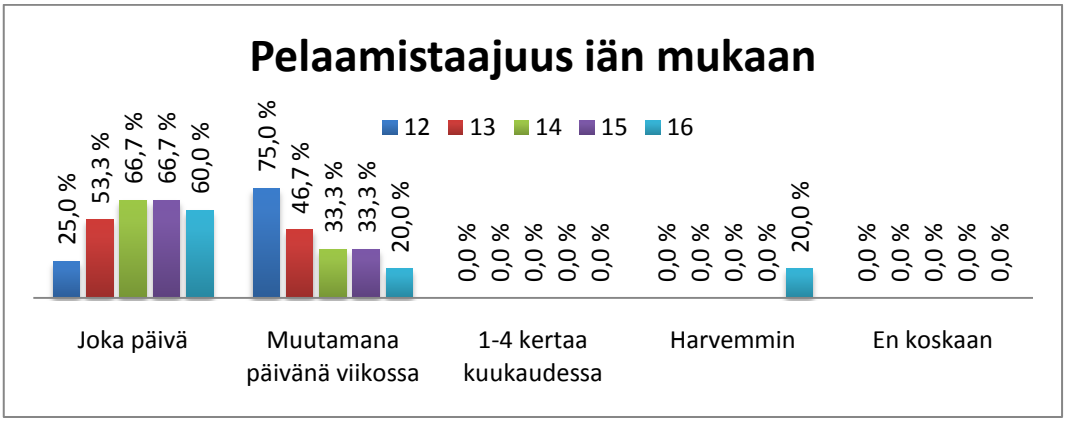
Kaavio 2-a



Kaavio 2-b

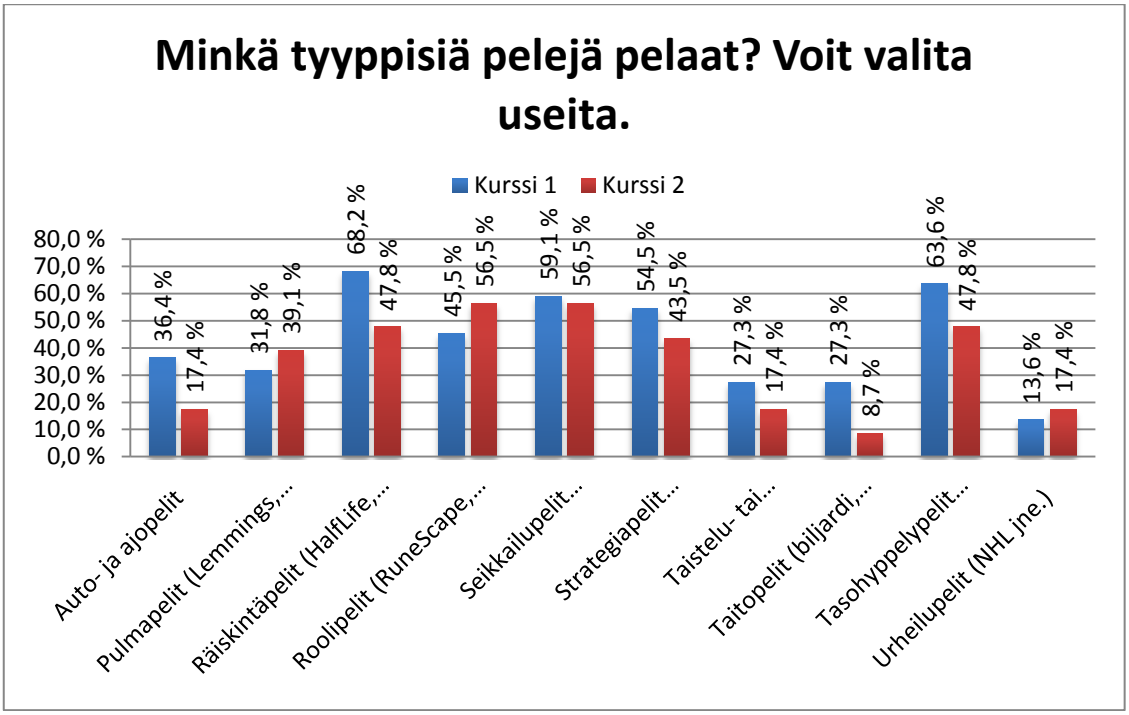


Kaavio 2-c

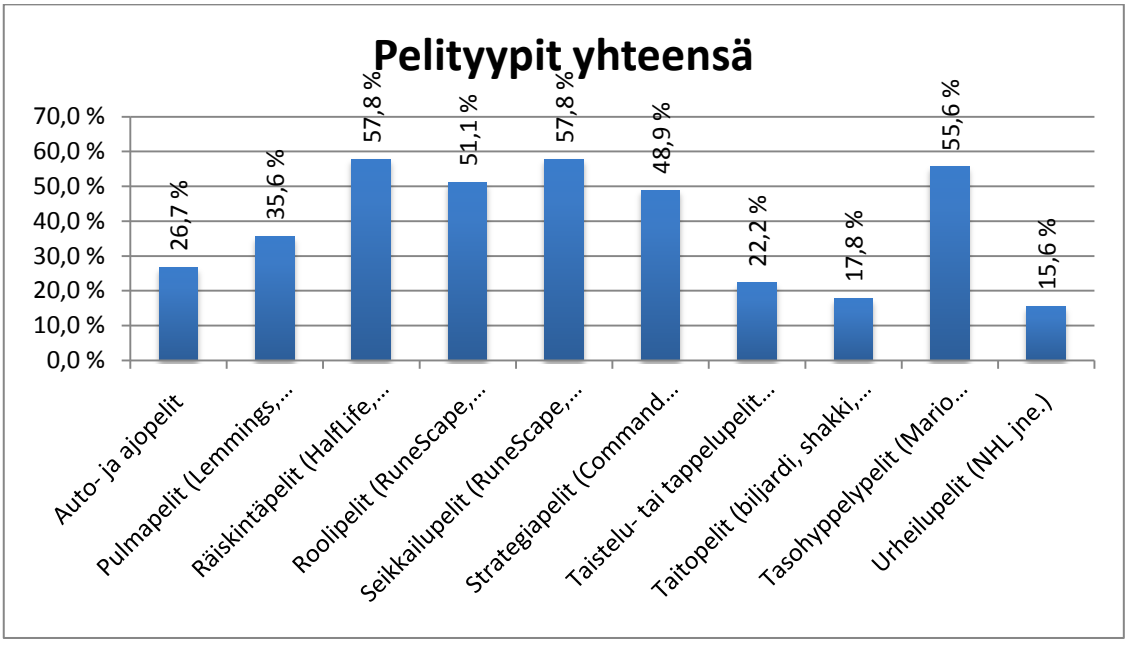


Kaavioparvi 3. *Minkä tyyppisiä pelejä pelaat (kysymys 4, n=45)*

Kaavio 3-a

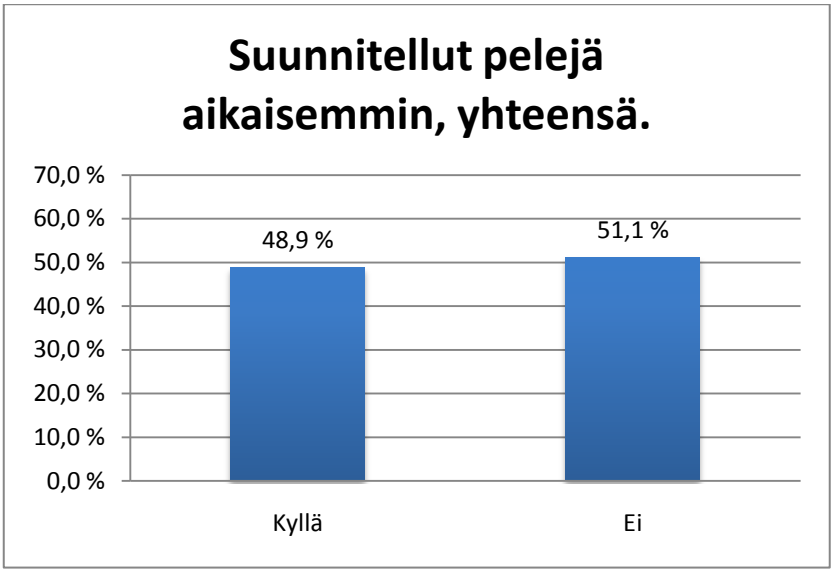


Kaavio 3-b

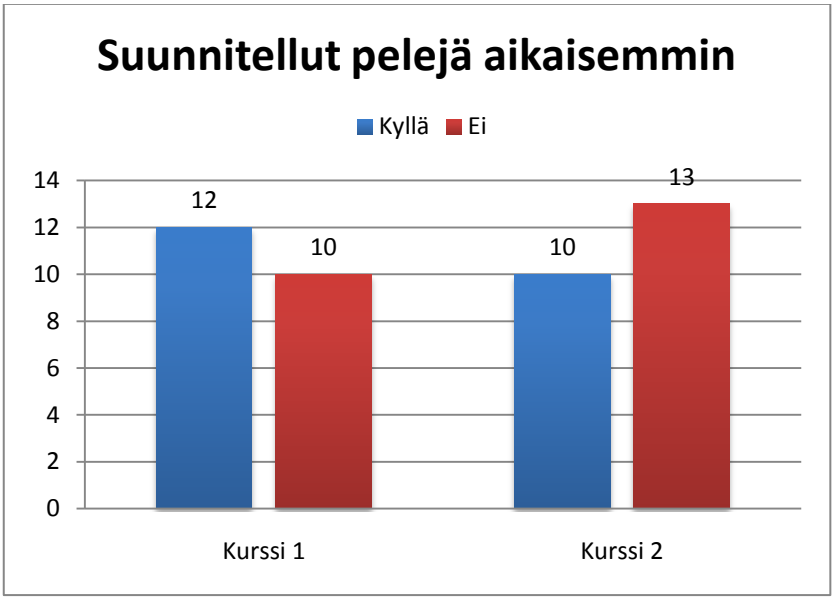


Kaavioparvi 4. *Suunnitellut pelejä aikaisemmin (kysymys 5, n=45)*

Kaavio 4-a

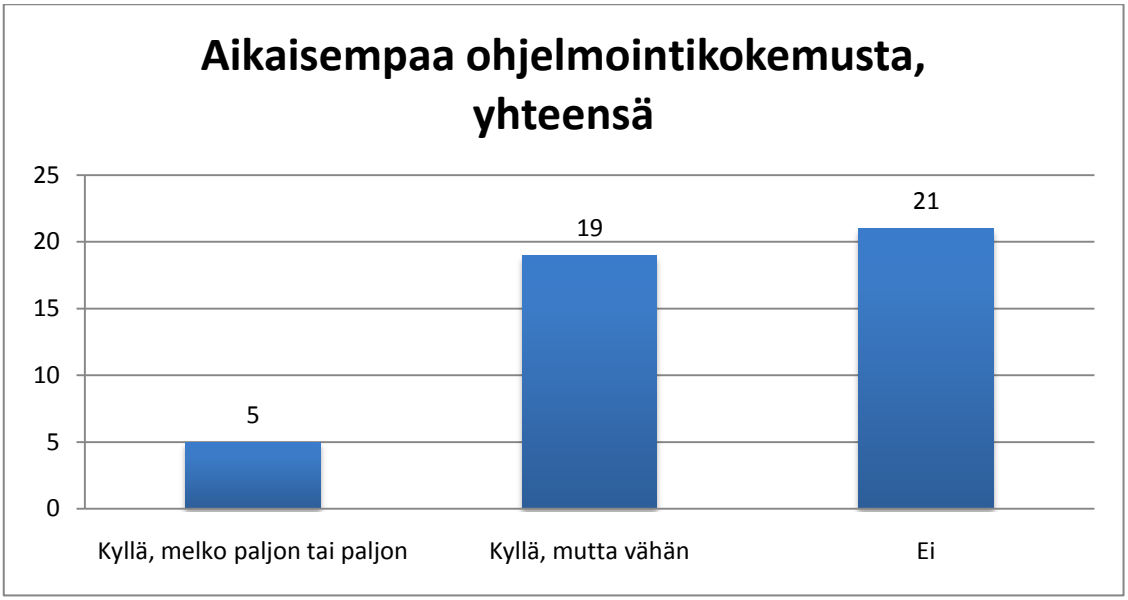


Kaavio 4-b

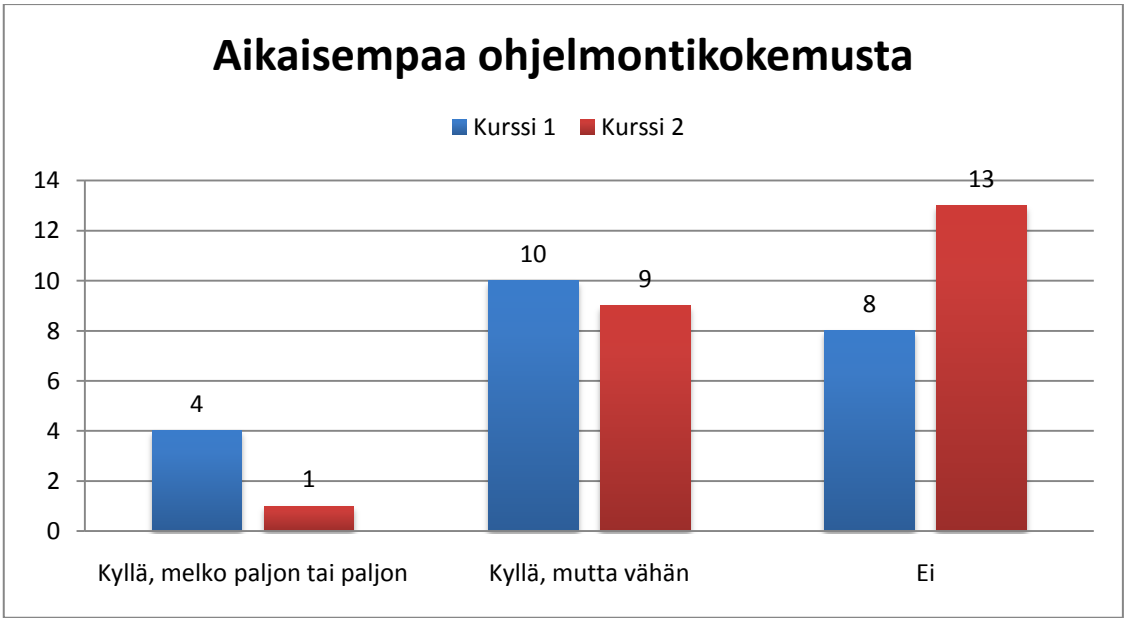


Kaavioparvi 5. *Aikaisempaa ohjelmointikokemusta (kysymys 6, n=45)*

Kaavio 5-a

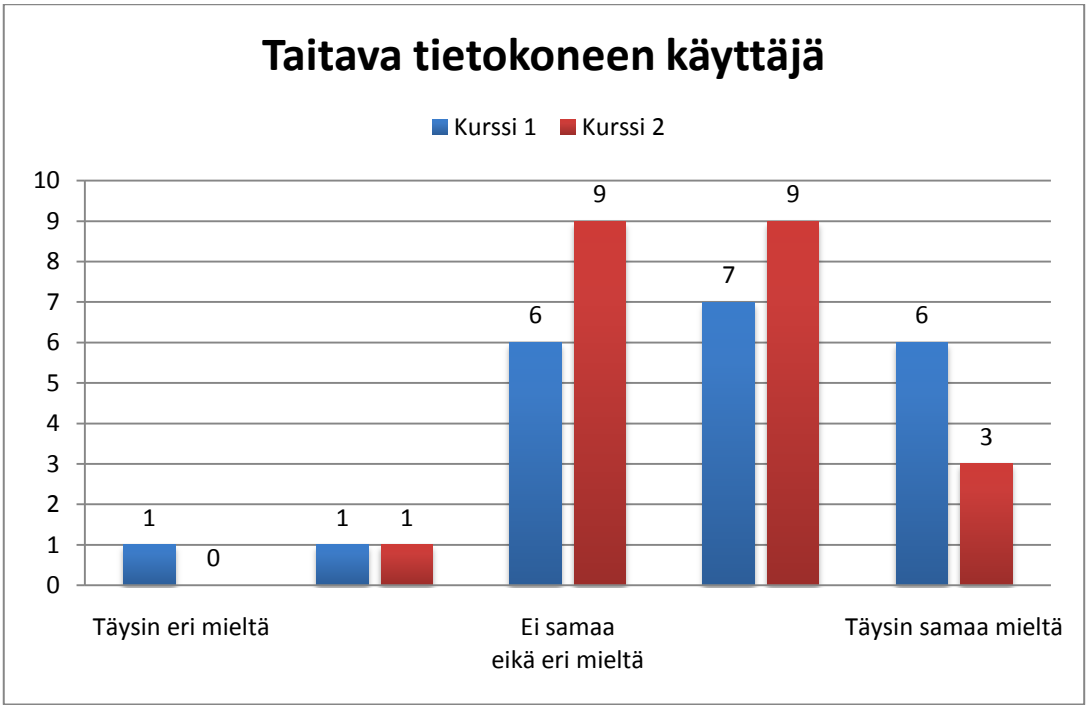


Kaavio 5-b

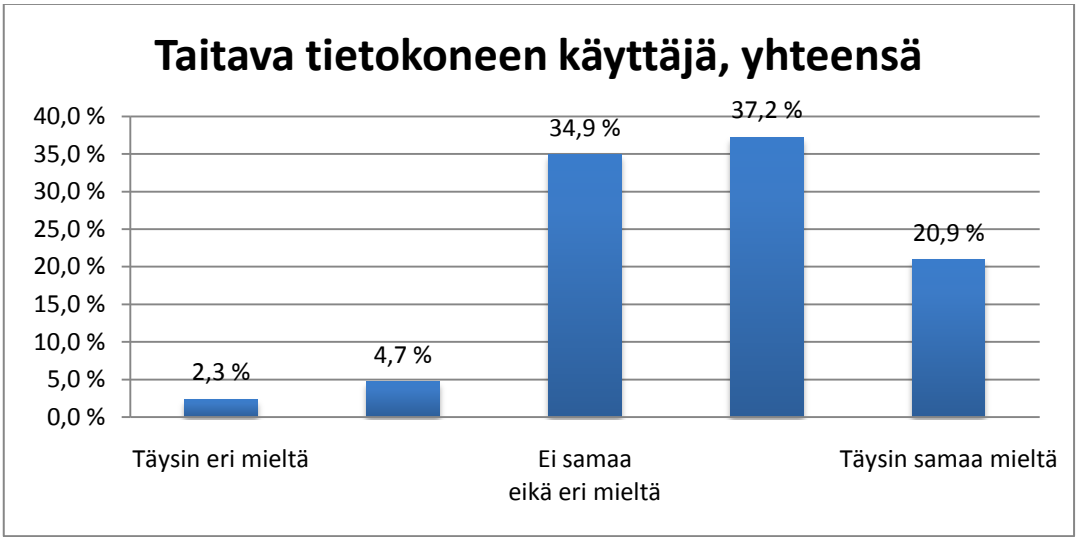


Kaavioparvi 6. *Tietokoneen käyttötaito (kysymys 7, n=45).*

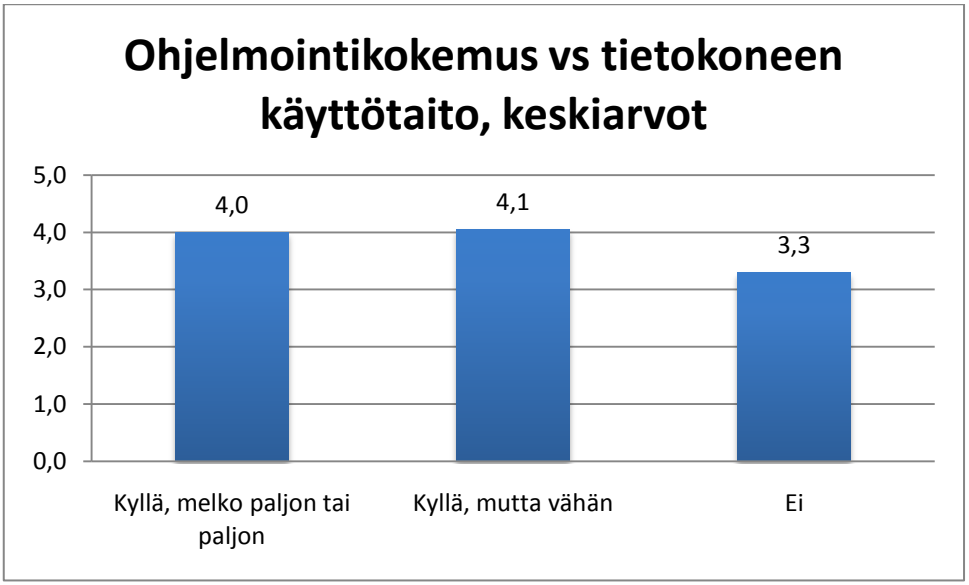
Kaavio 6-a



Kaavio 6-b

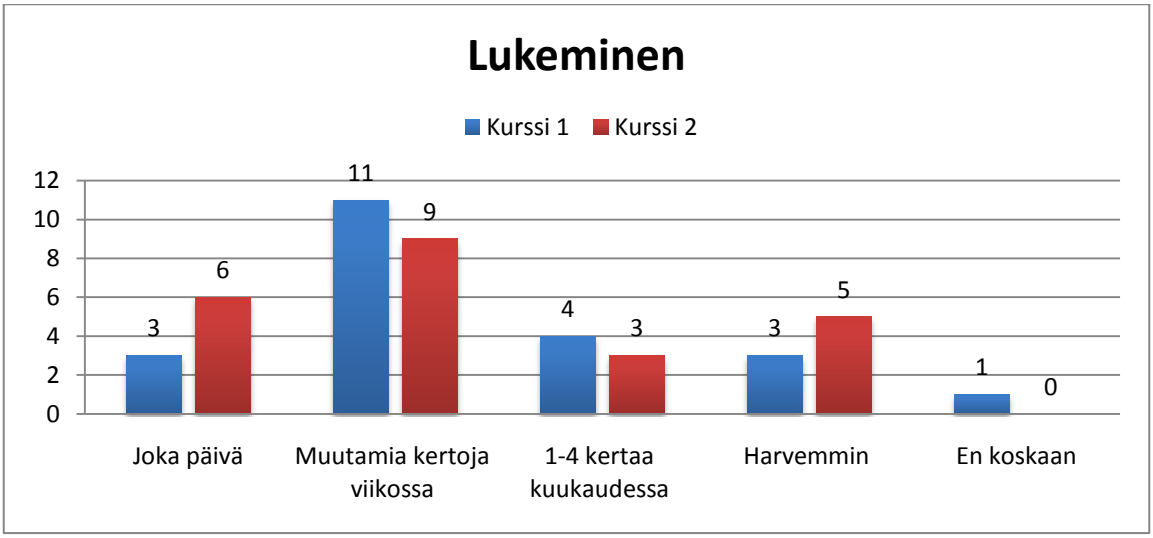


Kaavio 7. *Aikaisempaa ohjelmointikokemusta vs tietokoneen käyttötaito (kysymys 7, n=45).*

