



Pelikehityksen hyödyntäminen ohjelmoinnin opetuksessa

Oulun yliopisto
Tietojenkäsittelytiede
Kandidaatin tutkielma
Lauri Pudas
2024

Tiivistelmä

Ohjelmointitaidot ovat oleellinen osa monia tekniikan aloja. Tästä huolimatta näiden taitojen opiskelu on usein erityisesti aloitteleville opiskelijoille haastavaa. Opeteltävien uusien käsitteiden ja konseptien määrä on suuri ja ongelmat saattavat tuntua vaikeilta hahmottaa. Tämä herkästi johtaa siihen, että opiskelijan motivaatio ohjelmointia kohtaan heikkenee. Ohjelmointi voi myös vaatia opiskelijalta uusia ja erilaisia tapoja lähestyä ongelmia, kuin mihin hän on tottunut.

Tämän tutkielman tarkoituksena oli tarkastella pelikehityksen hyödyntämistä ohjelmoinnin opetuksessa. Tutkielma toteutettiin suorittamalla kirjallisuuskatsaus. Tutkielmassa käytiin läpi erilaisia opetusmenetelmiä, jotka hyödyntävät pelikehitystä ohjelmoinnin opetuksessa ja näiden opetusmenetelmien tuottamia tuloksia. Erilaisia tapoja hyödyntää pelikehitystä on paljon ja niillä on omat vahvuutensa ja haasteensa. Tästä huolimatta aikaisempien tutkimusten tuottamat havainnot ovat yleisesti yhtenäisiä. Pelikehityksellä on merkittävä positiivinen vaikutus ohjelmointia opiskelevien henkilöiden motivaatioon ja osaamiseen. Lisäksi pelikehityksen havaittiin olevan hyvä tapa kehittää opiskelijoiden ongelmanratkaisutaitoja.

Avainsanat

Pelikehitys, Ohjelmointi, Opetus

Ohjaaja

Jouni Lappalainen

Sisällysluettelo

1. Johdanto.....	4
2. Tutkimusmenetelmät	6
3. Aiempi tutkimus	8
3.1 Motivaatio ohjelmoinnin ja pelikehityksen opettamiselle	8
3.1.1 Kognitiivisten taitojen kehittäminen	8
3.1.2 Olio-ohjelmointi	9
3.1.3 Korkea keskeyttämisprosentti ja matala läpikäyminen	9
3.1.4 Tietojenkäsittelyn houkuttelevuuden parantaminen	10
3.2 Pelikehityksen käyttö ja menetelmät ohjelmoinnin opetuksessa	10
3.2.1 Opetusympäristöt.....	10
3.2.2 Teollisuudessa käytetyt kehitysympäristöt.....	12
3.2.3 Muut	13
3.3 Pelikehityksen hyödyt ja haasteet ohjelmoinnin opetuksessa.....	14
3.3.1 Motivaatio ja opiskelun mielekkyys.....	14
3.3.2 Menestys myöhemmissä opinnoissa.....	15
3.3.3 Kognitiiviset kyvyt	15
3.3.4 Ohjelmointitaitojen kehitys	15
3.3.5 Haasteet	17
4. Pohdinta.....	18
5. Johtopäätökset	21
Lähteet.....	23

1. Johdanto

Tietoteknisellä osaamisella ja ohjelmoinnilla on yhä suurempi merkitys nyky-yhteiskunnassa. Ohjelmoinnin osaajia tarvitaan nykyään lähes jokaisella alalla. Ohjelmoinnin perusteita on alettu opettamaan myös jo varhaisemmilla koulutusasteilla lukioissa ja peruskouluissa. Johdannolliset ohjelmointikurssit koetaan kuitenkin usein haastaviksi. Tästä kertoo esimerkiksi se, että noin kolmasosa johdannollisten kurssien opiskelijoista ei läpäise kurssia (Bennedsen & Caspersen, 2007). Opiskelijoiden kokemat haasteet ajan puutteen, ohjelmoinnin vaikeuden ja motivaation kanssa ovat osa merkittävistä syistä sille, miksi opiskelijat keskeyttävät tietojenkäsittelyn opiskelun (Vivian ym., 2013). Tekijöitä, jotka vaikuttavat opiskelijoiden opintojen keskeyttämiseen tietojenkäsittelyn alalla ovat, muun muassa opiskelijoiden heikko ymmärrys siitä, mitä tietojenkäsittely todellisuudessa sisältää ennen opiskelujen aloittamista, sekä puutteet opiskelijoiden matemaattisissa taidoissa ja opetusmenetelmissä (Beaubouef & Mason, 2005). Yksi tapa kehittää opiskelijoiden kiinnostusta ja motivaatiota alaa kohtaan on parantaa ohjelmoinnin opetuksessa käytettyjä menetelmiä.

Yksi merkittävä alue ohjelmoinnin opetusta on olio-ohjelmointi. Olio-ohjelmoinnissa opiskelijoilla on usein vaikeuksia abstraktion käsitteiden ymmärtämisessä. Pelikehitys tarjoaa opiskelijoille aktiivisen ja käytännönläheisen opetustavan. Pelikehitys on houkutteleva tapa opettaa ohjelmointia, sillä usein yksinkertaistenkin pelien toteuttaminen vaatii olio-ohjelmoinnin käsitteiden ymmärtämistä ja soveltamista (Giordano & Maiorana, 2013).

Tämän työn tarkoituksena on vastata seuraaviin tutkimuskysymyksiin:

”Miten pelikehitystä voidaan hyödyntää ohjelmoinnin opetuksessa?” ja

”Millaisia vaikutuksia pelikehityksellä on opiskelijoiden osaamiseen ja motivaatioon ohjelmoinnissa?”

Työssä syvennytään tutkimuksissa käytettyihin pelikehitystä hyödyntäviin oppimateriaaleihin ja menetelmiin ja niiden hyötyihin ja haasteisiin, sekä motivaatioihin kehittää ohjelmoinnin opetusta pelikehityksen avulla. Pelikehitystä hyödyntäviä oppimateriaaleja ja menetelmiä tarkasteltaessa keskitytään erityisesti niiden tekniseen toteutukseen. Tähän kuuluvat erilaiset kehitysympäristöt, pelimoottorit, kirjastot ja ohjelmointikehykset sekä perustelut niiden valinnalle opetuksen opetustavoitteiden ja kohderyhmän kannalta. Tutkielmassa käydään myös läpi pelikehityksen vaikutusta opiskelijoiden motivaatioon, opiskelumenestykseen, kognitiivisiin kykyihin ja ohjelmointitaitojen kehitykseen. Lisäksi tarkastellaan yleisiä haasteita, joita pelikehityksen hyödyntäminen ohjelmoinnin opetuksessa kohtaa.

Tutkielmassa tutkimusmenetelmänä on kirjallisuuskatsaus. Käymällä läpi aiheesta julkaistu tieteellinen kirjallisuus, voidaan tutkimuksista koota yhteen toistuvat ja keskeiset havainnot sekä aiheeseen liittyvät haasteet ja puutteet. Näin pyritään selvittämään aiheen nykytila tutkimuskysymysten kannalta sekä tuomaan esiin mahdollisia jatkotutkimusaiheita.

Tutkielman seuraavassa luvussa käydään läpi yksityiskohtaisesti kirjallisuuskatsauksessa käytetyt menetelmät, hakuprosessit ja rajaukset. Kolmannessa luvussa käydään läpi ja paneudutaan tarkemmin aiheesta tehtyyn tieteelliseen kirjallisuuteen. Aikaisemmista tutkimuksista tuodaan esiin motivaatioita ohjelmoinnin opettamiselle sekä syitä, miksi

ohjelmoinnin opetusta tulisi kehittää. Lisäksi tehdään kattava katsaus eri menetelmiin ja opetusmateriaaleihin, joita aikaisemmissa tutkimuksissa on käytetty ja niiden tuottamiin tuloksiin. Luvun lopuksi käydään läpi esiin nousseita keskeisiä hyötyjä, mitkä tekevät pelikehityksen hyödyntämisestä kannattavaa. Samalla kuitenkin tuodaan esiin myös pelikehityksen hyödyntämiseen ohjelmoinnin opetuksessa liittyviä haasteita ja tekijöitä, joita tulisi ottaa huomioon opetusta suunniteltaessa. Neljäs luku keskittyy tutkimuksessa tehtyjen havaintojen pohdintaan. Aikaisempien tutkimuksien läpikäynnin perusteella tehtyjä havaintoja tuodaan yhteen ja tarkastellaan laajemmassa kontekstissa. Tämän pohdinnan perusteella pyritään tarkastelemaan tehdyn työn merkitystä ja vastaamaan tutkimuskysymyksiin. Lopuksi yhteenvedossa tiivistetään tutkimuksen tulokset tutkimuskysymysten avulla sekä tuodaan esiin mahdollisia kysymyksiä jatkotutkimukselle.

2. Tutkimusmenetelmät

Tämä tutkielma toteutettiin kirjallisuuskatsauksena. Kirjallisuuskatsauksen tarkoituksena on syntetisoida aiheesta tehdyn kirjallisuuden perusteella oleellinen tieto, selvittää aiheen nykytila sekä löytää mahdolliset aukot aikaisemmissa tutkimuksissa. Tämä tutkielma siis pyrkii keräämään ja yhdistämään merkittävän tieteellisen kirjallisuuden tiedon ja puutteet pelikehityksestä työkaluna ohjelmoinnin opetuksessa ja löytämään siitä kysymyksiä jatkotutkimukselle. Tarkoituksena on myös kuvata ne menetelmät, joilla tämä kirjallisuuskatsaus on toteutettu, jotta sen tulokset olisivat toistettavissa.

Kirjallisuuskatsauksen ytimessä on aiheesta tehty tieteellinen kirjallisuus. Näiden teosten löytämiseksi tässä kirjallisuuskatsauksessa hyödynnettiin tieteellisiä tietokantoja kuten IEEE Xplore, ACM ja Scopus. Tietokantojen hyödyntämisessä oleellista on hakulausekkeiden ja -kriteerien käyttö. Avonaiset ja yleisiä hakutermejä käyttävät hakulausekkeet tuottavat valtavan määrän tuloksia, jolloin tulokset sisältävät suuren määrän täysin muihin aiheisiin liittyviä lähteitä. Toisaalta hyvin tarkkaan määritetyt haut puolestaan voivat helposti jättää pois lähteitä, joissa halutut asiat ovat esimerkiksi ilmaistu eri sanoilla.

Tämän kirjallisuuskatsauksen tekeminen alkoi erilaisten hakulausekkeiden kokeilemisella IEEE Xplore -tietokannassa. Hakusanoja, joita tässä vaiheessa hyödynnettiin, olivat muun muassa: ”game development”, ”programming”, ”learn*”, ”teach”, ”design” ja ”education”. Tarkoituksena oli löytää hakulausekkeitä, joilla tuloksia saatiin noin 50–200.

