Opinnäytetyö (AMK)

Tieto- ja Viestintätekniikka

2019

Juha Aarnio

OPETUSPELIN SUUNNITTELU JA TEKNINEN TOTEUTUS

– PELIMO -projekti

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja Viestintätekniikka

Ohjaaja: Yliopettaja Mika Luimula, dos.

16.06.2019 | 33 sivua, 0 liitesivua

Juha Aarnio

OPETUSPELIN SUUNNITTELU JA TEKNINEN TOTEUTUS

- PELIMO -projekti

Työn toimeksiantona oli opetuspelin toteutustavan valinta sekä demon kehittäminen valitulla teknologialla. Toimeksiantajana työssä oli projektitoimisto The Firma. Työ aloitettiin tarkastelemalla opetuspelien perusteoriaa jotta saatiin parempi ymmärrys niistä seikoista ja vaatimuksista, jotka opetuspelin tulee täyttää. Teoriaa ymmärtämällä pystyttiin varsinaisessa teknologiavertailussa varmistamaan, että valittu toteutustapa soveltuu toimeksiantajalta saadun projektin toteuttamiseen.

Teoriaosuus toteutettiin tutustumalla opetuspelejä ja niiden kehitystä käsitteleviin kirjallisiin lähteisiin. Näiden pohjalta saadun tiedon pohjalta voitiin aloittaa itse toteutustavan valinta- ja vertailuprosessi. Vertailuvaiheessa oli otettava huomioon lisäksi toimeksiantajan vaatimus käännösten toteuttamiseksi suomeksi ja ruotsiksi. Teoriaosuudessa tutustuttiin myös interaktiivisen tarinankerronnan teoriaan, sillä toteutettu peli oli vahvasti tarinapohjainen. Varsinainen vertailu suoritettiin eri pelimoottoreiden sekä tarinankerronnan työkaluihin tutustumalla ja ominaisuuksien vertailulla. Tämän vertailun jälkeen päädyttiin toteuttamaan prototyyppiä kahdella eri Unity-lisäosalla, RPGTalkilla ja Funguksella. Prototyyppivaiheen jälkeen valittiin lopulliseksi toteutustavaksi Fungus, jolla toteutettiin myös pelin käännökset.

Varsinaisen demon kehityksessä pelin tarinoiden käsikirjoitukset saatiin valmiina toimeksiantajalta, joista tehtiin Funguksen avulla pelattava demoversio toimivalla pisteytysmekaniikalla sekä vaihtoehtoisilla loppuratkaisuilla. Lopullinen peli on osa PELIMO -hanketta, jossa tavoitteena on kehittää peli 10–12 -vuotiaiden monikulttuurisuuskasvatuksen käyttöön. Lopputuloksena toimeksiantajalle toimitettiin Funguksella toteutettu demo sekä suositeltiin lopullisen version ja käännösten toteuttamista Funguksella. Työn loppupuolella teetettiin haastattelututkimus paikallisiin yrityksiin, jolla kartoitettiin valitun ratkaisun hyödyllisyyttä toimeksiantajalle.

ASIASANAT:

interaktiivinen tarinankerronta, opetuspeli, pelisuunnittelu

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and Communication Technology

supervisor: Principal Lecturer Mika Luimula, Adj.Prof

2019 | 33 pages, 0 appendices

Juha Aarnio

DESIGN AND TECHNICAL IMPLEMENTATION OF EDUCATIONAL GAME

- PELIMO -project

Goal of the thesis was to come up with an implementation method and practical demo for an educational game. The employer was The Firma, a learning environment located at Turku University of Applied Sciences. The project started with getting familiar with basic theory of educational games to get better understanding in what educational games are and which factors affect the effectiveness of educational game. Understanding the underlying theory made it possible to analyse how well the chosen method of implementation is suited for development of educational game.

Theoretical part of the project was conducted by reading literature on educational games and their design. Based on this knowledge the process for comparing and choosing the implementation method began. One requirement from employer's side was the inclusion of translations in Finnish and Swedish, which was considered during the comparison process. Theory of interactive storytelling was also covered during this process, as the planned game was heavy on storytelling. The comparison process started with comparing features of different game engines and storytelling tools. After this process, storytelling tools RPGTalk and Fungus were chosen as the best candidates for implementing the game. After prototyping with both tools, fungus was ultimately chosen as the tool for implementing the demo and final game.

Developing the demo began with creating playable implementations with Fungus from pre-existing story scripts delivered by the employer. The final game is part of the PELIMO -project, which goal is to create a game to be used in education of multiculturalism for 10–12 year old children as a part of social studies. At the end of the thesis, a set of interviews were conducted with local game companies to get an understanding on the usefulness of the selected implementation method.

KEYWORDS:

Interactive storytelling, educational game, game design

SISÄLTÖ

1 JOHDANTO

Opetuspelit ovat yhä kasvava osa opetusta ja niitä hyödynnetään moninaisesti erilaisten aiheiden opiskelussa ja havainnollistamisessa. Koska opetukselliset pelit ovat yhä kasvava ohjelmistokehityksen ala (Valentine 2018), on ajankohtaista tarkastella miten teknologisilla ratkaisuilla voidaan tukea pelien hyödyllisyyttä opetuksessa. Tämä opinnäytetyö tarkastelee opetuspelin teknistä toteutusta sekä käännösmekaniikan toteutusta. Opinnäytetyö ei ota kantaa varsinaiseen pedagogiseen sisältöön tai sen arviointiin, vaan pääpaino on yksinomaan sisällön pelillistämisessä ja esittämisessä videopelin muodossa. Työn toimeksiantaja oli The Firma, joka on Turun Ammattikorkeakoulun tiloissa toimiva projektitoimisto. Toimeksiantona oli valita toteutustapa PELIMO -peliin, jonka kohderyhmä ovat 10–12 -vuotiaat lapset. Peliä on tarkoitus käyttää osana lasten monikulttuurisuuskasvatusta osana yhteiskuntaopin tunteja. Toimeksiantoon kuului lisäksi demon toteuttaminen valitulla teknologialla sekä käännösten toteuttaminen suomeksi ja ruotsiksi. PELIMO on Opetushallituksen rahoituksella Turussa toteuttava hanke, jonka tarkoituksena on tuottaa monikulttuurisuuskasvatuksessa käytettävä peli.

Tavoitteena oli valita toteutustapa sekä tarinankerronnan että käännösten toteuttamiseksi. Toteutettava peli on vahvasti tarinapohjainen, joten painopiste teknologian valinnassa oli tarinakerronnan ja interaktiivisuuden toteuttamisessa. Tarinoiden rakenne oli haarautuva joten toteutustavan oli oltava sellainen, että se mahdollisti tarinarakenteen selkeän hallinnan ja rakentamisen jollain työkalulla tai käyttöliittymällä. Lisäksi oli tärkeää, että ratkaisulla pystyttiin toteuttamaan haluttu pisteytysmekaniikka ja toteuttamaan vaihtoehtoisia loppuratkaisuja sisältäviä tarinoita. Käännösmekaniikan osalta tavoiteltiin ratkaisua, jossa varsinainen tekstisisältö sijaitsisi erillisessä tekstitiedostossa jotta vältyttäisiin kovakoodaamisen haitoilta, joita ovat esimerkiksi muokkaamisen ja testausprosessin vaikeutuminen. Lisäksi tekstitiedostopohjaista ratkaisua olisi mahdollista muokata editorin ulkopuolella.

Työn alkuvaiheessa tutustuttiin opetuspelien teoriaan sekä sivuttiin interaktiivisen tarinankerronnan teoriaa. Teoriaosuudessa tutustuttiin pintapuolisesti myös opetuspeliteollisuuden nykytilaan ja tulevaisuuden trendeihin. Teoriaosuuden aikana saatua tietoa sovellettiin itse toteutustavan valinnassa. Itse valinta suoritettiin käytännössä vertailemalla ero työkalujen ominaisuuksia ja käytettävyyttä. Vertailun päätteeksi varteenotettavat toteutusvaihtoehdot olivat Unityn RPGTalk- sekä Fungus -lisäosat.

Prototyyppivaiheen kokemuksien perusteella lopullinen demo toteutettiin Funguksella. Demon oli tarkoitus toimia pohjana lopulliselle peliversiolle.

Perusteorian lähteiksi kelpuutettiin runsaasti eri julkaisijoiden teoksia. Suurin osa lähteistä valittiin Elsevierin kaltaisista tietokannoista julkaisijoilta, jotka tiedettiin luotettaviksi. Osa lähteistä on Internetissä julkaistuja artikkeleita eri teknologiasivustoilta. Näissä tapauksissa tutustuttiin kirjoittajan aikaisempaan tuotantoon jotta saatiin kuva kirjoittajan asiantuntemuksesta ja luotettavuudesta. Osa teoriasta perustuu melko vanhoihin lähteisiin. Näissä tapauksissa lähteitä on käytetty perusteoriana sellaisissa asioissa, joissa voidaan ajatella tiedon olevan niin perustavanlaatuista ettei sitä voida pitää vanhentuneena. Esimerkiksi Chris Crawfordin havaintoja interaktiivisesta tarinankerronnasta voidaan pitää alan perusteoriana, jonka keskeiset asiat tarinamaailmasta sekä interaktiivisen tarinankerronnan peruselementeistä ovat vielä nykyäänkin ajankohtaisia. Tarkasteltaessa teollisuuden ja teknologian nykytilaa lähteiden julkaisuajankohdalle annettiin suurempi painoarvo, sillä teknologiaa käsittelevät tietolähteet vanhenevat nopeasti teknologian nopean kehityksen vuoksi. Tavoitteena oli tehdä kohtuullisen kattava selvitys PELIMO -projektin teknisestä toteutuksesta ja toteuttaa demoprojekti valitulla teknologialla samalla pitäen mielessä kaikki opetuspelien teorian edellyttämät seikat.

Tekstin kolme ensimmäistä lukua sisältävät työn perustana käytetyn teorian. Samaan teoriaosuuteen sisältyvä neljäs luku pitää sisällään teoriaa opetuspelien pelimekaniikan toteutuksesta sekä eri teknologioiden vertailua. Viides luku sisältää varsinaisen projektin esittelyn ja työskentelyn vaiheet sekä perustelut valitun teknologian käyttämiselle. Loppuluku sisältää yhteenvedon työn lopputuloksista ja suorittamisesta, sekä arvion oman työpanoksen merkittävyydestä ja toimeksiantajalle tuotetusta lisäarvosta. Loppuluvussa esitellään myös teoriaosuuden pohjalta tehtyjä johtopäätöksiä.

2 OPETUSPELIEN SUUNNITTELU

Opetuspelien suunnitteluun pätevät hyvin pitkälti samat asiat kuin sovellusten suunnitteluun ylipäätään. Opetukseen ja harjoitteluun tarkoitetut pelit edellyttävät kuitenkin tietyiltä osa-alueilta tarkempaa testausta ja suunnittelua kuin viihdepelit. Kuten Nand ym.(2013, 4) toteaa, opetuspeleissä lapset arvostavat erityisesti haasteita, grafiikkaa sekä palautetta. Grafiikan osalta erityisen tärkeäksi Nandin tutkimuksessa nousevat värikkäät kuvat sekä tosielämään pohjautuvat hahmot. Suunnittelussa on otettava huomioon myös se, pohjautuuko pelaajan halu pelata peliä ulkoiseen tai sisäiseen motivaatioon. Ulkoinen motivaatio kumpuaa pelaajan ulkoisista tarpeista ja haluista, esimerkiksi halusta voittaa palkintoja tai saavuttaa rahallista hyötyä. Sisäisen motivaation ajama pelaaja taas pelaa peliä, koska pelin tavoitteet ja sisältö koetaan itsessään arvokkaiksi. (Kendra.) Sisäinen motivaatio on erittäin hyödyllinen motivaation lähde, sillä sisäisen motivaation kannustama pelaaja on todennäköisemmin sitoutunut pein tavoitteisiin ja ottaa innokkaammin osaa pelin haasteisiin (Wee, Choong 2019). Sisäiseen motivaation vaikuttavat pääsääntöisesti kolme tarvetta, jotka ovat itsenäisyys, kyvykkyys sekä samaistuminen (Shodhan 2017). Itsenäisyys tarkoittaa pelaajan kykyä tehdä omia päätöksiä pelissä. Mitä enemmän pelaaja kokee olevansa vapaa tekemään päätöksiä, sen tärkeämmältä valinnat tuntuvat. Kyvykkyys taas ilmenee pelaajan tarpeena kokea edistyvänsä pelissä ja saada konkreettista palautetta onnistumisista ja epäonnistumisista. (Shodhan 2017, Prieto, Medina-Medina et al. 2017.)

Tarkasteltaessa aiheita, joita lapset haluavat pelatessaan oppia, matematiikka on ylivoimainen ykkönen. Perusteluina on muun muassa se, että matematiikan opettelu ei ole hauskaa pelien ulkopuolella. Haasteen osalta tärkeä tekijä on progressiivisuus. Vaikeustason tulisi nousta pelin etenemisen mukaan, alkaen helpommilla haasteilla ja edeten kohti lopun vaikeita haasteita. Tämä parantaa oppimistuloksia ja lisää pelaajan onnistumisen tunnetta.(Nand, Baghaei et al. 2013.)

Vaikka Nandin ym. tutkimus rajoittuukin Haluatko Miljonääriksi -tyylisen pelin suunnitteluun, voidaan tutkimuksen yleisluontoisia tuloksia grafiikasta sekä pelimekaniikan haastavuudesta soveltaa mihin tahansa opetukselliseen peliin, sillä näiden elementtien voidaan ajatella olevan irrallisia varsinaisesta pelimekaanisesta toteutuksesta. Laajenpaa näkökulmaa opetuspelien suunnitteluun tarjoaa Clark N. Quinn (Engaging Learning: Designing e-learning Simulation Games, 2005). Quinn esittää näkemyksen, jonka mukaan opetuspelin kehittäjän ei pitäisi suunnitella sisältöä, vaan kokemuksia. Quinn

perustelee näkemystään sillä, että pelin loppukäyttäjä on tuotava ympäristöön jossa pelaaja ymmärtää paitsi tekemänsä valinnat, myös näiden valintojen vaikutukset sekä tärkeyden. Vain tällöin pelaaja saadaan sitoutumaan pelin tavoitteisiin ja uppoutumaan peliin, joka on olennainen osa oppimisprosessia. (Qiunn 2005, 10.)

Voidaan siis vetää johtopäätös, että ollakseen onnistunut, opetuspelin graafisen ulkoasun on vastattava kohderyhmän tarpeita. Lisäksi pelimekaniikka on tasapainotettava niin, että haaste on jatkuvasti tasapainossa pelaajan kykyjen ja oppimisen kanssa, jotta pelaaja kokee riittävästi onnistumisen tunteita ilman, että tämä turhautuu kohtuuttomasti. Lisäksi pelin tarinan tai tavoitteiden on luotava tarpeeksi painoarvoa pelaajan tekemille valinnoille sekä kommunikoitava näiden valintojen merkitys ja seuraukset pelaajalle mahdollisimman selvästi. Näin pelaaja saadaan oikeasti pohtimaan tekemiään valintojaan ja avattua tämän mieli oppimiselle. (Song, He et al. 2012.)