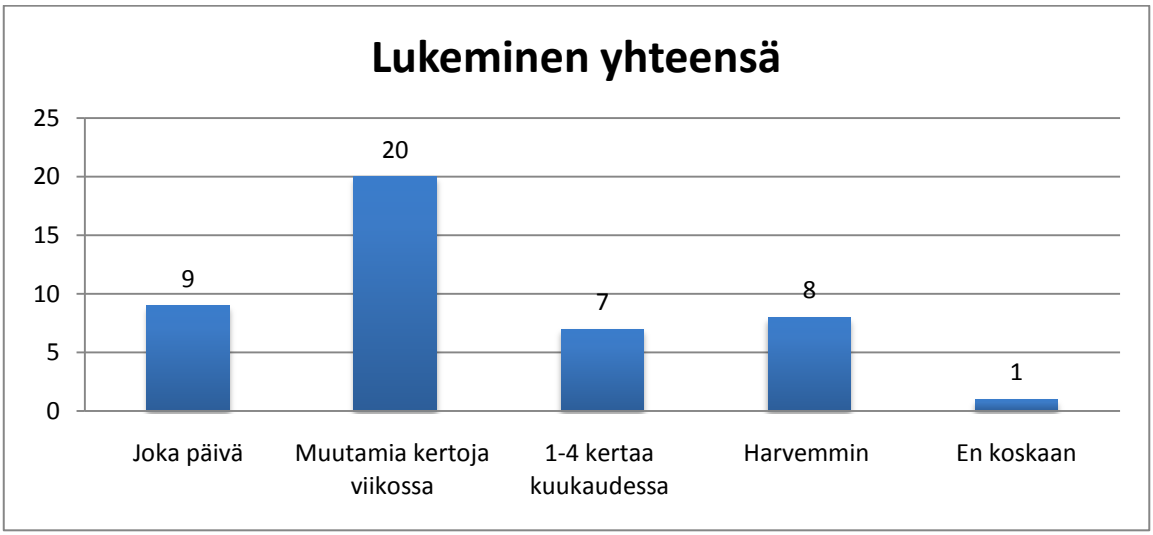


Kaavioparvi 8. *Harrastaa lukemista (kysymys 8, n=45).*

Kaavio 8-a

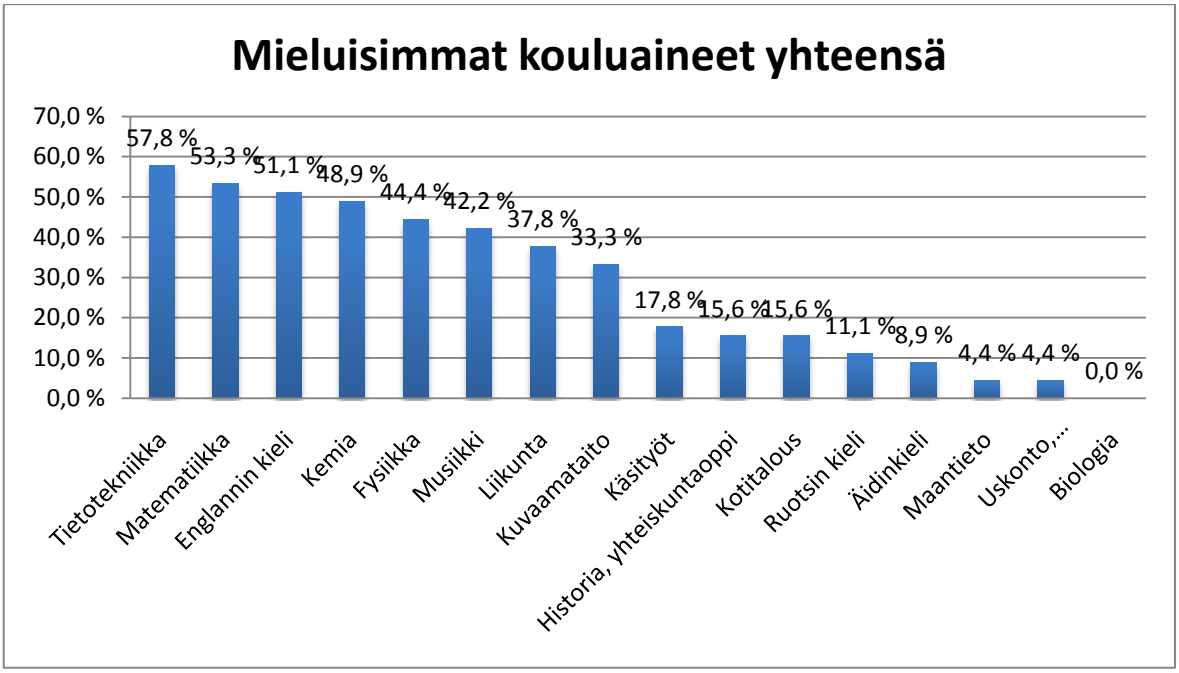


Kaavio 8-b

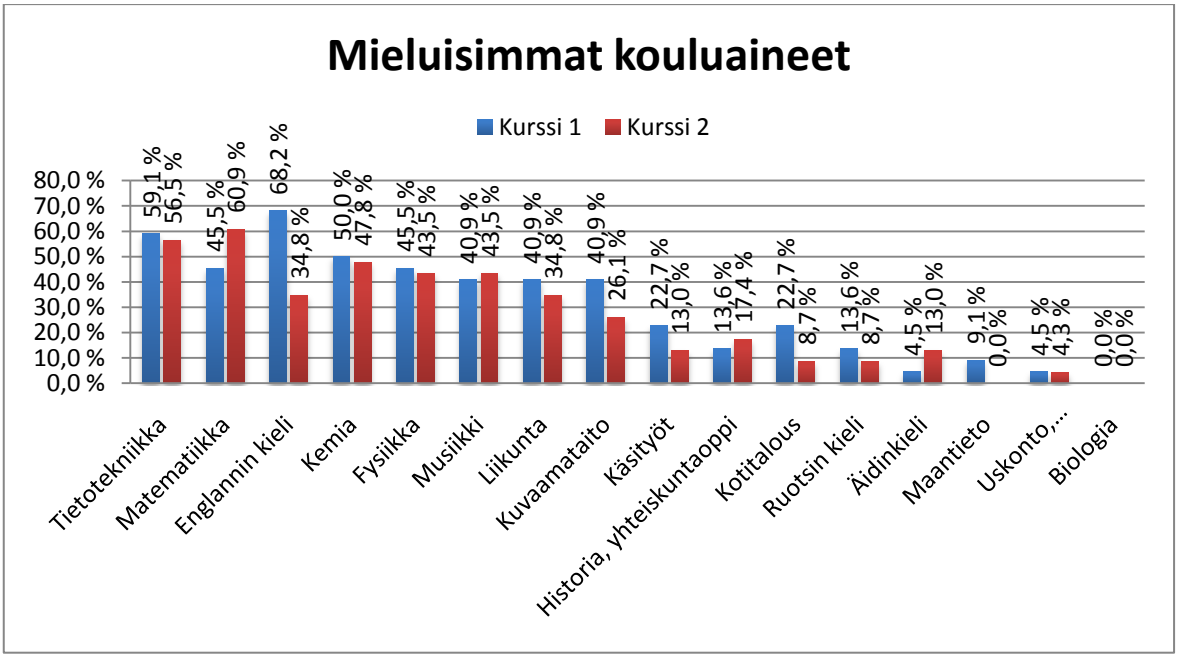


Kaavioparvi 9. *Mieluisimmat kouluaineet (kysymys 9, n=45).*

Kaavio 9-a

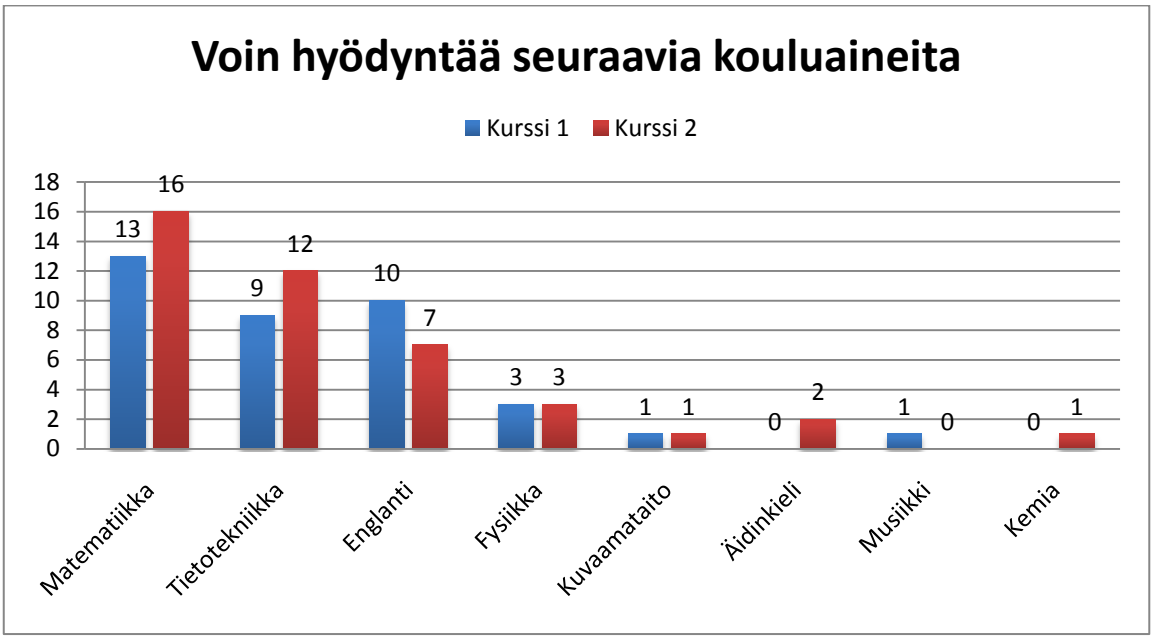


Kaavio 9-b

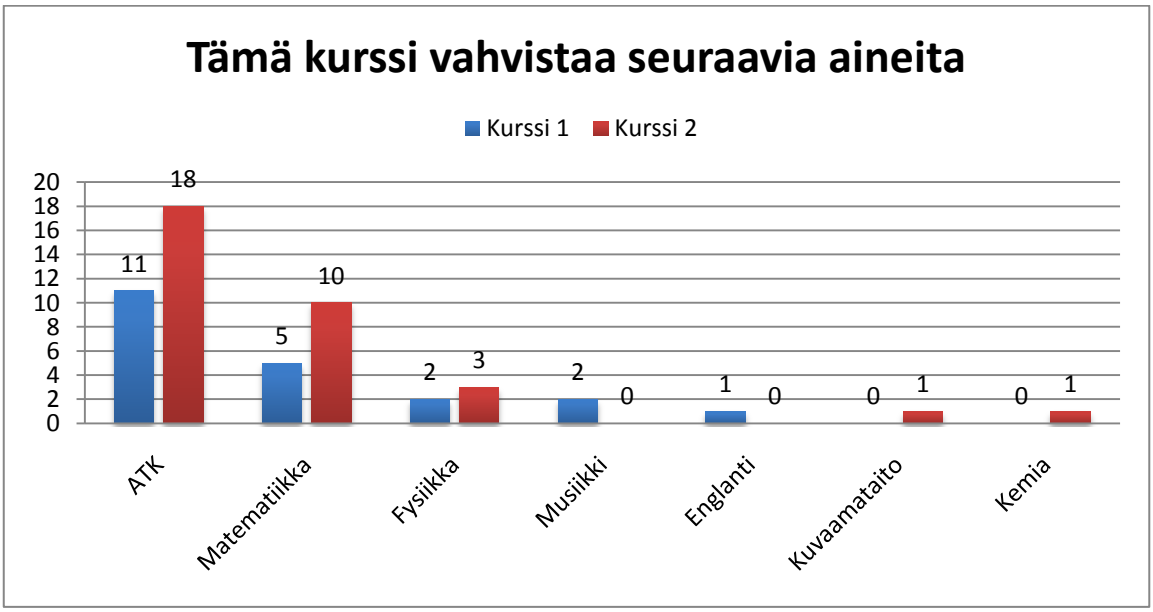


Kaavioparvi 10. *Oppiaineiden hyödyntäminen ja osaamisen vahvistuminen eri oppiaineissa kurssin myötä (kysymykset 10, n=43, ja 11, n=35).*

