Opinnäytetyö (AMK)

Tietotekniikka

Peliteknologia

2017

Mikko Pavén

VISUAALINEN TARINATYÖKALU VIDEOPELIEN KEHITTÄMISEEN

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka | Peliteknologia

2017 | 32

Ohjaaja: yliopettaja Mika Luimula

Mikko Pavén

VISUAALINEN TARINATYÖKALU VIDEOPELIEN KEHITTÄMISEEN

NordicEdu on turkulainen opetuspelejä asiakkaille kehittävä yritys. Monet NordicEdun projekteista ovat tarinapohjaisia pelejä, joissa pelaaja tekee valintoja ja vaikuttaa tarinan tapahtumiin. Pelien tarinat kirjoitetaan yhteistyössä tilaajan kanssa. Joskus tilaaja toimittaa NordicEdulle isoja osuuksia tarinasta valmiina, jolloin toimitettu materiaali täytyy siirtää käsityönä peliohjelman ymmärtämään muotoon.

Opinnäytetyössä kehitettiin sovellus, jolla NordicEdun työntekijät ja sen asiakkaat voivat helposti kirjoittaa haarautuvan tarinasisällön suoraan peliohjelman ymmärtämään muotoon. Sovelluksesta tehtiin helposti käytettävä ja kirjoitusprosessia parannettiin esittämällä tarinan rakenne visuaalisesti. Sovelluksesta tuli suunnitellun mukainen, mutta projektin aikana ilmeni useita mahdollisuuksia jatkokehitykselle.

Opinnäytetyössä myös tarkasteltiin erilaisia käytäntöjä keskustelujen esittämiseen tarinavetoisissa tietokonepeleissä ja tutustuttiin lyhyesti viiteen opinnäytetyön sovellusta muistuttavaan muiden kehittämään työkaluun interaktiivisen fiktion kirjoittamiseen.

ASIASANAT:

pelinkehitys, tarina, työkalu, visuaalinen ohjelmointi, interaktiivisuus

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information technology | Game technology

2017 | 32

Instructor: Principal Lecturer Mika Luimula

Mikko Pavén

GRAPHICAL STORY AUTHORING TOOL FOR VIDEO GAME DEVELOPMENT

NordicEdu is a Turku-based game development company specializing in educational games as a service. Many of NordicEdu's projects are story-based games where the player makes decisions can influence the story. Storylines for these games are co-written with the client. Occasionally a client delivers large chunks of the story, which then needs to be manually converted into a structure the game software can interpret.

In this thesis, an application was developed for NordicEdu and its clients to more easily write the story content directly into a usable format. The popular Unity game engine was used for the graphical user interface. The application is easy to use and improves the writing process using a flowchart representation of the story structure. Finally, a collection of features is presented to be implemented in the next version of this application.

Additionally, a look was taken into methods of presenting conversations in story-based computer games. Five interactive fiction tools with similarities to this thesis' application are also explored.

KEYWORDS:

video games, software development, interactivity

SISÄLTÖ

KUVAT

SANASTO

NPC	Mikä tahansa videopelissä esiintyvä hahmo, jota pelaaja ei ohjaa. (Non-Player Character)
Shader	Ohjelmakoodi, jolla näytönohjain piirtää jotain. Esineiden värit, pintamateriaalit ja valaistus riippuvat käytetyistä shader-ohjelmista.
Komentosarja	Ohjelmakoodi, joka tulkitaan ajon aikana eikä käännetä etukäteen. Joissain peleissä pelilogiikka on ohjelmoitu komentosarjoilla, mutta piirtäminen ja muut aikakriittiset toiminnot on käännetty konekieliseksi. (script)
Solmugraafi	Datan tai toiminnallisuuden esitystapa, jossa verkostoituja olioita kuvataan kaksi- tai kolmiulotteisina kappaleina (solmuina) ja niiden välisiä yhteyksiä kappaleita yhdistävinä viivoina tai säikeinä.
Seikkailupeli	Tarinapohjainen peligenre, jossa pelaaja ohjaa tarinan päähenkilöitä ja ratkaisee erilaisia pulmatilanteita.
Tekstiseikkailu	Seikkailupeli, jossa käyttöliittymä on pelkkää tekstiä. Pelin tapahtumat kuvaillaan pelaajalle ja pelaaja ohjaa hahmoaan tekstikomennoilla.
Visual novel	Erityisesti Japanissa suosittu tarinapelien genre, jossa yhdistetään tekstiä, kuvitusta, musiikkia ja ääninäyttelyä. Tyypillisesti visual novelien ainoa pelimekaniikka on valintojen tekeminen.
XML	Laajennettava merkintäkieli, jolla kirjoitetaan datan siirtämiseen ihmisen ja tietokoneen luettavia dokumentteja. (Extensible Markup Language)

1 JOHDANTO

Tässä opinnäytetyössä kehitetään NordicEdulle tarinankirjoitustyökalu, joka esittää tarinan rakenteen selkeästi ja tekee haarautuvien juonien kirjoittamisesta helppoa.

NordicEdu on Turussa toimiva yritys, joka kehittää hyöty- ja opetuspelejä. Vuodessa toteutetaan reilu kymmenen asiakasprojektia, joissa kaikissa asiakkaalla on pelistä selkeä visio. Projektin sisältö tulee usein suoraan asiakkaalta ja täytyy saada jotenkin lisättyä peliin. Sisältö vaihtelee laajasti monivalintakysymyksistä käsikirjoitukseen.

Sisällön siirtäminen asiakkaalta peliin on vaihteleva ja monivaiheinen prosessi. Pelkistetysti asiakkaalla on visio pelin sisällöstä mielessään ja se pitää muuntaa peliohjelmiston ymmärtämään muotoon. Esimerkiksi eräässä peliprojektissa asiakas on kirjoittanut kysymyksiä ja vastausvaihtoehtoja tekstidokumenttiin. Sitten NordicEdun työntekijä on siirtänyt dokumentin sisällön tietyllä tavalla muotoiltuun taulukkodokumenttiin, ja lopuksi peliohjelma tulkitsee kyseisen taulukkodokumentin projektia varten kehitetyllä ohjelmanosalla.

Vanhassa tuontijärjestelmässä on monia ongelmia. Taulukkodokumentin kirjoittaja saattaa tehdä muotoiluvirheitä. Haarautuvan käsikirjoituksen kulkua on vaikea seurata taulukosta. Formaatin muuttaminen ja erikoistilanteiden käsittely on hankalaa. Pienetkin muutokset vaativat työtä kaikissa prosessin vaiheissa. Sisältö pakataan osaksi peliä kääntämisvaiheessa eikä sitä voi muuttaa ilman kääntämisprosessin toistamista.

Tässä opinnäytetyössä kehitetään sovellus, joka pyrkii yksinkertaistamaan sisällöntuottamisen prosessia ja ratkaisemaan edellä kuvailtuja ongelmia. Tavoitteena on, että pienen ohjeistuksen avulla asiakas pystyy itse kirjoittamaan ja muotoilemaan sisällön uudella työkalulla. NordicEdun toiveesta työkalun käyttöliittymästä tehdään solmupohjainen ja kehitetään Unity-pelimoottorilla.

Teoreettisessa osuudessa tarkastellaan keskusteluita tietokonepeleissä sekä jo olemassa olevia kirjoitustyökaluja. Työn tutkimuskysymykset ovat seuraavat: millaisia ratkaisuja hahmojen välisten keskusteluiden esittämiseen peleissä käytetään? Mitä vahvuuksia tai ongelmia eri esitystavoissa on? Millaisia valmiita työkaluja pelinkehittäjille on tarjolla?