Lopulta päädyttiin kolmeen hakulausekkeeseen, joita lähteiden keräämisessä hyödynnettiin:

1. ”game development” AND teach* AND programming
2. video game teach* learn* programming development
3. ”game development” AND programming AND education

Alarajaa lähteiden julkaisuvedelle ei asetettu, sillä suurin osa lähteistä oli suhteellisen uusia ja tämä tutkielma käsittelee asioita, kuten ohjelmointia ja oppimista, joiden perusteet eivät ole juurikaan muuttuneet. ACM tietokannassa hakulausekkeet rajattiin abstraktien perusteella. Tämä tehtiin, jotta hakutulosten määrä saatiin pidettyä toivotussa määrässä.

Lähteiden arviointi noudatti seuraavaksi kuvattua menetelmää. Hakulausekkeilla löytyneistä lähteistä tarkistettiin ensin, ovatko ne kopioita. Toisin sanoen ensimmäisenä jokaisen löydetyn lähteen kohdalla selvitettiin, onko kyseinen lähde arvioitu jo aikaisemman haun yhteydessä. Tämä toteutettiin tuomalla jokaisen haun jälkeen saadut tulokset tiedostoon, josta niitä pystyttiin hakemaan uusien lähteiden arvioinnin yhteydessä.

Tämän jälkeen arviointia jatkettiin tarkastelemalla lähteiden otsikoita. Otsikoiden avulla pyrittiin valitsemaan lähteet, jotka vastasivat tämän tutkimuksen tutkimuskysymysten tavoitteita. Tärkeinä ominaisuuksina otsikoille olivat opetuskonteksti, pelikehitys tai siihen liittyvät ohjelmistot ja ohjelmointi tai ohjelmallinen ajattelu. Otsikoiden avulla valinnassa kuitenkin pyrittiin sisällyttämään myös lähteet, jotka vain osittain täyttivät nämä kriteerit. Tämä tehtiin, koska huomattiin tarkan otsikoiden perusteella tehdyn arvioinnin jättävän pois myös lähteitä, jotka sisälsivät relevanttia tietoa.

Otsikoiden perusteella valikoituneista lähteistä arvioitiin seuraavaksi abstraktit. Abstrakteista tuli käydä ilmi, että tutkimuksessa käsiteltiin pelikehitystä työkaluna ohjelmoinnin tai ohjelmoinnin kannalta tärkeiden taitojen opettamisessa. Esimerkiksi lähteet, jotka keskittyivät pelkästään oppimateriaalien pelillistämiseen tai peleillä opettamiseen jätettiin pois. Lähteissä käytettiin paljon termejä kuten pelillistäminen, peleillä opettaminen ja pelipohjainen oppiminen (game-based learning). Näiden termien merkitys vaihteli hieman eri lähteiden välillä, joten joidenkin lähteiden kohdalla arvioitiin myös johdanto, jotta varmistuttiin tutkimuksen käsittelevän haluttua aihetta.

Näiden vaiheiden jälkeen jäljellä oli 79 valittua lähdetä. Nämä lähteet arvioitiin vielä uudelleen relevanttiuden sekä luotettavuuden mukaan. Lisäksi lähteitä kerättiin hakujen perusteella valittujen lähteiden viittauksista. Tämän jälkeen jäljellä oli 43 lähdetä, joiden perusteella tämän tutkielman tutkimuskysymyksiin pyrittiin hakemaan vastauksia. Pelikehityksen hyödyntäminen ohjelmoinnin opetuksessa on suhteellisen tarkkaan rajattu aihe itsessään verrattuna esimerkiksi pelillistämiseen. Tämän vuoksi tutkimusta aiheesta ei ole olemassa valtavaa määrää. Tämän kirjallisuuskatsauksen avulla voidaan syventää näkemystä tästä aiheesta ja kehittää uusia jatkotutkimusaiheita. Ohjelmistojen ja laskentatehon jatkuva kehitys ja saatavuus tarjoaa myös jatkuvasti uusia mahdollisuuksia pelikehityksen hyödyntämiselle ohjelmoinnin opetuksessa, minkä vuoksi aihetta on tärkeää myös tutkia jatkossa.

3. Aiempi tutkimus

Tässä luvussa käydään läpi yksityiskohtaisesti kirjallisuuskatsauksen avulla löydettyjen tutkimusten löytämiä tuloksia ja havaintoja. Ensin käydään läpi motivaatioita sille, miksi ohjelmointia ja pelikehitystä halutaan opettaa eri koulutusasteilla. Tarkoituksena ei kuitenkaan ole jakaa tutkimusta koulutusasteiden mukaan, vaan tarkastella pelikehitystä yleisesti ohjelmoinnin opetuksessa. Samalla tarkastellaan syitä sille, miksi ohjelmoinnin opetusmenetelmiä pyritään kehittämään. Tämän jälkeen keskitytään aikaisemmissa tutkimuksissa käytettyihin opetusmenetelmiin, jotka hyödyntävät pelikehitystä ohjelmoinnin opettamisessa. Tämän osion tarkoituksena on luoda kattava käsitys tutkimuksissa käytetyistä erilaisista alustoista, oppimateriaaleista ja opetuksessa käytetyistä työkaluista ja menetelmistä. Lopuksi käydään läpi tutkimusten tuottamat tulokset. Viimeinen osio käsittelee tutkimuksissa ilmenneitä huomioita pelikehityksen vaikutuksesta opiskelijoiden motivaatioon ja osaamiseen ohjelmointia kohtaan. Tässä osiossa käydään myös läpi pelikehityksen kohtaamia haasteita. Haasteiden käsitteleminen on tärkeää, jotta opetusmateriaaleja ja -menetelmiä kehittäessä ymmärretään pelikehityksen vaatimukset ja rajoitteet ohjelmoinnin opetuksessa.

3.1 Motivaatio ohjelmoinnin ja pelikehityksen opettamiselle

Ohjelmoinnilla on merkittävä rooli monilla koulutusaloilla. Ymmärrys asioista, kuten erilaisista ohjelmointiparadigmoista, tietorakenteista, algoritmeista ja ohjelmistoarkkitehtuureista, on oleellista ohjelmistoalalla työskentelevien kannalta. Ohjelmointitaidoista hyötyvät myös monet muut teknillisten ja luonnontieteellisten alojen opiskelijat ja ammattilaiset. Lisäksi ohjelmoinnilla voidaan kehittää monia kognitiivisia taitoja, joista on usein hyötyä myös muissa konteksteissa. Ohjelmointi on kuitenkin aloittelijoille usein täysin uusi ympäristö. Ohjelmointi koetaan haastavana ja monilla on ongelmia pysyä motivoituneena. Tämän luvun tarkoituksena on tarkastella motivaatiota yleisesti ohjelmoinnin opettamiselle sekä ohjelmoinnin opetuksen kehittämiseksi pelikehityksen avulla.

3.1.1 Kognitiivisten taitojen kehittäminen

Akcaoglu (2014) mukaan ongelmanratkaisutaidot ovat yksi tärkeimpiä kognitiivisia taitoja nuorille heidän tulevaisuutensa kannalta. Kouluissa annetut tehtävät ovat kuitenkin usein tarkkaan strukturoituja ja rajattuja sekä keskittyvät enemmän symboliseen ajatteluun kuin suoraan vuorovaikutukseen esineiden ja objektien kanssa. Näistä syistä annetut tehtävät eivät usein kehitä oppilaiden taitoja tarpeeksi monimutkaisten tulevaisuuden ja koulun ulkopuolisten ongelmien kannalta. Pelikehityksen houkuttelevuuden ja sen mahdollistavien alustojen yleistymisen ansiosta pelikehitystä voitaisiin käyttää kehittämään oppilaiden ongelmanratkaisutaitoja vastaamaan enemmän niitä haasteita, joita he tulevaisuudessa kohtaavat. Myös Joel (2022) kertoo opiskelijoille annettujen tyypillisten tehtävien pakottavan heidät visualisoimaan abstrakteja käsitteitä ilman yhteyttä todellisiin asioihin. Haselberger ja kumppanit (2020) tuovat esiin myös ohjelmoinnin kognitiivisia hyötyjä. Vaikka nuorilla opiskelijoilla ei olisi tavoitteena ura tietojenkäsittelyn alalla, niin ohjelmointi kehittää hyödyllisiä taitoja, kuten ongelmien muodostamista ja abstraktiota.

3.1.2 Olio-ohjelmointi

Giordano ja Maiorana (2013) kertovat pelikehityksen soveltuvan olio-ohjelmoinnin opettamiseen, sillä yksinkertaistenkin pelien luomiseen vaaditaan useiden luokkien suunnittelua ja toteutusta. De Oliveira ja kumppanit (2017) havaitsivat tapaustutkimuksessaan opiskelijoiden kokevan olio-ohjelmoinnin opetuksen haasteiksi aiheen korkean käsitteellisyys ja vaikeuden muuntaa oikean elämän objekteja ohjelmallisiksi objekteiksi. Myös Shabalina ja kumppanit (2013) tuovat esiin ohjelmoinnin korkean abstraktiotason olevan usein haaste opiskelijoille. Phelps ja kumppanit (2009) väittävät pelikehityksen olevan tehokas keino eri ohjelmointiparadigmojen, kuten olio-ohjelmoinnin opettamiseen.

Ryoo ja kumppanit (2008) huomauttavat, että muiden kuin teknillisten alojen opiskelijat eivät välttämättä ole kiinnostuneita niinkään olio-ohjelmoinnin teoreettisesta puolesta kuin käytännön työskentelystä. Lisäksi Leutenegger ja Edgington (2007) kokevat ohjelmointiparadigmojen valinnan toissijaiseksi ongelmaksi opiskelijoiden motivaation ohella, kun kyseessä on aloittava opiskelija. Toisessa tutkimuksessaan Edgington ja Leutenegger (2008) käyttivät Rogue peliin perustuvaa projektia olio-ohjelmoinnin opettamiseen hyödyntäen UML-mallinnusta ja C++-ohjelmointikieltä. Beaubouef ja Mason (2005) kertovat havainneensa merkittävää vaihtelua mielipiteissä, miten ja milloin olio-ohjelmointia tulisi opettaa tietojenkäsittelyn alalla. Eri menettelytavoista huolimatta olio-ohjelmoinnin opettaminen koetaan usein haastavana opiskelujen alkuvaiheessa.

3.1.3 Korkea keskeyttämisprosentti ja matala läpipääsyprosentti

Beaubouef ja Mason (2005) tuovat esiin havaintojaan siitä, miksi opiskelijat päättävät pudottautua pois tietojenkäsittelyyn liittyvistä tutkinnoista. Esimerkiksi ennen tutkinnon aloittamista opiskelijoilla on usein heikko käsitys siitä, mitä tietojenkäsittelyn opiskelu todellisuudessa sisältää. Tähän liittyy myös opiskelijoiden heikko matemaattinen ja ongelmanratkaisutaitojen perusta. Opiskelun aikana syitä ovat puolestaan muun muassa saatavilla olevan harjoituksen vähyys ja siihen liittyvä opettajalta saatu palaute sekä opiskelijoiden heikot projektinhallintataidot.