Kaavio 10-a

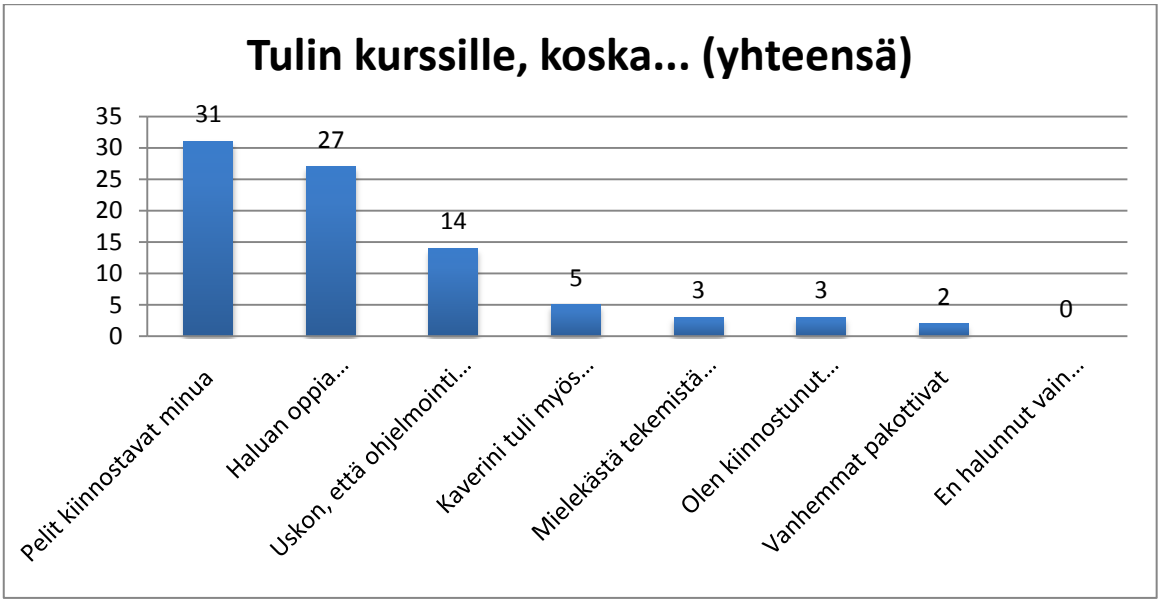


Kaavio 10-b

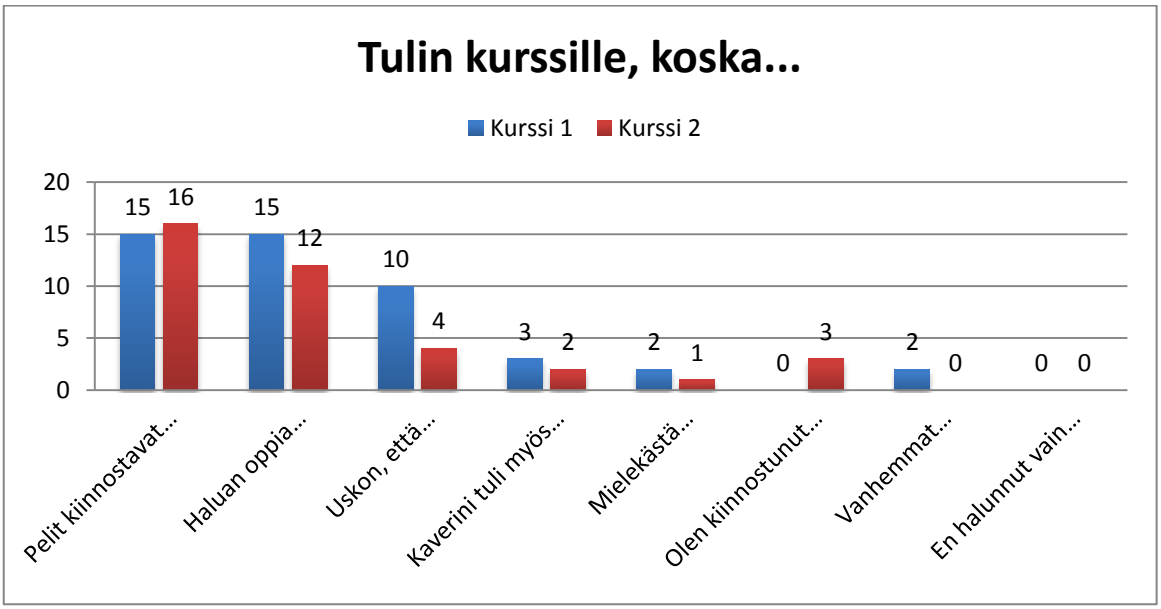


Kaavioparvi 11. *Kurssille tulemisen syitä (kysymys 12, n=44).*

Kaavio 11-a

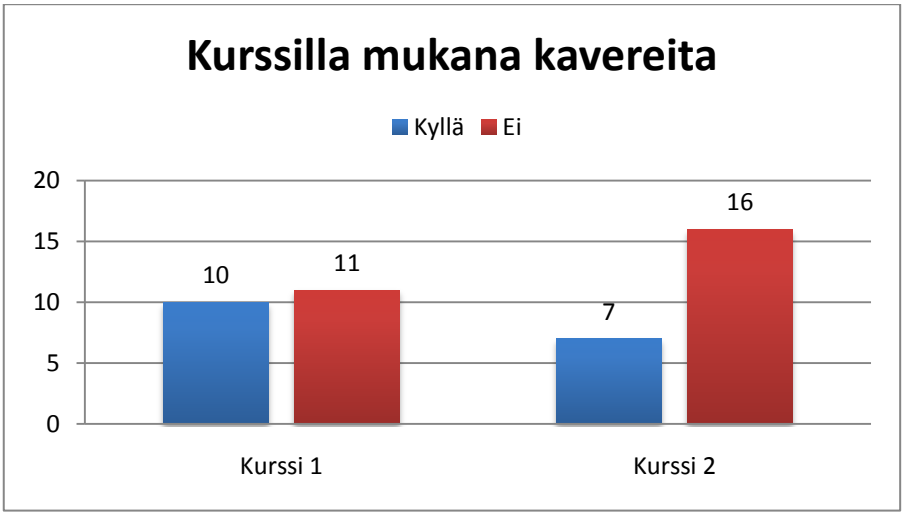


Kaavio 11-b

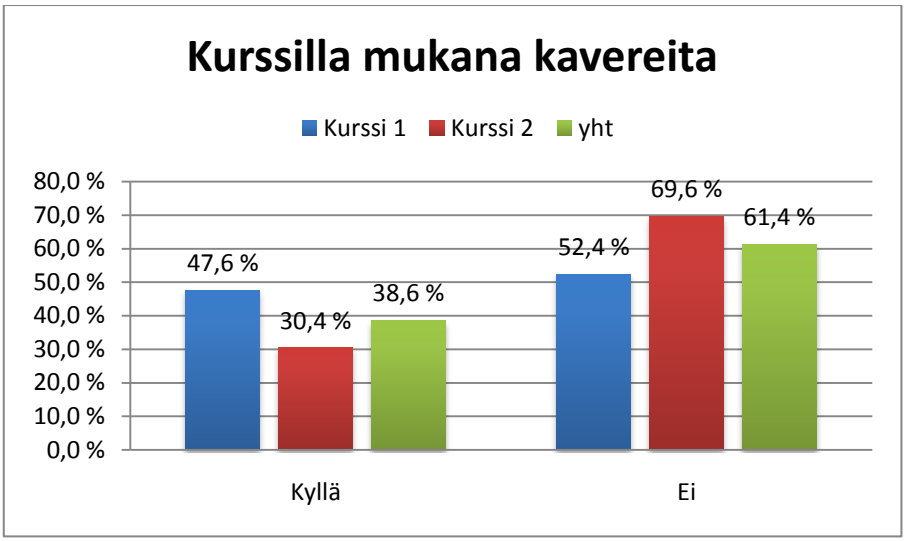


Kaavioparvi 12. *Kurssille mukana kavereita (kysymys 13, n=44).*

Kaavio 12-a

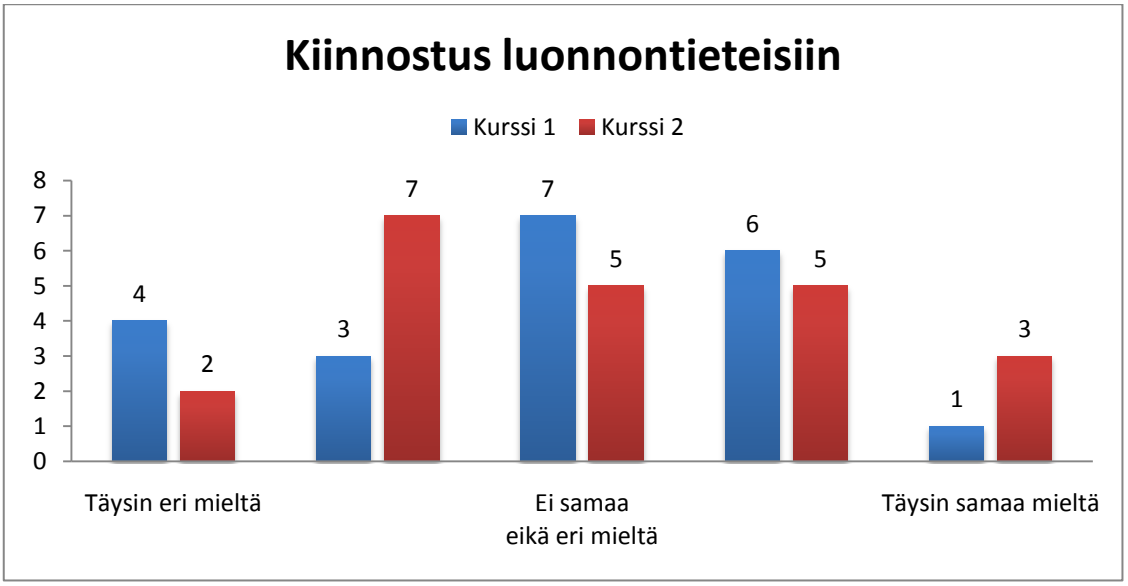


Kaavio 12-b

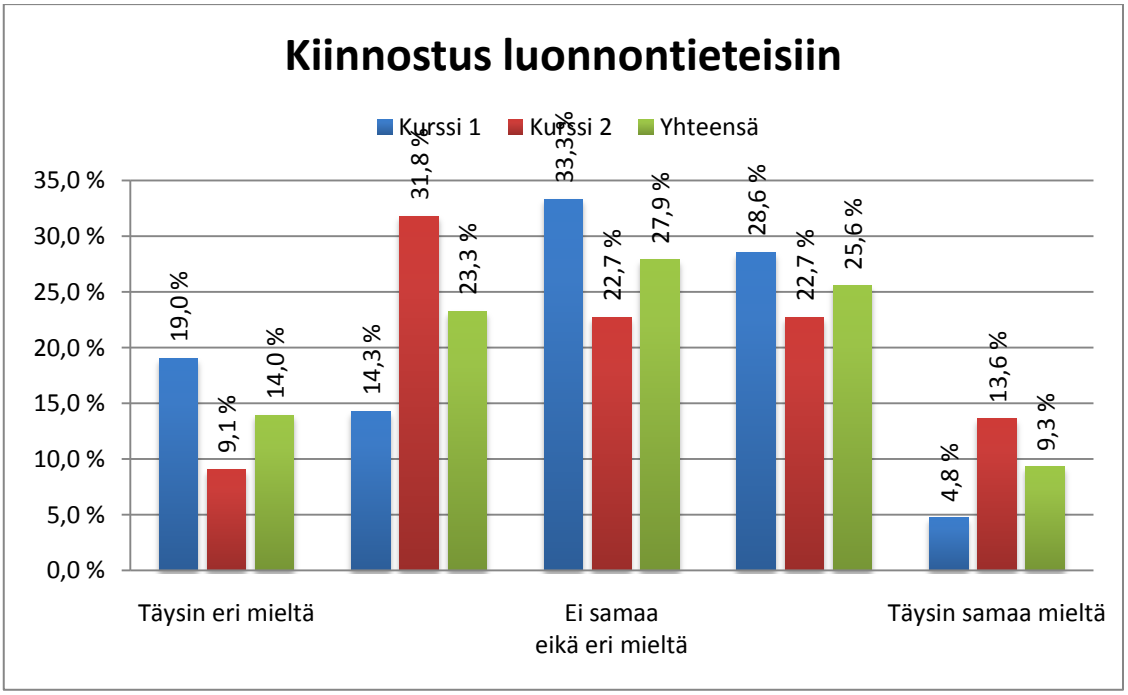


Kaavioparvi 13. *Kiinnostus luonnontieteiden opiskelua kohtaan (kysymys 14, n=43).*

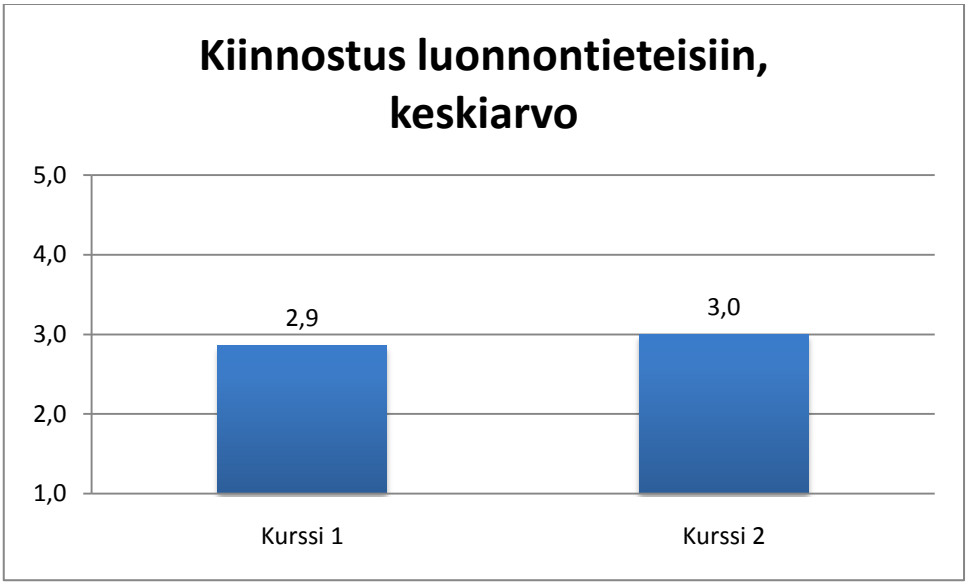
Kaavio 13-a



Kaavio 13-b

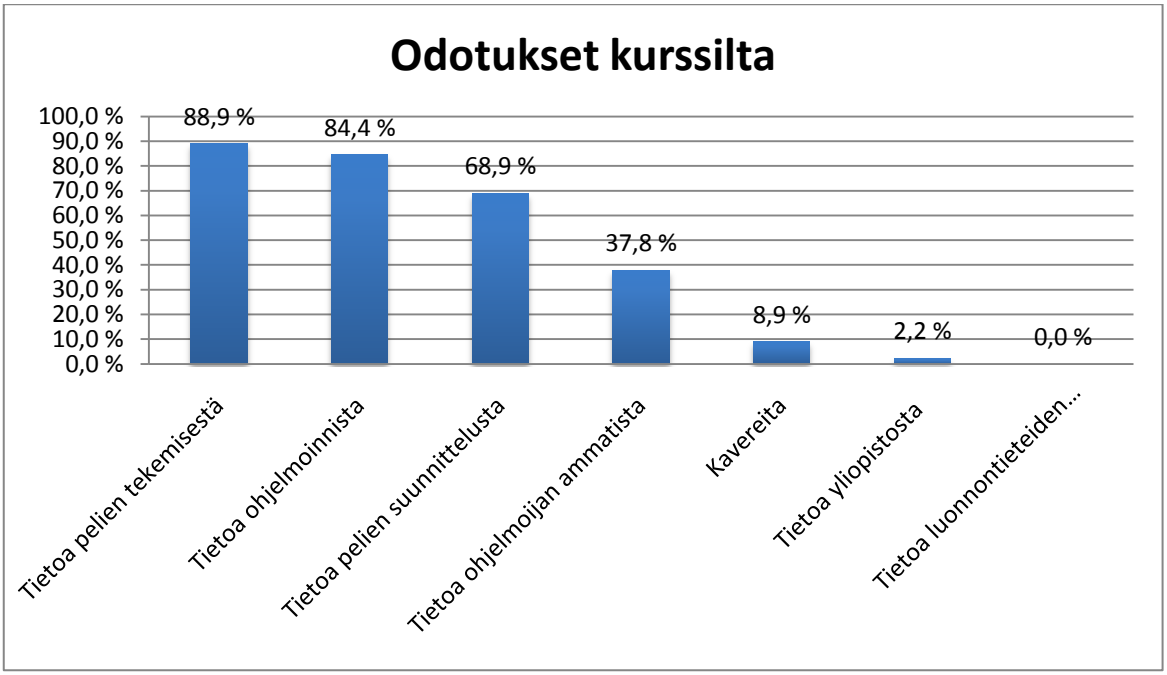


Kaavio 13-c

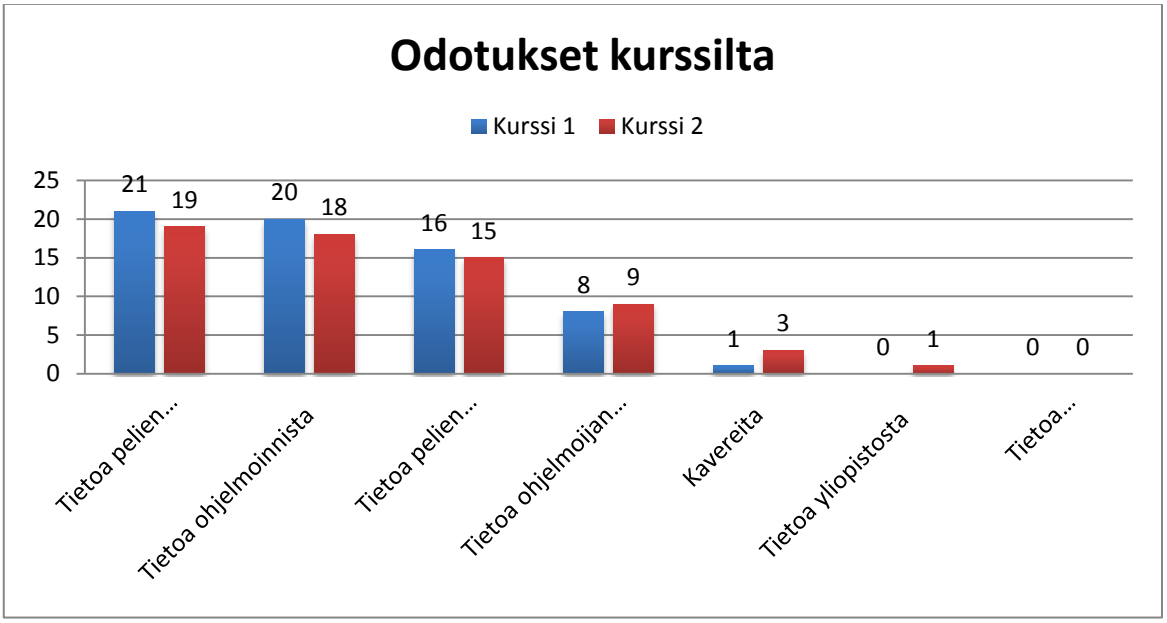

Kiinnostus luonnontieteisiin, keskiarvo
5,0
4,0
3,0
2,0
1,0
2,9
3,0
Kurssi 1
Kurssi 2

Kaavioparvi 14. *Odotukset kurssille (kysymys 15, n=45).*

Kaavio 14-a



Kaavio 14-b



Liite 8. Kurssien loppukyselyn kyselylomake.

KURSSIN LOPPUKYSELY

1. Nimi	_____ (etunimi ja sukunimi) Huomaa, että vaikka kyselyyn laitetaan nimi, henkilöllisyytesi **ei tule esille** tuloksia raportoitaessa. Voit siis huoletta vastata rehellisesti kyselyyn.	
2. Olen tyytyväinen kurssiin kokonaisuutena	Täysin eri mieltä 1 2 3 4 5 Täysin samaa mieltä	
3. Annetut ohjet olivat selkeitä	Täysin eri mieltä 1 2 3 4 5 Täysin samaa mieltä	
4. Oppituntien aloitukset olivat hyviä	Täysin eri mieltä 1 2 3 4 5 Täysin samaa mieltä	
5. Tehtävät olivat vaikeita	Täysin eri mieltä 1 2 3 4 5 Täysin samaa mieltä	
6. Tunnilla oli mukava ilmapiiri	Täysin eri mieltä 1 2 3 4 5 Täysin samaa mieltä	
7. Minulle on hyötyä siitä että osaan suunnitella ja ohjelmoida pelejä	Täysin eri mieltä 1 2 3 4 5 Täysin samaa mieltä	
8. Sain tältä kurssilta	☐ Tietoa ohjelmoinnista ☐ Tietoa pelien tekemisestä ☐ Tietoa pelien suunnittelusta ☐ Uusia kavereita ☐ Tietoa yliopistosta ☐ Tietoa luonnontieteiden opiskelusta ☐ Tietoa ohjelmoijan ammatista Muuta, mitä?:	
9. Millaisen pelin teit? (kerro lyhyesti)	_____ _____	
10. Miksi päädyit tekemään juuri sellaisen pelin?	_____ _____	
11. Parasta kurssilla oli	_____ _____	
12. Huonointa kurssilla oli	_____ _____	
13. Kehittäisin kurssia seuraavasti	_____ _____	
14. Kurssi vastasi odotuksiani	Täysin eri mieltä 1 2 3 4 5 Täysin samaa mieltä	
15. Miksi kurssi EI vastannut odotuksiasi (vastaa tähän vain jos annoit edellisessä kohdassa 1 tai 2 pistettä).	_____ _____	

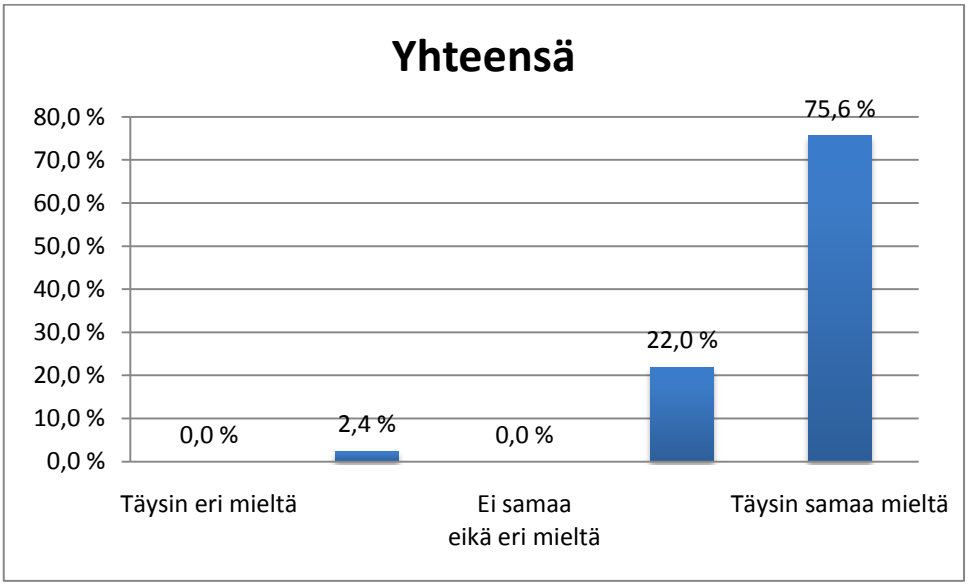
16. Aion hakeutua luonnontieteiden opiskelun pariin tulevaisuudessa (esim. lukiossa pitkä matematiik-ka/fysiikka/kemia/tietotekniikka)	Täysin eri mieltä	1	2	3	4	5	Täysin samaa mieltä
17. Olen valinnut (tai aion valita) koulun tarjoamia valinnaisia ATK-kursseja	Täysin eri mieltä	1	2	3	4	5	Täysin samaa mieltä
18. Minulle helpointa kurssilla oli	_____ _____						
19. Minulle vaikeinta kurssilla oli	_____ _____						
20. Mielestäni parhaiten kurssilla onnistuin	_____ _____						

Kiitos vaivannäöstäsi! Palauta valmis lomake opettajalle.