2 KESKUSTELUSYSTEEMIT

Monissa tarinavetoisissa tietokonepeleissä on kohtauksia, joissa tarinan hahmot vievät juonta eteenpäin vuoropuheluilla, aivan kuten kirjoissa tai elokuvissakin. Pelien tarinoissa on kuitenkin mahdollista, että keskusteluun liittyy myös pelaajan ohjailema hahmo. Joissain peleissä tämä ei eroa mitenkään muiden hahmojen välisistä keskusteluista, mutta jotkin pelit antavat pelaajan vaikuttaa vuoropuheluun.

Luonnollisen keskustelun aikaansaaminen tietokoneohjelman kanssa on toistaiseksi ratkaisematon ongelma. Pelien tapauksessa keskustelutekoälyn suunnitteleminen on erityisen vaikeaa. Keskusteluiden täytyy olla tarpeeksi uskottavia mutta myös kuljettaa tarinaa eteenpäin. Tietokoneen tulisi ikään kuin näytellä tarinan hahmoa pelaajalle. Tällainen sovellus ei ole vielä nykypäivää, joten peleissä täytyy tyytyä kompromissiratkaisuihin. Tällaisia keinoja simuloida keskusteluita videopeleissä kutsutaan keskustelusysteemeiksi.

Keskustelusysteemit voidaan karkeasti jakaa tekstinsyöttöä käyttäviin ja valikkopohjaisiin ratkaisuihin. Ne ovat toisistaan erilaisia tekniikan, pelisuunnittelun ja käyttäjäkokemuksen kannalta.

2.1 Tekstinsyöttöön perustuvat keskustelusysteemit

Tekstinsyöttöön perustuvissa ratkaisuissa pelaaja vaikuttaa pelin kulkuun kirjoittamalla komentoja tai repliikkejä avoimeen kenttään tai terminaaliin, jossa peliohjelma suoritetaan. Peliohjelma lukee syötteen ja yrittää tulkita sen joidenkin sääntöjen mukaisesti.

Koska keskustelun rajat ovat piilossa, tämän tyyppiset keskustelusysteemit voivat antaa pelaajalle vahvan vapauden tunteen. Pelihahmoille pystyy sanomaan mitä tahansa! Useimmilta pelaajilta illuusio kuitenkin murtuu nopeasti, kun kanssakeskustelija vastaa kaikesta tarinaan liittymättömästä saman en ymmärrä -vastauksen. (Roberts 2007.)

Keskustelun näkymättömät rajat voivat myös olla pelaajalle tuskastuttavia, jos tarinaa eteenpäin vievät avainsanat on piilotettu liian hyvin ja pelaajan on turvauduttava arvailuun. Ongelma on samankaltainen kuin klikkailuseikkailujen ”pikselinmetsästys”, eli tärkeät esineet eivät erotu tarpeeksi taustagrafiikasta. (Roberts 2007.)

2.1.1 Halpa tekoäly

Joissakin peleissä syötteen tulkinta perustuu enemmän tai vähemmän tarkkaan komentosyntaksiin. Toisissa peleissä pelaajan odotetaan käyttävän puheenvuorollaan luonnollista kieltä, ikään kuin hän kirjoittaisi toiselle ihmiselle (tai pelin hahmolle) jonkinlaisen pikaviestintäohjelman läpi.

Jälkimmäisen kaltaisista peleistä harvat kuitenkaan oikeasti pyrkivät tulkitsemaan aidosti luonnollista kieltä. Tyypillisesti koko tulkinta on silmänlumetta ja peli vain skannaa syötteestä tilanteen kannalta oleelliset avainsanat tai tarkistaa vastaako se jotakin säännöllistä lauseketta. Tällaista yksinkertaisia, mutta älykkääksi naamioituja ratkaisuja kutsutaan halvaksi tekoälyksi. (Roberts 2007.)

Tämänlaiset systeemit eivät ole kovin vakuuttavia, koska ne rikkoutuvat hyvin helposti jopa tarkoitetussa käytössä. Systeemin heikkoutena on ainakin homonyymit, taivutusmuodot ja lauseyhteydet. Peli ei voi esimerkiksi ymmärtää eroa avainsanan ”hyppää”, ja pelaajan syötteen ”älä hyppää” välillä.

Koska systeemi ei ole vakuuttava, harva pelaaja vaivaantuu leikkimään mukana. Kokeneet pelaajat hylkäävät luonnollisen kielen lauseet ja siirtyvät kirjoittamaan pelkästään avainsanoja. Onhan se kuitenkin tehokkain tapa käyttää kyseisen kaltaista käyttöliittymää. Vastapainoksi vähemmän tietokoneita ymmärtävä pelaaja saattaa hämmentyä, kun peli reagoi väärin selkeisiin lauseisiin. (Roberts 2007.)

Halpa tekoäly -ratkaisua käyttää esimerkiksi Douglas Adamsin surullisenkuuluisa *Starship Titanic* vuodelta 1998. Yleisen pulmanratkonnan lisäksi pelaaja keskustelee robottimiehistön kanssa kirjoittamalla luonnollisen kielen viestejä Personal Electronic Thing -laitteeseen. (Starship Titanic 1998.)

Myös vuosina 1964–66 Massachusetts Institute of Technologyssa kehitetty ELIZA käyttää yksinkertaista avainsanoihin perustuvaa tulkintamenetelmää. ELIZA on luonnollista kieltä niin sanotusti ymmärtävä terapeuttibotti, joka kehitettiin demonstraatioksi, kuinka pinnallista ihmisen ja tietokoneen välinen keskustelu on. ELIZA rikkoutuu helposti, mutta sopivalla käyttäjäsyötteellä se voi olla yllättävän vakuuttava. Anekdootin mukaan monet ELIZAa testanneet MIT:n vähemmän tekniset henkilökunnan jäsenet todella uskoivat ohjelman älykkyyteen (Weizenbaum 1976). ELIZA ei ole peli,

mutta se vaikutti moneen varhaiseen tietokonepeliin tuolloin omaperäisellä käyttöliittymällään.

2.1.2 Luonnollisen kielen prosessointi

Muutamat kokeelliset pelit ovat yrittäneet käyttää keskustelujen toteuttamiseen ja pelin ohjaamiseen monimutkaisempaa luonnollisen kielen prosessointia. Kalliimmat tuotannot ovat pysyneet kuitenkin etäällä, sillä siihen käytettävät teknologiat ovat monessa suhteessa jokseenkin epävakuuttavia.

Façade on 2005 ilmestynyt tekoälyyn nojaava tarinapeli, jossa pelaaja on kutsuttu cocktail-illalliselle ystäviensä Tripin ja Gracen asuntoon. Tunnelma on kiristynyt, kun pelaaja saapuu kutsuille kesken pariskunnan riidan. Pelissä ei ole varsinaista tavoitetta, vaan pelaaja pudotetaan sosiaaliseen hiekkalaatikkoon keskustelemaan robottien kanssa. Pelissä on seitsemän loppua, joista yksi on niin sanottu hyvä loppuratkaisu. (Façade, 2005.)

Tarinapelien käyttötarkoitukseen sopiva luonnollisen kielen tulkinta vaatii algoritmien lisäksi paljon dataa, sillä tekoälykeskustelijoiden täytyy käyttäytyä käsikirjoittajan tarkoituksen mukaisesti. Tekoälyä keskusteluiden simuloimiseen käyttävät pelit, kuten Façade, ovatkin läheisemmin sukua keskusteluboteille kuin interaktiiviselle fiktiolle.