2.1 Laajennettavuus

Opetuspeliä suunniteltaessa laajennettavuudelle ja muokattavuudelle on uhrattava aikaa suunnitteluprosessissa. Huolimatta tarkasta esivalmistelusta, pedagogiseen materiaaliin saattaa jäädä virheitä tai puutteita, jotka pitää pystyä korjaamaan vielä pelin julkaisun jälkeen. Toisaalta tutkimus tuottaa jatkuvasti uutta sisältöä, ja saattaa olla tarkoituksenmukaista päivittää pelin opetuksellista sisältöä vastaamaan uudempaa tietoa. Tämä edellyttää sitä, että pelin laajennettavuus otetaan huomioon jo suunnitteluvaiheessa. (Klečina 2019.)

Riittävä laajennettavuus on mahdollista saavuttaa modulaarisella pelisuunnittelulla, jossa pelin eri osa-alueet voidaan jakaa pienemmiksi kokonaisuuksiksi. Tässä tapauksessa varsinainen pelimekaniikka voidaan erottaa sisällöstä. (Klečina 2019.) Esimerkiksi visailutyylisessä pelissä varsinainen tekninen toteutus voidaan tehdä riippumatta kysymysten ja vastausten sisällöstä. Kysymysten sisältö voidaan hakea ajon aikana esimerkiksi tekstitiedostosta ja asettaa sille määritettyyn sisältöelementtiin dynaamisesti tarpeen mukaan. Tälläinen toteutus mahdollistaa kysymysten sisällön vaihtamisen ilman että varsinaiseen pelimekaniikkaan tarvitsee koskea lainkaan. Sama pätee laajemmin myös muunlaisiin opetuspeleihin. Jos sisältö on toteutettu riippumattomaksi pelimekaniikasta, voidaan sisältöä muuttaa ilman suuria muutoksia itse peliin. (Schöbel, Janson et al. 2017, Klečina 2019.)

2.2 Pelitestaus

Testattaessa opetuskäyttöön suunnattua peliä yksi testausprosessin eroista liittyy testauskohderyhmän valintaan. Siinä missä viihdepelin testauksessa pääpaino on aiotun kohderyhmän testaamisessa, opetuspelit on syytä testata myös opetusmateriaalin asiantuntijoiden kanssa (Bali 2016). Tällä varmistetaan että pelin sisältö ja mekaniikat vastaavat parhaalla tavalla opetettavan materiaalin vaatimuksia. Samoin myös testauksen kriteerien on otettava huomioon varsinainen oppiminen. Opetuspelille ei riitä, että kohderyhmään kuuluvilla pelaajilla oli hauskaa peliä pelatessa, vaan testausprosessin aikana on myös mitattava varsinaisen oppimisen taso. (Boller, Steven 2014.)

2.3 Interaktiivinen tarinankerronta

Interaktiivinen tarinankerronta on tarinankerronnan muoto, jossa lukija voi vaikuttaa tarinaan ja sen lopputulokseen (Song, Qiulian et al. 2012). Interaktiivista tarinankerrontaa on käytetty tehokeinona niin elokuvissa, kirjoissa kuin videopeleissäkin. Interaktiivisessa tarinankerronnassa maailma koostuu niin kutsutusta tarinamaailmasta (eng. Storyworld), jossa tarina tapahtuu. Tarinamaailmaan kuuluvat kaikki tarinan hahmot sekä tapahtumapaikat ja tapahtumat, ja pelaaja vaikuttaa tähän maailmaan valintojensa kautta. (Storyworlds.) Se, kuinka paljon valtaa pelaajalla on suhteessa kertojaan riippuu tarinasta ja pelisuunnittelun asettamista tarpeista. Yksinkertaisimmillaan pelaaja voi valita muutamasta ennalta määritellystä tarinapolusta haluamansa dialogipolun, jota pelaaja seuraa loppuun asti. Monimutkaisemmassa toteutuksessa pelaaja pystyy vaikuttamaan polkuun useassa kohdassa tarinaa ja näin ohjaamaan tarinaa hienovaraisesti kohti loppua. Yksinkertaisessa tarinassa pelaajan valinnat voivat muodostaa yksinkertaisen puurakenteen, jossa on vain muutama haarautumaton oksa. Monimutkaisemmasta tarinamaailmasta saattaa muodostua monimutkainen puu, jonka oksat haarautuvat ja yhdistyvät taas uudestaan joko itseensä tai muihin oksiin, luoden lukuisia rinnakkaisia tarinoita joiden välillä pelaaja voi valintojensa kautta liikkua. Pelin tarinarakenne voi olla myös piilotettu pelaajalta eikä pelin tarinarakennetta voi suoraan kuvata puurakenteella (Champandard.) Esimerkkinä tämän kaltaisesta toteutuksesta on Michael Mateaksen ja Andreas Sternin kehittämä Façade (Façade).

Façaden tarinarakenne perustuu luonnollisen kielen prosessointiin (eng. Natural Language Processing, NLP), jossa peli tulkitsee pelaajan kirjoittamia lauseita ja etsii niistä merkityksiä sekä yksittäisiä sanoja. NLP on tietojenkäsittelytieteen ja tekoälyn osa-

alue, joka tutkii tietokoneiden ja ihmisten vuorovaikutusta luonnollista kieltä prosessoimalla sekä analysoimalla (SAS). Pelin muut hahmot reagoivat pelaajan kirjoittamaan tekstiin eri tavoin riippuen siitä mitä pelaaja kirjoittaa, joko negatiivisesti tai positiivisesti. Tietokoneen ohjaamat hahmot ymmärtävät lukuisia erilaisia lausekonteksteja ja reagoivat niihin kuten tosielämän ihmiset. Pelaaja voi omilla vastauksillaan ohjata hahmojen keskinäistä kanssakäymistä sekä ohjailla peliä kohti useita mahdollisia lopputuloksia. Tämän toteuttamiseksi Façadessa käytetään luonnollisen kielen prosessoinnin lisäksi lukuisia muita tekoälypohjaisia teknologioita lauseiden merkitysten tulkitsemiseen ja oikean reagointitavan valintaan (Champandard). Mikäli hahmot eivät ymmärrä pelaajan lauseita, hahmot antavat geneerisen vastauksen joka yrittää ohjata pelaajaa kysymään ymmärrettävämpiä kysymyksiä. Vaikka Façade julkaistiin jo vuonna 2005, on se yhä hyvä esimerkki siitä mitä pitkälle viety interaktiivinen tarinankerronta voi videopeleissä olla. (Champandard.)

Interaktiivisen tarinankerronnan sovellus voidaan jakaa elementteihin, jotka muodostavat selkärangan tarinankerronalle. Näitä elementtejä ovat draamamanageri, käyttäjämalli sekä agenttimalli. Draamamanageri hallitsee tarinan tuottamista ja järjestelee tapahtumia pelaajan koettavaksi. käyttäjämalli sekä agenttimalli taas kuvaavat vastaavasti pelaajan hahmoa sekä tietokoneen ohjaamia hahmoja. Tietokoneen ohjaamat hahmot ovat olemassa viemässä tarinaa eteenpäin ja luomassa pelaajalle uskottavan taustan pelaajan kokemille tarinoille. Nämä hahmot saavat tietonsa draamamanagerilta, joka pitää kirjaa siitä mitä kukin hahmoista tietää. (Storyworlds.)

Toisaalta ongelmia voi syntyä myös, jos insinöörien ja taiteellisen sisällön tuottajien vastuualueita ei ole rajattu riittävän tarkasti. Crawford (2002) esittää, että vaikka teknologia interaktiivisten tarinoiden luomiseen on ollut olemassa jo pitkään, teosten taso ei ole erityisen korkea johtuen insinöörien ja artistien välisistä ideologisista eroista. Ratkaisuna esitetään työtehtävien jako niin, että insinööri keskittyy rakentamaan työkalun joka palvelee nimenomaan artistin tarpeita. Artistien on toisaalta ymmärrettävä teknologian asettamat rajoitteet ja opittava toimimaan näiden rajoitteiden lomassa. Käytännön projektissa tämä tarkoittaa huolellista ja usein toistuvaa vuorovaikutusta sisällöntuottajien ja kehittäjien välillä. (Crawford 2002.)

3 HYÖTYPELIT JA PELILLISTÄMINEN

Hyötypeleiksi voidaan kutsua kaikkia pelejä, joiden pelaamisella on tarkoitus kehittää jotain tosielämässä hyödyllisiä taitoja tai auttaa käyttäjää saavuttamaan hänelle tärkeitä tavoitteita. Hyötypeleiksi voidaan laskea muun muassa erilaiset liikuntapelit- ja sovellukset sekä kognitiiviset harjoitteet. Pelillistämisellä taas tarkoitetaan pelillisten elementtien lisäämistä sovelluksiin ja ympäristöihin. joilla ei suoraan ole mitään tekemistä videopelien kanssa. (Findlay 2016, Waterloon Yliopisto.)

Esimerkiksi tuloslistan ja pisteytysmekaniikan lisääminen liikuntapäiväkirjaan on yksi tapa toteuttaa pelillistäminen. Pelillistämisen tarkoitus on lisätä käyttäjän motivaatiota sovelluksen asettamien tavoitteiden saavuttamiseen sekä tehtävien suorittamiseen. Pelillistämisen ja hyötypelien erona on se, että hyötypelit on alusta asti suunniteltu ja kehitetty videopeleiksi, joihin on lisätty opettavaista tai koulutuksellista sisältöä kun taas pelillistämisessä opetustapahtumaan tai materiaaliin lisätään pelillisiä elementtejä. (Findlay 2016.)

3.1 Hyötypelien ja pelillistämisen hyödyt

Hyötypelien ja opetuspelien hyödyllisyydestä on olemassa runsaasti tutkimustietoa. Opetuspelien ansiosta käyttäjä voi epäonnistua ilman merkittäviä seuraamuksia ja kehittää taitojaan simuloidussa ympäristössä(Management Association 2018). Opetuspelien avulla käyttäjä voidaan totuttaa uuteen asiaan visuaalisien apukeinojen ja videoiden avulla. Opetuspelit mahdollistavat myös opetettavan asian välittömän soveltamisen käytännössä erilaisten interaktiivisten minipelien avulla, mikä tehostaa opitun muistamista ja soveltamiskykyä jatkossa.(Lamb, Annetta et al. 2018.)

Pelillistämisellä taas voidaan lisätä asiakkaiden tai käyttäjien sitoutumista tuotteeseen tai sovellukseen. Esimerkiksi urheilusovellus voi kannustaa käyttäjäänsä liikkumaan enemmän tarjoamalla käyttäjälle pisteitä palkinnoksi liikkumisesta, jolloin käyttäjän motivaatio jatkaa sovelluksen käyttöä on korkeampi kuin ilman pisteytyssysteemiä. (Larsson 2013.)

Vaikka simulaatioita ei tyypillisesti lasketa peleiksi, voidaan simulaatioiden katsoa olevan eräänlaisia hyötypelejä, jotka mallintavat mahdollisimman tarkasti ja realistisesti jonkin tosielämän tilanteen tai tehtävän (Petroski , S). Simulaatioiden avulla on

mahdollista mallintaa tosielämän tilanteita ja prosesseja ennen toteutusta. Näin pystytään hahmottamaan mahdollisten muutoksen vaikutuksia ja hyödyllisyyttä ennen varsinaista toteutusvaihetta. Simulaatioita käyttämällä voidaan helpottaa taitojen omaksumista tosielämää jäljittelevissä tilanteissa. Simulaatiota käyttämällä voidaan myös lievittää koulutettujen pelkoa uusista teknologioista ja prosesseista, sillä simulaatioilla voidaan tarjota koulutettaville turvallinen ympäristö epäonnistua. simulaatioilla on myös paikkansa opiskelussa, sillä simulaatioiden avulla voidaan esittää visuaalisesti erilaisia järjestelmiä ja käsitteitä. (Craig 1996, Lindenberger 2017, Rimon)

3.2 Hyötypelien ja pelillistämisen käyttökohteet

Hyötypelejä ja pelillistämistä voidaan soveltaa mitä moninaisimmissa tilanteissa sekä lisäämään asiakastyytyväisyyttä että tehostamaan oppimista erilaisissa valmennustehtävissä. Hyötypelejä voidaan käyttää opetuksessa havainnollistamaan opetettavaa aihetta käytännön esimerkeillä ja tehtävillä varsinaisen opetuksen aikana. Hyötypeleillä voidaan myös tehostaa erilaisia koulutustilanteita. Yritykset ja organisaatiot käyttävät hyötypelejä joko tehostamaan toimintojaan työntekijöiden valmennukseen suunnitelluilla hyötypeleillä tai valistamaan asiakkaitaan ja kohderyhmiään tarkoitukseen tehdyillä opetuspeleillä. Hyötypelejä voivat käyttää myös tavalliset kuluttajat esimerkiksi terveellisempien elämäntapojen saavuttamiseksi. (Pappas 2018, Pandley.)

Hyötypelejä voidaan käyttää myös mainonnassa ja markkinoinnissa mainospelien muodossa. Mainospelien tarkoitus on levittää tietoa yrityksen tai organisaation tuotteista ja palveluista potentiaalisille asiakkaille. Pääpaino mainospeleissä on yleensä viihteellisessä sisällössä. (Eskelinen 2005.) Myös useat messut käyttävät mainospelejä sekä pelillistettyjä sovelluksia messujen tarjonnan mainostamiseen. Näissä sovelluksissa on usein pelimäisiä ominaisuuksia, kuten esimerkiksi messuständeiltä skannattavia tarroja, jotka avaavat saavutuksia tai ominaisuuksia sovelluksessa. Sovellukset sisältävät yleensä myös muista hyödyllisiä interaktiivisia ominaisuuksia, kuten messualueen kartan, josta käyttäjä voi nähdä reaaliajassa oman sijaintinsa. Pelillistämistä voidaan soveltaa myös pelien ulkopuolella. (Pappas 2018.) Esimerkiksi liikuntapäiväkirjasovellus voi antaa pelaajalle mitaleita ja saavutuksia tämän suoritettua tietyn määrän liikuntasuorituksia. Toinen pelillinen motivaatiota lisäävä keino on esimerkiksi tavoitemittarin lisääminen sovellukseen. Mittarista käyttäjä pystyy visuaalisesti nähdä tehdyt suoritteet sekä sen, kuinka paljon tavoitteeseen on vielä matkaa.

3.2 Tulevaisuus

Tulevaisuudessa AR- ja VR -teknologiat tulevat korostumaan yhä enemmän hyötypelien kehityksessä teknologioiden tullessa yhä suuremman joukon saataville (The Future of Educational Games within Augmented Reality). Erityisesti AR on laajenemassa yhä suuremman käyttäjäkunnan saataville useimpien mobiililaitteiden tukiessa ainakin jotain AR -teknologiaa. Myös VR -laitteet ovat hiljalleen tulossa useampien kuluttajien saataville, laitteiden kovan hinnan ollessa yhä merkittävä este yleistymiselle. Tulevaisuudessa myös laitteiden koko tulee pienenemään, jolloin VR -laitteita pystytään käyttämään monipuolisemmissa ympäristöissä ja tilanteissa. Komponenttien pieneminen tekee myös mahdolliseksi laitteet, jotka eivät tarvitse erillistä tietokonetta toimiakseen, vapauttaen käyttäjät hyödyntämään laitteita esimerkiksi pihalla tai puistoissa. (Ffiske 2018, Fingas.)

Ilmiönä opetus- ja -hyötypelit tulevat yhä kasvamaan ja niiden rooli varsinkin opetuksessa tulee olemaan yhä keskeisempi. Opetuspelit mahdollistavat entistä joustavamman opetuksen eri tasoisille oppilaille tinkimättä opetuksen laadusta. Pidemmällä aikavälillä opiskelun suunnittelussa ja arvioinnissa voidaan käyttää massadataa ja datalouhintaa tutkimaan opetuspelien keräämää tietoa ja näin kehittää opetusta entistä yksilöllisemmäksi. (Ffiske 2018, Fingas, Tack.)