Xu & Jin (2021) kertovat tutkimuksessaan 40 % opiskelijoista saavan joko välttävän tai hylätyn arvosanan tai keskeyttävän ohjelmoinnin perusteita opettavan kurssin omassa oppilaitoksessaan. Xu ja Jin (2021) kertovat myös havainneensa turhautumista opiskelijoiden keskuudessa ongelmanratkaisun suhteen. Tämä turhautuminen johtaa helposti myös motivaation heikkenemiseen. Vivianin ja kumppaneiden (2013) tutkimuksessa motivaation puute oli yksi pääsyyistä opiskelijoiden päätökseen keskeyttää opiskelu. Phelps ja kumppanit (2009) kertovat ohjelmoinnin opetteluun vaativan ensin monien asioiden, kuten syntaksin, debuggaamisen ja ohjelmistosuunnittelun opettelu. Näiden asioiden konkreettista vaikutusta voi olla vaikeaa nähdä erityisesti opintojen alussa. Tämän seurauksena opiskelijat turhautuvat helposti ja menettävät mielenkiintonsa alaa kohtaan.

Ohjelmoinnin ja kielten opiskelun välillä voidaan myös tehdä vertauksia. Vieraita kieliä on helpompi opetella, kun käytetyt materiaalit keskittyvät opiskelijalle valmiiksi tuttuihin tai jokapäiväisiin asioihin. Tekstipohjaiset harjoitukset voivat tuntua erityisesti eitekniillisten alojen opiskelijoille vaikeilta hahmottaa. Tästä näkökulmasta myös ohjelmointia voitaisiin opettaa tehokkaammin, kun ohjelmointiin liitettäisiin esimerkiksi pelien kautta multimediaa, kuten kuvia, ääniä ja videoita. Näiden asioiden

hyödyntäminen ohjelmoinnissa on kuitenkin usein aloittelijoille liian haastavaa, jolloin tarvitaan työkaluja, joilla tätä prosessia voidaan yksinkertaistaa (Valente ym., 2020).

3.1.4 Tietojenkäsittelyn houkuttelevuuden parantaminen

Motivaationa pelikehityksen hyödyntämiselle on myös tehdä tietojenkäsittelyn alasta houkuttelevampaa opiskelijoille (da Silva & da Silva Aranha, 2015). Teknologian nopeasti kehittyvässä maailmassa perinteiset opetusmallit eivät välttämättä kykene enää vastaamaan opiskelijoiden tarpeisiin. Pelikehityksellä voitaisiin mahdollisesti auttaa opiskelijoita oppimaan asioita uudella tavalla (de Oliveira ym., 2017). Pelikehityksen etuna on myös sen kyky tarjota opiskelijalle visuaalista palautetta tekemästään työstä. Pelit ovat tuotteita, joita opiskelija pystyy jakamaan ja joilla voidaan vahvistaa teoreettisia käsitteitä. Tämän lisäksi osajien tarve monille ohjelmointitaitoja ja tietoteknisiä taitoja vaativille työpaikoille lisää myös oppilaitosten tarvetta kehittää tehokkaita opetusmenetelmiä (Doerschuk ym., 2013). Pelikehityksen avulla voidaan myös korostaa luovuutta ja antaa opiskelijoille vapaampia projekteja tarkoin määriteltyjen töiden sijaan, kunhan pelit täyttävät odotetut osaamistavoitteet (Ryoo ym., 2008).

Phelps ja kumppanit (2009) tuovat esiin myös tietojenkäsittelyyn liittyvien näkemysten vaikutuksen kiinnostukseen alaa kohtaan. Alalla työskentely nähdään usein epäsosiaalisena ja tylsänä. Pelikehityksen kautta ohjelmointi voidaan nähdä helpommin osana kokonaisuutta esimerkiksi taiteen ja musiikin kanssa.

3.2 Pelikehityksen käyttö ja menetelmät ohjelmoinnin opetuksessa

Pelikehityksen käyttöä opetuksessa voidaan toteuttaa monella tapaa. Tutkimuksissa käytetyt ohjelmointikielet ja pelikehityksen mahdollistavat resurssit kuten kirjastot, ohjelmistokehykset ja pelimoottorit vaihtelevatkin paljon eri tutkimusten välillä. Tässä luvussa tarkastellaan näitä erilaisia käytännön toteutustapoja ja niiden hyödyntämiä työkaluja sekä teknologioita pelikehityksen käyttämiseksi ohjelmoinnin opetuksessa. Tutkimuksissa käytetyistä menetelmistä tuodaan esiin myös perusteluja ja hypoteeseja kyseisten resurssien ja työkalujen käytölle ja soveltavuudelle ohjelmoinnin opetuksessa. Alaluvut on jaoteltu opetukseen tarkoitettuihin kehitysympäristöihin, teollisuudessa käytettyihin kehitysympäristöihin ja muihin. Opetuksessa ja teollisuudessa käytetyt kehitysympäristöt viittaavat teknologioihin, jotka ovat julkisesti laajan yleisön saatavilla. Muihin teknologioihin puolestaan luetaan esimerkiksi tiettyä kurssia tai tutkimusta varten kehitetty alustat sekä muut ohjelmistokehykset tai kirjastot.

3.2.1 Opetusympäristöt

Topalli ja Cagiltay (2018) käyttivät tutkimuksessaan Scratch-ohjelmointiympäristöä¹ tietotekniikan opiskelijoille tarkoitetulla johdannollisella ohjelmointikurssilla. Käytännössä Scratchin hyödyntäminen tapahtui varaamalla kurssin laboratorioharjoituksista viisitoista minuuttia Scratchin tärkeimpien konseptien esittelyyn. Tämän jälkeen opiskelijoiden täytyi valmistella peliprojekti. Vastaavasti Araujo ja kumppanit (2018) käyttivät Scratchia ohjelmoinnin perusasioiden opettamiseen

¹ <https://scratch.mit.edu/>

15-vuotiaille, jonka jälkeen mukaan otettiin myös Python ja Jython Environment for Students -ympäristö, joka mahdollistaa median käsittelyn Python kielellä. Huansong ja kumppanit (2021) puolestaan hyödynsivät Scratchin yksinkertaisuutta ja visuaalista ohjelmointia luodakseen yksinkertaisia pelikehitykseen perustuvia tehtäviä peruskoululaisille. Garneli ja kumppanit (2015) päättivät valita myös Scratchin sen yksinkertaisuuden takia peruskoululaisille tarkoitettussa tutkimuksessa.

Bittencourt ja kumppanit (2015) käyttivät tutkimuksessaan avuksi vertaisopiskelijoita. Tutkimuksessa kehitettiin aloitteleville korkeakouluopiskelijoille tarkoitettu viisipäiväinen Scratchilla toteutettu työpaja. Työpajan ajankohta oli ennen varsinaisen opetuksen aloitusta ja sen tarkoituksena oli helpottaa uusia opiskelijoita alkuvaikeuksissa. Moreno-León ja kumppanit (2020) analysoivat 250 Scratch-projektia ja jakoivat ne kolmeen kategoriaan niiden monimutkaisuuden mukaan. Yksinkertaisimpina olivat animaatiot, taide ja musiikki. Seuraavaksi tulivat tarinat ja monimutkaisimpana pelit. Näin opiskelijat voisivat aloittaa ohjelmallisen ajattelun aloittamalla ensin yksinkertaisemmista projekteista ja sitten siirtymällä kohti pelejä.

Snap!² on Scratchin tyylinen ohjelmointiympäristö, joka perustuu Scratchin tapaan lohkopohjaiseen ohjelmointikieleen, jossa ohjelmointi tapahtuu siirtelemällä ja yhdistelemällä ohjelman logiikkaa vastaavia visuaalisia elementtejä. Bevcic ja Rugelj (2020) käyttivät Snap!-ohjelmointia kannustaakseen nuoria ja erityisesti tyttöjä ohjelmointiin. Holenko Dlab ja Hoic-Bozic (2021) päättivät myös valita Snap!-ohjelmointiympäristön 11–15-vuotiaille suunnatussa ohjelmassa. Tätä valintaa perusteltiin aloittelijaystävyydellä sekä Snap!-ohjelmoinnin kyvyllä hyödyntää myös edistyneempiä toimintoja kuin Scratchilla.

Doerschuk ja kumppanit (2013) pyrkivät opettamaan oliopohjaista ohjelmointia pelikehityksen avulla hyödyntämällä Java-ohjelmointikieleen perustuvaa Greenfoot-opetusympäristöä³. Opetusmateriaalit jaettiin moduuleihin, joissa jokaisen tarkoituksena on opettaa tiettyjä ohjelmoinnin peruskäsitteitä. Jokainen moduuli sisälsi yhden keskeneräisen pelin, johon opiskelijan tuli kehittää ratkaisu. Tämän tueksi moduuleissa oli oppimateriaaleja ja sisältöön perustuvia kyselyitä. Al-Bow ja kumppanit (2008) valitsivat myös nuorille tarkoitettussa kesäohjelmassa Greenfootin. Greenfootin valintaa perusteltiin muun muassa sen alustariippumattomuudella ja ilmaisuudella.

Akcaoglu (2014) tutki pelikehityksen käytön vaikutusta nuorten oppilaiden ongelmanratkaisukykyihin. Tutkimuksessa käytettiin Microsoft Kodu⁴ -pelikehitysympäristöä 10 päivää kestävässä kesäohjelmassa. Perusteeksi Kodun valitsemiseen kerrottiin mahdollisuus luoda kolmiulotteisia pelejä, mikä tekee peleistä visuaalisesti houkuttelevampia. Lisäksi Kodu tarjoaa yksinkertaistetun ohjelmointikielen sekä mahdollisuuden luoda simulaatioita, joista käyttäjä pystyy tarkkailemaan määrittämiään tapahtumia. Kenelläkään kesäohjelmaan osallistuneista oppilaasta ei ollut aikaisempaa kokemusta Kodusta tai pelikehityksestä. Myös Fowler ja Khosmood (2018) käyttivät Kodua nuorille tytöille suunnatussa ohjelmointi- ja pelikehitysohjelmassa.

² <https://snap.berkeley.edu/>

³ <https://www.greenfoot.org/door>

⁴ <https://www.kodugamelab.com/>

Kodun valintaa perusteltiin sen sarjakuvamaisen ulkonäön avulla sekä sen kyvyllä suorittaa pelejä, vaikka niiden toteutuksessa olisi virheitä.