Artificial Intelligence Markup Language (AIML) on XML-pohjainen merkkauskieli keskustelubottien käyttäytymisen määrittelemiseen. AIML perustuu syöte-tulostepareihin tai sääntöihin, ja niiden linkittämiseen rekursiiviseksi valintapuuksi. Yhdessä säännössä on villikortilla varustettu syötekenttä, jota käytetään säännön valitsemiseen, sekä tulostekenttä, joka sisältää joko valmiin vastauksen tai uuden syötteen botille. Rekursiota käytetään monimutkaisempien syötteiden pureskeluun helpommin vastattavaan muotoon. (Wilcox 2011.)

AIML-esimerkki

Bruce Wilcoxin antamassa esimerkissä käyttäjä antaa AIML-pohjaiselle keskustelubotille syötteen

> Can you please tell me what Linux is right now?

Botti löytää ensimmäiseksi säännön, jossa on kaavana

* RIGHT NOW

Säännön mukaisesti botti antaa itselleen uuden, muokatun syötteen:

can you please tell me what Linux is

Rekursio jatkuu:

CAN YOU PLEASE * → please tell me what Linux is

PLEASE TELL ME WHAT * → tell me what Linux is

TELL ME WHAT * IS → what is Linux

Lopulta syöte on niin yksiselitteinen, että AIML-datan joukosta löytyy lopullinen, käyttäjälle tarkoitettu vastaus:

Linux is an operating system.

(Wilcox 2011.)

2.1.3 Komentopohjaiset tulkit

Tekstinsyöttöön perustuvista ratkaisuista suositumpi on kuitenkin komentosyntaksin käyttäminen. Luonnollisen tekstin sijasta pelaaja antaa tarkasti muotoiltuja käskyjä. Tämä tyyli on luonnollinen jatke tekstiseikkailupelien käyttöliittymälle, jossa pelaaja suorittaa kaikki muutkin toiminnot antamalla selkeitä komentoja.

Komentojen rakenne on suoraan peritty tekstiseikkailuista. Ne näyttävät yksinkertaisilta englanninkielisiltä lauseilta, esimerkiksi ”JACK, PASS ME THE KETCHUP”. Keskusteluiden rakenne on tyypillisesti, että pelaaja aloittaa valitsemalla jonkin puheenaiheen ja sitten keskustelukumppani vastaa jotain aiheesta.

Komentopohjaisille keskusteluille on ominaista, että keskustelut ovat tilattomia. Vastaukset samaan kysymykseen voivat toki muuttua, mutta keskustelukumppani ei välitä, jos pelaaja tekee jotain muuta kahden sananvaihdon välissä. (Roberts 2007.)

Tätä tyyliä edustavia pelejä on esimerkiksi koko Infocomin tuotanto. Infocom oli 80-luvulla merkittävä interaktiivisen fiktion tuottaja, joka kehitti muun muassa *Zork*-sarjan. Infocomin tuotantoon kuuluu myös tekstiseikkailusovitus Douglas Adamsin *The Hitchhiker’s Guide to the Galaxy*sta, jonka kehittämiseen myös kirjailija itse osallistui.

Varsinkin *The Hitchhiker's Guide* on klassinen esimerkki interaktiivisesta fiktiosta. (Montfort 2007.)

2.2 Valikkopohjaiset keskustelusysteemit

Valikkoon perustuvissa keskusteluissa pelaaja valitsee vuorosanansa pelin antamasta listasta vaihtoehtoja (Kuva 1). Sitten pelaajan kanssa keskusteleva hahmo vastaa ja pelaaja saa taas listan repliikkejä. Silmukka jatkuu, kunnes pelaaja päättää lopettaa keskustelun, keskustelunaiheet loppuvat listasta tai jokin erikoistapaus, kuten juonta edistävä valinta päättää keskustelun. Valikkopohjaiset keskustelut ovat luonnollinen ratkaisu graafisissa peleissä, mutta niitä esiintyy toisinaan myös tekstiseikkailuissa.

Pelaajalle tarjottavat vaihtoehdot voivat olla listassa kokonaisia repliikkejä tai pelkästään jonkinlaisia repliikkiä edustavia symboleita, kuten tiivistelmiä tai repliikin sävyä kuvaavia ikoneita.

Kuva 1. Pelaajan valitsee repliikkinsä pelissä *The Secret of Monkey Island.*

Nykyinen trendi on käyttää entistä enemmän ääninäyttelyä valikkopohjaisten systeemien kanssa. Ääninäyttelyn lisääminen saattaa pelaajalle merkitä, että hän saa kuulla jatkuvasti ääneen juuri hetkeä ennen lukemansa repliikit. Toiston vähentämiseksi ja varsinkin pelikonsolien ohjaimien tukemiseksi ison budjetin pelit kuten *Mass Effect* ja *Fallout 4* näyttävät pelaajalle hyvin tiivistetyn, joskus vain yhden sanan symbolin repliikeistä.

Vaihtoehtojen esittämisessä symboleilla on aina väärinymmärryksen riski. Viime vuosilta erikoinen tapaus on *The Wolf Among Us*. Pelissä on keskustelu, jossa voi yhdessä kohtaa aloittaa baaritappelun valitsemalla ”glass him”. Keskustelun sävy ei viittaa minkäänlaiseen vihamielisyyteen, ja jotkut englantia vieraana kielenä puhuvat pelaajat luulivat fraasin merkitsevän juoman tarjoamista. Virheen kuitenkin huomaa nopeasti, kun pelaajahahmo Bigby yhtäkkiä särkee juomalasin keskustelukaverinsa päähän. (The Wolf Among Us 2013.)

Graafisissa, animoiduissa peleissä pelaajan miettimistaukojen aiheuttamat tauot keskusteluissa saattavat antaa oudon vaikutelman. Monissa peleissä keskustelu ei yksinkertaisesti vain jatku ennen kuin pelaaja on tehnyt valintansa. Jotkin pelit pyrkivät korjaamaan tämän pienen omituisuuden antamalla pelaajalle vain rajallisesti aikaa vastata. Käyttöliittymän keventämisen lisäksi vaihtoehtojen tiivistäminen tai symbolien taakse piilottaminen voi auttaa myös nopeuttamaan pelaajan vastausaikaa. (Ellison 2008.)

Esimerkiksi *The Walking Dead* lisää vuoropuhelun nopeuttamiseksi pelaajan puheenvuoroihin aikarajan. Rajan ylittäminen ei kuitenkaan ole varsinaisesti epäonnistuminen, sillä pelaaja voi myös aikarajan sisällä valita vaikenemisen. Tällä valinnalla on yleensä sama vaikutus kuin aikarajan ylittämisellä. (The Walking Dead 2012.)

Aikarajan sijasta *Mass Effect*-pelit näyttävät pelaajalle vastausvalikon jo ennen kuin muut hahmot ovat päättäneet repliikkinsä. Keskustelu jatkuu tauotta lähes luonnollisesti, jos pelaaja on tarpeeksi nopea valitsemaan. (Mass Effect 2007.)

Pelin kirjoittajalle valikkopohjaiset keskustelut tuovat vaikutusvaltaa tekstinsyöttömenetelmiin verrattuna. Ensinnäkin pelaajan vaihtoehtojen lukumäärä kaikissa tilanteissa rajoittuu merkittävästi. Nyt kirjoittajan tarvitsee enää pahimmassakin tapauksessa käsitellä vain kourallinen pelaajan vastausvaihtoehtoja, jotka kirjoittaja saa itse päättää. Toinen etu kirjoittajalle on, että pelaajan repliikkejä voi tyylitellä ja ne voivat

olla juuri niin yksityiskohtaisia tai hienovaraisia kuin halutaan. Esimerkiksi jossakin tilanteessa pelaajan vaihtoehdot saattaisivat erota toisistaan vain sävyltään. (Roberts 2007.)