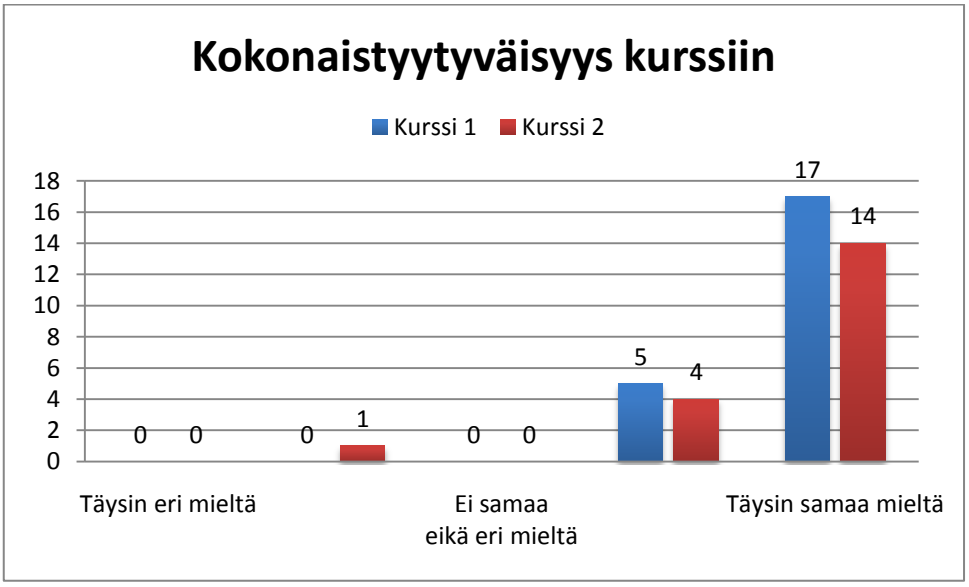
Liite 9. Loppukyselyiden tulokset (n=41).