Yksi tapa yhdistää useita opetusmetodeja, on hyödyntää vertaismentoreita pelikehityksen kanssa. Xu ja Jin (2021) toteuttivat pelikehityksen käyttämisen ohjelmoinnin perusteita käsittelevällä kurssilla keräämällä kolme yhden tai kaksi vuotta vanhempaa opiskelijaa vertaismentoreiksi. Nämä vertaismentorit suunnittelivat ja kehittivät pelin, jonka jälkeen he loivat ohjeistuksen tuleville opiskelijoille kyseisen pelin kehittämiseen itse. Kurssilla opiskelijat tutustuivat pelikehityksen aikana työpajojen ja kotitehtävien avulla ohjelmoinnin peruskäsitteisiin. Kotitehtävien tarkoituksena oli toimia siltana työpajoissa esiteltujen tärkeimpien käsitteiden välillä. Vertaismentoreiden tarkoituksena oli motivoida opiskelijoita osoittamalla, että heitä vain muutamia vuosia edellä olevat opiskelijat kykenivät toteuttamaan ja ohjaamaan näitä työpajoja. Tutkimuksessa hyödynnettiin Processing-ohjelmointiympäristöä⁵. Processing on Javaan perustuva ja se tarjoaa työkalut esimerkiksi graafisten elementtien käsittelyyn.

3.2.2 Teollisuudessa käytetyt kehitysympäristöt

Da Silva ja da Silva Aranha (2015) käyttivät tutkimuksessaan Construct2-pelimoottoria⁶. Tutkimus suoritettiin avoimena verkko-opetuksena, jonka kohteena olivat toisen asteen opiskelijat. Opetukseen osallistuminen ei vaatinut aikaisempaan osaamista ohjelmoinnista. Seaborn ja kumppanit (2012) käyttivät tutkimuksessaan GameMaker-pelimoottoria⁷. GameMakerin vahvuudeksi kerrottiin sen aloittelijoille ystävällinen käyttöliittymä, sekä mahdollisuus käyttää lohkopohjaisen ohjelmoinnin lisäksi myös tekstipohjaista skriptauskieltä monimutkaisemman pelilogiikan luomiseksi.

Giordano ja Maiorana (2013) keskittyivät työssään olio-ohjelmoinnin opettamiseen. 15 viikkoa kestäväällä opintojaksolla opiskelijat perehdyttiin ensin mahdollisimman nopeasti oliopohjaiseen suunnitteluun UML-kaavioiden avulla. Opintojakson lopussa oppilaille esiteltiin Microsoftin kehittämä pelikehitykseen tarkoitettu XNA-kehitysympäristö. Pelikehitys tapahtui toteuttamalla yksinkertainen peliprojekti mobiililaitteille. Wang ja Wu (2011) hyödynsivät myös XNA kehitysympäristöä ohjelmistoarkkitehtuureihin keskittyvällä kursilla. Pelikehitys integroitiin olemassa olevaan kurssiin käyttämällä XNA:n päälle kehitettyä ohjelmointirajapintaa ja keskittymällä kaksiulotteisiin peleihin. Näitä valintoja perusteltiin sillä, että opiskelijat haluttiin saada nopeasti työskentelemään kurssiprojektin parissa ja keskittymään ohjelmiston rakenteeseen muiden teknisten vaatimusten, kuten suorituskyvyn, sijasta.

Leutenegger ja Edgington (2007) käyttivät kolmivaiheisella kurssillaan ensin Flash-kehitysympäristöä ja ActionScriptiä, joiden avulla opiskelijat pystyivät nopeasti aloittamaan pelien kehityksen. Tämän jälkeen siirryttiin C++-ohjelmointikieleen ja OpenGL rajapintaan, jotka ovat laajasti käytettyjä ja joilla pystyttiin opettamaan tärkeitä käsitteitä, kuten muistin hallintaa. Kurkovsky (2009) puolestaan päätti hyödyntää mobiililaitteita pelikehityksessä käyttäen Java 2 Platform Micro Editionia (J2ME).

⁵ <https://processing.org/>

⁶ <https://www.construct.net/en>

⁷ <https://gamemaker.io/en>

Kurkovsky (2009) perusteli valintaansa mobiililaitteiden kasvavalla suosiolla sekä mobiilipelien yksinkertaisuudella.

Haselberger ja kumppanit (2020) tutkivat eri tekijöiden vaikutusta opiskelijoiden kokemukseen heidän omista ohjelmointitaitoistaan. Tutkimuksessa hyödynnettiin Unity-pelimoottoria⁸. Unity on laajalti peliteollisuudessa ammattilaisten käyttämä pelikehitystyökalu ja eroaa näin merkittävästi esimerkiksi opetukseen tarkoitettu Scratchista. Issae ja kumppanit (2021) valitsivat myös Unityn vertaillessaan pariohjelmoinnin ja yksilöohjelmoinnin vaikutuksia opiskelijoiden ohjelmointitaitoihin.

3.2.3 Muut

Pelikehityksen hyödyntämistä kursseilla mietittäessä täytyy valita, kuinka suuri rooli sillä on kurssin kokonaisuuden kannalta. Joel (2022) muodosti johdannollisen ohjelmointikurssin täysin pelikehityksen ympärille. Kurssilla opiskelijat muodostivat ryhmiä, jotka kehittivät ideoita pelille annettujen rajoitusten mukaan. Ryhmissä kehitetyistä malleista valittiin äänestämällä yksi kurssilla toteutettavaksi. Xu ja kumppanit (2022) puolestaan käyttivät tutkimuksessaan animaatioita ja pelien luomista kahtena ohjelmoinnin opetuksen pääkeinona. Xun ja kumppaneiden (2022) toteutus keskittyi web-kehitykseen ja opetuskieleksi valittiin tämän myötä JavaScript.

Duch ja Jaworski (2018) pyrkivät tutkimuksessaan tukemaan ohjelmoinnin opetusta hyödyntämällä Arduino-pohjaista opetukseen kehitettyä elektroniikka-alustaa. Arduinot varusteltiin erilaisilla syöttö- ja tulostuslaitteilla, joiden tarkoituksena oli antaa opiskelijoille erilaisia tapoja olla vuorovaikutuksessa laitteiston kanssa ohjelmoinnin avulla. Kahteen lukukauteen jaetulla vuoden mittaisella Ohjelmoinnin perusteet -kurssilla opiskelijat oppivat C-ohjelmointikielen perusteet ja työskentelevät Arduinoilla luodakseen erilaisia pelejä mutta myös muita projekteja.

Djelil ja Sanchez (2023) loivat tutkimustaan varten olio-ohjelmoinnin opetukseen tarkoitettun ympäristön nimeltä Progo. Progo tarjoaa opiskelijalle kolmiulotteisen näkymän, jossa hän voi rakentaa ja animoida robotteja ja rakennelmia komponenteista kirjoittamansa koodin avulla. Koodin ja kolmiulotteisen näkymän tarjoaman vuorovaikutuksen avulla Progon tarkoituksena on antaa opiskelijalle kokemusta luokkien ja objektien välisistä suhteista ja attribuuteista sekä metodien kutsumisesta. Tämän tarkoituksena on muuntaa olio-ohjelmoinnin käsitteet graafiseen muotoon. Myös de Oliveiran ja kumppaneiden (2017) kehittämän *Gaia Abstraction Game Object Oriented* -pelin tarkoituksena oli auttaa opiskelijoita olio-ohjelmoinnin opettamisessa.

Valente ja kumppanit (2020) esittelevät tutkimuksessaan kehittämäänsä ohjelmoinnin opetukseen tarkoitettua Pygame-moduuliin perustuvaa Python-kirjastoa. Tämän kirjaston tarkoituksena on yksinkertaistaa monien asioiden, kuten graafisten elementtien piirtämistä. Näin opiskelijat pystyvät mahdollisimman nopeasti luomaan heille tuttuja pelejä ja käyttöliittymiä ilman, että heidän tarvitsee ensin opetella suurta määrää näiden järjestelmien alla olevasta teoriasta. Myös Holsapple ja Bart (2022) toteuttivat samantyyllisen Pygameen perustuvan Python-kirjaston.

⁸ <https://unity.com/>

3.3 Pelikehityksen hyödyt ja haasteet ohjelmoinnin opetuksessa

Tässä luvussa tarkennutaan pelikehityksen vaikutuksiin ohjelmoinnin opetuksessa. Pelikehityksen merkitystä ohjelmoinnin opetuksessa tarkastellaan useista näkökulmista aikaisemmissa tutkimuksissa toistuvasti nousseiden keskeisten teemojen ja aiheiden kautta. Pelikehityksen vaikutuksesta pyritään tuomaan esiin sen merkittävimmät vahvuudet sekä haasteet.

3.3.1 Motivaatio ja opiskelun mielekkyys

Wilson ja Shrock (2001) toivat tutkimuksessaan esiin eri tekijöiden vaikutuksia opiskelijoiden menestykseen johdannollisilla ohjelmointikursseilla. Tutkimuksen tulosten perusteella opiskelijoiden mukavuustaso oli merkittävin positiivinen ennuste opiskelijoiden menestyksestä kurssilla. Mukavuustaso määriteltiin kyselyn avulla. Mukavuustasoon vaikuttivat muun muassa koettu tehtävien haastavuus, tehtävien aiheuttama stressi ja kysymysten ja vastausten toteutus kurssilla. Toiseksi merkittävin ennustaja oli opiskelijan matemaattinen tausta.

Scratchia käytettiin työkaluna useissa tutkimuksissa. Bittencourtin ja kumppaneiden (2015) toteuttamassa tutkimuksessa ryhmä aloittavia korkeakouluopiskelijoita osallistui viiden päivän mittaiseen työpajaan. Opiskelijat kuvailivat opetusmenetelmää muun muassa stimuloivaksi ja oppimista edistäväksi. Opiskelijat arvioivat Scratchin käyttäjäystävälliseksi ja mekaniikoiltaan hyväksi. Tutkimukseen osallistuneista 18 opiskelijasta 70 % kertoi työpajan auttaneen heitä oppimaan ohjelmointia ja 80 % arvioi työpajan auttaneen heitä ensimmäisen varsinaisen ohjelmointiprojektin toteuttamisessa. Jokainen opiskelija myös koki pelien auttaneen heidän oppimismotivaatioonsa. Shabalinan ja kumppaneiden (2013) tutkimuksessa pelikehityksen päähyödyiksi nostettiin myös esiin motivaation kasvu. Vastaavasti Holenko Dlab ja Hoic-Bozic (2021) kertovat oppilaiden kokeneen ohjelmoinnin opiskelun pelikehityksen avulla motivoivaksi ja tehokkaaksi keinoksi. Tutkimuksessa mukana olleiden opettajien ja ulkoisten asiantuntijoiden havaintojen mukaan ohjelmointitehtävien yhdistäminen oikean elämän konsepteihin teki näistä tehtävistä mielenkiintoisempia ja motivoivampia.

Ryoo ja kumppanit (2008) kuvailivat pelikehityksen saaneen yleisesti hyvin positiivista palautetta. Samoin Duchin ja Jaworskin (2018) Arduinoja hyödyntävällä kurssilla käytetyt opetusmenetelmät koettiin yleisesti hyvin positiivisina. Leuteneggerin ja Edgingtonin (2007) kyselyiden perusteella enemmistö opiskelijoista piti peleihin painottuvaa opiskelua hyvänä. Lisäksi pelikehityksen huomattiin houkuttelevan enemmän myös uusia opiskelijoita alalle. Al-bow ja kumppanit (2008) havaitsivat myös nuorten asenteen teknistieteellisiä aloja kohtaan parantuneen.