Valikkopohjaiset keskustelusysteemit ovat tyypillisesti tilakoneeseen sidottuja. Pelaajalle voidaan näyttää eri vastausvaihtoehdot muun edellisestä repliikistä tai muun pelin tilanteesta riippuen. Vaihtoehdot voi tehdä riippuvaisiksi esimerkiksi säästä ja kanssakeskustelijan mielialasta! Keskustelujen mallintaminen puurakenteena mahdollistaa myös punaisen langan kirjoittamisen keskusteluihin.

Liian rönsyilevä keskustelupuu saattaa olla haitallinen pelin immersiivisyyden kannalta. Keskustelutilanteesta tuleekin sokkelo, joka pitää koluta läpi. Tämä on harmillista varsinkin, jos kyseiseen keskustelupuuhun on piilotettu jotain tarinan etenemisen kannalta välttämätöntä tietoa. (Roberts 2007.)

Tekstinsyöttömenetelmien sana-arvailuongelma eliminoituu valikkopohjaisella systeemillä täysin, mutta samalla keskustelun rajat tulevat selkeästi nähtäviksi. Lienee makuasia, onko keskusteluiden selkeä rajallisuus hyvä vai huono asia.

2.3 Interaktiivisen fiktion työkaluja

Tässä työssä kehitetään työkalu interaktiivisen fiktion tai tarinapelien kirjoittamiseen, joten tarkastellaan seuraavaksi paria samankaltaista työkalua.

2.3.1 Twine

Twine on avoimen lähdekoodin tarinankirjoitustyökalu, joka esittää tarinan rakenteen solmukuviona. Alun perin Twine oli Python-pohjainen työpöytäsovellus Windowsille ja Macille, mutta versio 2 on rakennettu kokonaan uudelleen Javascriptilla. Sitä voi käyttää selaimella Twinen verkkosivuilla tai ilman verkkoyhteyttä, jos sovelluksen lataa omalle tietokoneelle.

Twinen käyttöliittymä on hyvin helppokäyttöinen, eikä peruskäyttö vaadi laisinkaan ohjelmointiosaamista. Käytännössä kuka tahansa voi kirjoittaa sillä interaktiivista fiktiota tai muunlaisia teoksia. Kalliit studiopelit jatkavat kallistumistaan, mutta myös pienemmän

budjetin pelejä on yhä enemmän. Onkin mielenkiintoista nähdä, millaisia töitä Twinen kaltainen äärimmäinen pelinkehittämisen demokratisointi saa aikaan. (Petit 2013.)

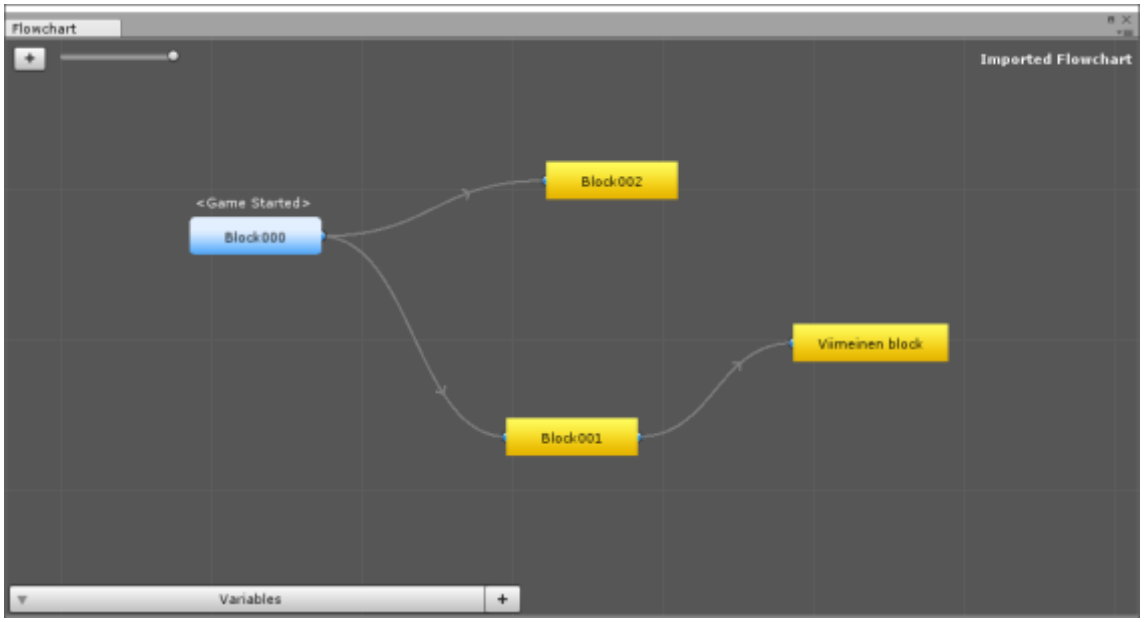
Twine-teosten julkaisumuoto on itsenäinen html-tiedosto, joka sisältää kaiken tarinadatan, tarinan esittämiseen tarvittavan Javascript-pohjaisen moottorin sekä muotoilumääritykset. Formaatin ansiosta teoksia on todella helppo levittää ja pelata. Tarinadataa lukuun ottamatta tiedosto on pienimmillään noin 200 kt. Tiedosto kasvaa nopeasti, jos teokseen lisätään uutta koodia tai multimediaa, mutta vaihtoehtoisella yhteisön luomalla pelimoottorilla on mahdollista lisätä esilataustoimintoja.

2.3.2 Fungus

Fungus on avoimen lähdekoodin Unity-liitännäinen, jota ylläpitää irlantilainen pelifirma **Snozbot**. Fungus lisää Unityyn komponentteja, joiden avulla voi helposti oletusasetuksilla esittää visual novel -tyylisiä monivalintakeskusteluja. Fungus laajentaa myös Unityn editoria lisäämällä visuaalisen tarinankirjoitustyökalun (kuva 2).

Fungus on melko helppokäyttöinen, mutta vaatii Unityn editoriin tutustumista. Liitännäisen mukana tulevilla tiedostoilla on mahdollista luoda visual novel -tyylisiä pelejä hyvin vähäisellä Unity-tuntemuksella. Avoimen lähdekoodin ansiosta edistyneemmät kehittäjät voivat kuitenkin muokata ja laajentaa Fungusta täysin omaan peliinsä sopivaksi.

Fungus tarjoaa myös monia helppokäyttötoimintoja monimutkaisten keskustelujärjestelmiin liittyvien käyttöliittymäominaisuuksien kehittämiseen. Liitännäinen tukee täysin Unityn uudempaa UI-järjestelmää, joka lanseerattiin versiossa 4.6. Fungus myös sisältää yleisesti Unityn kanssa toimivan Lua-kielen tulkin. Sitä voi käyttää monimutkaisten tarinaskriptien luomiseen koskematta varsinaiseen pelikoodiin.



Kuva 2. Funguksen vuokaavionäkymä.

2.3.3 Ren'Py

Ren'Py on Python-pohjainen pelimoottori visual novelien kehittämiseen. Sekin perustuu avoimeen lähdekoodiin ja toimii Windowsin lisäksi Macilla, Linuxilla, Androidilla ja iOS:llä. Vaikka Ren'Py ei sisälläkään mitään graafista editoria, tuodaan se tässä esille sen käyttämän skriptauskielen vuoksi.