Kaavioparvi 1. *Kokonaistyytyväisyys kurssiin (kysymys 2, n=41).*

Kaavio 1-a

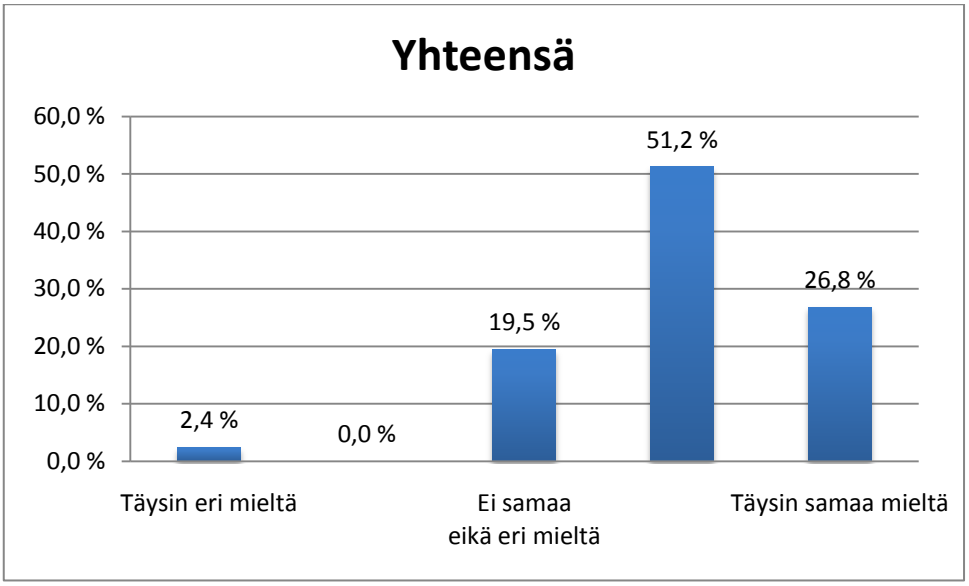


Kaavio 1-b

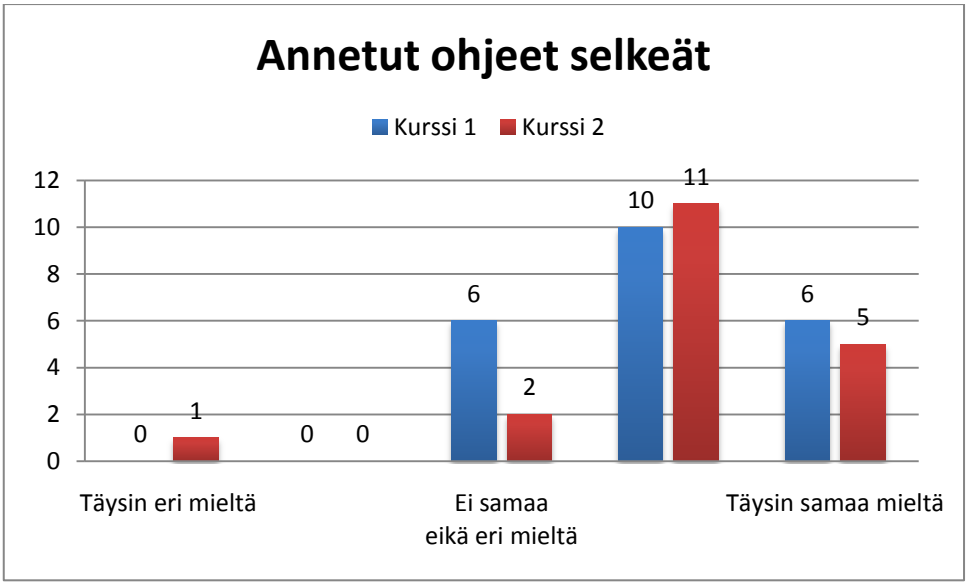


Kaavioparvi 2. *Annettujen ohjeiden selkeys (kysymys 3, n=41).*

Kaavio 2-a

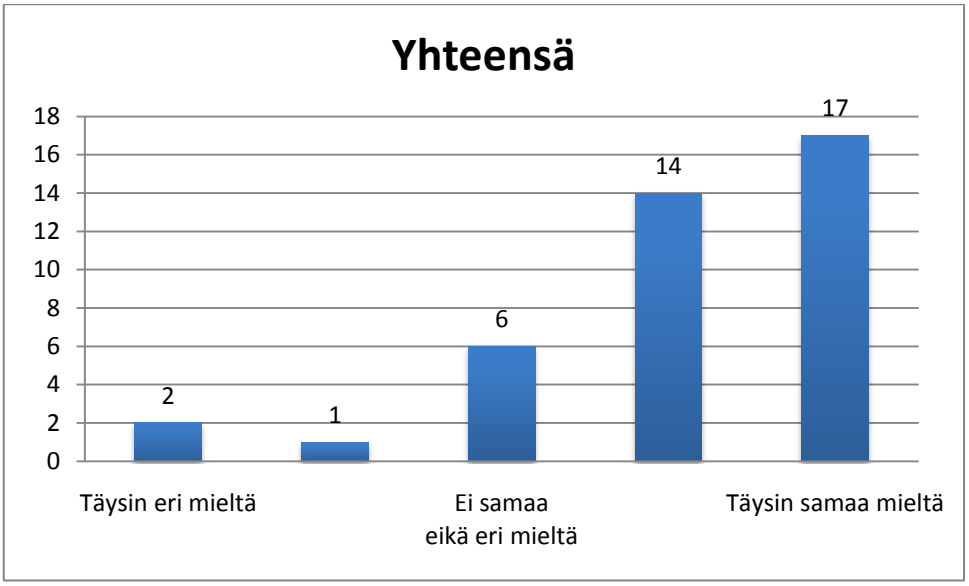


Kaavio 2-b

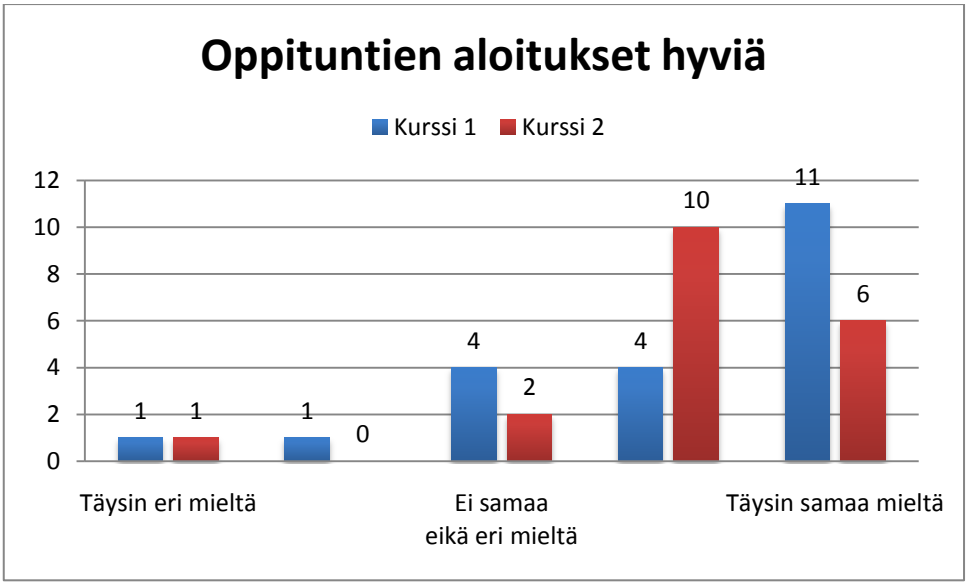


Kaavioparvi 3. *Oppituntien aloitukset olivat hyviä (kysymys 4, n=40).*

Kaavio 3-a

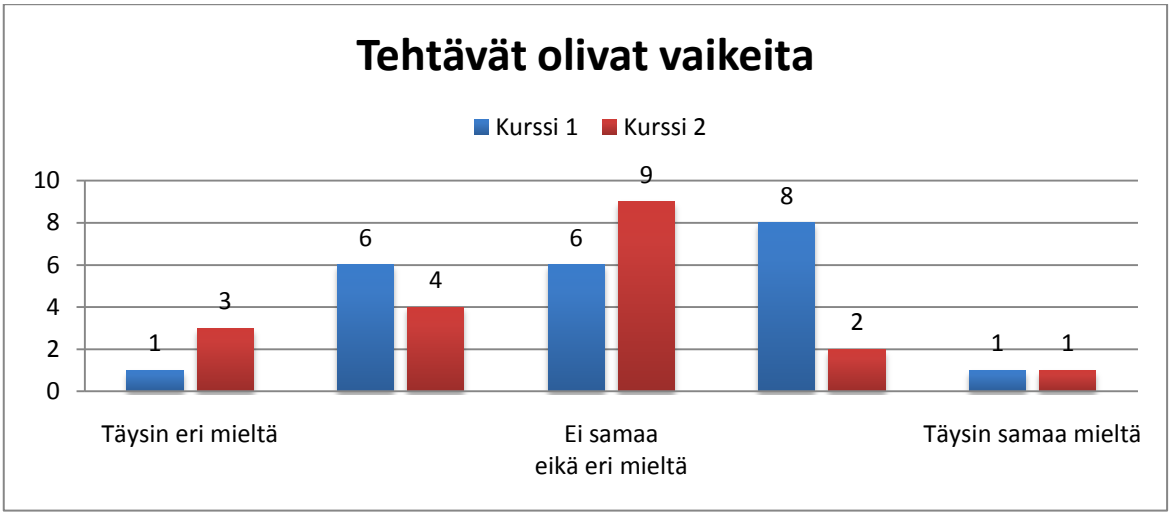


Kaavio 3-b

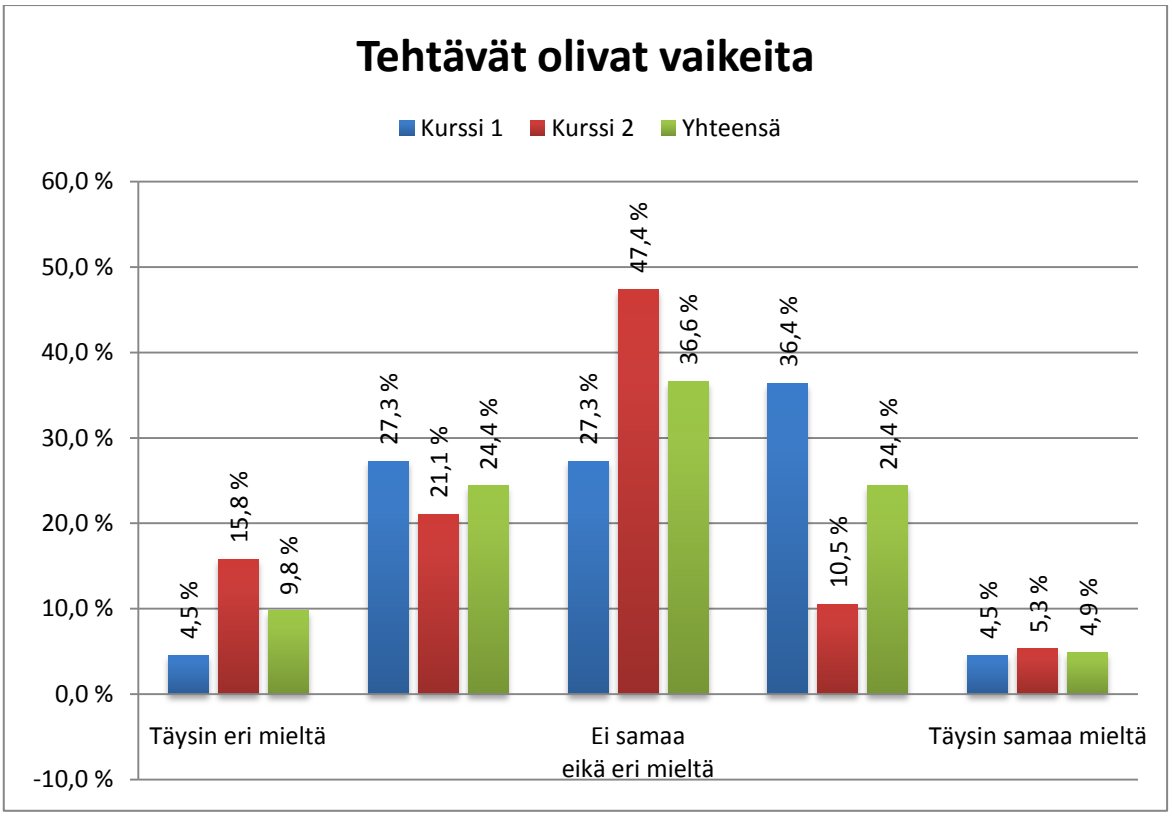


Kaavioparvi 4. *Tehtävien vaikeus (kysymys 5, n=41).*

Kaavio 4-a

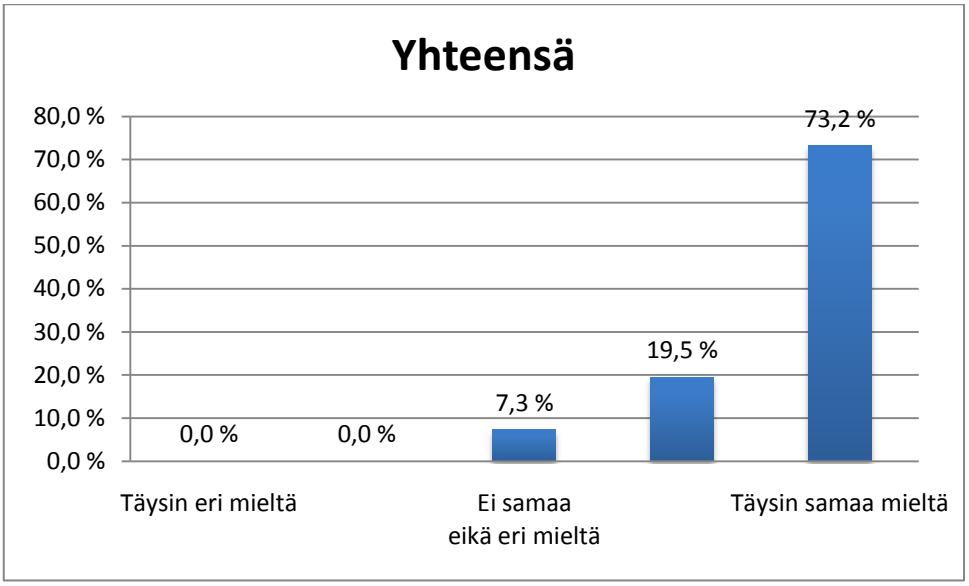


Kaavio 4-b

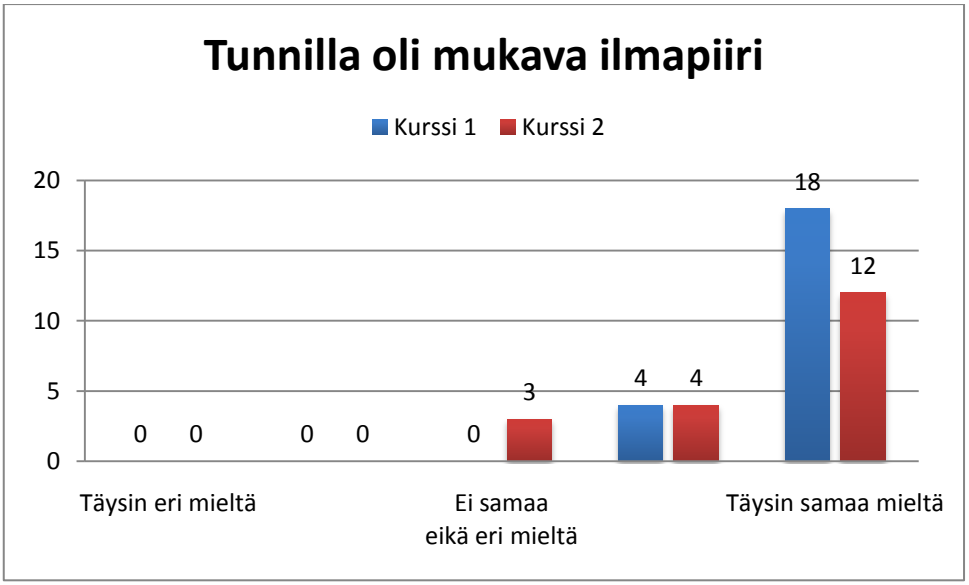


Kaavioparvi 5. *Tunnilla oli mukava ilmapiiri (kysymys 6, n=41).*

Kaavio 5-a

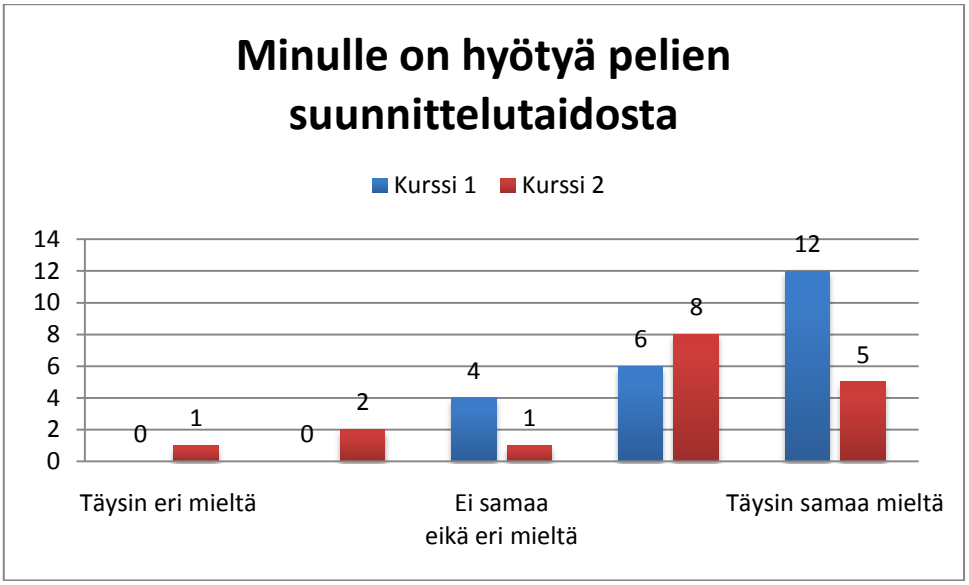


Kaavio 5-b

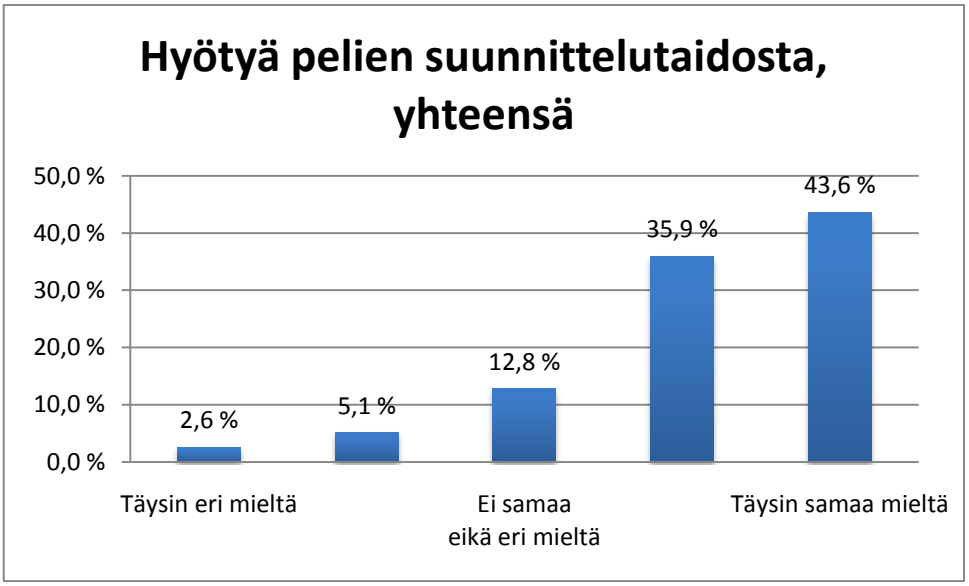


Kaavioparvi 6. *Hyötyä pelien suunnittelutaidosta (kysymys 7, n=41).*

Kaavio 6-a

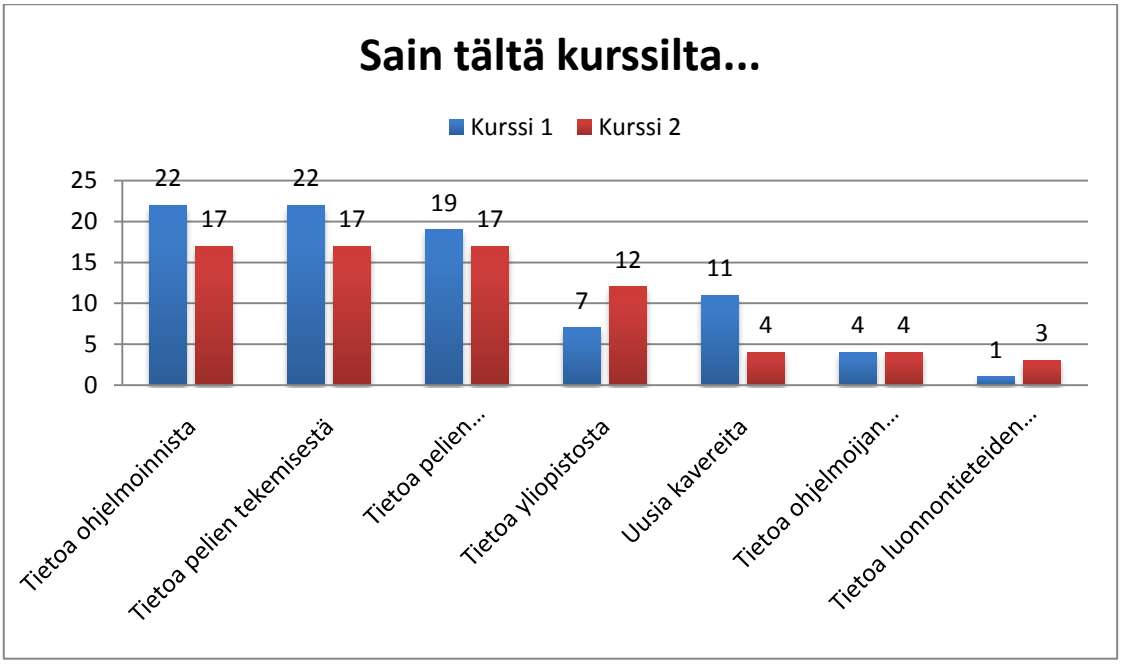


Kaavio 6-b

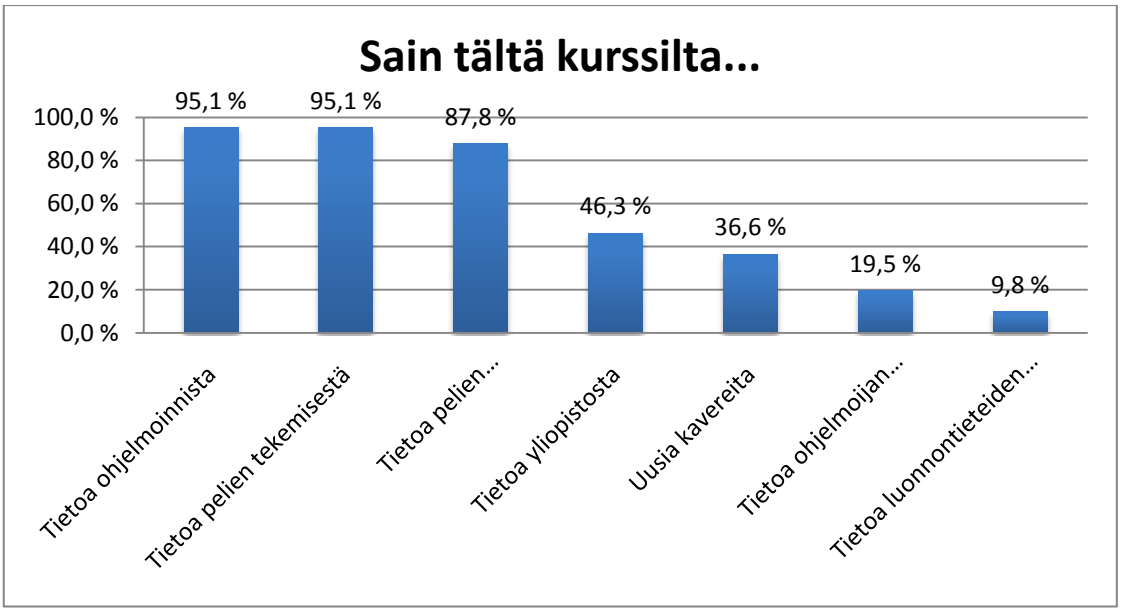


Kaavioparvi 7. *Sain tältä kurssilta (kysymys 8, n=41).*

Kaavio 7-a

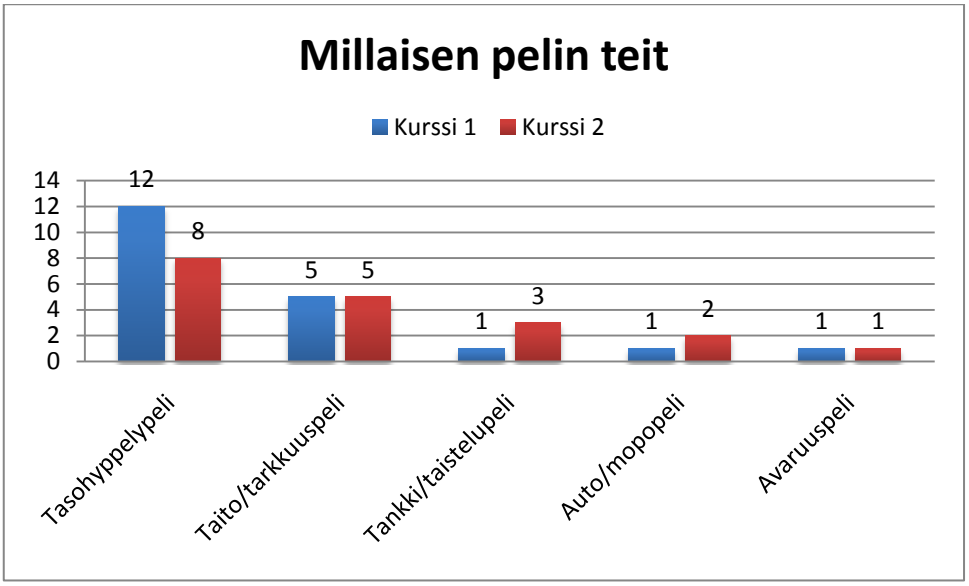


Kaavio 7-b

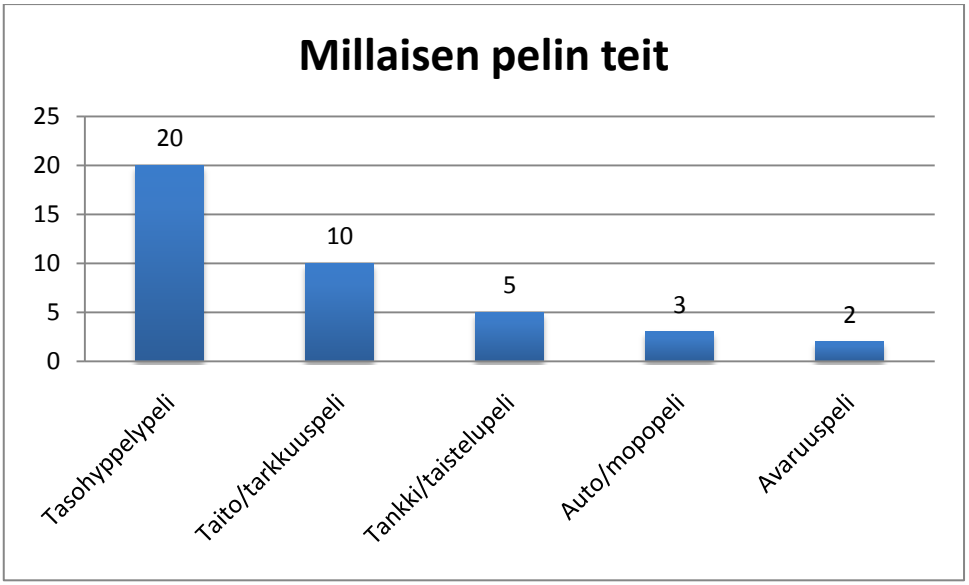


Kaavioparvi 8. *Millaisen pelin teit (kysymys 9, n=41).*

Kaavio 8-a



Kaavio 8-b



***Taulukko 1**. Oppilaiden vastaukset ja niiden ryhmittely kysymyksessä 9: Millaisen pelin teit?*

Tasohyppelypeli	Taito- tai tarkkuuspeli	Tankki- tai muu taistelupeli
• *Yksinkertaisen tasohypplypellin (jossa seikkailee ystävämme Maaaariooo ja Luigi)* • *Tein Super Marion-nimisen pelin omasta mielestäni se on iha ok.* • *tarkoitus oli olla pulmanratkaisu tasohyppely* • *tasohyppely* • *Tikku-ukko pelin* • *tasohyppely* • *Tasohyppelypelin, jossa hypitään ruudun yläpäähän* • *Minä tein tsohyppelypelin.* • *Tasoleijunta (tasohyppely ilman pohjaa)* • *Pitää liikutaa palloa ja kerätä kultakimpaleita.* • *tasohyppelypelin* • *seikkailu-tasohyppely* • *tasohyppely, ammuskelu* • *kaksi pelaajaa yrittävät saada itselleen liekin, jolla sytyttää kentän ylälaidassa oleva kynttilä* • *Tasohyppelypelin*	• *Hiirellä ammutaan liikkuvia kohteita* • *Pallopelin* • *tornipuolustuspeli* • *Labyrinttipelin (en osaa selittää millaisen)* • *Labyrintti, jonka läpi ohjataan vaikeasti ohjattava pallo, 4 tasoa* • *pientä neliötä liikutellaan hiiren avulla eikä saa osua muihin palikoihin tai seiniin* • *skeittipelin, missä hypitään tavaroiden yli* • *Defence pelin* • *Pelasta kaupunki tulvalta, hypi kattoja pitkin* • *Ammuntapelin, jossa pitää ampua palloja.*	• *Kaksin pelattavan tankkiräiskinnän.* • *tankki* • *parin kanssa tankkipelin* • *Norsut ampuu toisia.* • *Laivapeli*