4 PELIMEKANIIKKA JA TEKNOLOGIAT

Opetuspelin pelimekaniikkaa suunniteltaessa on tärkeää huolehtia siitä, että pelin tehtävät ja niiden sisältö sekä mekaniikat vastaavat tilanteita joihin pelaajaa yritetään valmentaa. Pelimekaniikan on testattava ja kehitettävä niitä taitoja, joita pelaaja tarvitsee. Myös teknologisten ratkaisujen on tuettava pelaajan oppimista ja opetustapatumaa. (Boller, Steven 2014.) Eräitä opetuspeleille tyypillisiä elementtejä ovat muun muassa avattavat saavutukset, aikarajoitukset sekä tiedon tai sisällön jakaminen pienempiin kokonaisuuksiin (Kadle 2010).

4.1 Pelimekaniikka

Pelimekaniikalla tarkoitetaan tyypillisesti kokoelmaa sääntöjä, joiden pohjalta muodostuu pelin varsinainen toiminnallisuus. Pelimekaniikka voidaan ajatella rajapintana pelaajan ja pelin lähdekoodin välillä, jonka kautta pelaaja vuorovaikuttaa pelitapahtumiin ja toimii pelin maailmassa. (Sicart 2008.) Opetuspelin pelimekaniikkaa suunniteltaessa on tärkeää tiedostaa pelin kohteena olevan käytännön tilanteen vaatimukset sekä onnistumisen kannalta tärkeät seikat. Esimerkiksi jos tarkoituksena on valmentaa käyttäjää asiakaspalvelun tehtävissä, voi tuntua houkuttelevalta lisätä peliin haastetta ja kiireen tuntua aika rajalla. (Boller, Steven 2014.)

Tämä ei kuitenkaan ole perusteltua opetuksellisesta näkökannasta tarkasteltuna. Vaikka aikarajan lisääminen lisääkin haastetta ja käyttäjän stressitasoa kuten oikeassakin tilanteessa, ei keinotekoisella aikarajalla ole yhtymäkohtaa tosielämän asiakaspalvelutilanteisiin. Aikaraja siirtää pelin fokuksen pois niistä osa-alueista, jotka todellisessa tilanteessa ovat keskiössä, esimerkiksi kuuntelemisen ja eleiden ymmärtämisen taidosta, tekniseen suorittamiseen ja ajankäytön optimointiin. Tämä siirtymä johtaa siihen, että pelin opettamat taidot eivät enää ole hyödyllisiä varsinaisessa tilanteessa, pahimmillaan käyttäjä oppii huonoja toimintatapoja, jos pelin perusmekaniikka palkitsee pelaaja näistä toimintatavoista. (Boller, Steven 2014.)

Myös pelin pisteytyssysteemin on palkittava pelaajaa nimenomaan niistä teoista, jotka ovat olennaisia myös tosielämän tilanteessa pärjäämiseen. Esimerkiksi yksinkertainen pisteytyssysteemi, jossa pelaajaa palkitaan pisteillä vain ajan perusteella, ei välttämättä aja asiaansa jos parhaan ajan saavuttaakseen pelimekaniikka ohjaa pelaajaa

tekemään tositilanteen kannalta huonoja ratkaisuja. (Beginners' Guide: How to Assess the Learning Outcomes of Educational Games.)

4.2 Teknologiset ratkaisut

Opetuspelin teknologista toteutusta suunniteltaessa on tiedettävä millaista opetuspeliä ollaan tekemässä sekä mille kaikille eri alustoille peli on tarkoitus julkaista. Peliin suunniteltu pelimekaniikka, käyttöliittymä sekä toiminnallisuus asettavat vaatimuksia, jotka peliin valittujen teknologisten ratkaisujen tulee täyttää. Tyypillisessä projektissa valinta tehdään valmiiden pelimoottoreiden välillä. Jotkut pelimoottorit sopivat paremmin 2D-grafiikan esittämiseen ja sisältävät valmiit työkalut 2D-elementtien muokkaamiseen. Toinen tärkeä seikka pelimoottorin valinnassa on tuen sekä lisäosien saatavuus. Korkeatasoinen opetusmateriaali helpottaa alkuun pääsemistä sekä nopeuttaa ongelmatilanteiden ratkomista. Myös pelin julkaisualustan valinta on otettava huomioon jo suunnittelun alkuvaiheessa, sillä esimerkiksi suunniteltaessa peliä mobiililaitteille on tärkeää, että valittu käyttöliittymätoteutus sekä itse pelimoottori tukevat suoraan kosketuskontrolleja. (Narain 2016, MightyFingers.)

4.2.1 Pelimoottorit

Pelimoottori on ohjelmistokehys, joka tarjoaa kehittäjälle valmiita työkaluja grafiikan ja pelimekaniikan toteuttamiseen. Useimmista pelimoottoreista löytyy valmiit kirjastot ja toiminnot myös esimerkiksi verkko-ominaisuuksien ja äänimaailman toteuttamiseen. Pelimoottori määrittää pelin koodaamisessa käytettävän koodikielen, mille alustoille peli voidaan kääntää sekä millaisia tekniikoita pelin osa-alueiden suunnittelussa voidaan käyttää. Pelimoottorin valinta vaikuttaa myös siihen, millaista graafista ulosantia pelin on mahdollista tehdä. Pelimoottorin peruskomponentteihin kuuluu myös fysiikkamoottori, joka huolehtii pelifysiikan interaktioista ja fysiikkaan liittyvien laskutoimitusten suorittamisesta. (How Do Game Engines Work? 2016, Game engines - how do they work?.)

Useimmiten käytettyjä pelimoottoreita ovat muun muassa Unity, Unreal Engine, Godot ja Defold. Näistä Unity, Unreal Engine sekä Godot sopivat 3D-pelien kehitykseen. 2D -kehitykseen parhaiten sopivat Defold, Godot sekä Unity. Myös Unreal Engine tukee 2D -grafiikkaa, mutta sen Paper2D -komponenttia ei ole päivitetty pitkään aikaan. Myös

suorituskyvyn optimoinnissa 2D-peleissä on Unreal Enginessä nähtävä ylimääräistä vaivaa. (r/unrealengine - WHY exactly is Paper2D not recommended?.)

Godot taas sisältää täyden tuen 2D-grafiikalle, kuten myös Unity. Pelimoottorin valintaan vaikuttaa merkittävästi se, millaista opetuspeliä on tarkoitus. VR- ja AR-tuen suhteen Unity ja Unreal ovat jonkin verran Godotia edellä, joten tälläisten opetuspelien kehityksessä Unreal ja Unity ovat hyviä vaihtoehtoja. Pelimoottori määrittää myös pelinkehityksessä käytettävän ohjelmointikielen. Sekä Unityn käyttämä C# sekä Unrealin käyttämä C++ ovat ohjelmistokehityksessä yleisesti käytettyjä kieliä (TIOBE Index). Kummastakin kielestä on myös olemassa runsaasti materiaalia käyttäjien saatavilla. Godot taas käyttää virallisesti omaa GDScript-ohjelmointikieltä, jota ei käytetä muussa ohjelmistokehityksessä. GDScript on syntaksiltaan kuitenkin melko yksinkertainen, muistuttaen Python-ohjelmointikieltä. Godotissa on myös mahdollista käyttää useita muita ohjelmointikieliä, esimerkiksi edellämainittuja C#- sekä C++ -ohjelmointikieliä. (Godot Engine latest documentation, Unity Technologies, Epic Games.)

Defold taas on puhtaasti 2D-pelien kehittämiseen suunniteltu pelimoottori jonka koodikielenä käytetään Lua-koodikieltä. Defoldin pääpaino on mobiili- ja selainpohjaisten videopelien kehityksessä ja julkaisussa Andoid- ja iOS -käyttöjärjestelmille vaikkakin Defoldilla pystyy koostamaan projekteja myös Windows-, Linux- sekä MacOS -käyttöjärjestelmille. Defold on vertailluista pelimoottoreista ainoa, joka ei tue pelien julkaisemista Nintendon, Sonyn tai Microsoftin pelikonsoleille. (Learn to make 2D games with Defold.) Myöskään Godot ei virallisesti tue konsoleille julkaisemista, vaikka UWP-alustan avulla onkin mahdollista julkaista pelejä Microsoftin Xbox-pelikonsolille. Godotiin on myös saatavilla kolmannen osapuolen palveluita, joita käyttämällä on mahdollista kääntää Godotilla kehitettyjä pelejä Sonyn ja Nintendon konsoleille. (Console support in Godot - Godot engine latest documentation 2018.)

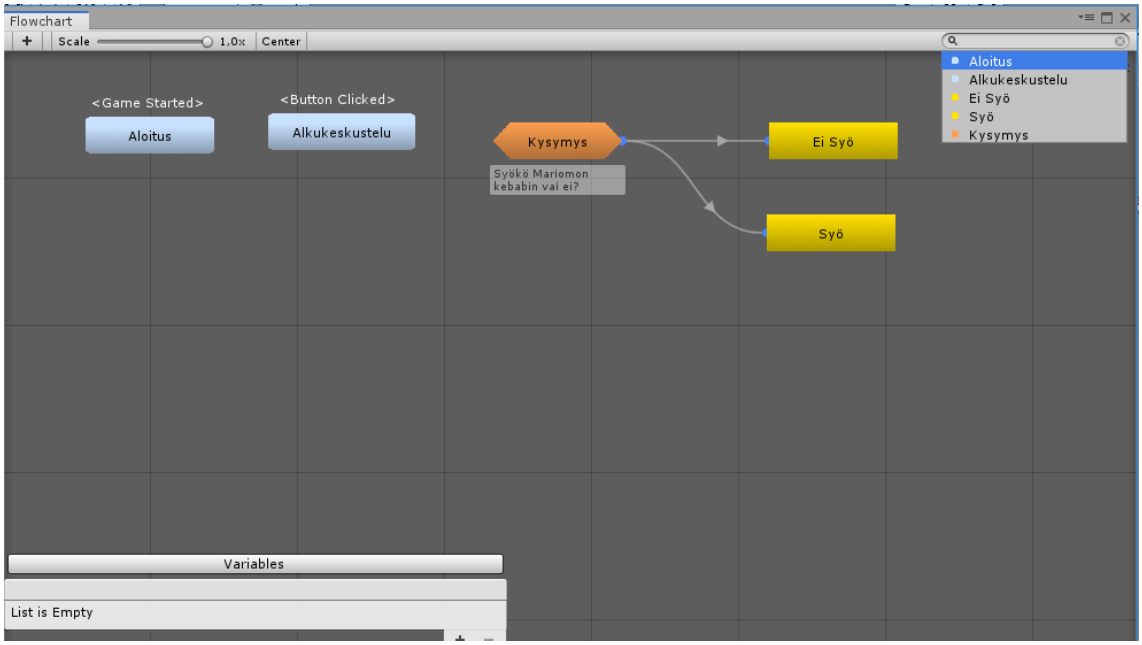
Kehitettäessä peliä, jonka pääpaino on interaktiivisessa tarinankerronnassa ja suhteellisen yksinkertaisessa grafiikassa, suureen rooliin nousee tarinarakenteen luomiseen tarkoitettujen työkalujen saatavuus. Tässä suhteessa Unityn Asset Store-kauppapaikka on monipuolisempi verrattuna esimerkiksi Godotiin ja Unrealiin. Unityn Asset Storesta löytyy erilaisia lisäosia joko maksullisina tai ilmaisina versioina. Tarinankerronnan kannalta eräitä ilmaisena saatavilla olevia lisäosia ovat muun muassa RPGTalk sekä Fungus. Vastaavia maksuttomia lisäosia ei ole saatavana Unrealille, joten vastaava toiminnallisuus on kehitettävä itse tai maksettava valmiin lisäosan käytöstä. Godotilla on mahdollista käyttää ilmaisia runkoja interaktiivisen tarinankerronnan toteuttamiseen,

esimerkkinä Godotin oma Escoria. (Godot Engine - Our point'n'click framework is finally out!.)

Escoria on alun perin point and click -tyylisten seikkailu- ja pulmapelien kehitykseen tarkoitettu kehys, josta löytyy myös interaktiivisen tarinan toteuttamiseen tarvittavat ominaisuudet. Toisin kuin esimerkiksi Unityn Fungus, Escoria ei ole kokonainen editori eikä sisällä omia työkaluja tai käyttöliittymää. Escoria ei myöskään sisällä työkaluja toiminnallisuuden lisäämiseen ilman koodia, vaan Ecscoriaa on tarkoitettu käytettäväksi osana Godotin omaa editoria toiminnallisuuden toteuttamiseen. Escoriasta puuttuu muun muassa Funguksen vuokaaviotyökalu, vaan tarinan etenemislogiikka on ohjelmoitava peliin itse. (Godot Engine - Our point'n'click framework is finally out!.)

4.2.2 Tarinankerronnan työkalut Unity-pelimoottorissa

Unity-pelimoottoriin sisältyy mahdollisuus käyttää useita erilaisia tarinankerronnan toteuttamiseen suunnattuja työkaluja, joista tässä osassa tutustutaan RPGTalkiin sekä Fungukseen. Eräitä visuaalisten novellien tekemiseen tarkoitettuja Unity-lisäosia ovat muun muassa Fungus, Visual Novel Toolkit sekä RPGTalk (RPGTalk 2016, Fungus, SOL-TRIBE). Projektissa tullaan myös tarvitsemaan käännöksiä suomesta ruotsiksi, tässä osiossa tarkastellaan erilaisia työkaluja ja avoimen lähdekoodin käännöskirjastoja. Yksi vaihtoehdoista, Visual Novel Toolkit, jätettiin pois vertailusta, sillä sitä ei ole päivitetty pitkiin aikoihin eikä sen yhteensopivuudesta uudempien Unity-versioiden kanssa ole mitään takeita(SOL-TRIBE). Fungus mahdollistaa interaktiivisten tarinoiden luomisen helppokäyttöisillä työkaluilla. Funguksessa on lisäksi tuki hahmoille, joita voidaan käyttää joko keskustelukuvakkeina tai hahmoina itse pelialueella. Erityisen hyödyllinen on Funguksen flowchart-toiminto, jonka kautta hallitaan keskustelun kulkua sekä tarinan haarautumista. Vuokaavion avulla kehittäjä voi helposti lisätä keskusteluja ja määrittää haarautumiskohtia keskusteluihin pelaajan valintojen perusteella sekä nähdä yhdellä vilkaisulla koko keskustelukartan. Fungus on lähdekoodiltaan avoin, joten siihen on mahdollista tehdä pelikohtaisia muutoksia tarpeen mukaan.



Kuva 1. Esimerkki Funguksen flowchart-vuokaaviosta.