Keskustelusisällöt Ren'Pyä varten kirjoitetaan erityisellä moottoria varten kehitetyllä käsikirjoituskielellä. Kieli on kehitetty helpottamaan sisällöntuottajien työtä, siis samasta syystä kuin tämän työn sovellus.

Komentosarjakielen ansiosta käsikirjoittajien ei tarvitse opiskella Pythonia tai muita varsinaisia ohjelmointikieliä, mutta kaikki keskusteluihin liittyvä data kirjoitetaan suoraan pelimoottorin ymmärtämään muotoon. Vuorosanojen lisäksi komentosarjassa voi määritellä kohtauksen siirtymät, esiintyvät hahmot sekä hahmojen eleet ja ilmeet. (Ren'Py 2017.)

Ren'Pyn kehittäjät ovat valinneet komentosarjapohjaisen ratkaisun sisällöntuottamiseen, koska se ei sido kehittäjiä mihinkään tiettyyn editoriin. Käsikirjoittaja voi työskennellä millä tahansa normaalilla tekstieditorilla. Ren'Pyn kehittäjien mukaan sisällöntuottaminen on myös tehokkaampaa, kun hiiripohjaiset käyttöliittymät otetaan pois laskelmista ja

kaikki työskentely tapahtuu näppäimistöllä (Ren'Py 2017). Tekstipohjaisessa ratkaisussa tietysti menetetään graafisen esityksen mahdollistama rakenteen selkeys.

2.3.4 Dialog Editor 2.0

Dialog Editor on Game Maker -pelimoottorilla kehitetty graafinen tarinankirjoitustyökalu, jonka perusideat ovat hyvin samanlaiset kuin Funguksen ja tämän työn sovelluksen. Se on kaupallinen tuote, joka on saatavilla Game Makerin kehittäjän **Yoyo Gamesin** kauppapaikalta 4,99 dollarin hintaan (Yoyo Games 2017).

Dialog Editorin käyttöliittymä perustuu samankaltaiseen solmuvisualisointiin kuin Fungus tai tämän työn sovellus. Se on hyvin suoraviivainen ja helppo käyttää Fungukseen verrattuna. Dialog Editoria pystyy käyttämään myös erikseen Game Makerin editorista. Editori tukee myös joitain mukavuuksia kuten hahmojen asettamista repliikeihin ja solmujen värikoodaamista.

Valitettavasti Dialog Editorissa on jokin paha suunnitteluvirhe, jonka seurauksena työkalun käyttöliittymä hidastuu ja muuttuu käyttökelvottomaksi, kun keskustelupuu kasvaa. Hidastelu on ongelma varsinkin, koska työkalu ei tunnista kaikkia näppäinpainalluksia, jolloin kirjoittaessa tekstistä jää puuttumaan merkkejä. Käytännössä ainoa tapa käyttää Dialog Editoria onkin kirjoittaa kaikki teksti jollakin erillisellä tekstieditorilla ja sitten kopioida ja liittää se solmuihin.

2.3.5 Chat Mapper

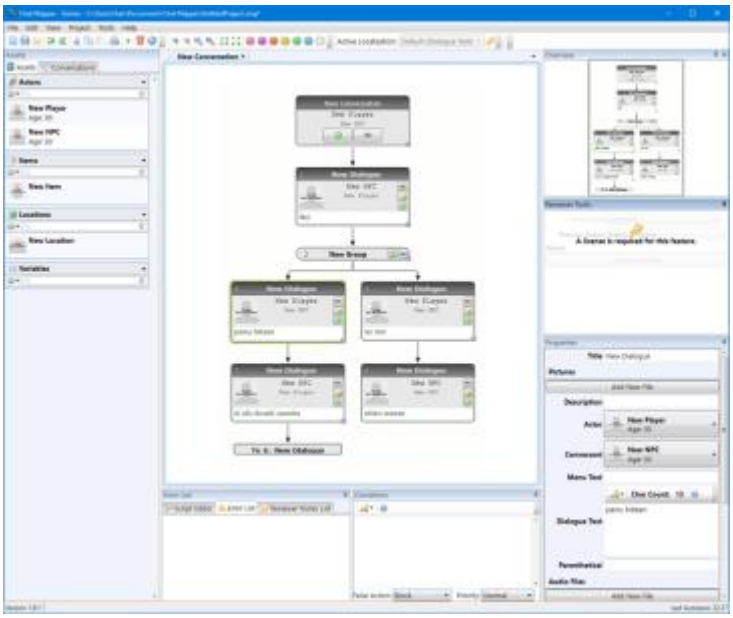
Chat Mapper on kaupallinen haarautuvien keskusteluiden ja käsikirjoitusten kirjoittamiseen tarkoitettu työkalu. Se on saatavilla vain Windowsille ja lisenssi maksaa 495-1395 dollaria riippuen käyttäjien määrästä ja halutuista lisäpalikoista. Kalliimman tiimilisenssin mukana tulee valmis keskustelujärjestelmä Unitylle. (Chat Mapper 2017b.)

Chat Mapperissa on vakuuttava määrä toimintoja ja kenttiä, joiden ansiosta se varmasti sopii monenlaisten sovellusten sisällöntuottamiseen. Esimerkiksi hahmoja, esineitä, paikkoja ja muuttujia varten voi luoda projektikohtaisia mallipohjia, jolloin kaikilla olioilla on varmasti tarvittavat kentät. Sovelluksen mukana tulee valmiiksi mallipohjat muun muassa yleisluontoiseen peliprojektiin ja lääketieteelliseen projektiin. Jälkimmäisessä esimerkiksi on hahmoilla kentät iälle, veriryhmälle ja oireille. Sovelluksessa on myös

kätevä keskustelusimulaattori testaamista varten ja valmiit tuotokset voi viedä sovelluksesta monissa data-, teksti-, kuva- ja taulukkoformaateissa. (Chat Mapper 2017a.)

Chat Mapperissa onkin niin paljon toimintoja, että monet niistä jäävät todennäköisesti käyttämättä pienemmissä projekteissa. Näille projekteille myös sovelluksen hinta saattaa olla hieman korkea. Kehittäjä mainostaakin myös indie-lisenssiä kehittäjille, joiden vuosittainen liikevaihto on alle 50 000 dollaria.

Sovelluksen käyttöliittymä on toimiva, mutta hieman kankea. Esimerkiksi keskustelupuun solmuja ei pysty järjestelemään raahaamalla, ja visualisointia ei voi suurentaa hiiren rullalla. Tuntuu myös kyseenalaiselta muotoiluvalinnalta esittää puurakenne pystysuunnassa ylhäältä alas, kun vaakasuuntainen esitys täyttäisi leveän tietokonemonitorin tehokkaammin (Kuva 3).



Kuva 3. Chat Mapperin visualisointi esittää puurakenteen pystysuunnassa.

3 SOVELLUS

Tämän työn sovellus on graafinen tarinankirjoitustyökalu. Työkalu on suunniteltu käytettäväksi Unity-pelimoottorin kanssa. Työkalu ei ole täysin riippuvainen Unitysta, sen voisi siirtää jollekin toiselle pohjalle tai kokonaan itsenäiseksi tarvittaessa. Tässä työssä työkalu päädyttiin kuitenkin rakentamaan Unitylla, jotta se olisi helpompi integroida NordicEdun työskentelytapoihin. Työkalua on tarkoitus käyttää NordicEdulla sisäisesti, mutta myös jakaa asiakkaille sisällön tuottamista varten.