• *Tasohyppelyn, jossa ammutaan.* • *Tasohyppely aikaa vastaan* • *tasohyppely, jossa norsu pomppii ja vesi nousee* • *Munan, joka pomppii pilvillä* • *tasohyppely/taistelupelin, jossa tarkoitus on teurastaa kummitukset*		
Auto- tai mopopeli	**Avaruuspeli**	**Hylätty vastaus**
• *Autopelin, jossa ajetaan kahta autoa. Se on kuvattu ylhäältä päin.* • *mopopeli* • *mopolla ajoa*	• *Avaruuspelin jossa pitää suojella maata.* • *avaruuslentelyn*	• *Hienon*

***Taulukko 2**. Oppilaiden vastaukset ja niiden ryhmittely kysymyksessä 11: Parasta kurssilla oli…*

Oman pelin tekeminen ja suunnittelu	**Ohjelmointi**	**Kurssilla saatu oppi**
• *Oman pelin koodaaminen* • *Oman pelin teko* • *oman pelin teko* • *pelin tekeminen* • *Opin paljon pelien tekemisestä.* • *Oman pelin teko.* • *Onnistumisen ilo ja pelin saaminen valmiiksi.* • *oman pelin tekeminen* • *Pelin teko.* • *Pelin teko.* • *oman pelin tekeminen* • *pelien teko* • *tehdä peli* • *Oman pelin teko* • *Pelin teko.* • *tankkipeli*	• *ohjelmoinnin oppiminen* • *kun ohjelmat sai toimimaan* • *ohjelmoiti itse ja luennot deltta huoneessa* • *Ohjelmointi* • *ohjelmointi* • *Koodaaminen* • *koodaus* • *Kivaa ja rentoa + pääsi koodaamaan.*	• *ohjelmoinnin oppiminen* • *en osaa sanoa. Oppiminen oli mahtavaa!* • *Hmm…Että oppi niin paljon* • *Opin paljon pelien tekemisestä.*
Haasteet, haastavuus	**Kaverit ja ohjaajat**	**Pelisuunnittelu**
• *Haasteellisuus.* • *Kun näki työn tuloksen.* • *kun alkoi muistamaan*	• *ohjaajat* • *kaverit pelikurssilla* • *Kaverit ja sai olla*	• *pelisuunnittelu* • *Pelien suunnittelu.*

miten objekti luodaan • *Looginen peli* • *Looginen peli* • *kun ohjelmat sai toimimaan*	*koneella.*	
Pelien esittely	**Grafiikan tekeminen**	**Hylätyt tai ei-informatiiviset vastaukset**
• *Pelien esittely*	• *grafiikat*	• *Ruokalassa Wieninleike* • *kaikki*

***Taulukko 3**. Oppilaiden vastaukset ja niiden ryhmittely kysymyksessä 12: Huonointa kurssilla oli…*

Kurssi oli liian lyhyt tai kurssin rytmitys epäonnistui	**Oppilas ei mielestään oppinut riittävän hyvin opetettuja asioita tai ei ymmärtänyt annettuja ohjeita**	**Luennot**