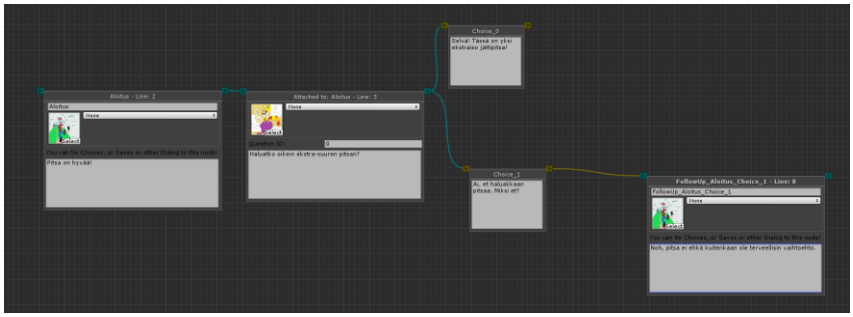
Vuokaavio koostuu lohkoista, (eng. Block), joihin kehittäjä voi määrittää erilaisia tapahtumia ja ehtoja jotka ohjaavat pelin kulkua. Lohkoja voi luoda ja poistaa vapaasti ja niiden avulla onkin helppo rakentaa melko monimutkaisiakin keskusteluja ja rakenteita. Lohkoihin voi myös liittää Unityn peruselementtejä, kuten animaatioita, tekstiä ja grafiikkaa. Funguksen keskustelujen teksti kirjoitetaan jokaiseen palikkaan erikseen tai se voidaan hakea Lua -tiedostoista (Fungus: Controlling Fungus 2018). Erityisesti voimakkaasti haarautuvaan tarinaan on todennäköisesti tehokkaampaa kirjoittaa tekstit suoraan lohkoihin, sillä Lua -tiedostoon kirjoitettaessa jokainen valinta vaatii kummankin vaihtoedon sisällyttämistä tiedostoon sekä perättäisiä if-else -rakenteita. Tämän vuoksi LUA-tiedoston tekstimäärä varsinkin isommissa projekteissa kasvaa suureksi, tehden tarinarakenteen hahmottamisesta vaikeaa. Toisaalta, jos kaikki tarinan tekstielementit sijaitsevat samassa tiedostossa, on tekstiä helpompi muuttaa myöhemmin. Erityisesti jos sisältöä tarvitsee kääntää toiselle kielelle, voidaan tekstitiedostoja tehdä erikseen jokaiselle kielelle ja ladata ne käyttöön valitun kielen mukaan. Käytännössä tällä ei kuitenkaan ole juuri merkitystä, sillä Fungus mahdollistaa myös vuokaaviossa olevan tekstin automaattisen tuomisen CSV- tai JSON -muotoisena tekstitiedostona, johon on helppo lisätä käännökset sarake kerrallaan (Fungus: String table 2019). CSV (Lyhenne sanoista Comma Separated Values) on tekstitiedostoformaatti, jota käytetään datan tallentamiseen ja siirtämiseen tietokoneiden ja verkkojen välillä. CSV on yksinkertainen taulukkomallinen tallennustapa, jossa kentät erotellaan toisista rivinvaihdoilla ja

pilkuilla. CSV -tiedostoja voidaan avata ja muokata useimmilla taulukkolaskentaohjelmilla. (Y. Shafranovich 2005.)

```
local choice = 0

choice = choose { "Option A", "Option B" }
if choice == 1 then goto optionA end
if choice == 2 then goto optionB end

::optionA::
say "Chose option A"
goto endoptions

::optionB::
choice = choose { "Option C", "Option D" }
if choice == 1 then goto optionC end
if choice == 2 then goto optionD end
goto endoptions

::optionC::
say "Chose option C"
goto endoptions

::optionD::
say "Chose option D"
goto endoptions

::endoptions::
say "End options"
```

Kuva 2. Esimerkki Funguksen Lua-syntaksista. (Kuvakaappaus sivustolta http://fungus-docs.snozbot.com/Lua_controlling_fungus.html, Huhtikuu 23. 2019)

Vuokaaviotyökalu mahdollistaa tarinarakenteen esittämisen visuaalisesti, joka helpottaa hahmottamista. Tekstipohjaisessa ratkaisussa etuna on helpompi laajennettavuus ja muokattavuus Unityn ulkopuolella, mutta CSV-tiedostojen tuonti- ja vientiominaisuus antaa lähes saman muokattavuuden kuin puhdas Lua -pohjainen tarina ja on helpompi ylläpitää. Toinen vaihtoehto tarinankerronnan toteuttamiseen on RPGTalk (RPGTalk 2016). Kuten Fungus, myös RPGTalk mahdollistaa keskustelujen luomisen kohtuullisen pienellä vaivalla. RPGTalk on Funguksen tavoin avoimeen lähdekoodiin perustuva ratkaisu, joten sen toiminnallisuutta voi vapaasti muokata ja laajentaa vastaamaan paremmin kulloisenkin peliprojektin tarpeita. Merkittävin ero Funguksen ja RPGTalkin välillä on vuokaaviotyökalun puuttuminen RPGTalkista. RPGTalkin uusin versio (keväällä 2019, 1.3) sisältää Node Editor-työkalun, joka toimii samaan tapaan kuin Funguksen vuokaaviotyökalu, mutta vielä keväällä 2019 kyseinen työkalu oli vielä testausvaiheessa sisältäen toimintavirheitä.



Kuva 3. Esimerkki RPGTalkin Node Editor-työkalusta.

Esimerkiksi valintojen luominen ja haarautuminen tuotti ongelmia Node Editoria käytettäessä. Valintarakenteita tallennettaessa käyttäjä saattaa saada virheilmoituksen, jossa keskustelun pohjana toimiva tekstitiedosto pyydetään lataamaan uudestaan. Uudelleenlatauksen yhteydessä Node Editor hukkasi kaikki valintalohkot ja hahmojen keskustelukuvakkeet. Node Editorin kautta ei myös ainakaan toistaiseksi ole mahdollista lisätä suoraan uusia hahmoja, vaan nämä on lisättävä manuaalisesti tekstitiedostoon. Perustilassa RPGTalkin dialogit luetaan suoraan tavallisesta tekstitiedostosta, josta dialogialueelle luettava teksti valitaan joko rivinumeron tai otsikon perusteella. Otsikoilla dialogi voidaan jakaa pienempiin kokonaisuuksiin, jolloin tekstitiedostosta saadaan selkeämpi. Itse pelin koodissa RPGTalkin dialogifunktion parametriksi voidaan antaa haluttu otsikko, jolloin RPGTalk toistaa automaattisesti koko otsikon rajaaman alueen sisällön.

```
1  [title=Aloitus_begin]
2  Pitsa on hyvää![newtalk start=3 break=Aloitus_end]
3  [title=Aloitus_end]
4  [title=FollowUp_Aloitus_Choice_1_begin]
5  Noh, pitsa ei ehkä kuitenkaan ole terveellisin vaihtoehto.
6  [title=FollowUp_Aloitus_Choice_1_end]
7  [title=FollowUp_Aloitus_Choice_-1_begin]
8  Aahh kyllä, pitsa antaa minulle varsinaiset supervoimat.\nEn kyllä tiedä jaksanko syödä aivan kaikkea, mutta ainahan voin laittaa loput pakastimeen.
9  [title=FollowUp_Aloitus_Choice_-1_end]
10
```

Kuva 4. Esimerkki RPGTalk-dialogin tekstitiedostosta.

Myös edellä mainittu Node Editor -työkalu käyttää tekstitiedostoa pohjanaan. Node Editorilla pystyy joko luomaan kokonaan uuden tekstitiedoston tai avaamaan olemassaolevan RPGTalkin syntaksia noudattavan tekstitiedoston. Tallennettaessa uutta tekstitiedostoa Node Editor luo automaattisesti kaaviorakennetta vastaavan rakenteen itse tekstitiedostoon. Vastaavasti tekstitiedostoa avattaessa Node Editor luo tekstitiedoston rakennetta vastaavan vuokaavion. Node Editor voidaan siis nähdä graafisena käyttöliittymänä RPGTalkin tekstitiedostoille. Laajennettavuuden ja muokattavuuden kannalta ratkaisu on toimiva, sillä se yhdistää vuokaavion helppolukuisuuden sekä tiedostopohjaisen ratkaisun modulaarisuuden. Tekstitiedostoa pystyy tarvittaessa muokkaamaan

Unityn ulkopuolella normaalilla tekstieditorilla ja Unityn sisällä pystyy käyttämään Node Editoria, joka luo vuokaavion muokatusta tekstitiedostosta. Muokkaustapojen välillä pystyy vaihtamaan vapaasti. Myös RPGTalkista löytyy mahdollisuus dialogikäännösten hallintaan ja tekoon.

4.2.3 Käännösmekaniikka

Ennen käännösten toteutustavan valintaa on syytä perehtyä lokalisaatioon konseptitasolla. Tarkoituksena on saada käsitys siitä, millaisten seikkojen huomioon ottamista pelikäännösten tekeminen vaatii. Lokalisaatio tarkoittaa tuotteen tai pelin sisällön sovittamista tietylle kielellä ja kulttuurialueelle. Käytännössä tämä tarkoittaa useimmiten tekstien ja äänitiedostojen kääntämistä kohdekielelle, mutta pitää usein myös sisällään sisällön muun muokkaamiseen kohdekulttuuriin sopivammaksi. Itse lokalisaatioprosessi voidaan jakaa useampaan vaiheeseen, jotka rytmittävät prosessia. Työskennelläkseen tehokkaasti ja tuottaakseen kontekstiin sopivaa käännöstekstiä kääntäjien on ensiksi tutustuttava sisältöön itse pelissä. Pelkkien käsikirjoitusten antaminen käännettäväksi ei yleensä riitä, sillä pelien dialogi on yleensä rakenteeltaan melko monimutkaista ja vaikeasti ymmärrettävää pelin ulkopuolella. Siksi kääntäjille on varattava aikaa tutustua käsikirjoitukseen itse pelissä toteutettuna. Seuraavaksi tehdään tyyliohje, joka sisältä tärkeimmät termit sekä niiden lausumisoppaan. Tyylioppaan ja sanaston tarkoituksena on varmistaa mahdollisimman tasalaatuisten ja yhteensopivien käännösten tekeminen. Tyyliohjetta ja sanastoa voidaan muokata projektin edetessä uusien tarpeiden ja käännössääntöjen mukaan. (Honeywood, Fung 2011.)

Näiden vaiheiden jälkeen voidaan aloittaa varsinainen käännöstyö eli tekstien ja muiden elementtien kääntäminen kohdekielelle. Käännöstyön laadun varmistamiseksi varsinaiset tekstit ovat mahdollisuuksien mukaan kannattavaa luetuttaa erikoistuneilla toimittajilla. Kun käännökset ovat valmiina ne on syytä asettaa demoon ja antaa kääntäjien testata käännöksiä pelissä. Näin kääntäjät saavat paremman kuvan siltä, miltä lopulliset käännökset näyttävät lopullisessa pelissä. Mikäli pelissä käytetään ääninäyttelyä, varsinaisen käännösvaiheen jälkeen tehdään repliikkien äänitys. Lopussa pelin tarkastetut käännökset hyväksytetään ja tarkistetaan mahdollisten virheiden varalta, jonka jälkeen käännökset ovat julkaisukunnossa. (Honeywood, Fung 2011.)

Myös käännösten tekniseen toteutukseen on olemassa suositeltavia toimintatapoja, joita on syytä noudattaa jotta prosessi pysyy mahdollisimman hallittavana. Ohjelmoinnin osalta on suositeltavaa, että kaikki varsinainen käännettävä teksti sijaitsee erillisissä tiedostoissa eikä kovakoodattuna itse pelikoodissa (Honeywood, Fung 2011, s.11). Kovakoodatun tekstin muokkaaminen on hankalaa, sillä se edellyttää ohjelmoijan läsnäoloa tai työpanosta, vaikka tekstiin haluttaisiin tehdä vain pieniäkin muutoksia. Kovakoodattu teksti aiheuttaa ongelmia myös tekstisisällön jakamisessa kääntäjille ja kehitystiimin ulkopuolisille tahoille. Jos varsinainen teksti on kovakoodattuna peliin, täytyy tekstin lisäksi lähettää koko pelin lähdekoodi. Tämä tekee tekstin muokkaamisen vaikeaksi käännöstyöntekijöille ja saattaa myös johtaa siihen, että kääntäjät tekevät vahingossa muutoksia pelin koodiin, jolloin ohjelmoijien on käytettävä aikaansa näiden muutosten korjaamiseen. (Honeywood, Fung 2011.)

Käännösten toteuttamiseen on olemassa useita valmiita työkaluja sekä valmiita käännöksiä sisältäviä listoja, jotka mahdollistavat käännösten toteuttamisen tekstitiedostoihin pohjautuen. Tarkastelussa tutustuttiin Polyglot-, Common Game Translations- sekä Game Text Translations -projekteihin (Damien 2017). Yksi vaihtoehto tämän toteuttamiseen on Polyglot-työkalu, joka on saatavilla Unity-yhteensopivana lisäosana (Polyglot, Unity 2016). Polyglot käyttää käännösten lataamiseen niin sanottua Master-tiedostoa, joka on CSV -tiedosto, johon on valmiiksi sisällytetty peruselementtien käännökset useilla eri kielillä. Käännetyt sanat ovat peleissä yleisesti esiintyviä sanoja, esimerkiksi ”Pelaaja”, ”Taso” sekä ”Painike” jotka voidaan lukea automaattisesti Master-tiedostosta oikeille paikoilleen. Polyglotilla on myös mahdollista luoda omia käännöksiä sekä laajentaa olemassa olevia käännöksiä.

Tärkein puoltava tekijä Polyglotin valinnalle projektiin on ruotsin ja suomen kielen käännösten sisältyminen Master-tiedostoon, eli ne ovat valmiina käytettäviksi heti. Polyglot on myös avoimista ratkaisuista suurten kielien osalta arvioitu luotettavimmaksi ja käyttökelpoisimmaksi ratkaisuksi käännösten tekemiseen(Damien 2017). Polyglot on myös muista tarkastelluista vaihtoehdoista ainoa, josta on olemassa Unity-lisäosa. Lisäosa nopeuttaa huomattavasti käännösten käyttöönottoa ja muokkausta, sillä sen avulla muutokset voi tehdä suoraan editorissa. Muita vaihtoehtoja vastaavien käännösten tekoon ovat muun muassa Common Game Translations ja Game Text Translations. Kummatkin näistä toimivat samalla tavalla kuin Polyglot, eli ne sisältävät Master-tiedoston, jossa on useimmiten peleissä tarvittuja sanoja käännettynä eri kielille. Sekä Common Game Translationin että Game Text Translationin ongelma on kuitenkin siinä, että

kummankaan master-tiedostossa ei ole käännöksiä suomeksi eikä ruotsiksi, joita projektissa tarvitaan. Kummastakaan ei myöskään löydy Unity-yhteensopivaa lisäosaa, joten kehittäjän on tehtävä kaikki muutokset manuaalisesti CSV-tiedostoihin. Näin ollen päädyttiin ratkaisuun, että mikäli lisäosaa käytetään, tullaan projektiin valitsemaan Polyglot.

Myös tarkastelussa olleet tarinankerronnan lisäosat sisältävät käännöstoiminnallisuutta, jolla dialogitekstit voidaan näyttää useammalla kielellä. Näiden ratkaisujen etuna on se, että ei tarvita erillistä lisäosaa käännösten hallintaan, vaan kaikki tarvittava toiminnallisuus löytyy samasta lisäosasta. Funguksen lokalisaatiokomponentti luo tarvittaessa automaattisesti CSV-muotoisen lokalisaatiotiedoston pelin dialogista, johon kehittäjän on helppo lisätä käännökset kullekin tarvittavalle kielelle.