Construct2-pelimoottoria hyödyntävässä verkkokurssissa da Silva ja da Silva Aranha (2015) kertoivat pelikehityksen hyödyiksi mielenkiinnon lisäämistä tietojenkäsittelyä kohtaan sekä kognitiivisten ja sosiaalisten taitojen kehittymisen. Xu ja Jin (2021) järjestivät vertaismentoreiden toteuttaman pelikehitystyöpajan. Myös tässä tapauksessa havaittiin yleisesti kiinnostuksen ja opiskelun mielekkyyden kasvua ohjelmointia kohtaan.

Holliday (1995) tuo tutkimuksessaan esiin näkökulman, jonka mukaan pelikehitystä hyödyntävien kurssien tulisi pyrkiä tuottamaan pelejä, joiden toiminnallisuus vastaa opiskelijoiden itse pelaamia pelejä. Tutkimuksessa opiskelijat loivat vaihteittain etenevien ohjeiden avulla kaksi peliä. Holliday (1995) havaitsi, että jakamalla pelin

luomisen osiin, peliprojektit pystyttiin aloittamaan aikaisin kursseilla ja opiskelijoiden oli helpompi hallita niitä.

3.3.2 Menestys myöhemmissä opinnoissa

Topallin ja Cagiltayn (2018) mukaan, kun pelikehitystä hyödynnettiin Scratchin avulla johdannollisella ohjelmointikursseilla, havaittiin merkittävä positiivinen vaikutus opiskelijoiden menetyksessä myöhemmissä opinnoissa. Tuloksia verrattiin kontrolliryhmään, jonka opetuksessa ei hyödynnetty pelikehitystä. Shabalina ja kumppanit (2013) havaitsivat myös merkittävää kasvua pelikehitykseen osallistuneiden opiskelijoiden arvosanoissa verrattuna muihin. Leutenegger ja Edgington (2007) puolestaan havaitsivat kurssien keskeytyksien vähentyneen.

3.3.3 Kognitiiviset kyvyt

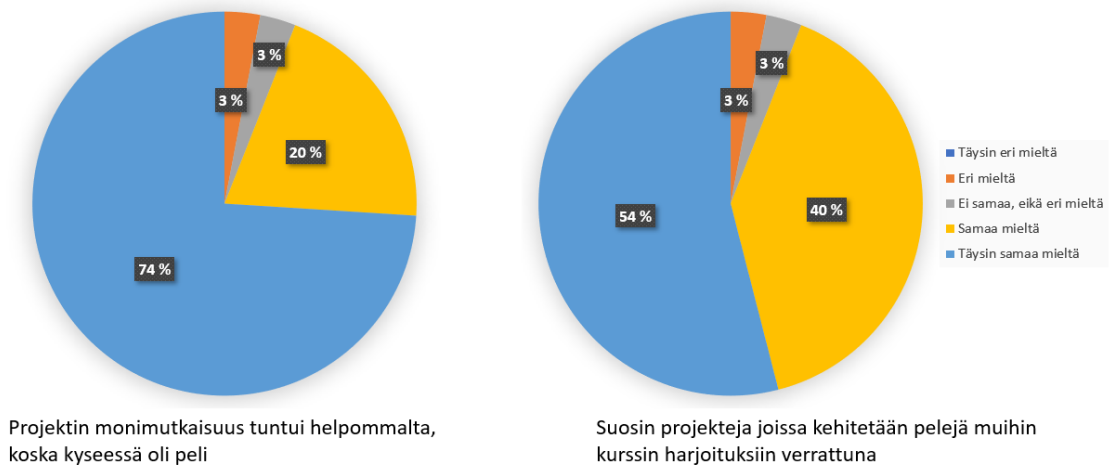
Fowler ja Khosmood (2018) havaitsivat pelikehityksen lisäävän nuorien tyttöjen ohjelmallisen ajattelun taitoja sekä parantavan heidän kokemustaan omasta kyvykkyydestä ohjelmoinnissa. Akcaoglun (2014) tutkimuksessa havaittiin merkittävää kasvua lasten järjestelmänalyysi-, suunnittelu- ja päätöksentekotaidoissa. Tutkimuksessa lapset suorittivat ongelmanratkaisutaitoja kartoittavan testin ennen pelikehitystä ja sen jälkeen. Vianetsintätaitoissa ei kuitenkaan havaittu merkittävää kasvua. Näihin tuloksiin saattoi kuitenkin vaikuttaa pelikehityksen aikana kohdattujen ongelmien ja testien esittämien ongelmien väliset eroavaisuudet. Vastaavia tuloksia tuli esiin myös de Jesuksen ja Silveiran (2022) tutkimuksessa. Pelikehitystä hyödyntävässä 12–15-vuotiaille suunnatussa työpajassa havaittiin osallistuneiden laskennallista ajattelua vaativien ongelmanratkaisukykyjen kehittyneen.

3.3.4 Ohjelmointitaitojen kehitys

Garnellin ja kumppaneiden (2015) tutkimuksessa opiskelijat jaettiin kolmeen ryhmään, joista yhdelle opetettiin ensin teoria ja sitten projekti. Kahdelle muulle ryhmälle projekti annettiin heti alusta, jossa toisen ryhmän tarkoituksena oli simuloida jotain ilmiötä, kun taas toisen ryhmän piti esittää sama ilmiö pelikehityksen avulla. Simulaatio- ja pelikehitysryhmät tekivät merkittävästi vähemmän virheitä, kun taas perinteisesti opetettu ryhmä käytti enemmän edistyneempiä konsepteja. Pelikehitysryhmällä oli kuitenkin huonoin projektien valmistumisprosentti. Lewis ja Massingill (2007) havaitsivat pelikehitysprojektien toimineen vähintään yhtä hyvin kuin aikaisemmin käytetyt tehtävänannot. Yksi merkittävimpiä hyötyjä pelikehityksestä oli kuitenkin opiskelijoiden kehitys dokumentaation käytössä.

Doerschukin ja kumppaneiden (2013) moduuleihin perustuvassa opetustavassa haluttiin selvittää moduulien suorittamisen vaikutusta käsitellyn aiheen osaamiseen ja mielenkiintoon. Pelikehitystä hyödyntäviä materiaaleja käytettiin kahdessa lukioikäisille tarkoitettussa ohjelmassa. Molemmissa tapauksissa opiskelijoiden vastaamien kyselyiden perusteella havaittiin merkittävää kasvua opettujen aiheiden osaamisessa ja mielenkiinnossa tietojenkäsittelyä kohtaan. Xu ja kumppanit (2022) raportoivat vastaavia tuloksia omassa tutkimuksessaan. Opetusmateriaalien arvioimiseksi toteutetussa jälkikyselyssä 86 % oli samaa tai vahvasti samaa mieltä animaatioiden opettamisen hyödyllisyydestä JavaScriptin ymmärtämisessä. Vastaavasti 75 % oppilaista oli samaa tai vahvasti samaa mieltä pelikehityksen hyödyllisyydestä. 77 % oppilaista myös koki

pelikehityksen mielenkiintoiseksi opetusmenetelmäksi. Martinsin ja kumppaneiden (2018) tutkimuksen mukaan ohjelmoinnin opiskelijat kokivat toteutetun peliprojektin monimutkaisuudet helpommaksi selvittää ja oppineensa enemmän, koska kurssilla käytettiin pelikehitystä (Kuva 1). Suurimmaksi haasteeksi kurssilla koettiin pelikehityksen vaatima aika.



Kuva 1. Pelien vaikutus harjoitusten haasteellisuuteen ja kiinnostavuuteen (Martins ym., 2018).

Xu ja Jin (2021) raportoivat kurssilla olleiden vertaismentoreiden kokeneen heidän ohjelmointi- ja johtamistaitojensa kehittyneen. Tutkimuksessa havaittiin myös lähiopetuksen hyödyttäneen enemmän naisia ja aliedustettuja vähemmistöjä. De Oliveiran ja kumppaneiden (2017) olio-ohjelmointiin keskittyvässä tutkimuksessa suurin osa opiskelijoista koki kehittyneensä paljon tai melko paljon abstraktion, objektien, luokkien ja suhteiden käsitteiden ymmärtämisessä. Vastaavasti Seaborn ja kumppanit (2012) havaitsivat oppilaiden ohjelmointitaitojen kehittyneen merkittävästi pelikehitystä hyödyntävän kurssin aikana.

Duchin ja Jaworskin (2018) mukaan Arduinoja hyödyntävällä kurssilla käytetyt opetusmenetelmät koettiin yleisesti hyvin positiivisina. Aloitteijoille suunnatussa kurssissa ensimmäisen vuoden opiskelijat kokivat hyötyneensä hieman enemmän laboratorioharjoituksista kuin toisen vuoden opiskelijat. Holenko Dlab ja Hoic-Bozic (2021) havaitsivat nuorten 11–15-vuotiaiden kokemuksen omista ohjelmointitaidoistaan kehittyneen merkittävästi pelikehitysaktiviteettien jälkeen. Hainey ja kumppanit (2020) havaitsivat Scratchilla toteutetun pelikehityksen olleen tehokas tapa auttaa 8–11-vuotiaita oppilaita kehittämään heidän ohjelmointitaitojaan. Hainey ja kumppanit (2020) kuitenkin huomauttavat, että tutkimuksen keskittyessä Scratchiin tutkimuksen tulokset eivät välttämättä heijastu pelikehityksen käyttöön ohjelmoinnin opetuksessa yleisesti.

Agrawal ja kumppanit (2012) tuovat esiin tutkimuksessaan useita esimerkkejä siitä, miten erilaisia tietorakenteita ja algoritmeja hyödynnetään pelikehityksessä. Näihin kuuluvat muun muassa erilaiset nopeat lajittelu- ja hakualgoritmit, tietorakenteet sekä polunetsintä- ja tiedonpakkausalgoritmit. Agrawal ja kumppanit (2012) perustelevatkin pelien käyttöä esimerkkeinä ja toteutettavina tehtävinä sillä, että erilaiset pelit sisältävät erilaisia vaatimuksia algoritmien valinnan ja toteutuksien suhteen, jotta peli voidaan toteuttaa tehokkaasti. Myös Kurkovsky (2009) käyttää vastaavia perusteluita. Pelien toteuttaminen kattaa laajan alueen ohjelmistokehitykseen liittyviä käsitteitä ja taitoja. Näitä alueita ovat muun muassa tietorakenteet ja algoritmit, ihmisen ja tietokoneen välinen vuorovaikutus, grafiikat ja tietokannat. Pelien avulla näitä käsitteitä voidaan tuoda opiskelijoiden tietoisuuteen jo varhaisessa vaiheessa opintoja.