• *lyhyt kesto* • *Että se oli niin lyhyt.* • *Kun se loppui* • *Sen loppuminen.* • *Ei ehtinyt tehdä loppuun asti* • *Saisi olla pidempi* • *Liian pitkät ruokatauot.* • *pitkät tauot* • *aikaiset hertykset*	• *Kun ei ymmärtänyt.* • *Kun ei osaa, mutta nyt osaan vähän enemmän kuin alussa.* • *Alussa ainakin oli tosi hankalaa.* • *ohjeet eivät olleet niin helppoja* • *Se kun ohjeita ei tajunnut, mutta ne selitettiin kyllä heti selvällä suomenkielellä, kun kysyi.* • *joskus ei ymmärtänyt mitään koodeista*	• *Tylsät luentoaloitukset* • *Aamuluennot* • *Aamuluennot* • *Luennot*
Opetus, ohjaus tai ohjeistus oli puutteellista	**Ei juuri mikään**	**Kyselylomakkeita oli liian paljon**
• *perusteiden pieni opetus* • *Kun joutui odottamaan välillä pitkään neuvoa* • *Hiukan epäselvät ohjeet* • *epäselviksi jääneet komennot ja niiden*	• *ei mikään* • *ei mikään* • *ei juuri mikään* • *ei mikään*	• *Tällaiset palaute jutut* • *tällaiset kyselyt*

tarkoitukset		
Tekniset ongelmat	**Hylätyt tai ei-informatiiviset vastaukset**	
• *Kone kaatui 5 kertaa*	• *kallis ruoka* • *en tiiä* • *tankkipeli*	

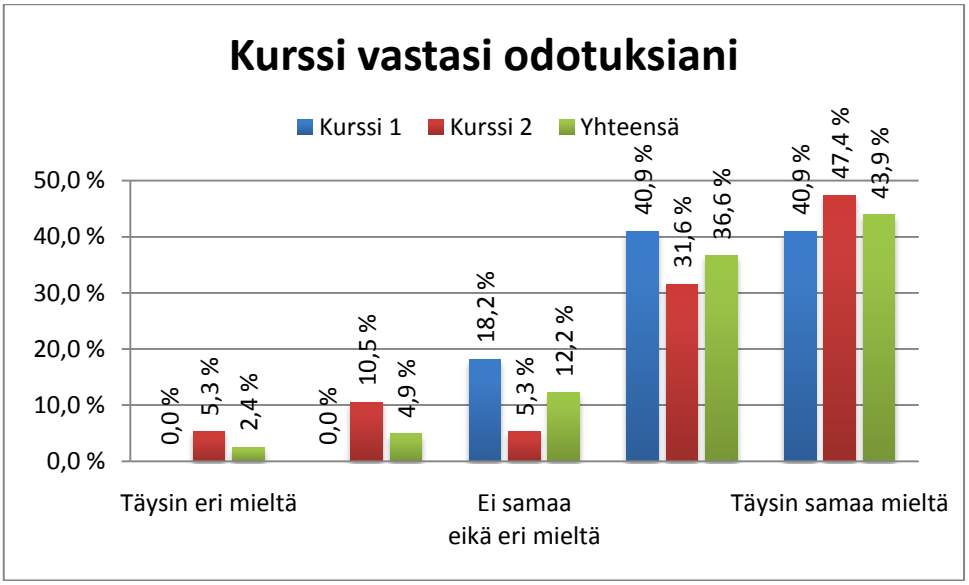
***Taulukko 4**. Oppilaiden vastaukset ja niiden ryhmittely kysymyksessä 13: Kehittäisin kurssia seuraavasti*

Kurssin pitäisi olla pidempi, monipuolisempi tai sen pitäisi sisältää enemmän opetusta	**Annettujen ohjeiden pitäisi olla selkeämmät**	**Aikataulua tai rytmitystä tulisi kehittää**
• *Pidempi, monipuolisempi.* • *opettaisin perusteita enemmän. Muokkaisin pelikirjastoa* • *pitemmät kurssit* • *pidempi kurssi* • *Vois olla jopa vähän pidempi kurssi* • *Vähän pidempi (?)* • *lisää aikaa, jotta voisi tehdä nettipelin* • *vois olla pidempiki ois enemmän aikaa tehä peli* • *pidempi* • *Saisi olla pidempi*	• *selkeitä ohjeita* • *Vähän paremmat ohjeet* • *ohjeet parempia* • *ohjeet selitettäisiin ymmärrettävämmästi.* • *opastettaisiin enemmän perusasioissa* • *jos aika sallii, niin enemmän tietoa koodeista tai kunnon selitykset, vaikka nettiin, ettei käy niin, että vaan kopsaa koodia tajuamatta, mitä se tekee*	• *Aamusta ei luentoja* • *Alkaisi myöhemmin päivällä* • *Lyhyemmät ruokatauot*
Ei kehitettävää	**Ohjelmointia harrastaneille pitäisi järjestää oma, vaativampi kurssi**	**Kirjaston pitäisi olla monipuolisempi**

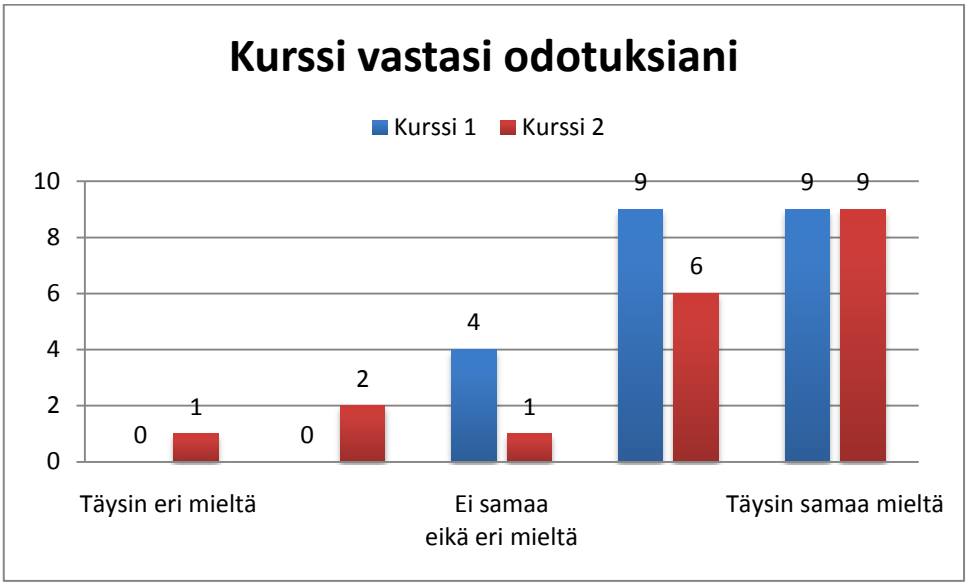
• *ei ole kehitettävää* • *En mitenkään.* • *Se on hyvä näin*	• *Tarvittaisiin myös toinen vaikeampi kurssi.* • *Laittaisin ehkä 2 ryhmää. Toisessa oi ne joilla on kokemusta ja toisessa aloittelijat.*	• *opettaisin perusteita enemmän. Muokkaisin pelikirjastoa* • *korjaisin/laajentaisin kirjastoa*
Tiettyjä pelityyppejä lisää	**Enemmän pelaamista**	
• *Panostaisin roolipelien tekemiseen.*	• *split screeni ja Lanit mahdolliseksi*	