NordicEdu tekee asiakastyönä erilaisia hyötypelejä, joissa on toisinaan interaktiivinen haarautuva tarina. Pelit suunnitellaan yhdessä NordicEdun kanssa, mutta tyypillisesti asiakas tuottaa ja toimittaa lopullisen tekstisisällön peliin. Asiakkaiden tuottama sisältö on aiemmin toimitettu NordicEdulle esimerkiksi tekstiasiakirjana tai taulukkolaskentaohjelman tiedostona. Tässä muodossa interaktiivinen tarina on kuitenkin jaettava pieniin osiin ja esitettävä usein epälineaarisessa järjestyksessä. Nämä käsikirjoitukset muistuttavat 80- ja 90-lukujen pelikirjoja (*Choose Your Own Adventure, Fighting Fantasy*).

Pelikirjoissa tarina keskeytyy välillä ja lukijaa pyydetään tekemään jonkin hahmon puolesta valinta. Valinta tehdään siirtymällä vaihtoehtoa vastaavalle sivulle, josta tarina sitten jatkuu halutulla tavalla.

Palasiin jaettuna haarautuvan tarinan rakennetta on vaikea hahmottaa, ja se on myös tarpeettoman työlästä ottaa käyttöön. Asiakkaan toimittama sisältö täytyy usein kopioida manuaalisesti varsinaiseen peliohjelmaan. Kopioimistyön joutuu käytännössä aina tekemään ohjelmoija. Työkalun avulla tämä vaihe poistuu työnkulusta, koska sisällöntuottaja on kirjoittanut tarinan suoraan pelimoottorilla helposti käsiteltävään muotoon.

Sovelluksen graafinen käyttöliittymä perustuu puurakenteeseen, joka on hyvin luonnollinen metafora haarautuvan tarinan esittämiseen. Tarinanpalat kirjoitetaan tarralappuja muistuttaviin suorakulmioihin, jotka toimivat puurakenteen solmuina. Tarinanpalat yhdistetään toisiinsa siimoilla, joita kirjoittaja raahaa solmusta toiseen. Erilaisilla solmutyypeillä voidaan esittää erikoistiloja, kuten tarinan alun ja valintatilanteet.

Jonkin valmiin työkalun sijasta NordicEdulla päätettiin kehittää oma sovellus muutamasta syystä. Ensimmäinen syy on riippumattomuus. Omaa työkalua käyttäessä

voi luottaa siihen, että sovellus ei muutu päivityksen seurauksena epäyhteensopivaksi kesken projektin. Lisäksi lähdekoodi ja toimintaperiaate ovat omassa työkalussa tuttuja, ja korjauksia ja muutoksia ei tarvitse odottaa kolmannelta osapuolelta.

Toinen syy on asiakasyhteistyö. Sovelluksen yksi käyttötarkoitus on antaa NordicEdun ulkopuolisten yhteistyökumppanien käyttää työkalua ja tuottaa projekteihin sopivaa sisältöä. Lisensoidulla työkalulla tätä ei sallittaisi. Omalla työkalulla varmistetaan, että käyttöliittymä on selkeä ja myös maallikolle ymmärrettävä.

Kolmas syy on sovelluksen mukauttaminen. NordicEdu pystyy laajentamaan työkalua tai integroimaan sen osaksi jotain kokonaisuutta. Sovelluksesta voi myös karsia kaikki turhat ominaisuudet ja keskittyä vain NordicEdun ja sen asiakkaiden tarpeisiin.

3.1 Toimintaperiaate

Ohjelma toimii niin, että Unityssa on GUI-systeemi, jonka metodeita täytyy kutsua tietyn nimisen metodin sisällä tietyn luokan perivässä oliossa, jotta se piirtää käyttöliittymäelementtejä (Unity 2017a). Sovelluksessa on yksi keskeinen MonoBehaviourin perivä luokka, joka toteuttaa tuon GUI-metodin ja pitää listaa kaikista solmuista. Tämän lisäksi solmut pitävät kirjaa kaikista yhteysobjekteistaan muihin solmuihin. Kaikilla erilaisilla solmuilla on omat piirtofunktionsa, joita MonoBehaviourin perivä ohjelman keskusolio kutsuu joka ruudunpäivityksellä solmujen piirtämiseksi.

Kaikki sovelluksen syötteenotto, kuten klikkaukset ja raahaamiset, tulevat Unityn GUI-systeemin läpi. Lähes kaikkialla syötteet luetaan sellaisenaan, mutta solmujen piirtämiseen käytettiin Unityn sisäänrakennettua GuiWindow-oikotietä. Tämän ominaisuuden käyttäminen säästi hieman aikaa, mutta myös hieman hankaloitti kehitystyötä myöhemmin. GuiWindow on helppokäyttöiseksi suunniteltu tapa luoda raahattavia ikkunoita Unityn vanhassa GUI-järjestelmässä (Unity 2017b). Sen yksinkertaiseksi tarkoitettu käyttöliittymä kuitenkin hankaloitti yhdessä vaiheessa muun sovelluksen kehitystyötä, kun GuiWindow'lla luodut ikkunat hukkasivat tärkeitä tapahtumia.

Unitya ei ole sellaisenaan tarkoitettu normaalien työpöytäsovellusten rakentamiseen. Tyypillisesti myös tätä sovellusta vastaavat työkalut rakennettaisiin laajennuksiksi Unityn editoriin. Sovellus on tehty Unitylla ajonaikana käytettäväksi, jotta sitä voisi käyttää ilman Unityn työkalujen asentamista, jotta sovelluksella luodun datan pystyisi helposti

siirtämään Unityyn ja jotta mahdolliset jatkokehityssuunnitelmat voivat käyttää hyväksi pelimoottorin ominaisuuksia. Nykytilassa sovelluksen voisi muokata toimimaan jollain tyypillisemmällä käyttöliittymäratkaisulla tai laajennukseksi Unityn kehitysympäristöön.

3.2 Toiminnollisuus

Sovelluksen päänäkymä on tyhjä tila, johon tarinasolmuja sijoitellaan. Solmuja yhdistämällä voidaan luoda puurakenteita, jotka mallintavat yksinkertaista haarautuvaa tarinaa tai keskustelua (kuva 4). Solmuja on neljä erilaista. Kolme niistä vaikuttaa puun rakenteeseen ja neljäs ainoastaan säilyttää tietoa.

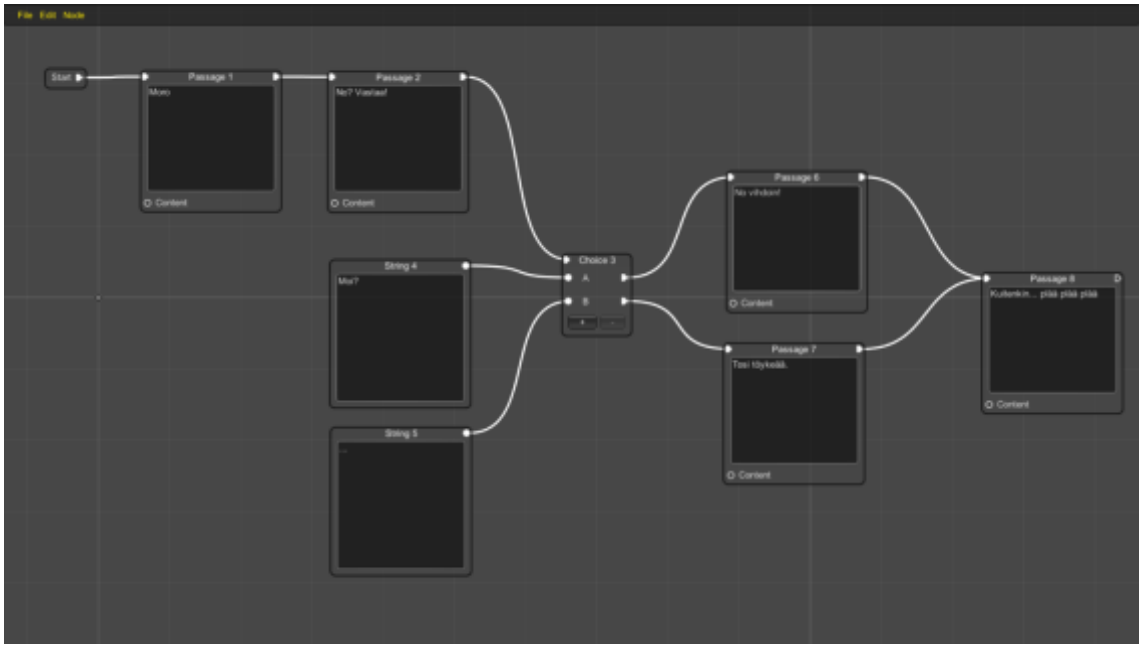
Aloitussolmu on ensimmäinen rakennesolmu. Sen tehtävä on kertoa missä puurakenteen juuri on. Se sisältää viittauksen kappale- tai valintasolmuun, josta tarina alkaa.