	Key	Description	Standard	Swedish
2	SAY.FIN.7		Fatima on matkalla kouluun. Hän istuu bussissa ystävänsä Saran viereen.	Fatima är på väg till skolan. Han sitter på bussen bredvid sin vän Sara.
3	SAY.FIN.8		Niko huutaa taaempaa:	Niko ropar högre:
4	SAY.FIN.9.Niko		Hei rättipää, onko meillä matikan koe tänään?	Hej ragdoll, har vi matteprov idag?
5	SAY.FIN.1.Fatima		Herään isän rukouskutsuun. Mene olohuoneeseen, missä isä rukoilee.	Jag vaknar till min fars böneanrop. Gå till vardagsrummet där fadern ber.
6	SAY.FIN.3.Fatima		Isä, miksi rukoilemme aamuisin?	Fader, varför ber vi på morgonen?
7	SAY.FIN.4.Fatiman isä		Perheitä on erilaisia. Me olemme muslimeita, siksi rukoilemme.	Det finns olika familjer. Vi är muslimer, så vi ber.
8	SAY.FIN.5.Fatima		Rukoilevatkohan kaveritkin aamuisin…?	Ber du för killarna på morgonen …?
9	SETTEXT.FIN.167			0
10	SETTEXT.FIN.165			0
11	SAY.FIN.16		Mitä teen?	Vad gör jag?
12	MENU.FIN.12		Rättipää, kuulostaapa tyhmältä. En kuitenkaan uskalla sanoa mitään.	Rättipää, låter som en dåre. Men jag vågar inte säga något.
13	MENU.FIN.14		Sanon Saralle; "Aika tyhmää, älä välitä".	Jag säger till Sarah; "Dum dum, oroa dig inte".
14	MENU.FIN.15		Vastaan Nikolle; "Ei se ole rätti, vaan huiv i".	Mot Niko; "Det är inte en trasa, det är en halsduk."
15	SAY.FIN.26		Tytöt selaavat kännykköitään. Tuntematon tyyppi on lähettänyt viestin Saralle, joka näyttää sen Fatimalle.	Tjejer surfar på sina mobiltelefoner. En okänd typ har skickat ett meddelande till Sara, vilket visar det till Fatima.
16	MENU.FIN.30		Kysyn Saralta, onko hän kertonut vanhemmilleen	Jag frågar Sarah om hon berättade föräldrarna
17	MENU.FIN.31		Ehdotan, että vastaamme tyypille yhdessä.	Jag föreslår att vi svarar på typen tillsammans.
18	SETTEXT.FIN.171			0
19	SAY.FIN.29.Fatima		Hups, kannattaakohan tuntemattomille vastata…?	Oj, ska jag känna till de okända …?
20	SETTEXT.FIN.169			0
21	SETTEXT.FIN.173			0
22	SAY.FIN.45		Luokalla on alkamassa uskonnon tunti. Muut jäävät luokkaa, Fatima ja Sara lähtevät islamin tunnille.	Det finns en timmes religion i klassen. Andra är i klassen, Fatima och Sara lämnar för islam.
23	SAY.FIN.46.Matilda		Te ette oo suomalaisia kun käytte eri uskonnon tunnilla.	Du är inte finländare när du deltar i en annan religionslektion.
24	MENU.FIN.48		Matildan kommentti tuntuu pahalta, sillä tunnen itseni suomalaiseksi, mutta en tee mitään.	Matildas kommentar känns dålig för att jag känner finska, men jag gör ingenting.
25	MENU.FIN.50		Vastaan Matildalle, "Oon ihan yhtä suomalainen kuin sä!"	Mot Matilda: "Jag är lika finsk som vädret!"
26	SAY.FIN.37		Tytöt lähettävät viesti tyypille; "Moi :)"	Flickor skickar meddelande för att skriva; "Hej :)"
27	SAY.FIN.38		Puhelin kilahtaa	Telefonen klämpar
28	SAY.FIN.39.Tuntematon tyyppi		No moi, missä oot? Lähetäks selfien.	Tja moi, var är du? Sendks selfien.
29	SAY.FIN.40		Tytöt säikähtävät ja laittavat puhelimen pois	Tjejerna skrämmer och lägger ut telefonen
30	SAY.FIN.41.Fatima		Ei ehkä olisi kannattanut vastata. Pitäisköhän sun jutella tästä kotona?	Det kanske inte varit värt att svara. Längtar att prata om detta hemma?
31	SAY.FIN.34.Sara		En vielä, mutta täytyyki kertoa.	Inte än, men jag var tvungen att berätta.
32	SETTEXT.FIN.175			0
33	SAY.FIN.54		Ruokana on perunoita ja lihapullia.	Maten är potatis och köttbullar.
34	SAY.FIN.55.Fatima		Taas eri ruokaa kuin muille.	Återigen olika livsmedel än andra.
35	MENU.FIN.58		Menen luokan mukana lihapullajonoon. Otan vain perunaa ja näkkäriä.	Jag går med klassen med kött kött kö. Jag tar bara potatis och hane.
36	MENU.FIN.59		Menen Saran kanssa hakemaan kasvisruokaa, ainakin jono on lyhyempi.	Jag går med Sara för att få en vegetarisk måltid, åtminstone är kön kortare.
37	SAY.FIN.83		Fatima vahtii pikkuveljeä olohuoneessa.	Fatima tittar på småbror i vardagsrummet.
38	SAY.FIN.85		Fatima saa viestin Matildalta ja Saralta	Fatima får ett meddelande från Matilda och Sara
39	SAY.FIN.86		Tuletko pelaamaan futista?	Ska du spela fotboll?
40	SAY.FIN.91		Mitä teen?	Vad gör jag?
41	MENU.FIN.90		Joo, nähdään vartin päästä!	Ja, låt oss se dig gå!
42	MENU.FIN.92		Hoidan nyt pikkuveljeä, mutta tulen puolen tunnin päästä.	Nu ska jag ta hand om lillebror, men branden i en halvtimme.
43	SAY.FIN.76.Fatima		Ääh, mummolta tuli viesti. Mun pitäisi mennä kauppaan ostamaan viiniinlehtiä.	Åh, min mormor fick ett meddelande. Låt oss gå till affären för att köpa vinbladen.
44	SAY.FIN.77.Matilda		Mä tuun mukaan.	Det är det.
45	SAY.FIN.78		Tytöt lähtevät kävelemään. Matilda ehdottaa, että he menevät matkan varrella olevaan kauppaan.	Flickor går en promenad. Matilda föreslår att de går till affären längs vägen.
46	SAY.FIN.79.Fatima		Meidän perhe käy aina toisessa kaupassa	Vår familj går alltid till en annan butik
47	SAY.FIN.80.Matilda		Ai miks?	Åh, varför?

Kuva 5. Esimerkki Funguksen lokalisaatiotiedostosta. Käännökset tulevat kukin omiin sarakkeisiinsa, joita voi olla useampia kuin kaksi.

Kun käännökset on lisätty tiedostoon, tiedosto luetaan Unityssa takaisin lokalisaatikomponenttiin, jonka jälkeen käännökset ovat käytettävissä pelissä kehittäjän määrittämien maakoodien avulla.

Uudempi tapa toteuttaa lokalisaatio Funguksella perustuu JSON -tiedostoon, johon dialogiteksti kirjoitetaan (Ks. Kuva 6). Jokaiselle dialogipätkälle määritetään maakoodi, ja tämän maakoodin perusteella valitaan mikä version tekstistä näytetään. JSON (lyhenne sanoista JavaScript Object Notation) on tiedonvälitykseen käytettävä avoimen standardin tiedostomuoto, jota voidaan CSV -tiedoston tavoin käyttää datan säilömiseen sekä siirtämiseen. Nimestään huolimatta JSON ei ole riippuvainen

JavaScript -ohjelmointikielestä. JSON -tiedostossa data tallennetaan sulkumerkkien ja lainausmerkkien sisään ja tallennusmuoto tukee useita muuttujatyyppejä. (Introducing JSON.)

```
{
    "alku" : {
        "fin" : "Ääh, kasin aamu, pitää nousta. Äiti ja isä ovat varmaan jo lähteneet. Onkohan kaapissa vielä muroja…",
        "swe" : "Åh, morgon måste du gå upp. Mamma och pappa har förmodligen redan lämnat. Finns det en spannmål i garderoben?"
    },
    "alku2" : {
        "fin" : "Jes, täällä on vielä, jaetaanko nämä?",
        "swe" : "Jes, är det fortfarande här för att dela dem?"
    },
    "alku3" : {
        "fin" : "Anna tänne, sä voit etsiä muuta syötävää!",
        "swe" : "Ange här, du kan hitta något annat att äta!"
    },
    "alku_val1" : {
        "fin" : "Et sä voi niitä omia, munkin pitää saada!",
        "swe" : "Du kan inte äga dem, munken behöver få!"
    },
    "alku_val2" : {
        "fin" : "(ajatuksissaan)Voi ei, mä en pärjää sille. Toivottavasti leipä ei ole homeessa.",
        "swe" : "(i hans tankar) Åh nej, jag kan inte göra det. Förhoppningsvis brödet är inte i formen."
    },
    "alku_val3" : {
        "fin" : "En sano mitään, vaan menen kaatamaan muroja itselleni."
        "swe" : "Jag säger ingenting, men jag ska hålla spannmålen till mig själv."
    },

    "jami_vie_murot" : {
        "fin" : "Isoveli nappaa murot Nikolta, Niko menee harmistuneena katsomaan löytyisikö laatikosta leipää.",
        "swe" : "Storebror tar tag i smulorna från Niko, Niko blir ledsen för att se om det fanns bröd i lådan."
    },

    "niko_syö_leipää" : {
        "fin" : "Niko menee harmistuneena katsomaan löytyisikö laatikosta leipää.",
        "swe" : "Niko går fel för att se om det fanns bröd i lådan."
    }
```

Kuva 6. Esimerkki JSON -käännöstiedostosta

Tässä ratkaisussa dialogitekstiä ei kirjoiteta itse vuokaaviotyökaluun, vaan vuokaavioon kirjoitetaan tekstitagi, jonka perusteella JSON -tiedostosta noudetaan oikea tekstipätkä näytettäväksi itse peliin. Varsinaisen kielen vaihtaminen tapahtuu kutsumalla Lua -funktiota, joka vaihtaa käytettävän kielen.

Commands	
Execute Lua	*setlanguage("fin")*
Say	*Niko: "{$alku}"*
Say	*Niko: "{$alku2}"*
Say	*Niko: "{$alku3}"*
Menu	*{$alku_val1} : Jami vie murot*
Menu	*(ajatuksissaan)Voi ei, mä en pärjää sille. Toivottavasti leipä ei ole homeessa. : Niko alistuu*
Menu	*En sano mitään, vaan menen kaatamaan muroja itselleni. : Jami vie murot*

Kuva 7. Esimerkki JSON -tekstin kutsumisesta vuokaavion komentolohkoissa.

Funktiokutsua voi käyttää joko vuokaavion kautta komentolohkossa tai Lua -kooditiedostosta. JSON -pohjainen ratkaisu vaatii jonkin verran enemmän suunnittelua JSON -tiedoston rakenteen suunnittelussa, mutta on varmatoimisempi kuin CSV -tiedostoilla toteutettu ratkaisu tilanteissa, joissa tekstiä muokataan editorin ulkopuolella (Skult

2019). CSV:llä toteutettu ratkaisu saattaa aiheuttaa toimintavirheitä jos tekstiä on muokattu runsaasti ennen kuin tiedosto on ladattu takaisin Unityyn. Tällaisten tilanteiden välttämiseksi on CSV:llä toteutettua ratkaisua käytettäessä käytettävä varsinaista käännöstiedostoa vain käännöstekstien asettamiseen ilman, että alkuperäiseen tekstiin tehdään muutoksia. Varsinaiset tekstimuutokset tulee tehdä itse Funguksen vuokaaviotyökalussa. Jos tekstiä on tarve muuttaa Unityn ulkopuolella, on JSON:lla toteutettu ratkaisu luotettavampi ja joustavampi.

RPGTalkin sisältämä käännöstyökalu vastaa melko pitkälti Funguksen vastaavaa työkalua. Merkittävin ero on siinä, että RPGTalkin käännöstyökalu vaatii toimiakseen erillisen tekstitiedoston jokaisesta käännöskielestä. Käännöskielten tiedostot asetetaan RPGTalkin lokalisaatiokomponenttiin, minkä jälkeen niitä voi kutsua koodissa kielen vaihtamiseksi.

5 ESIMERKKIPROJEKTI PELIMO

PELIMO on Unity -pelimoottorilla toteuttava opetuspeli, jonka kohderyhmänä ovat 10–12 -vuotiaat lapset. PELIMO on perusrakenteeltaan tarinallinen peli, jossa pelaaja tekee valintoja ja vaikuttaa näin tarinan lopputulokseen. Pelissä on useita päähenkilöitä, joiden tarinaa pelissä seurataan. Pelin tarkoituksena on opastaa kohderyhmän lapsia kohtaamaan monikulttuurisuutta ja reagoimaan kiusaamis- ja syrjimistilanteissa. Pelin hahmot ovat myös kouluikäisiä lapsia jotta pelaajan on helpompi samaistua tarinaan. Kehitettävä demo ja teknologiavertailu ovat osa Turussa toteutettavaa PELIMO -hanketta, jossa tavoitteena on kehittää peli monikulttuurisuuskasvatukseen. PELIMO on Opetushallituksen rahoittama ja Turun Ammattikorkeakoulun, Humanistisen Ammattikorkeakoulun sekä Turun Yliopiston toteuttama hanke, jonka lopputuloksena kehitettyä peliä on tarkoitus hyödyntää yhteiskuntaopin opetuksessa. (PELIMO | Pelillistäminen mahdollisuutena monikulttuurisuuskasvatuksessa.)

Varsinainen työskentelyvaihe alkoi pelimoottorien vertailuilla. Painopisteenä tässä vertailussa oli pelimoottoriin saatavan opastusmateriaalin saatavuus sekä sopivuus tarinankerronnallisen pelin toteuttamiseen. Tässä tärkeäksi tekijäksi nousivat lisäosat ja niiden saatavuus. Tarinavetoista peliä tehtäessä korkea painoarvo oli pelimoottoriin saatavilla lisäosilla, joilla pelikehitystä voitiin virtaviivaistaa. Tarinavetoisen pelin kehitysprosessissa oli tärkeää pystyä hallitsemaan monimutkaistakin tarinarakennetta tehokkaasti ja selkeän käyttöliittymän kautta. Tämän tutkimiseksi työn aikana tutustuttiin pelimoottoreiden omiin kauppapaikkoihin sekä Internetin keskustelupastoilta kerättyyn palautteeseen. Lisäksi pelimoottoreita ja niiden lisäosia kokeiltiin käytännössä PELIMOa vastaavan pelin kehittämisessä. Samalla perehdyttiin opetuspelien ja hyötypelien sekä myös interaktiivisen tarinankerronnan teoriaan. Teoriaosuuden tarkoituksena oli syventää ymmärrystä opetus- ja hyötypelien suunnittelusta ja kehittämisestä jotta saatiin parempi kuva siitä, millaisia vaatimuksia valitun toteutustavan tulee täyttää. Kun opetuspelien yleiset vaatimukset ja suunnittelulliset yksityiskohdat olivat selvillä, pystyttiin tarkasteltavia toteutusvaihtoehtoja arvioimaan monipuolisemmin, sillä puhtaasti teknologisten seikkojen lisäksi pystyttiin myös arviomaan kuinka tehokasta opetuspelien vaatimukset täyttävää peliä on kehittää valitulla toteutustavalla. Teoriaosuuden perusteella voidaan todeta, että Funguksella on mahdollista toteuttaa opetuspeli, jossa on toimiva pisteytysmekaniikka sekä interaktiivinen tarinarakenne.