Kaavioparvi 9. *Kurssi vastasi odotuksiani (kysymys 14, n=41).*

Kaavio 9-a

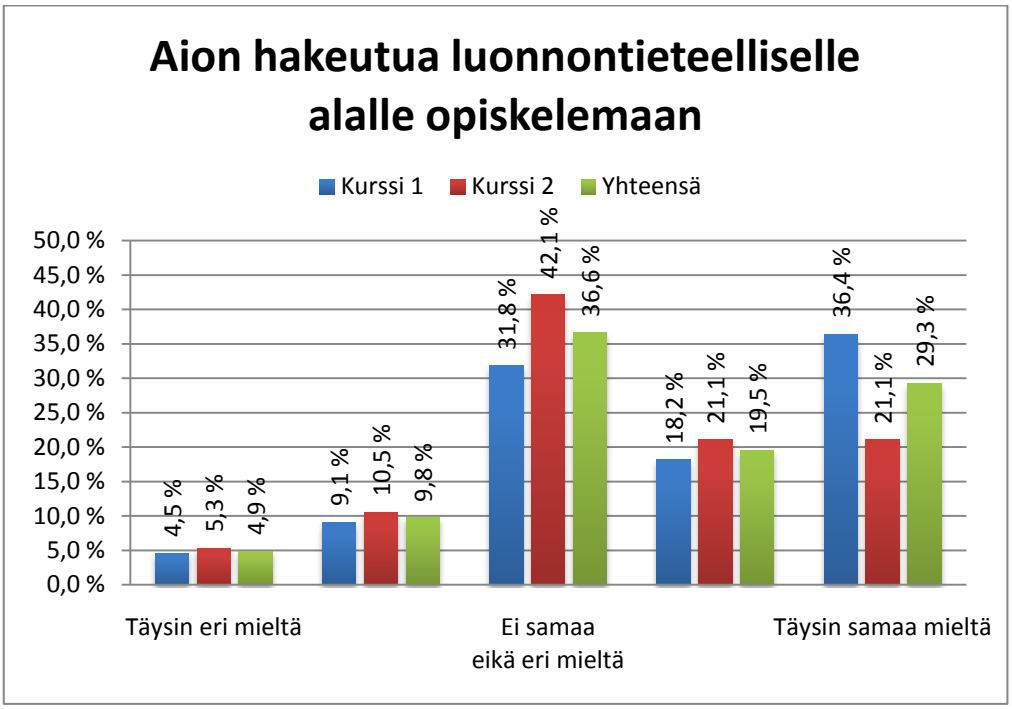


Kaavio 9-b

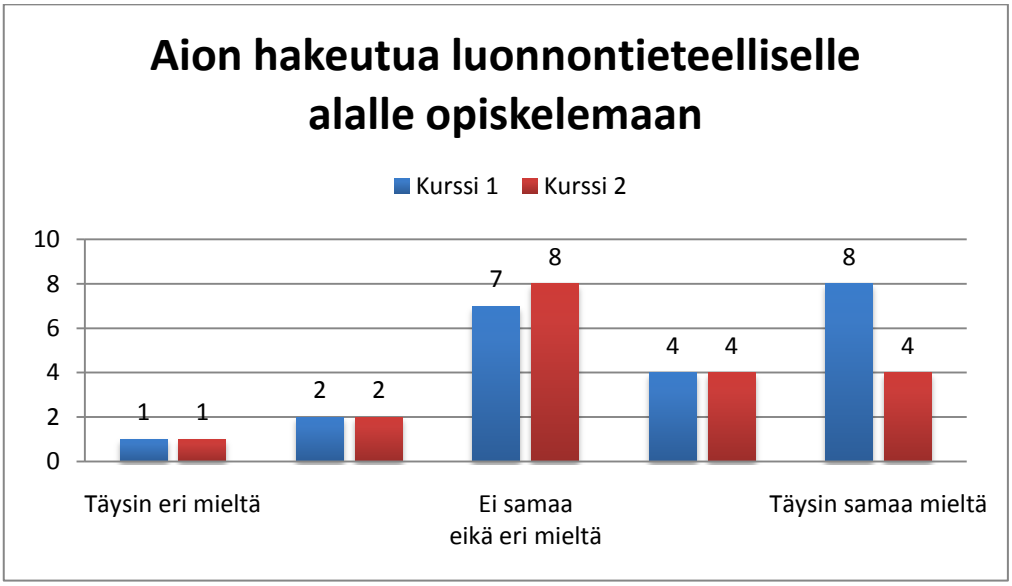


Kaavioparvi 10. *Aion hakeutua luonnontieteelliselle alalle opiskelemaan (kysymys 16, n=41).*

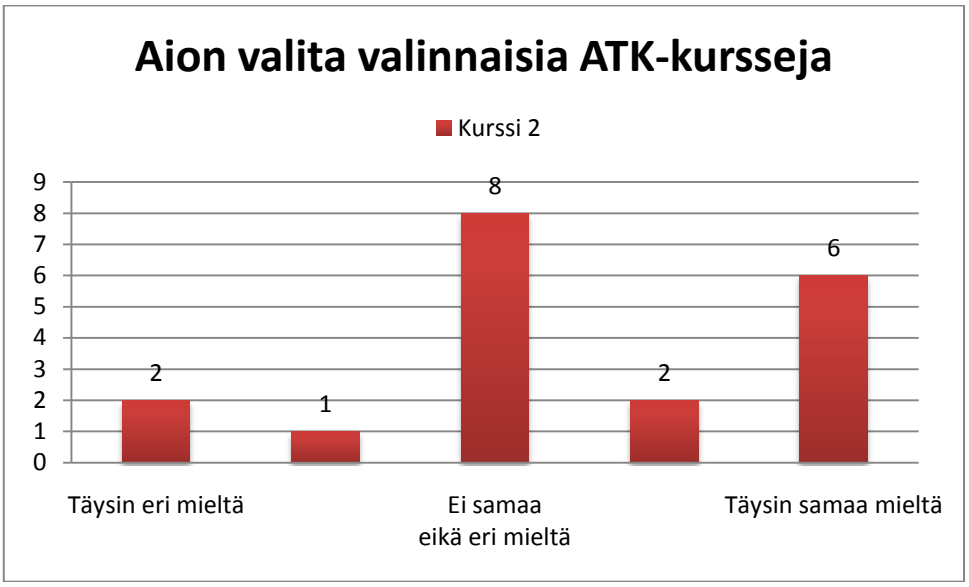
Kaavio 10-a



Kaavio 10-b



Kaavio 11. *Aion valita kouluni mahdollisesti tarjoamia valinnaisia ATK-kursseja (kysymys 17, n=19).*



***Taulukko 5**. Oppilaiden vastaukset ja niiden ryhmittely kysymyksessä 17 (jälkimmäisellä kurssilla 18): Helpointa kurssilla oli…*

Pelin suunnittelu	**Ohjelmointi, sen osa-alueet ja orientaatio ohjelmointiajatteluun**	**Luennot tai luennolla esiintyneet yksittäiset asiat**
• *Suunnittelu* • *suunnittelu* • *suunnittelu* • *Suunnittelu ja piirtäminen* • *pelien suunnittelu.* • *suunnittelu* • *suunnitteleminen paperille* • *kun teimme pelien suunnitelmia* • *suunnittelu* • *oman pelin suunnittelu* • *Pelin teko*	• *sopeutuminen ohjelmointiin* • *Alun ohjelmointi* • *ohjelmointi* • *Pelin teko* • *ohjelmointi, erityisesti ehtolauseet* • *ohjelmointi*	• *koordinaatit* • *Luentojen kuunteleminen.* • *Kuunnella luentoja* • *vektorit*
Pelin toteuttamiseen liittyvät oheistoiminnot	**Harjoitustehtävät ja niiden tekeminen**	**Valmiin koodin hyväksikäyttäminen ja koodin muuntelu**
• *biisin tekeminen peliin* • *pelin testaus* • *piirtäminen* • *testaaminen ja muitten*	• *Harjoitukset* • *tehtävät* • *tehtävät* • *tehdä ohjeitten mukaan*	• *valmiiden koodien muuntelu…* • *valmiin koodin editointi.* • *valmiin koodin*

kanssa työskentely		*editointi.*
Pelien pelaaminen	**Yleinen työskentely kurssilla**	**Lähes kaikki**
• *Pelaaminen* • *pelata pelejä* • *pelata pelejä*	• *Se että tein paritöitä* • *testaaminen ja muitten kanssa työskentely*	• *kaikki* • *melkein kaikki*
Hylätyt tai ei-informatiiviset vastaukset		
• *en osaa sanoa* • *runaway* • *Hmm… En osaa sanoa*		

Taulukko 6*. Oppilaiden vastaukset ja niiden ryhmittely kysymyksessä 18 (jälkimmäisellä kurssilla 19): Vaikeinta kurssilla oli…*

Ohjelmointi yleisesti	**Pelin suunnittelu ja tuottaminen**	**Työskentely kurssilla ja aikataulussa pysyminen**
• *Itse koodien käsittely. (Kokemusta vain HTML:stä ja CSS:stä, joka ei liity pelien tekoon mitenkään)* • *Yksityiskohtien ohjelmointi.* • *jotkut kohdat ohjelmoinnista* • *tehdä koodeja* • *osa koodauksesta* • *jotkut koodit* • *Koodaaminen* • *kun piti itse keksiä, että mikähän koodinpätkä tohon tulis, ettei koko roska hajoais…* • *jotkut koodit* • *koodaaminen* • *Ehkäpä ohjelmointi* • *ohjelmointi* • *Alussa hankalat ohjelmoinnit.* • *joidenkin asioiden ohjelmointi (jota ei opetettu)*	• *pelin suunnittelu* • *palikkatesti* • *peli* • *Peli-idean keksiminen* • *Tikku-ukkojen liikuttaminen*	• *tehdä joku uusi asia peliin* • *tarttuu toimeen, aloittaa.* • *Oman pelin aikataulussa pysyminen* • *aikataulussa pysyminen*
Ohjelmointikielen kieliopin ja rakenteen omaksuminen	**Virheiden jäljittäminen**	**Ei juuri mikään**
• *Kielen rakenne*	• *virheiden*	• *Ei mikään sen jälkeen*

• *Koodin oikeinkierjoitus.* • *kielen omaksuminen* • *Muistaa kaikkea koodia*	*paikantaminen* • *virheiden korjaus* • *virheitten etsiminen* • *Bugien ja muiden virheiden poisto*	*kun sen tajusi* • *ei mikään* • *ei juuri mikään*
Yksittäiset harjoitustehtävät	**Ohjeiden seuraaminen**	**Hylätyt tai ei-informatiiviset vastaukset**
• *toisen päivän ekä tehtävä* • *kaikki tietotekniikka jutut*	• *Ohjeiden seuraaminen* • *ohjeitten ymmärtämiset*	• *En tiedä* • *En tiedä.*

Taulukko 7*. Oppilaiden vastaukset ja niiden ryhmittely kysymyksessä 19 (jälkimmäisellä kurssilla 20): Parhaiten onnistuin…*

Ohjelmointi yleisesti	**Pelin suunnittelu ja tuottaminen**	**Työskentely kurssilla ja aikataulussa pysyminen**
• *Itse koodien käsittely. (Kokemusta vain HTML:stä ja CSS:stä, joka ei liity pelien tekoon mitenkään)* • *Yksityiskohtien ohjelmointi.* • *jotkut kohdat ohjelmoinnista* • *tehdä koodeja* • *osa koodauksesta* • *jotkut koodit* • *Koodaaminen* • *kun piti itse keksiä, että mikähän koodinpätkä tohon tulis, ettei koko roska hajoais…* • *jotkut koodit* • *koodaaminen* • *Ehkäpä ohjelmointi* • *ohjelmointi* • *Alussa hankalat ohjelmoinnit.* • *joidenkin asioiden ohjelmointi (jota ei opetettu)*	• *pelin suunnittelu* • *palikkatesti* • *peli* • *Peli-idean keksiminen* • *Tikku-ukkojen liikuttaminen*	• *tehdä joku uusi asia peliin* • *tarttuu toimeen, aloittaa.* • *Oman pelin aikataulussa pysyminen* • *aikataulussa pysyminen*
Ohjelmointikielen kieliopin ja rakenteen omaksuminen	**Virheiden jäljittäminen**	**Ei juuri mikään**
• *Kielen rakenne*	• *virheiden*	• *Ei mikään sen jälkeen*

• *Koodin oikeinkierjoitus.* • *kielen omaksuminen* • *Muistaa kaikkea koodia*	*paikantaminen* • *virheiden korjaus* • *virheitten etsiminen* • *Bugien ja muiden virheiden poisto*	*kun sen tajusi* • *ei mikään* • *ei juuri mikään*
Yksittäiset harjoitustehtävät	**Ohjeiden seuraaminen**	**Hylätyt tai ei-informatiiviset vastaukset**
• *toisen päivän ekä tehtävä* • *kaikki tietotekniikka jutut*	• *Ohjeiden seuraaminen* • *ohjeitten ymmärtämiset*	• *En tiedä* • *En tiedä.*

Liite 10. Esimerkki oppilaan tekemästä pelisuunnitelmasta.

Tekijät. X ja Y

Tietoja pelistä. Pelin nimi: Tasohyppelypeli Pelialusta: Windows Pelaajien lukumäärä: 1 pelaaja

Pelin tarina. Random tyypit ovat kidnapanneet Bobot ja vanginneet sinut jonnekkin random paikkaan?

Pelin idea ja tavoitteet. Kentästä pitää pelastaa kaikki Bobot ja sitten mennä portista ulos. Maailmassa on kenttiä ja kun pääsee yhden kentän läpi niin avaa maailmasta toisen kentän. Pahat random tyypit pitää ampuu seulaks mutta ei oo pakko jos pääsee niiden ohi menee. Ohjattava tyypi on nimeltään Rampe "Rambo" Randominen. Hän on kovis, jolla on pyssy. Jos ampuu laatikoita niin niistä voi saada pisteitä\elämiä\stuffia. Stuffista tulee kuolemattomaksi(hetkeksi). Rampella on myös puukko, koska onhan hän oikea suomalainen junttard. Jos Rampe osuu pahikseen niin hän menettää elämän ja aloittaa kentän alusta, mutta säilyttää pisteet. Jos hän menettää kaikki elämät niin menettää kaikki pisteet ja peli loppuu, mutta voi aloittaa taas siitä kentästä mihin jäi.

Hahmotelma pelistä. Kentät tehdään .txt muotoon, jotta kaikki voivat tehdä kenttiä ja maailmoja.

Toteutuksen suunnitelma. Keskiviikko: Aloittaisin tekemään ensin pelihahmot, joita voi liikuttaa. Pelihahmoilla ei tarvitse aluksi vielä olla todellista ulkoasua. Seuraavaksi tekisin kentän, jossa on tasoja. Tasojen päälle pystyy hyppäämään. Kaikki Bobot pitäsisi pystyä keräämään jotta siirryt seuraavalle tasolle. Torstai: Lisätään peliin maali, johon meneminen pelin lopuksi kertoo, että kenttä on läpäisty ja pääsee seuraavaan kenttään. Lisään ampumisen. Perjantai: Lisätään peliin ääniä ja taustamusiikki. Hiotaan grafiikkaa. Viimeistellään peli. Jos aikaa jää: Tehdään lisää kenttiä peliin.

Liite 11. Esimerkki oppilaan tekemästä pelisuunnitelmasta.

Tekijä. Oppilas X.

Tietoja pelistä. Pelin nimi: Hyppijä Pelialusta: Windows tai Xbox 360 Pelaajien lukumäärä: 1

Pelin tarina. Suurkaupunkiin on tullut suuri tulva. Kaupungin pääviemärin voi avata vain kaupugin toiselta laidalta. Parkour-Pekan täytyy päästä kaupungin toiselle puolelle avaamaan viemäri, mutta hän ei osaa uida. Ainoa tie on kattoja pitkin. Pekka lähtee siis matkalle hyppien kerrostalojen katolta toiselle.

Pelin idea ja tavoitteet. Pelaaja pomppii kerrostalojen katoilta toiselle. Mitä pidemmälle pelaaja pääsee, sitä enemmän pisteitä hän saa. Kun pelaja pääsee tarpeeksi pitkälle, hän voittaa pelin (tai pääsee seuraavalle tasolle jos jää aikaa). Jos pelaaja putoaa alas (ei osu seuraavalle katolle), hän kuolee. Kamera liikkuu itsestään tasaista vauhtia (oikealle) ja pelaaja yrittää pysyä ruudun sisällä.

Toteutuksen suunnitelma. Ke Tehään pelaaja, talot ja liikkuva kamera. To Pisteenlaskut, tavoitteet ja seuraukset. Pe Lisäätään grafiikat ja viimeiställään peli hyväksi. Lisätään tasoja jos jää aikaa.