Merkkijonosolmu säilyttää merkkijonoa, johon yksi tai useampi kappale- tai valintasolmuja voi viitata.

Kappalesolmu sisältää yhden tekstikappaleen verran sisältöä ja viittauksen järjestyksessä seuraavaan solmuun. Se on toinen kolmesta rakennesolmusta. Seuraava solmu voi olla kappale- tai valintasolmu. Kappalesolmu on tarkoitus piirtää peleissä yksittäisenä tekstiruutuna tai repliikkinä. Kappalesolmun sisällön voi määrittää kirjoittamalla tekstiä suoraan kappalesolmun omaan tekstilaatikkoon tai yhdistämällä kappalesolmun haluttuun merkkijonosolmuun.

Viimeinen rakennesolmu on valintasolmu. Se tekee puurakenteesta puun eikä jonoa. Valintasolmu säilyttää listaa merkkijono-rakennesolmupareista, jotka kuvaavat pelaajalle tarjottavia vaihtoehtoja.

Pelin sisällä tämän voi esittää valintaa edeltävänä repliikkinä (kappalesolmu) ja listana valintasolmun vaihtoehdoista. Pelaaja valitsee tarinahaaran esimerkiksi klikkaamalla haluamaansa vaihtoehtoa listasta.



Kuva 4. Yleiskuva sovelluksen käyttöliittymästä.

Käyttöliittymän yläreunaan on sijoitettu monista käyttöjärjestelmistä tutun ikkunaparadigman mukaisia laajentuvia valikoita. Täältä löytyvät tallennus- ja avaustoiminnot sekä uusien solmujen luominen.

Työkalupalkki on sovelluksessa käyttömukavuuden vuoksi. Se on monille käyttäjille tuttu tapa käyttää tietokoneohjelmia.

3.3 Rakenne

Sovelluksen keskiössä on luokka *GuiRenderer*, joka lähes ainoana osana perii Unityn *MonoBehaviour*-luokan. *MonoBehaviour* on Unityn tarjoama pohja kaikille olioille, joiden tilaa pelimoottorin täytyy päivittää ennen jokaista ruudunpäivitystä. Siihen kuuluu metodi *OnGUI*, jota kutsutaan aina kun Unity päivittää käyttöliittymäkomponenttinsa tilaa. (Unity 2017c)

OnGUI on tyhjä metodi, joka täytyy tarvittaessa määrittää *MonoBehaviourin* perivässä luokassa. Sen sisällä piirretään graafisia käyttöliittymäkomponentteja ja käsitellään kaikki käyttöliittymän havaitsemat syötteet. (Unity 2017a)

Sovelluksen *GuiRenderer* pitää listaa kaikista olemassa olevista solmuolioista. Solmuoliot käydään läpi ennen jokaista ruudunpäivitystä *OnGUI*:n aikana ja piirretään niiden omilla piirtofunktioilla.

Erilaiset solmuoliot periytyvät kaikki *NodeWindow*-luokasta, jotta niitä voidaan käsitellä sekaisin kokoelmissa. *NodeWindow*'ssa toteutetaan monia kaikille solmuolioille tärkeitä metodeja sekä määritellään joitain signatuureja varsinaisten solmuolioiden toteutettavaksi.

Solmuoliot yhdistetään toisiinsa *Socket*-olioilla. Solmut omistavat *Socket*teja, jotka voivat yhdistää toisiinsa, jos ne ovat yhteensopivia. Solmuolioita ei yhdistetä suoraan toisiinsa, koska *Socket*it ovat helpommin yleistettävä ratkaisu. Kahden *Socket*in välistä yhteensopivuutta on yksinkertaisempi testata kuin yrittää kehittää sääntökirjaa kaikkien erilaisten solmuolioiden erilaisista liitoskohdista. Tämän sovelluksen *Socket*-nimisiä olioita ei pidä sekoittaa verkkoliikennettä käyttävien ohjelmien socket-tekniikoihin.

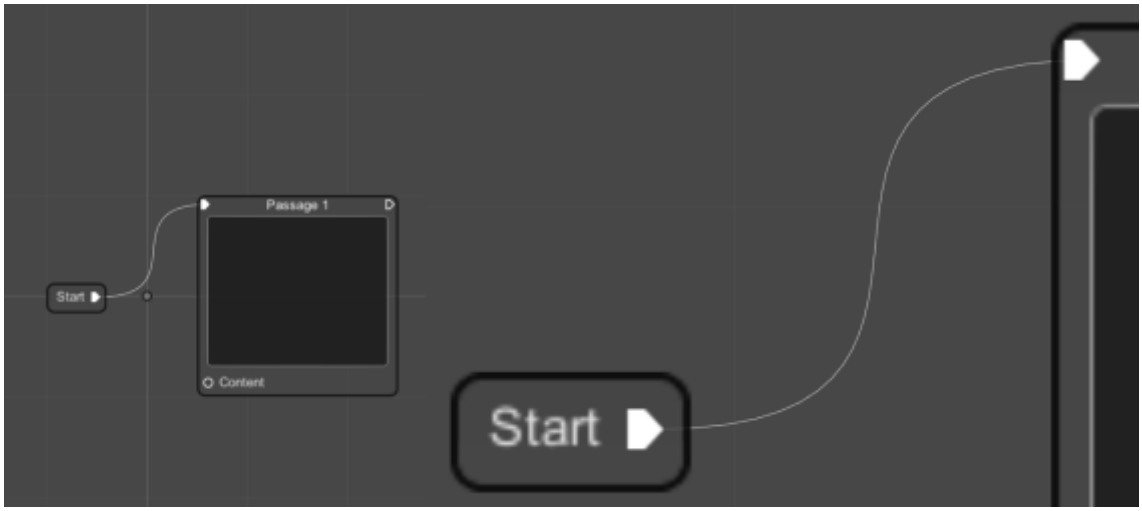
3.4 Solmuja yhdistävät langat

Selkeä solmuvisualisointi vaatii solmujen yhteyksien esittämiseen siistit ja selkeät viivat. Unityn UI-systeemi ei kuitenkaan tarjoa mitään valmista ratkaisua mielivaltaisten viivojen piirtämiseen ruudulle. Tässä luvussa tarkastellaan ratkaisua, johon päädyttiin editorin käyttöliittymässä. Tavoitteena oli saada solmujen välille selkeitä kaartuvia viivoja.