Interaktiivisen tarinankerronnan teoriaa sovellettiin pohdittaessa tarinankerronnan toteutusta ja sen asettamia vaatimuksia projektissa. Koska peli on vahvasti tarinavetoinen, oli teorian tuntemuksesta hyötyä myös arvioitaessa pelitarinan toteutuksen teknistä haastavuutta sekä pelaajan ja pelimaailman välisen vuorovaikutuksen toteutusta. Interaktiivisen tarinankerronnan teoria toi esiin myös seikkoja itse kehitysprosessin työnjaosta kehittäjien sekä tarinan tuottajien välillä. Tämä oli otettava huomioon toteutustapaa valittaessa.

Vertailuvaiheessa vaihtoehdot olivat pelidemon toteuttaminen joko Funguksella tai RPGTalkilla. Käytännössä kummatkin vaihtoehdot mahdollistavat samankaltaisen sisällön luomiseksi tarinapohjaiseen peliin. Lopullisessa valinnassa päädyttiin kuitenkin Fungukseen, sillä se oli näistä vaihtoehdoista monipuolisempi. Toisin kuin RPGTalk, Fungusta on mahdollista käyttää muun muassa pelilogiikan sekä animaatioiden ohjaamiseen. Tärkein yksittäinen syy oli kuitenkin RPGTalkin vuokaaviotyökalun puutteet. RPGTalkin vuokaaviotyökalua käytettäessä oli mahdollista menettää lohkoihin määritetyt tarinan haarautumiskohdat toimintavirheiden takia.

Teoria- ja vertailuosuuden jälkeen toteutustavaksi valittiin Unityn Fungus -lisäosa, joka täytti parhaiten toimeksiantajan tarpeet interaktiivisesta pelimekaniikasta sekä teoriaosuudessa esille tulleet seikat käännösmekaniikasta. Fungusta käyttäen käännökset on mahdollista toteuttaa siten, että varsinainen teksti sijaitsee ulkoisessa tekstitiedostossa, jolloin tekstiä ei tarvitse kovakoodata itse peliin. Demon rakentaminen aloitettiin lisäämällä ensimmäinen tarina Fungusta käyttämällä Unity -projektiin. Tarkoituksena oli tutustua Fungukseen työkaluna sekä testata tarinarakenteen loogisuutta niin että kaikki loput on mahdollista saavuttaa valintoja tehden. Ensimmäisessä versiossa olikin epäloogisuus, jonka vuoksi pelaajan oli mahdollista päätyä tilanteeseen, jossa pelaaja ei saa avatuksi yhtäkään loppuratkaisua. Kehitystyö jatkui kolmen muun tarinan lisäämisellä peliin. Lopullisessa demossa oli neljä eri tarinaa ja päävalikko, josta haluamansa tarinan pystyy valitsemaan.

5.1 Prototyypin toteutus

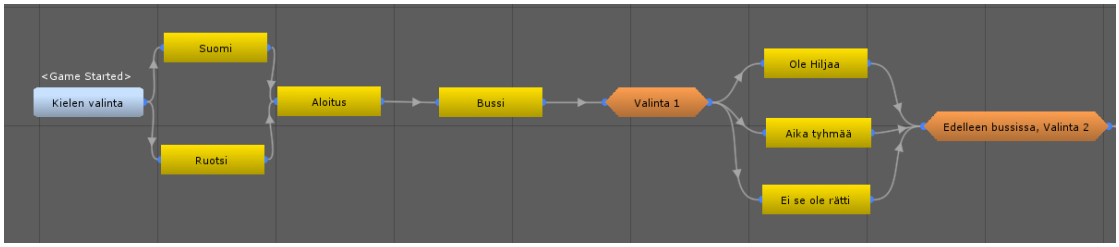
Pelin sisällön suunnittelu alkoi aikaisemmin tehtyjen käsikirjoitusten sovittamisella pelistä tehtyyn demoversioon. Demoversio toteutettiin Unityn 2019 -versiolla. Demoversion tarkoituksena oli testata ensinnäkin Funguksen toiminnallisuutta ja myös selvittää, onko käsikirjoitusten nykyinen rakenne looginen. Demoversiossa kokeiltiin myös

pisteytysmekaniikkaa, jonka perusteella pelin loppu määritetty. Loppuja on erilaisia riippuen siitä, millaisia valintoja pelaaja on pelin aikana tehnyt, ja eri lopputulokset käsittelevät yleensä hahmojen suhteita toisiinsa ja koostavat yhteen pelikerran tapahtumia. Pelin lopussa pelaaja saa myös sanallisen palautteen pelisuorituksestaan, jossa pelaajaa kehutaan onnistuneesta suorituksesta tai kehotetaan esimerkiksi ottamaan muut paremmin huomioon jos pelaajan valinnat olivat kovin itsekeskeisiä. Koska pelin on tarkoitus myös toimia monikulttuurisuuden opetusvälineenä, peli sisältää elementtejä eri kulttuureista ja uskonnoista. Esimerkiksi tiettyjä esineitä klikkaamalla voi saada lisätietoa esineestä ja sen merkityksestä kyseisessä kulttuurissa. Teoriaosuuden perusteella peliin lisättiin pistemittari, josta pelaaja näkee reaaliajassa valintojen muutokset pisteisiin. Demon pohjalta toimeksiantaja voi aloittaa varsinaisen pelin jatkokehityksen.

5.2 Pelimekaniikka

Peli tulee perusmekaniikaltaan olemaan visual novel-tyyppinen peli, jossa pelaaja seuraa tarinaa ja tekee valintoja ennalta määrätyissä kohdissa. Pelaajan tekemät päätökset vaikuttavat tarinaan ja sen lopputulokseen. Valinnat voivat olla joko neutraaleja, negatiivisia tai positiivisia. Positiivisista ja negatiivisista vastauksista pelaaja saa ja menettää pisteitä, neutraaleista ei menetä eikä saa pisteitä. Valinnoista ei etukäteen näe, kuinka paljon pisteitä niistä saa tai menettää, joten pelaaja joutuu miettimään tilannetta ja vaihtoehtojaan ennen lopullista valintaa. Pelaaja kuitenkin näkee välittömästi valintansa vaikutuksen pisteisiinsä valinnan tehtyään, joten palaute valinnasta on välitön. Pelaajan valinnoista saamat pisteet on suunniteltu niin, että parhaat pisteet on mahdollista saavuttaa vain oikeita vastauksia tekemällä. Toisin sanoen, pelaaja ei voi kompensoida alussa tekemäänsä väärää valintaa myöhemmin oikeita valintoja tekemällä. Lopullisessa pelissä on tarkoitus olla viisi eri tarinaa, joita pelaaja voi seurata. Pelin loppu riippuu siitä kuinka paljon pisteitä pelaaja on kerännyt. Osassa tarinoita on myös minipelejä, joiden lopputulos vaikuttaa siihen, miten tarina jatkuu. Varsinaisessa demossa on yksi rankkarinlauonta-minipeli, jossa pelaajan tarkoitus on saada pallo maaliin.

PELIMO-projektissa interaktiivisen tarinankerronnan toteutus on melko tavanomainen, jonka tarinarakenne on haarautuva ja valintapohjainen. Projektin tarinarakenne edustaa puumaista rakennetta, jossa tarina haarautuu oksiin ja yhdistyy sitten tarpeen vaatiessa takaisin.



Kuva 8. Esimerkki PELIMO:n tarinarakenteesta

Pelaaja vuorovaikuttaa tarinaan yksinkertaisesti valitsemalla valikosta haluamansa vaihtoedon, jonka jälkeen tarina jatkuu seuraavaan valintakohtaan asti ilman pelaajan vuorovaikutusta peliin. PELIMO -projektissa suuri osa varsinaisesta pelisuunnittelusta liittyy tarinaan ja käsikirjoitusten toteutukseen, joten Funguksen valita toteutustavaksi ei aseta mitään rajoitteita pelimekaniikan toteutukselle. Funguksen käyttö mahdollistaa pelimekaniikan vapaan suunnittelu Unityn työkaluja käyttäen.

5.3 Laajennettavuus

Pelin tarinarakenne luodaan Funguksen vuokaaviotyökalulla. Käännökset tarinaan voidaan toteuttaa joko tuomalla Funguksen lokalisaatiokomponentin luoma CSV -tiedosto ja lisäämällä siihen käännöstekstit tai toteuttamalla tarinan dialogi JSON -tiedostoon pohjautuvalla tekniikalla, jossa tarinan teksti kirjoitetaan erilliseen tiedostoon ja luetaan tästä tiedostosta itse vuokaaviotyökaluun. Kummassakin tapauksessa käännösten lisääminen onnistuu helposti myös Unityn ulkopuolella yksinkertaisesti lisäämällä käännökset tiedostoon ja vaihtamalla käytettyä kieltä Funguksen komentolohkoissa. JSON -pohjainen ratkaisu mahdollistaa lisäksi tekstin muokkaamisen Unityn ulkopuolella. JSON:lla toteutettu ratkaisu edellyttää JSON -syntaksin tuntemusta, mutta toisin kuin CSV:llä toteutettu ratkaisu, varsinainen tekstisisältö sijaitsee kokonaan omassa tiedostossaan ja sitä voi turvallisesti muokata. CSV:llä toteutetussa ratkaisussa alkuperäisen tekstin muuttaminen saattaa aiheuttaa virheitä luettaessa tiedostoa takaisin Unityyn.

Alkuperäinen suunnitelma oli toteuttaa projektin käännökset Polyglotilla. Projektin suunnitelman edetessä kävi kuitenkin ilmi, että käännökset tehdään kääntäjän toimesta, eikä Polyglotin tarjoamia valmiita käännöksiä välttämättä tarvitakaan. tällaisessa tilanteessa ei välttämättä ole perusteltua käyttää kokonaista kääntämiseen tarkoitettua lisäosaa, sillä loppujen lopuksi käännettäväksi jäisi vain muutamia

käyttöliittymäelementtejä, jotka ovat tehokkaampaa tehdä suoraan Unityssä ilman lisäosia tai CSV-tiedostoja. Polyglot työkaluna on tarkoitettu pääasiassa pelin lokalisointiin tilanteessa jossa tarvitaan valmiiksi tehtyjä käännöksiä, joten sen hyödyllisyys on selvästi pienempi projekteissa, joissa on käännösten tekijä omasta takaa.

Prototyypin käännösmekaniikka toteutettiin CSV:tä käyttäen, eikä mitään ennalta tiedettyjä ongelmia kohdattu kehitystyön aikana. CSV -pohjainen ratkaisu on myös hieman yksinkertaisempi ja nopeampi toteuttaa verrattuna JSON -pohjaiseen ratkaisuun. Yksi tarinoista kuitenkin käännettiin kokeeksi JSON -tiedostolla paremman kokonaiskuvan saamiseksi. Lopullisessa projektissa voidaan melko vapaasti valita käytettäväksi kumpi tahansa ratkaisu. JSON -pohjainen on uudempi ja luotettavampi tapa toteuttaa käännökset. Ratkaisu on vain hieman CSV -pohjaista monimutkaisempi mutta mahdollistaa tekstin muokkaamisen riskittömästi Unityn ulkopuolella. Ratkaisu on myös uudempi ja Funguksen kehittäjien suosittelema tapa toteuttaa käännökset ja siksi tuki kyseiselle ominaisuudelle tulee oletettavasti jatkumaan pidempään kuin CSV -pohjaisen ratkaisun. Toisaalta demon kehityksen aikana ei noussut erityisiä ongelmia CSV -ratkaisun käytössä, joten mitään varsinaista estettä myös tämän toteutustavan valintaan ei ole. Projektissa ei ole juuri tarvetta tehdä merkittäviä muutoksia tekstiin suoraan tekstitiedostosta käsin.

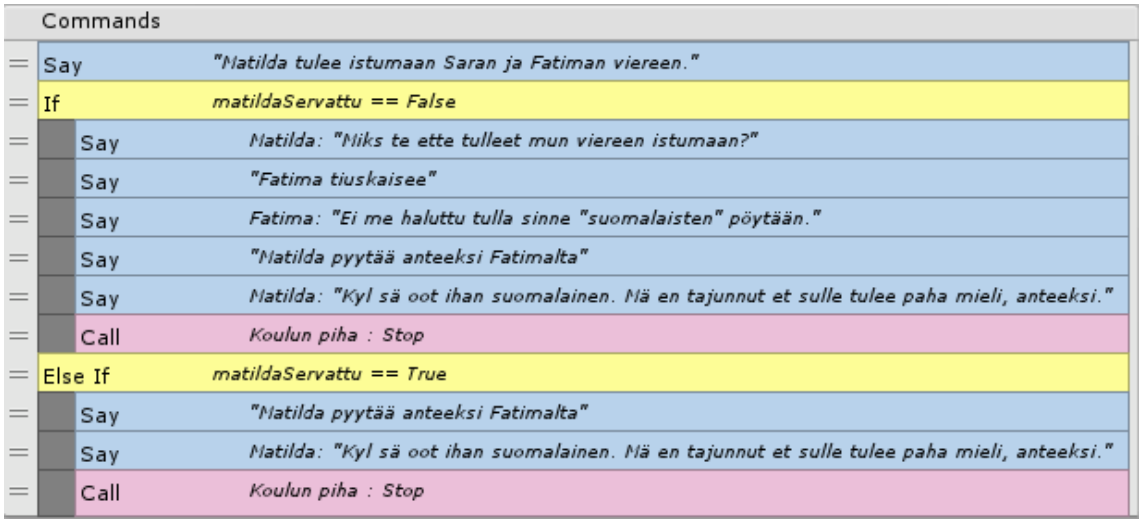
5.4 Funguksen hyödyllisyyden arviointi

PELIMO-projektin puitteissa testauksen kohteena on teknologisten ratkaisujen sopivuus pelin toteuttamiseen. Tarkoituksensa on varmentaa että valittu toteutustapa mahdollistaa käännösten ja sisältää tarvittavat ominaisuudet interaktiivisen pelimekaniikan toteutukseen. Tämän lisäksi varmistetaan, että valittu ratkaisu on mahdollisimman helppokäyttöinen ja luotettava. Tämä toteutetaan vertailemalla valittua ratkaisua vastaavaan projektiin, joka toteutetaan ilman valittua teknologiaa. Toinen potentiaalinen toteutustapa RPGTalk suljettiin pois aikaisemmassa vaiheessa teknisten ongelmien takia.

Projektin pelimoottoriksi oli jo aikaisemmassa vaiheessa toimeksiantajan puolelta valittu Unity, joten pelimoottorien ominaisuuksien vertailu ei lopullisessa teknologiavalinnassa osoittautunut tarpeelliseksi. Muut tarpeet helppokäyttöisyydestä sekä laajennettavuudesta olivat kuitenkin yhä olemassa, joten päätös käytettävistä teknologioista tehtiin näitä vaatimuksia peilaten. Testausvaiheen ja teknologiavertailun perusteella

päädyttiin käyttämään Unityn Fungus-lisäosaa, joka mahdollistaa interaktiivisten tarinoiden luomisen ilman ohjelmointia. Fungus mahdollistaa myös käännösten lisäämisen nopeasti ja ilman ylimääräistä ohjelmointia. Verrattuna ilman Fungusta toteutettuun projektiin, pystytään Fungusta käyttämällä säästämään tunteja kehitysaikaa. Pelkästään dialogin näyttämiseen tarvitaan koodia, jolla voidaan joko lukea suoraan Unityyn syötettyä kovakoodattua tekstiä tai tekstitiedostoon kirjoitettua tekstiä. Helppokäyttöisyyden ja laajennettavuuden kannalta tekstitiedostosta lukeminen on parempi ratkaisu, sillä tekstitiedostoa pystyy tarpeen mukaan muokkaamaan myös Unityn ulkopuolella. C# -kieleen sisältyy funktioita tekstitiedoston lukemiseen, eikä tekstitiedoston lukeminen itsessään ole kovinkaan työläs toteuttaa, mutta ongelmaksi muodostuu tekstin asettelu Unityn käyttöliittymäelementteihin. PELIMO:n tapauksessa dialogitekstiä halutaan näyttää puhekuplissa sekä erilaisissa dialogilaatikoissa joiden koko käyttöliittymässä on rajallinen. Tästä johtuen tekstiä voidaan näyttää vain vähän kerrallaan ilman että teksti valuu ulos elementistä tai jää ruudun ulkopuolelle.