3.3.5 Haasteet

Muuttajat koettiin yhdeksi vaikeimmista aiheista ohjelmoinnissa huolimatta niiden käytöstä pelikehityksen aikana (Bittencourt ym., 2015). Giordano ja Maiorana (2013) kertoivat yleisiksi opiskelijoiden haasteiksi olio-ohjelmoinnissa syntaksiset ja semanttiset virheet sekä suunnitteluvirheet sekvenssi- ja luokkakaavioissa. Al-bow ja kumppanit (2008) huomauttavat, että vaikka nuorten näkemys teknistieteellisistä aloista oli taiteeseen, pelikehitykseen ja ohjelmointiin keskittyvällä kesäleirillä kehittynyt positiivisesti, niin heidän ohjelmointitaitonsa eivät kehittyneet merkittävästi. Agrawal ja kumppanit (2012) kertovat opiskelijoiden suurimmaksi haasteeksi osoittautuneen pelien sisäisen logiikan ja graafisen käyttöliittymän yhdistämisen opettaessaan tietorakenteita ja algoritmeja.

Kehittäessään pelejä opetustarkoituksiin Arnez ja kumppanit (2014) havaitsivat haasteiksi opetuksen ja pelikehityksen välisen ajan tasapainottamisen. Lisäksi peliprojektien vaatimat suuret määrät digitaalista aineistoa, kuten ääniä ja animaatioita, koettiin haasteelliseksi. Da Silva ja da Silva Aranha (2015) sekä Shabalina ja kumppanit (2013) nostivat haasteiksi myös tiukan aikataulun sekä kommunikointiin liittyvät ongelmat. Samoja huomioita tekivät myös Haselberger ja kumppanit (2020). Unityä hyödyntävässä tutkimuksessa tutkijat nostivat esiin 14–16-vuotiaiden opiskelijoiden kritiikin Unityn monimutkaisuudesta sekä ajan puutteesta Unityyn tutustumiseen. Seaborn ja kumppanit (2012) tekivät samanlaisia havaintoja oppilaiden avoimen palautteen perusteella. Oppilaat kokivat aliarvioineensa pelien kehitykseen vaadittavan työmäärää.

Erilaisten ohjelmoinnin opettamiseen tarkoitettujen ympäristöjen käyttämisessä on myös haasteita, sillä simuloidut ohjelmointiympäristöt eivät vastaa tavallista ohjelmointikielien kanssa työskentelyä (Ryoo ym., 2008). Vastaavasti Haselberger ja kumppanit (2020) huomauttavat, että vaikka lohkopohjaiset kielet tarjoavat käyttäjäystävällisen ja helpon lähtöpisteen ohjelmointiin, niin ne eivät yleensä anna kehittäjälle saman tason mukauttamismahdollisuuksia ja pääsyä käytettyyn järjestelmään. Seaborn ja kumppanit (2012) eivät havainneet testeissään merkittävää kehitystä oppilaiden omista kokemuksista ohjelmointitaidoissa ja materiaalien mielekkyydessä.

4. Pohdinta

Tämän tutkimuksen tavoitteena oli tutkia, miten pelikehitystä voidaan hyödyntää ohjelmoinnin opetuksessa ja millaisia vaikutuksia sillä on opiskelijoiden osaamiseen ja motivaatioon. Aikaisemman tutkimuksen perusteella yleinen huomio oli, että pelikehityksen havaittiin olevan tehokas työkalu ohjelmoinnin opetuksessa, ongelmanratkaisutaitojen kehityksessä ja opiskelijoiden motivoimisessa sekä ohjelmoinnin opetuksen mielekkyydessä. Pelikehitystä olisikin näistä syistä syytä harkita, kun valmistellaan opetusmateriaaleja ohjelmointiin. Pelikehitys ei kuitenkaan ole ainoa toimiva opetusmenetelmä ja opetuksessa käytetyt projektit ja oppimateriaalit tulisi aina suunnitella niiden tarkoituksen mukaan. Pelikehityksen tehokkuutta ohjelmoinnin opetuksessa olisikin mielenkiintoista nähdä verrattuna muihin opetusmenetelmiin, kuten pelillistämiseen. Pelikehitystä ei myöskään yleensä käytetty ainoana opetusmenetelmänä vaan se yhdistettiin perinteiseen teorian opiskeluun oppimateriaalien ja luentojen avulla. Pelien tekeminen vaatii joka tapauksessa tietyn tason ymmärtämistä ohjelmoinnin peruskäsitteistä (Giordano & Maiorana, 2013).

Opetukseen käytettyjä erilaisia ohjelmistoja, ohjelmointikieliä ja kehitysalustoja löydettiin paljon. Pelikehitykseen käytettyjen ohjelmistojen, kuten pelimoottoreiden välillä oli myös paljon eroavaisuuksia niiden tarkoituksien ja toiminnallisuuksien suhteen. Käytetyt kehitysalustat vaihtelivat opetukseen tarkoitetuista alustoista, teollisuudessa yleisesti käytettyihin työkaluihin sekä mukautettuihin ohjelmistokehyksiin ja kirjastoihin. Näiden ohjelmistojen ja resurssien valintaa ohjelmoinnin opetukseen perusteltiin myös monin eri tavoin. Valittuja työkaluja ja menetelmiä perusteltiin muun muassa opiskelijoiden koulutustasolla ja aikaisemmalla kokemuksella ohjelmoinnista. Lisäksi vaikuttavana tekijänä oli usein tietyn kurssin tai työpajan tavoitteet. Nämä havainnot heijastavat yleisen yhteisymmärryksen ja menettelytapojen puutteesta ohjelmoinnin opetuksessa sekä ohjelmoinnin valtavasta määrästä erilaisia käyttötarkoituksia ja sovelluksia. Ohjelmointi käsittää hyvin suuren määrän asioita, jonka seurauksena sitä voidaan myös opettaa monella tapaa. Laaja määrä erilaisia menettelytapoja ja yhtenäisyyden puuttuminen ei kuitenkaan suoranaisesti ole negatiivinen asia. Kuten aikaisemmin mainittiin, eri työkalujen käyttöä voitiin perustella eri tavoin. Esimerkiksi Huansong ja kumppanit (2021) sekä Garneli ja kumppanit (2015) perustelivat Scratchin käyttöä peruskouluikäisille Scratchin yksinkertaisuudella, kun taas Wang ja Wu (2011) perustelivat XNA:n käyttöä yliopisto-opiskelijoille sen kypsyydellä sekä korkean tason ohjelmointirajapinnalla. Kaiken ohjelmoinnin perustana toimii kuitenkin hyvin usein samat periaatteet kuten muuttujat, funktiot, ehtorakenteet ja silmukat. Ohjelmoinnin opetuksessa oleellisinta on löytää keinoja, joilla näitä perusteita voidaan opettaa tavoilla, jotka motivoivat opiskelijoita ja kannustavat heitä oppimaan lisää sekä kehittämään ohjelmoinnin perusteita tukevia taitoja. Pelikehitys tarjoaa yhden vaihtoehdon auttaa aloittelevia, mutta myös kokeneita ohjelmoijia pysymään kiinnostuneina ohjelmoinnista haasteista huolimatta. Pelit tarjoavat ympäristön, jossa ohjelmoija pystyy selvemmin näkemään ja olemaan vuorovaikutuksessa toteuttamansa ohjelman kanssa, mikä vahvistaa myös teorian ymmärtämistä (Doerschuk ym., 2013).

Pelikehityksen yhtenä hyötynä ohjelmoinnin opetuksessa on sen soveltuvuus moneen käyttökontekstiin. Esimerkiksi Akcaoglu (2014) tutkimuksessa pelikehitystä käytettiin peruskouluikäisten kanssa, kun taas Xun ja Jinin (2021) tutkimuksessa mukana olivat korkeakouluopiskelijoita. Tämä kertoo pelikehityksen skaalautuvuudesta ohjelmoinnin opetuksessa. Pelikehitys sopii myös erilaisten ohjelmointiparadigmojen opetukseen. Olio-ohjelmointia käytetään usein pelikehityksessä pelien rakenteen vuoksi.

Giordano ja Maiorana (2013) mainitsevat tutkimuksessaan yksinkertaistenkin pelien luomisen vaativan useiden toistensa kanssa vuorovaikutuksessa olevien luokkien suunnittelua ja toteuttamista. De Oliveiran ja kumppaneiden (2017) tutkimuksessa opiskelijat kokivat olio-ohjelmoinnin perusteiden olevan hyvin käsitteellisiä, mikä tekee niiden oppimisesta haastavaa. Pelikehitys voisi tarjota tähän keinon auttaa olio-ohjelmoinnin opiskelijoita yhdistämään teoreettisia käsitteitä konkreettisiin asioihin. Pelikehitys vaatii usein monien erilaisten resurssien, kuten kuvien ja äänien, käsittelyä. Aloittelevalle ohjelmoijalle, jonka alustavana tavoitteena on usein oppia datatyypeistä, muuttujista, ehtorakenteista ja silmukoista, kuvien ja äänien käsittely ohjelmallisesti voi olla hieman tavoitteiden ja osaamisen ulkopuolella. Tämä on yksi syy, miksi valmiiden pelikehitysympäristöjen tai -kirjastojen käyttäminen voi olla kannattava vaihtoehto, kun kyse on aloittelevista ohjelmoijista. Toisaalta kokeneetkin ohjelmoijat harvoin rakentavat kaikkia tarvittavia kirjastoja ja työkaluja alusta alkaen. Tärkeää on siis miettiä, mitä tavoitteita opintojaksoon kuuluu. Pelikehityksen tulisi toimia opetuksen tukena, eikä ylimääräisenä opittavana materiaalina, kun tarkoituksena on oppia ohjelmoinnin perusteita. Yleinen havainto aikaisemmissa tutkimuksissa olikin opiskelijoiden kokema ajan puutteen vaikutus kehitysympäristöjen parissa (Arnez ym., 2014; da Silva & da Silva Aranha, 2015; Haselberger ym., 2020; Seaborn ym., 2012; Shabalina ym., 2013).

Käytetyistä työkaluista ja alustoista huolimatta tutkimusten tulokset olivat laajasti yhtenäisiä tulosten suhteen. Tutkimusten tulokset perustuivat kuitenkin pitkälti tutkimuksissa käytettyjen opetusmenetelmien tehokkuuteen opiskelijoiden antamien mielipidemittauksien mukaan. Pelikehitys koettiin yleisesti motivoivaksi ja ohjelmoinnin käsitteiden osaamista edistäväksi opetusmenetelmäksi. Kuitenkin vain yhdessä tutkimuksessa pelikehityksen tehokkuutta verrattiin samanaikaisesti kontrolliryhmään, joka suoritti perinteiseen ohjelmointikurssiin (Topalli & Cagiltay, 2018). Olisi tärkeää ja mielenkiintoista nähdä myös muita suoria vertauksia kahden eri opetusmenetelmän välillä.

Aikaisemmissa tutkimuksissa raportoitiin myös pelikehityksen haasteita ohjelmoinnin opetuksessa. Kaikki tutkimukset eivät olleet yhtenäisiä pelikehityksen positiivisten tulosten suhteen. Al-bow ja kumppanit (2008) eivät havainneet merkittävää kehitystä ohjelmointitaidoissa. Vastaavasti Seaborn ja kumppanit (2012) eivät havainneet oppilaiden kokemuksen heidän ohjelmointitaidoistaan kehittyneen merkittävästi. Giordanon ja Maioranan (2013) tutkimuksessa opiskelijoiden yleisimmiksi virheiksi raportoitiin esimerkiksi syntaksiset ja semanttiset virheet. Agrawal ja kumppanit (2012) puolestaan havaitsivat opiskelijoiden suurimmaksi haasteeksi pelilogiikan ja graafisten elementtien yhdistämisen. Näitä havaintoja voidaan heijastaa sen kanssa, että harvat tutkimukset tekivät suoria vertauksia muihin opetuksen menetelmiin.

Pelikehityksen vaatima aika ja resurssit olivat kuitenkin yleisin esiin noussut haaste ohjelmoinnin opetuksessa. Työpajat voivat olla hyvinkin lyhyitä ja ohjelmointikursseissa on usein paljon sisältöä, joka tulisi saada opetettua kurssin aikana. Tämä voi olla esteenä sille, kuinka paljon aikaa pelien tekemiseen voidaan realistisesti käyttää. Esimerkiksi Haselberger ja kumppanit (2020) nostavat esiin 14–16-vuotiaiden antaman kritiikin Unityn monimutkaisuudesta ja siihen tutustumiseen varatun ajan vähyydestä. Kun kyse on teollisuudessa ammattilaisten käyttämästä ohjelmistosta ja nuorista opiskelijoista, täytyy oppimistavoitteita miettiä tarkkaan ajallisten resurssien kanssa. Näillä kursseilla voi myös olla hyvinkin suuria määriä opiskelijoita. Pelien testaaminen ja arvioiminen voi viedä todella paljon aikaa ja arvioimisprosessia voi olla vaikea automatisoida. Nämä ovat asioita, joita kurssien rakennetta mietittäessä joudutaan ottamaan huomioon. Usein rajallisten resurssien ja valmiiksi kiireellisten aikataulujen säätämien rajoitteiden

puitteissa voi olla vaikeaa löytää sopivaa ratkaisua pelikehityksen integroimiseen oppimateriaaleihin.

Tärkeää on myös huomata, että jokainen tässä työssä käsitelty pelikehitykseen keskittyvä tutkimus tarkasteli pelikehityksen hyödyntämiseen ohjelmoinnin opetuksessa, kun opetusta ohjattiin jollain tavalla. Tämä voitiin toteuttaa esimerkiksi valmiiksi valitulla kehitysympäristöllä, opettajilla tai valmiilla oppimateriaaleilla. Oppilaitoksissa toteutetuilla kursseilla tai ohjelmointipajoissa tapahtuvassa opiskelussa yleensä kaikilla näillä tavoilla. Olisi mielenkiintoista nähdä myös, miten ohjelmointia täysin itseopiskelevat henkilöt kokevat pelikehityksen tehokkuuden oppimisessa ja miten se eroaa ohjatusta opiskelusta. Itseopiskelussa opiskelija joutuu itse valitsemaan esimerkiksi, mitä pelimootoria tai ohjelmointikieltä hän käyttää ja millaista projektia hän lähtee kehittämään. Tämä tuo ohjelmoinnin opiskeluun enemmän kuormitusta ja valintoja, joilla voi olla vaikutusta opiskelijan motivaatioon. Lisäksi monissa tutkimuksissa käytettiin esimerkiksi tiettyä kurssia varten kehitettyä ohjelmistokirjastoa, jonka tarkoituksena oli nimenomaan helpottaa opiskelijoita pääsemään nopeammin käytännön harjoituksiin (de Oliveira ym., 2017; Djelil & Sanchez, 2023; Holsapple & Bart, 2022; Valente ym., 2020). Itseopiskelussa pääsy vastaaviin heille mukautettuihin resursseihin voi olla hyvinkin vähäistä. Toisaalta vapaus valita voi myös olla joillekin juuri se, mikä motivoi heitä eniten. Usein kurssien vaatimukset asettavatkin rajauksia sille, mitä lopputulosta esimerkiksi peliprojektilta odotetaan ja miten se tulisi toteuttaa.

Pelikehityksen tarjoamien hyötyjen lisäksi täytyy muistaa, että uusien taitojen kehittäminen vaatii aina aikaa. Vaikka opetusmenetelmillä on ehdoton vaikutus opiskelijoiden motivaatioon ja eri opetusmenetelmien tehokkuutta voidaan vertailla toisiinsa, vaatii ohjelmointitaitojen kehittäminen joka tapauksessa sitoutumista ja vaivannäköä. Näihin asioihin vaikuttavat opiskelijoiden ennakkoluulot ja oletukset siitä, mitä ohjelmointi ja tietojenkäsittely on, ja millaisia taitoja ja perusteita nämä vaativat (Beaubouef & Mason, 2005). Yksilöiden välillä on myös eroavaisuuksia mieltymysten suhteen. Pelikehitys saattaa kuulostaa monille houkuttevalta tavalla oppia, mutta ei voida olettaa, että pelikehitys olisi jokaiselle yhtä tehokas ratkaisu. Pelikehitys ei myöskään ole itsessään yksittäinen asia, vaan sen toteutus täytyy mukauttaa tavoitteiden ja käyttökontekstin mukaan.

5. Johtopäätökset

Tämän työn tavoitteena oli vastata seuraaviin tutkimuskysymyksiin:

”Miten pelikehitystä voidaan hyödyntää ohjelmoinnin opetuksessa?” ja

”Millaisia vaikutuksia pelikehityksellä on opiskelijoiden osaamiseen ja motivaatioon ohjelmoinnissa?”

Tutkimus toteutettiin kirjallisuuskatsauksen avulla etsimällä ja analysoimalla aiheesta tehtyä aiempaa tieteellistä kirjallisuutta. Tutkimus aloitettiin oletuksella, että pelikehityksellä voidaan lisätä opiskelijoiden motivaatiota ja osaamista ohjelmoinnin oppimisessa.

Tapoja hyödyntää pelikehitystä ohjelmoinnin opetuksessa löydettiin useita. Miten pelikehitystä käytännössä hyödynnetään ohjelmoinnin opetuksessa, on vahvasti riippuvainen opetusmateriaaleista sekä valituista teknologioista. Tässä tutkimuksessa pelikehityksen mahdollistavat teknologiat lajiteltiin kolmeen kategoriaan: opetusympäristöt, teollisuudessa käytetyt kehitysympäristöt ja muut. Jokaisen kategorian kohdalla esiin nousi niin vahvuuksia kuin heikkouksiakin. Lisäksi oppilaitoksilla on usein valmiiksi olemassa olevat oppimistavoitteet, jolloin pelikehityksen integroiminen opetukseen tulee pystyä huomioimaan myös nämä tavoitteet. Aikaisemmissa tutkimuksissa käytettyjen pelikehitysalustojen valintaa perusteltiin usein niiden sopivuudella tietyille kohderyhmälle, esimerkiksi lapsille tai nuorille. Eri kategorioihin lajiteltuja alustoja käytettiin kuitenkin usein myös sekaisin tutkimusten kohderyhmän iästä ja kokemuksesta riippumatta.

Pelikehityksen vaikutuksia opiskelijoiden osaamiseen ja motivaatioon ohjelmoinnissa tarkasteltiin useista lähtökohdista. Tehdyn tutkimuksen perusteella pelikehitys havaittiin yleisesti tehokkaaksi tavaksi opettaa ohjelmointia. Pelikehityksen avulla pystyttiin myös usein lisäämään opiskelijoiden mielenkiintoa tietojenkäsittelyä kohtaan sekä kehittämään heidän ymmärrystensä ohjelmoinnista. Toinen yleinen havainto oli, että opiskelijat kokivat pelikehityksen mieluisana ja motivoivana opetusmenetelmänä. Pelikehityksellä havaittiin myös olevan merkittävä vaikutus opiskelijoiden kognitiivisten taitojen, kuten ohjelmoinnin kannalta oleellisten ongelmanratkaisukykyjen kehitykseen. Pelikehityksellä havaittiin kuitenkin myös useita haasteita ohjelmoinnin opetuksessa. Esiin nousivat muun muassa valitun kehitysympäristön tuottamat haasteet ja sopivuus kyseiseen opetuskontekstiin, ajanhallintaongelmat ja todellinen vaikutus opiskelijoiden ohjelmointitaitoihin verrattuna perinteisiin opetusmenetelmiin.

Tutkimusta aiheesta ei myöskään ole merkittävän paljon. Esimerkiksi verrattuna pelillistämiseen tai pelien avulla opettamiseen, tämän tutkimuksen aiheeseen liittyvää tieteellistä kirjallisuutta on vähän. Lisäksi tietojenkäsittelyn alalla muutos on nopeaa ja uusia teknologioita kehitetään jatkuvasti. Pelikehitykseen luodut työkalut ovat kehittyneet vuosien varrella paljon. Tämän vuoksi on tärkeää, että aiheen tutkimusta jatkettaisiin myös tulevaisuudessa. Tätä tutkielmaa varten kerätty tieteellinen kirjallisuus keskittyy hyvin pitkälti opiskelijoihin, jotka ovat lapsista ja nuorista korkeakouluopiskelijoihin. Vaikka korkeakouluopiskelijat sisältävät laajan ikäjoukon, useimmat ovat kuitenkin nuoria aikuisia. Tästä syystä jatkotutkimusta voitaisiin tehdä pelikehityksen vaikutuksista ohjelmoinnin opetuksessa oppilaitosten ulkopuolella, hyödyntäen esimerkiksi vapaita verkkomateriaaleja. Myös eroavaisuuksia ikäjoukkojen välillä pelikehitystä hyödyntäessä ohjelmoinnin opetuksessa voitaisiin tarkastella

tarkemmin. Lisäksi harvat tutkimukset käyttivät kontrolliryhmiä tehdessään arviointia käytettyjen opetusmenetelmien tehokkuudesta.