Tämän takia tiedostosta luettavaa tekstiä on pilkottava sopivan pituisiksi tekstipätkiksi, jotta vältytään yhteensopivuusongelmilta käyttöliittymän kanssa. Tämä voidaan saavuttaa myös tekemällä skaalautuvan dialogilaatikon, joka vierittää automaattisesti tekstiä mikäli sen pituus ylittää dialogilementin pituuden. Tämä toiminnallisuus on mahdollista toteuttaa Unityn scroll rect -komponentilla. Scroll rect on vieritettävä käyttöliittymäkomponentti, jonka sisällä voi esittää esimerkiksi tekstiä. Scroll rect voidaan ohjelmoida vierittämään itseään automaattisesti niin kauan kuin uutta tekstiä syötetään elementin sisälle, jolloin koko teksti voidaan näyttää ongelmitta. Fungusta käytettäessä on myös otettava huomioon edellä mainitut seikat käyttöliittymän yhteensopivuusongelmista, mutta Funguksen etuna on tekstin helppo hallinta vuokaaviotyökalulla. Funguksessa halutun pituinen teksti voidaan syöttää yksittäisiin komentolohkoihin, ja tekstiä voidaan vapaasti jakaa pienempiin osiin luomalla lisää objekteja (Ks. Kuva 7). Fungus hoitaa automaattisesti tekstin ketjuttamisen lohkosta toiseen. Ilman Fungusta toteutetussa projektissa tekstin jakaminen sopivan pituisiksi kappaleiksi on tehtävä joko itse luettavaan tekstitiedostoon tai tehtävä ohjelmallisesti tekstin lukemisen jälkeen. Ohjelmallisen jakamisen ongelmana oli tarpeeksi monipuolisen algoritmin kehittäminen. Jos tekstiä jaettiin vain merkkimäärän mukaan, saattoi teksti jakautua kesken lauseen, jolloin tekstistä tuli vaikeasti luettava itse pelissä. Toisaalta älykkäämmän tavan ohjelmointi osoittautui aikaa vieväksi, joten verrokkiprojektissä päädyttiin jakamaan teksti yksinkertaisesti tekstitiedostossa.



Kuva 9. Funguksen komentolohko ja muuttujien käyttö.

Funguksen oma käyttöliittymä sisältää myös suoran tuen muuttujille, joka mahdollistaa esimerkiksi pelaajan tekemien valintojen käyttämisen myöhemmin pelin aikana. Vastaavan toiminnallisuuden ohjelmoiminen ilman Fungusta toteutettuun projektiin vaatii game manager -komponentin ohjelmointia. Singleton -suunnittelumallilla toteutetun game managerin tehtävänä on varastoida pelissä käytettävät muuttujat sekä hallita pelilogiikkaa. Game manageria voidaan kutsua koodissa kun halutaan tarkistaa jonkun muuttujan arvo ja tämän perusteella näyttää muuttujan osoittama dialogi. Tälläinen ratkaisu on toimiva, mutta työläs ylläpidettävä, sillä jos tarinaa halutaan muokata ja lisätä uusia muuttujia, joudutaan muutokset ja uudet muuttujat lisätä ensin game manageriin ja vasta sitten kutsut itse pelin koodiin. Ratkaisu vaatii myös muokkaajalta kohtuullista ymmärrystä ohjelmoinnista ja pelikoodi toteutuksesta. Funguksessa varsinainen koodi on piilotettu helppokäyttöisten valikoiden ja käyttöliittymän taakse, jolloin muokkaaja pystyy tekemään muutoksia tarinan kulkuun ilman että tämän tarvitsee puuttua taustalla olevan koodin toteutukseen. Tämän ansiosta artistit voivat tarvittaessa tehdä suoraan muutoksia tarinaan ilman, että kehittäjien on oltava suuressa roolissa. Vastaavan käyttöliittymäkerroksen toteuttaminen vaatisi vähintäänkin useita tunteja kehitystyötä, sillä muuttujat on sidottava käyttöliittymän valikkopainikkeisiin. Myös varsinaisen pelikoodin dialogisiirtymät on linkitettävä käyttöliittymään, jotta niitä voidaan muokata ilman koodia.

Tässä vaiheessa verrokkiprojektista puuttui kuitenkin yhä yksi Funguksen hyödyllisimmistä ominaisuuksista, eli tarinarakennetta kuvaava vuokaavio. Vuokaavion lisääminen edellytti huomattavia panostuksia käyttöliittymän ja itse pelikoodin ohjelmointiin.

Toteutusta lähdettiin tekemään oliopohjaisella toteutuksella, jossa jokainen vuokaavio palikka on olio, josta löytyy kentät tekstille, muuttujille sekä funktiot palikoiden välillä siirtymiseen. Tämän jälkeen olisi vielä luotava graafinen käyttöliittymä, jonka kautta olioita pystyttäisiin luomaan ja muokkaamaan. Aikataulun kireyden takia toteutusta ei kuitenkaan ehditty saada alkua pidemmälle. Selvä oli kuitenkin se, että vastaavan toteutuksen koodaaminen tyhjästä vaatii huomattavan määrän aikaa itse kehitykseen sekä testaamiseen.

Tärkein peruste Funguksen valinnalle oman järjestelmän kehityksen tai RPGTalkin sijaan on toimintavarmuus. Avoimen lähdekoodin ratkaisuna Funguksella on kohtuullisen suuri käyttäjäjoukko, jonka toimesta koodiin ja käyttöliittymään tehdään parannuksia sekä korjauksia. Fungusta on käyttäjäkuntansa pitkään käyttämä ja huolellisesti testattu ratkaisu, jonka toimintavarmuus osoittautui projektin aikana erinomaiseksi. Myös bugikorjaukset mahdollisiin bugeihin tulevat yleensä nopeasti, sillä bugeja pystyvät korjaamaan myös muut kuin varsinaiset ylläpitäjät. Fungukseen on myös saatavilla melko hyvin tukea erilaisiin ongelmatilanteisiin joko Funguksen omilta foorumeilta tai muilta Internetsivustoilta. Avoimen lähdekoodin ratkaisuna Fungukseen on myös mahdollista tehdä omia korjauksia tarpeen mukaan. Omaan koodiin perustuva ratkaisu vaatisi runsaasti testaamista toimintavarmuuden varmistamiseksi. RPGTakin kohdalla ongelmalliseksi osoittautui epävarmasti toimiva vuokaaviotyökalu, jota käytettäessä oli mahdollista menettää tarinaan tehdyt haarautumiskohdat toimintavirheiden takia.

Laajemman kuvan saamiseksi ratkaisun hyödyllisyydestä projektin loppupuolella teetettiin haastattelututkimus, jossa paikallisilta yrityksiltä kysyttiin niiden kokemuksia vastaavien ratkaisujen käytöstä sekä hyödyistä, joita yritykset kokivat saavuttaneensa valmiita tarinankerronnan työkaluja käyttämällä. Haastattelupyynnöt lähetettiin pelialan yrityksiin, joiden tiedettiin käyttävän tai käyttäneen Fungusta tai jotakin vastaavaa työkalua omissa projekteissaan. Lopulliseen haastatteluun saatiin vastaukset kahden yrityksen edustajilta, Snowfallin sekä Red Nettle Studion kehittäjiltä. Haastatellut henkilöt olivat rooliltaan pelikehittäjiä, joiden työnkuvaan kuului toiminnallisuuden toteuttaminen Fungusta käyttäen. Haastattelukysymyksissä painotettiin työkalun käyttötarkoitusta yrityksen projekteissa sekä hyötyjä, joita yritykset kokivat saavuttaneensa työkalua käyttämällä. Haastattelussa tarkasteltiin myös sitä millaisiin tarkoituksiin työkaluja käytettiin. Haastatteluissa mainittiin syyksi Funguksen tai vastaavan ratkaisun käyttöön työkalun hyvä sopivuus kyseiseen projektiin. Valmiiden työkalujen eduksi mainittiin myös ajansäästö, sillä aikaa ei tarvinnut käyttää perusmekaniikan kehittämiseen, vaan työkaluilla

voitiin tehokkaasti ja nopeasti toteuttaa tarvittavat tarinankerronnalliset toiminnot. Näin ohjelmoijien aika säästyi varsinaisen pelisisällön kehittämiseen. Yksi usein mainittu hyöty oli myös mahdollisuus tuottaa sisältöä ilman merkittävää ohjelmointiosaamista. Erityisesti Funguksen ollessa kyseessä, haastateltavat kokivat suureksi eduksi sen, että myös taiteellisempaa sisältöä tuottavat työntekijät pystyivät lisäämään projektiin sisältöä itsenäisesti. Haastatelluissa yrityksissä Fungusta käytettiin myös muun kuin tarinankerronnan toteuttamiseen, esimerkiksi kenttäeditorina. Esiin nousi myös mahdollisuus käyttää Fungusta koko pelilogiikan ohjaamiseen kustomoitujen komentojen avulla. Myös Funguksen muokattavuus mainittiin haastatteluiden aikana. (Aho 2019, Skult 2019.)

Haastatellut yritykset olivat käyttäneet Fungusta tai vastaavaa ratkaisua erilaisissa peliprojekteissa, niin pulmanratkontapeleissä kuin puhtaasti tarinallisissa peleissä. Funguksen koodin muokattavuus mahdollisti myös edellä mainitun tasoeditori -ratkaisun toteuttamisen. Huonoina puolina Funguksesta mainittiin se, että joskus Funguksen uudet versiot rikkoivat vanhaan versioon tehtyjä muokkauksia, ja päivitykset vaativat usein pientä korjailua. Funguksen lähdekoodista oli myös löytynyt joitain virheitä, joiden korjaaminen ei ollut yksinkertaista. Hyötyihin verrattuna nämä haittapuolet arvioitiin kuitenkin melko pieniksi, varsinkin projektissa, jossa ei edellytetä oman toiminnallisuuden lisäämistä Fungukseen. (Aho 2019, Skult 2019.)

Yhteenvetona voidaan todeta Funguksen sisältävän kaikki PELIMO -projektin kannalta tärkeät ominaisuudet valmiina, joten tarvetta oman toiminnallisuuden lisäämiseen on vain vähän. Fungusta käyttämällä voidaan siis PELIMOn tapauksessa säästää huomattavasti kehitysaikaa, joka voidaan käyttää varsinaisen pelisisällön tuottamiseen. Haastateltujen yritysten perustelut ja syyt Funguksen valintaan vastasivat hyvin PELIMO -projektissa käytettyjä perusteluja, joten voidaan vetää johtopäätös, että Funguksen käyttö PELIMOssa on perusteltu ratkaisu joka tuottaa toimeksiantajalle lisäarvoa.

6 LOPUKSI

Työn lähtökohtana oli tarkastella eri vaihtoehtoja toteuttaa interaktiiviseen tarinankerrontaan pohjautuva opetuspeli. Lopputuloksena päädyttiin Unitylla sekä Funguksella toteutettuun ratkaisuun. Työn lopputuloksena oli toimiva demoversio, joka sisälsi kaikki neljä käsikirjoitusta sekä toimivan pisteytysmekaniikan. Demo noudattaa kohtuullisen hyvin teoriaosuudessa linjattuja vaatimuksia pelimekaniikasta sekä opetuksellisen materiaalin optimaalisesta esitystavasta. Funguksen käyttö ei aseta rajoitteita pelimekaniikan toteuttamiselle, joten toimeksiantaja voi vapaasti soveltaa haluamiaan osia teoriaosuudesta pelin suunnitteluun. Teoriaosuuden ehkä tärkein havainto olikin se, että suuri osa onnistuneen opetuspelin teoriasta liittyy peruspelimekaniikkaan. Vertailua tehdessä voitiin tällöin melko vapaasti vertailla helppokäyttöisyyttä sekä tehokkuutta projektin toteuttamisessa, sillä missään tapauksessa minkään yksittäisen työkalun valinta ei rajoita pelimekaniikan toteutusta esimerkiksi pisteytysmekaniikan tai grafiikan suunnittelussa.

Pelimekaniikan osalta voidaan todeta, että peli ei saa asettaa pelaajaa tilanteeseen, jossa kompromissien tekeminen jollain osa-alueella esimerkiksi ajan säästämiseksi johtaa loppupisteissä parempaan lopputulokseen, jos kyseiset kompromissit todellisessa tilanteessa johtaisivat epäonnistumiseen. Esimerkiksi ensihoidon tehtäviin valmentava peli ei saa asettaa pelaajaa tilanteeseen, jossa esimerkiksi hoitotoimenpiteen ohittaminen tuottaa pienemmän pistesakon kuin säästetty aika tuottaa lisäpisteitä. Visuaalisen novellin muottiin tehdyssä pelissä näitä havaintoja voidaan soveltaa siten, että väärä valinta ei saa koskaan johtaa tapahtumaketjuun, jonka kautta pelaajan on mahdollista saada enemmän pisteitä kuin suoraan oikean valinnan tekemällä. Pelitilanteiden on myös vastattava oikeaa tilannetta, johon liittyen pelaajaa halutaan opettaa.

Laajennettavuuden ja ylläpidon huomioimisen arvoinen seikka on päivitys- tai muuttamisprosessin tekninen vaativuus. Parhaimmillaan prosessi saattaa olla niin yksinkertainen, että sisällön päivitys on mahdollista suorittaa ilman merkittävää panosta pelin suunnitteluun osallistuneilta kehittäjiltä. Tämä säästää aikaa ja resursseja, sillä sisällön vaihtamiseen tarvitaan lähinnä sisällön tuottajan työpanosta. Tietenkään tällainen lähestymistapa ei ole aina mahdollinen, vaan joskus sisällön muutokset edellyttävät merkittäviä muutoksia itse peliin. Tälläinen on esimerkiksi tilanne, jossa puutteet sisällössä liittyvät enemmän pelimekaniikkaan kuin itse sisältöön. Jos vaikka halutaan muuttaa tapaa, jolla pelaaja vaikuttaa pelin maailmaan esimerkiksi hiiriohjauksesta

äänikomentoihin, joudutaan pelin mekaniikkaan tekemään muutoksia. Tärkeää on myös analysoida valitun toteutustavan sekä työkalujen tehokkuutta lopullisessa projektissa. Työssä käytettävien teknologioiden ja työkalujen on mahdollistettava opetuspelien teoriavaatimukset täyttävän pelin kehitys. Tämän lisäksi työkalujen on oltava helppokäyttöisiä ja vakaita. Tarinallisessa pelissä tarinan rakenteen hallinta on tärkeässä roolissa, sillä runsaasti dialogia sisältävässä pelissä tarinarakenteesta saattaa tulla hyvinkin monimutkainen. Tässä auttavat erilaiset vuokaaviotyökalut joilla tarinan kulkua voi havainnoida visuaalisesti. Interaktiivisen tarinankerronnan osalta esitetyt monimutkaiset agenttimallin johtamat tarinamaailmat ovat vielä nykyteknologialla melko teoreettisia käsitteitä. Käytännössä teknologian rajoitteet eivät mahdollista ideaalin interaktiivisen tarinamaailman toteutusta, vaan kehittäjien on usein tehtävä kompromisseja tarinarakenteen suunnittelussa. Rajoitteet liittyvät lähes poikkeuksetta tekoälyyn, sillä ideaali toteutus vaatii erittäin pitkälle kehitettyä tekoälyä, joka pystyy ymmärtämään paitsi sanoja ja lauseita, myös lauseiden tarkoituksia eri konteksteissa. Ongelmaksi muodostuu myös tarpeeksi monimutkaisen järjestelmän kehitys, joka pystyy huolehtimaan hahmoista, hahmojen tietämistä asioista sekä pelaajan vuorovaikutuksesta tarinamaailman kanssa.