Lähteet

- Agrawal, R. K., Kurmas, Z., Gudivada, V. N., El-Bathy, N., & Seay, C. (2012). Motivating students to learn programming using game assignments. *ASEE Annual Conference and Exposition, Conference Proceedings*. <https://doi.org/10.18260/1-2-21707>
- Akcaoglu, M. (2014). Learning problem-solving through making games at the game design and learning summer program. *Educational Technology Research and Development*, 62(5), 583–600. <https://doi.org/10.1007/S11423-014-9347-4/TABLES/1>
- Al-Bow, M., Austin, D., Edgington, J., Fajardo, R., Fishburn, J., Lara, C., Leutenegger, S., & Meyer, S. (2008). Using Greenfoot and games to teach rising 9th and 10th grade novice programmers. *Proceedings of Sandbox 2008: An ACM SIGGRAPH Videogame Symposium, Sandbox '08*, 55–60. <https://doi.org/10.1145/1401843.1401853>
- Araujo, L. G. J., Bittencourt, R. A., & Santos, D. M. B. (2018). Contextualized spiral learning of computer programming in brazilian vocational secondary education. *Proceedings - Frontiers in Education Conference, FIE, 2018-October*. <https://doi.org/10.1109/FIE.2018.8658456>
- Arnez, F., Pace, J., & Sung, K. (2014). Learning while building games for teaching. *Computer*, 47(4), 88–91. <https://doi.org/10.1109/MC.2014.91>
- Beaubouef, T., & Mason, J. (2005). Why the high attrition rate for computer science students. *ACM SIGCSE Bulletin*, 37(2), 103–106. <https://doi.org/10.1145/1083431.1083474>
- Bennedsen, J., & Caspersen, M. E. (2007). Failure rates in introductory programming. *ACM SIGCSE Bulletin*, 39(2), 32–36. <https://doi.org/10.1145/1272848.1272879>
- Bevcic, M., & Rugelj, J. (2020). Game design based learning of programming for girls. *2020 43rd International Convention on Information, Communication and Electronic Technology, MIPRO 2020 - Proceedings*, 576–580. <https://doi.org/10.23919/MIPRO48935.2020.9245175>
- Bittencourt, R. A., Dos Santos, D. M. B., Rodrigues, C. A., Batista, W. P., & Chalegre, H. S. (2015). Learning programming with peer support, games, challenges and scratch. *Proceedings - Frontiers in Education Conference, FIE, 2015*. <https://doi.org/10.1109/FIE.2015.7344222>
- da Silva, T. R., & da Silva Aranha, E. H. (2015). Online game-based programming learning for high school students - A case study. *Proceedings - Frontiers in Education Conference, FIE, 2015*. <https://doi.org/10.1109/FIE.2015.7344131>
- de Jesus, Â. M., & Silveira, I. F. (2022). A Collaborative Learning Framework for Computational Thinking Development through Game Programming. *Informatics in Education*, 21(2), 253–281. <https://doi.org/10.15388/INFEDU.2022.14>

- de Oliveira, E. D., Camargo, M. C., Barbosa, C. R. S. C., Brancher, J. D., de Oliveira Barros, V. T., de Campos, V. V. S., & de Barros, R. M. (2017). The power of the game as a mediator tool paradigm of object oriented teaching-learning process. *Proceedings - Frontiers in Education Conference, FIE, 2017-October*, 1–8. <https://doi.org/10.1109/FIE.2017.8286341>
- Djelil, F., & Sanchez, E. (2023). Game design and didactic transposition of knowledge. The case of progo, a game dedicated to learning object-oriented programming. *Education and Information Technologies*, 28(1), 283–302. <https://doi.org/10.1007/S10639-022-11158-6>
- Doerschuk, P., Juarez, V., Liu, J., Vincent, D., Doss, K., & Mann, J. (2013). Introducing programming concepts through video game creation. *Proceedings - Frontiers in Education Conference, FIE*, 523–529. <https://doi.org/10.1109/FIE.2013.6684879>
- Duch, P., & Jaworski, T. (2018). Enriching computer science programming classes with arduino game development. *Proceedings - 2018 11th International Conference on Human System Interaction, HSI 2018*, 148–154. <https://doi.org/10.1109/HSI.2018.8430994>
- Edgington, J., & Leutenegger, S. (2008). Using the ancient game of rogue in CS1. *Journal of Computing Sciences in Colleges*. <https://doi.org/10.5555/1409763.1409803>
- Fowler, A., & Khosmood, F. (2018). The Potential of Young Learners Making Games: An Exploratory Study. *2018 IEEE Games, Entertainment, Media Conference, GEM 2018*, 189–195. <https://doi.org/10.1109/GEM.2018.8516486>
- Garneli, V., Giannakos, M. N., Chorianopoulos, K., & Jaccheri, L. (2015). Serious Game Development as a Creative Learning Experience: Lessons Learnt. *Proceedings - 4th International Workshop on Games and Software Engineering, GAS 2015*, 36–42. <https://doi.org/10.1109/GAS.2015.14>
- Giordano, D., & Maiorana, F. (2013). Object Oriented Design through game development in XNA. *Proceedings of the 3rd Interdisciplinary Engineering Design Education Conference, IEDEC 2013*, 51–55. <https://doi.org/10.1109/IEDEC.2013.6526760>
- Hainey, T., Baxter, G., & Ford, A. (2020). An evaluation of the introduction of games-based construction learning in upper primary education using a developed game codification scheme for scratch. *Journal of Applied Research in Higher Education*, 12(3), 377–402. <https://doi.org/10.1108/JARHE-02-2018-0031>
- Haselberger, D., Motschnig, R., Comber, O., Mayer, H., & Horbe, M. (2020). Experiential Factors Supporting Pupils' Perceived Competence in Coding - An Evaluative Qualitative Content Analysis. *Proceedings - Frontiers in Education Conference, FIE, 2020-October*. <https://doi.org/10.1109/FIE44824.2020.9274217>
- Holenko Dlab, M., & Hoic-Bozic, N. (2021). Effectiveness of game development-based learning for acquiring programming skills in lower secondary education in Croatia. *Education and Information Technologies*, 26(4), 4433–4456. <https://doi.org/10.1007/S10639-021-10471-W>
- Holliday, M. A. (1995). Incremental game development in an introductory programming course. *Proceedings of the Annual Southeast Conference*, 170–175. <https://doi.org/10.1145/1122018.1122049>

- Holsapple, K., & Bart, A. C. (2022). Designing Designer: The Evidence-Oriented Design Process of a Pedagogical Interactive Graphics Python Library. *SIGCSE 2022 - Proceedings of the 53rd ACM Technical Symposium on Computer Science Education, 1*, 85–91. <https://doi.org/10.1145/3478431.3499363>
- Huansong, Y., Jia'En, W., & Mengting, S. (2021). The Practice and Exploration of Scratch Programming Instruction in Elementary School Based on Game Design. *2021 IEEE Conference on Telecommunications, Optics and Computer Science, TOCS 2021*, 186–190. <https://doi.org/10.1109/TOCS53301.2021.9688834>
- Issaee, A., Motschnig, R., & Comber, O. (2021). Pair- versus solo-programming of mini-games as a setting for learning to program: An Action Research approach. *Proceedings - Frontiers in Education Conference, FIE, 2021-October*. <https://doi.org/10.1109/FIE49875.2021.9637178>
- Joel, W. (2022). A Game-Development Paradigm for Building Programming Intuition. *Proceedings - SIGGRAPH 2022 Educator's Forum*. <https://doi.org/10.1145/3532724.3535601>
- Kurkovsky, S. (2009). Engaging students through mobile game development. *SIGCSE'09 - Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, 44–48. <https://doi.org/10.1145/1508865.1508881>
- Leutenegger, S., & Edgington, J. (2007). A games first approach to teaching introductory programming. *SIGCSE 2007: 38th SIGCSE Technical Symposium on Computer Science Education*, 115–118. <https://doi.org/10.1145/1227310.1227352>
- Lewis, M. C., & Massingill, B. (2007). Graphical game development in CS2: A flexible infrastructure for a semester long project. *Proceedings of the Thirty-Seventh SIGCSE Technical Symposium on Computer Science Education*, 505–509. <https://doi.org/10.1145/1121341.1121499>
- Martins, V. F., de Almeida Souza Concilio, I., & de Paiva Guimarães, M. (2018). Problem based learning associated to the development of games for programming teaching. *Computer Applications in Engineering Education*, 26(5), 1577–1589. <https://doi.org/10.1002/CAE.21968>
- Moreno-León, J., Robles, G., & Román-González, M. (2020). Towards Data-Driven Learning Paths to Develop Computational Thinking with Scratch. *IEEE Transactions on Emerging Topics in Computing*, 8(1), 193–205. <https://doi.org/10.1109/TETC.2017.2734818>
- Phelps, A. M., Egert, C. A., & Bayliss, J. D. (2009). Media impact: Games in the classroom: Using games as a motivator for studying computing: Part 1. *IEEE Multimedia*, 16(2), 4–8. <https://doi.org/10.1109/MMUL.2009.40>
- Ryoo, J., Fonseca, F., & Janzen, D. S. (2008). Teaching Object-Oriented software engineering through Problem-Based Learning in the context of game design. *Software Engineering Education Conference, Proceedings*, 137–144. <https://doi.org/10.1109/CSEET.2008.26>
- Seaborn, K., Seif El-Nasr, M., Milam, D., & Yung, D. (2012). Programming, PWned: Using digital game development to enhance learners' competency and self-efficacy in a high school computing science course. *SIGCSE'12 - Proceedings of the 43rd*

ACM Technical Symposium on Computer Science Education, 93–98.
<https://doi.org/10.1145/2157136.2157169>

- Shabalina, O., Sadovnikova, N., & Kravets, A. (2013). Methodology of teaching software engineering: Game-based learning cycle. *Proceedings - 2013 IEEE 3rd Eastern European Regional Conference on the Engineering of Computer Based Systems, ECBS-EERC 2013*, 113–119. <https://doi.org/10.1109/ECBS-EERC.2013.22>
- Topalli, D., & Cagiltay, N. E. (2018). Improving programming skills in engineering education through problem-based game projects with Scratch. *Computers and Education*, 120, 64–74. <https://doi.org/10.1016/J.COMPEDU.2018.01.011>
- Valente, A., Marchetti, E., & Wang, J. (2020). Design of an educational multimedia library to teach Python to non-technical university students. *Proceedings - 2020 9th International Congress on Advanced Applied Informatics, IIAI-AAI 2020*, 169–175. <https://doi.org/10.1109/IIAI-AAI50415.2020.00041>
- Vivian, R., Falkner, K., & Falkner, N. (2013). Computer science students' causal attributions for successful and unsuccessful outcomes in programming assignments. *ACM International Conference Proceeding Series*, 125–134. <https://doi.org/10.1145/2526968.2526982>
- Wang, A. I., & Wu, B. (2011). Using game development to teach software architecture. *International Journal of Computer Games Technology*. <https://doi.org/10.1155/2011/920873>
- Wilson, B. C., & Shrock, S. (2001). *Contributing to success in an introductory computer science course*. 184–188. <https://doi.org/10.1145/364447.364581>
- Xu, S., Lai, S., & Pollacia, L. (2022). Integrating Animation and Game-making in Teaching JavaScript. *2022 IEEE Integrated STEM Education Conference, ISEC 2022*, 318–323. <https://doi.org/10.1109/ISEC54952.2022.10025078>
- Xu, X., & Jin, W. (2021). Game development workshops designed and delivered by peer mentors to increase student curiosity and interest in an introductory programming course. *Proceedings of the 2021 ACMSE Conference - ACMSE 2021: The Annual ACM Southeast Conference*, 87–92. <https://doi.org/10.1145/3409334.3452046>