Haastattelututkimuksen aikana saadut vastaukset ja kommentit vastaavien työkalujen käytöstä vastasivat hyvin myös PELIMOssa käytettyjä kriteerejä toteutustavan valinnassa. Myös näkemykset valitun ratkaisun hyödyllisyydestä olivat yhteneväisiä opinnäytetyön arviointiosion kanssa. Haastatteluissa mainittiin hyötyinä muun muassa koodarien ajan säästyminen varsinaisen sisällön tuottamiseen sekä mahdollisuus tuottaa sisältöä ilman merkittävää ohjelmointikokemusta. Näin myös käsikirjoittajien on helpompi tehdä muutoksia itse tarinankerrontaan. Koska useat yritykset kokevat ratkaisun hyödylliseksi ja tehokkaaksi, voidaan todeta työn tuottaneen aitoa lisäarvoa toimeksiantajalle.

Projektin aikataulutuksen takia varsinainen pelitestaus ei tähän opinnäytetyön toimeksiantoon ehtinyt mukaan, ja tämä onkin yksi keskeisistä jatkokehitysmahdollisuuksista. Opetuspelin testausprosessin haasteena on muun muassa varsinaisen oppimisen todentaminen. Yhtenä tutkimuskohteena voisikin olla sen pohtiminen, millaisilla mittaustavoilla ja työkaluilla voidaan mitata pelaajien oppimista pelaamisen aikana ja jälkeen. Myös ulkopuolisten asiantuntijoiden tarve on tutkimisen arvoinen asia. Jos opetuspelin opetusmateriaali vaatii ulkopuolisten asiantuntijoiden käyttämistä projektissa, voi olla tarpeen pohtia sitä, miten nämä asiantuntijat sekä heidän tietonsa otetaan parhaalla

tavalla käyttöön sekä kehitys- että testausprosessissa. Kaikki nämä seikat edellyttävät varsinaisen testausprosessin huolellista suunnittelua jo pelikehityksen alkuvaiheessa. Varsinainen pelitestaus on kuitenkin järjestettävä vasta kun pelimekaniikka ja grafiikat ovat niin valmiita, että pelin toiminnasta ja käyttökokemuksesta saa realistisen kuvan.

Arvioitaessa varsinaisen työn suoritusta ja lopputuloksen arvoa kokonaisuutena, voidaan työn täyttävän hyvin kaikki toimeksiantajan vaatimukset ja toiveet toteutustavan valinnassa ja demon toteutuksessa. Toteutettu demo sisältää niin käännökset kuin pisteytysmekaniikan jolla pelaajaa palkitaan oikeista valinnoista. Teoriaosuudessa kerrytetyllä tiedolla pystyttiin varsinaisen toteutustavan valinnassa pohtimaan toteutustapaa laajemmassa kontekstissa ja varmistamaan toteutustavan sopivuus paitsi PELIMO -projektiin, myös laajemmassa mittakaavassa opetuspelien toteutukseen ylipäätään. Demoversiolla varmistettiin paitsi käsikirjoitusten looginen eteneminen, myös valitun toteutustavan sopivuus peliin. Omaa työntekoa arvioitaessa ja kriittisesti pohdittaessa ainoana mainittavana puutteena ja parannuskohteena voidaan nähdä haastattelututkimuksen aikataulutus. Myöhäisen aloittamisajankohdan vuoksi kaikkia tavoiteltuja yrityksiä ei ollut mahdollisuutta haastatella opinnäytetyön aikataulun puitteissa. Haastattelut kuitenkin saatiin Snowfallin sekä Red Nettle Studion edustajilta. Varsinainen vertailu ja demoprojektin toteutus onnistuivat suunnitellusti ja aikataulussa eikä mitään ennakoimattomia ongelmia ilmennyt kehitystyön aikana. Myös työnantajalta saatu palaute tukee johtopäätöstä siitä, että varsinainen toteutustyö onnistui odotusten mukaisesti.

LÄHTEET

ADAMS, E., b-last update, The Designer's Notebook: You Must Play Fa�ade , Now!. Saatavilla: https://www.gamasutra.com/view/feature/130771/the_designers_notebook_you_must_.php. Viitattu 17.05.2019

AHO, J., 2019. *Haastattelu, Snowfall.*

BALI, M., Päivätty 2016, 9 Mistakes to Avoid When Designing Educational Games. Saatavilla: https://clalliance.org/blog/9-mistakes-avoid-designing-educational-games/. Viitattu 28.04.2019

Beginners' Guide: How to Assess the Learning Outcomes of Educational Games. Saatavilla: https://store.teachergaming.com/blog/beginners-guide-how-to-assess-the-learning-outcomes-of-educational-games-n6. Viitattu 20.05.2019

BOLLER, S., 2014. *Games vs Simulations: Choosing the Right Approach for Learning.*

BOLLER, S., 2013. *Learning Game Design: Rewards and Scoring.*

BOLLER, S., Päivätty 2014, 4 Mistakes That Ruin A Serious Game's Learning Value. Saatavilla: https://elearningindustry.com/4-mistakes-that-ruin-a-serious-game-learning-value. Viitattu 29.04.2019

CHAMPANDARD, A.J., , Chatting Up Façade's AI: 23 Ideas to Talk Your Game Into. Saatavilla: http://aigamedev.com/open/review/facade-ai/. Viitattu 17.05.2019

CHAUDY, Y. and CONNOLY, T., 2019. Specification and evaluation of an assessment engine for educational games: Integrating learning analytics and providing an assessment authoring tool. *Entertainment Computing,* **30**, pp. 100294.

Console support in Godot - Godot engine latest documentation. Päivätty 08.08.2018. Saatavilla: https://docs.godotengine.org/en/3.1/tutorials/platform/consoles.html Viitattu 18.05.2019

CRAIG, D., 1996. Advantages of Simulation.

CRAWFORD, C., 2002. Artists and engineers as cats and dogs: implications for interactive storytelling. *Computers & Graphics,* **26**(1), pp. 13-20.

CSAPÓ, B., LÖRINCZ, A. and MOLNÁR, G., 2012. Innovative Assessment Technologies in Educational Games Designed for Young
Students .

DAMIEN, Päivätty 09.03.2017, Can You Localize a Small Indie Game for Free?. Saatavilla: https://www.leveluptranslation.com/single-post/can-you-localize-a-small-indie-game-for-free [Apr 25, 2019]. Viitattu 25.04.2019

ESKELINEN, M., 2005. *Pelit ja pelitutkimus
luovassa taloudessa.*

EPIC GAMES, Programming Guide. Saatavilla: https://docs.unrealengine.com/en-US/Programming/index.html. Viitattu 17.06.2019.

Façade. Saatavilla: https://www.playablstudios.com/facade. Viitattu 15.05.2019

FFISKE, T., 2018. *VR in 2019 - What is the Future?* .

FINDLAY, J., 2016. *Game-Based Learning vs. Gamification: Do You Know the Difference?* .

FINGAS, J., , Oculus predicts a VR future that includes ultra-thin headsets. Saatavilla: https://www.engadget.com/2018/09/26/oculus-predicts-ultra-thin-vr-headsets/ [May 22, 2019]. Viitattu 22.05.2019

Godot Docs – 3.1 branch — Godot Engine latest documentation. Saatavilla: https://docs.godotengine.org/en/3.1/. Viitattu 17.06.2019

Godot Engine - Our point'n'click framework is finally out!. Saatavilla: https://godotengine.org/article/our-point-click-framework-finally-out. Viitattu 18.05.2019

HOGUET, B., 2014. *What is interactive storytelling?* .

HONEYWOOD, R. and FUNG, J., 2011. *Best Practices for Game Localization.*

Introducing JSON. Saatavilla: https://www.json.org/. Viitattu 17.06.2019

Fungus: String table, Saatavilla: http://fungusdocs.snozbot.com/Lua_string_table.html. Viitattu 23.04.2019

Fungus: Controlling Fungus, Päivätty 16.06.2018. Saatavilla: http://fungusdocs.snozbot.com/Lua_controlling_fungus.html. Viitattu 23.05.2019

Fungus. Saatavilla: http://fungusgames.com/. Viitattu 29.04.2019

ISLAM, A., 2018. *The Future of Augmented Reality.*

KENDRA, C., Extrinsic vs. Intrinsic Motivation: What's the Difference?. Saatavilla: https://www.verywellmind.com/differences-between-extrinsic-and-intrinsic-motivation-2795384 [Jun 16, 2019].

KADLE, A., Päivätty 2010, 3 Game Mechanics To Include In Learning Games. Saatavilla: https://www.upsidelearning.com/blog/index.php/2010/12/15/3-game-mechanics-to-include-in-learning-games/. Viitattu 29.04.2019

KLEČINA, D., Päivätty 2019, MODULAR DESIGN in Educational Systems. Saatavilla: https://poweringkidslearning.com/modular-design-in-educational-systems/. Viitattu 21.05.2019

LARSSON, R., 2013. *Motivations in Sports and Fitness Gamification*, UMEÅ UNIVERSITET.

Learn to make 2D games with Defold. Saatavilla: https://www.defold.com/learn/. Viitattu 17.06.2019

LINDENBERGER, J., 2017. *8 Top Benefits of Training Simulations in the Workplace.*

NARAIN, A., 2016-last update, 6 Crucial Questions to Ask Before Choosing Your Game Engine. Saatavilla: https://blackshellmedia.com/2016/09/29/6-crucial-questions-ask-choosing-game-engine/. Viitattu 16.06.2019

MANAGEMENT ASSOCIATION, I.R., 2018. *Gamification in Education.* Hershey: Information Science Reference (Isr).

MightyFingers. Saatavilla: http://mightyfingers.com. Viitattu 17.06.2019

O'FLANAGAN, J., , Game Engine Analysis and Comparison. Saatavilla: https://www.gamesparks.com/blog/game-engine-analysis-and-comparison/. Viitattu 12.05.2019

PANDLEY, A.,Top 6 Benefits Of Gamification In eLearning, Päivätty 2015. Saatavilla: https://elearningindustry.com/top-6-benefits-of-gamification-in-elearning. Viitattu 09.05.2019

PAPPAS, C., Päivätty Elokuu 2018, 23 Effective Uses Of Gamification In Learning: Part 1. Saatavilla: https://elearningindustry.com/23-effective-uses-gamification-in-learning-part-1. Viitattu 21.05.2019

PELIMO | Pelillistäminen mahdollisuutena monikulttuurisuuskasvatuksessa. Saatavilla: https://pelimo.turkuamk.fi/. Viitattu 14.06.2019

PETROSKI, A., , Games vs. Simulations: When Simulations May Be a Better Approach. Saatavilla: https://www.td.org/magazines/td-magazine/games-vs-simulations-when-simulations-may-be-a-better-approach. Viitattu 20.05.2019

Polyglot Unity Päivätty 27.09.2016. Saatavilla: https://github.com. Viitattu 24.04.2019

r/unrealengine - WHY exactly is Paper2D not recommended?. Saatavilla: https://www.reddit.com/r/unrealengine/comments/7kwt2a/why_exactly_is_paper2d_not_recommended/. Viitattu 12.05.2019

PRIETO, R., MEDINA-MEDINA, N., MONTES, R., MORA, A. and VELA, F.L., 2017. *Designing educational games: Key elements and methodological approach.*

RIMON, G., 4 Benefits Of Learning Simulations. Saatavilla: https://elearningindustry.com/4-benefits-of-learning-simulations . Viitattu 17.06.2019

RPGTalk, Päivätty 2016. Saatavilla: http://www.seizestudios.com/developer/rpgtalk/. Viitattu 24.05.2019

S, P., *Difference Between Games and Simulations | Difference Between.*

SAS, What is Natural Language Processing?. Saatavilla: https://www.sas.com/en_us/insights/analytics/what-is-natural-language-processing-nlp.html. Viitattu 17.06.2019

SCHUETZ, J.,Comparing Game Engines: Unity vs Unreal vs Corona vs GameMaker. Saatavilla: https://www.pubnub.com/blog/comparing-game-engines-unity-unreal-corona-gamemaker/. Viitattu 12.05.2019

SCHÖBEL, S., JANSON, A., ERNST, S. and LEIMEISTER, J.M., 2017. *How to Gamify a Mobile Learning Application – A Modularization Approach.*

SHODHAN, S., 2017. *Intrinsic vs Extrinsic Motivation in Games.*

SICART, M., 2008. Defining Game Mechanics. *Game Studies,* **8**(2).

SKULT, P., 2019. *Haastattelu, Red Nettle Studio.*

Storyworlds. Saatavilla: http://convergenceishere.weebly.com/storyworlds.html. Viitattu 17.05.2019

SOL-TRIBE, , Visual Studio Toolkit Free. Saatavilla: https://assetstore.unity.com/packages/templates/visual-novel-toolkit-free-9416. Viitattu 29.04.2019

SONG, Q., HE, L. ja HI, X., 2012. To Improve the Interactivity of the History Educational Games with Digital Interactive Storytelling. *Physics Procedia,* **33**, pp. 1798-1802.

TACK, D., Serious Games And The Future Of Education. Saatavilla: https://www.forbes.com/sites/danieltack/2013/09/12/serious-games-and-the-future-of-education/. Viitattu 22.05.2019

TIOBE Index. Saatavilla: https://www.tiobe.com/tiobe-index/. Viitattu 12.05.2019

The Future of Educational Games within Augmented Reality. Saatavilla: https://www.ciklum.com/blog/future-educational-games-within-augmented-reality/. Viitattu 21.05.2019

UNITY TECHNOLOGIES. Game engines - how do they work?. Saatavilla: https://unity3d.com/what-is-a-game-engine. Viitattu 17.06.2019

UNITY TECHNOLOGIES, Unity - Manual: Creating and Using Scripts. Saatavilla: https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html. Viitattu 17.06.2019

UNITY RECHNOLOGIES, Unity - Manual: Unity User Manual (2019.1). Saatavilla: https://docs.unity3d.com/Manual/index.html. Viitattu 17.06.2019

Waterloon Yliopisto, Gamification and Game-Based Learning. Saatavilla: https://uwaterloo.ca/centre-for-teaching-excellence/teaching-resources/teaching-tips/educational-technologies/all/gamification-and-game-based-learning. Viitattu 22.04.2019

WEE, S. and CHOONG, W., 2019. Gamification: Predicting the effectiveness of variety game design elements to intrinsically motivate users' energy conservation behaviour. *Journal of Environmental Management,* **233**, pp. 97-106.

Y. SHAFRANOVICH, 2005, Common Format and MIME Type for Comma-Separated Values (CSV) Files. Saatavilla: https://www.ietf.org/rfc/rfc4180.txt. Viitattu 17.06.2019