Unity tarjoaa GL-luokan alla kokoelman matalan tason piirtofunktioita. Nämä funktiot sallivat Unityn milloinkin käyttämän 3D-kirjaston käyttämisen melko suorasti. GL-luokan rajapinta mukailee löyhästi OpenGL-kirjaston käyttörajapintaa, mutta toimii samalla tavalla myös, jos Unity on DirectX-moodissa Windowsilla tai Xboxilla. GL-kokoelman joukossa on myös menetelmä, joka on tarkoitettu viivojen piirtämiseen. (Unity 2017d.)

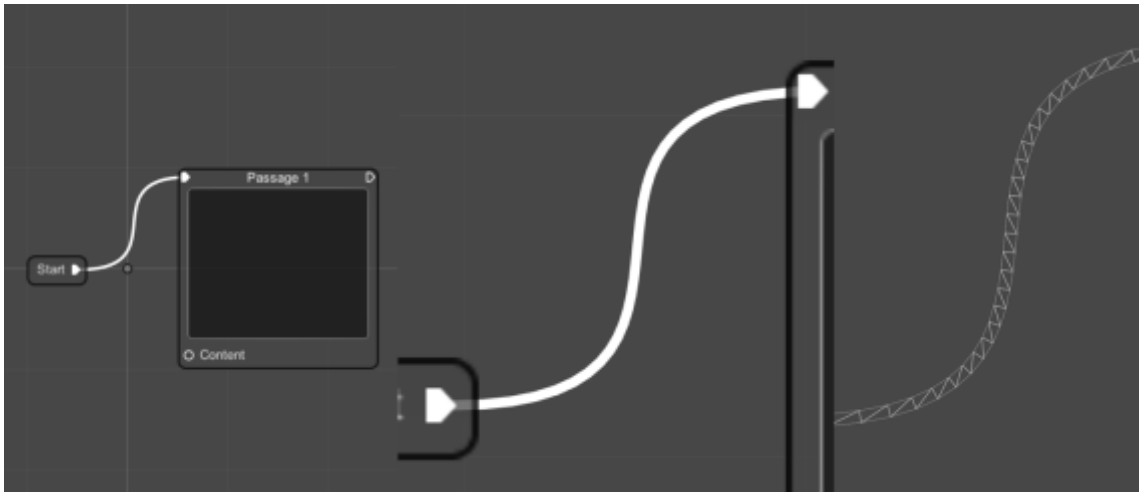
3.4.1 Viivojen piirtäminen

GL-luokan viivanpiirtostrategia valitettavasti tuottaa kuitenkin tämän projektin kannalta epätoivotun näköisiä viivoja. Viivat piirretään aina suoraan kuvan pikselimatriisiin ”yhden pikselin” levyisenä (Kuva 5). Menetelmä tuottaa karkean näköisiä viivoja, joiden paksuus on riippuvainen käyttäjän valitsemasta piirtoresoluutiosta.



Kuva 5. GL-viivat ovat yhden pikselin levyisiä suurennoksesta huolimatta.

GL-luokka tarjoaa onneksi muitakin piirtotoimintoja. Suora viiva, jolla on leveys, on nelikulmio. GL pystyy piirtämään nelikulmioita. (Oikeasti ne ovat yhteen liitettyjen kolmioiden pareja.) Piirretään viiva siis litteänä 3D-muotona. Myös kaikki Unityn omat GUI-elementit piirretään sisäisesti tällä tavalla. Bonuksena viivojen piirtämiseen voi nyt käyttää kaikkia normaaleita shader-tekniikoita.



Kuva 6. Nelikulmioilla piirretty käyrä pysyy suurennettuna oikean kokoisena muihin elementteihin nähden. Oikealla kuva käyrän muodostavista kolmioista.

Nelikulmioilla voi kuitenkin piirtää vain suoria viivoja. Mikä avuksi? Kaartuvan viivan saa tietysti aikaan piirtämällä monta viivaa peräkkäin. Pelkästään asettelemalla suorakaiteita peräkkäin tulee rumaa jälkeä osaviivojen saumakohdissa. Tämä korjaantuu, kun

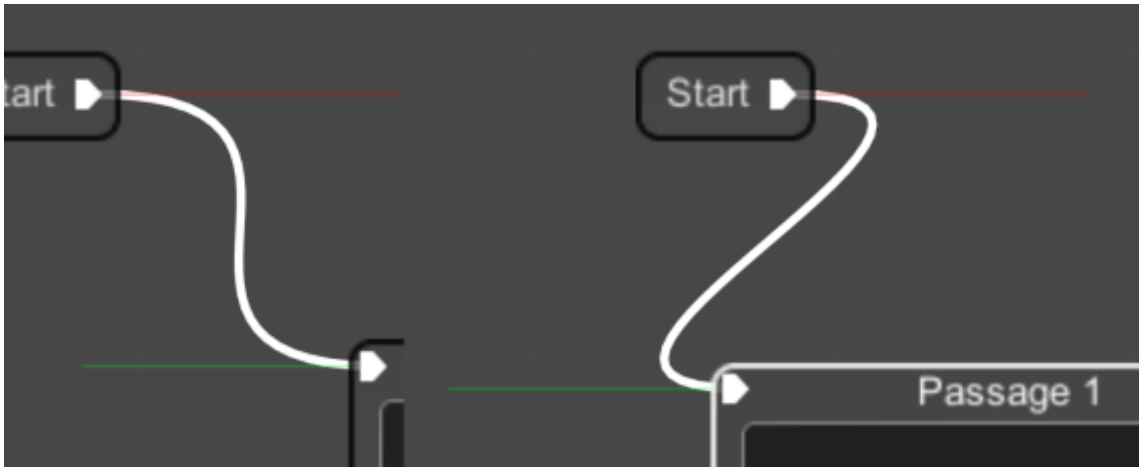
yksittäisten suorakulmioiden sijaan piirretään epäsäännöllisistä nelikulmioista muodostuvia kaaria (Kuva 6).

3.4.2 Bézier-käyrät

Mistä siiman kaaren muoto sitten saadaan selville? Käytetään hyväksi neljän pisteen Bézier-käyriä. Asetetaan käyrän alku ja loppu solmugrafiikan reunalle ja kontrollipisteet samalle korkeudelle hieman kauemmas solmugrafiikasta. Näin saadaan mukiinmenevä, vaikkei täydellinen polku yhdistämään kahta solmua (Kuva 7).

Kontrollipisteitä älykkäästi siirtelemällä voisi selkeyttää yhteysviivojen polkuja entisestään. Nykyisessä toteutuksessa viivat kulkevat liian usein solmujen takaa, kun solmut on asetettu erityisen hankalaan asentoon toisiinsa nähden.

Viivan muodostamiseksi implementoidaan helppokäyttöinen Bézier-funktio, jolle annetaan argumenteiksi vektoreina käyrän päät ja kontrollipisteet, sekä liukuluku väliltä [0,1]. Syötetty liukuluku kuvaa käyrän ”aikaa”, eli arvo 0 palauttaa käyrän alkupisteen, 1 palauttaa loppupisteen ja väliarvoilla voidaan selvittää mikä tahansa piste halutulta Bézier-käyrältä.



Kuva 7. Punainen ja vihreä viiva ovat piirretty Bézier-käyrän päistä kontrollipisteisiin.

Viivasegmenttien rajoilla tasaisen viivanpaksuuden aikaansaamiseksi täytyy suorittaa lisää laskuja. Bézier-funktion lisäksi implementoidaan funktio, joka palauttaa Bézier-käyrän derivaatan halutussa kohdassa. Derivaatta palautetaan suuntavektorina, josta

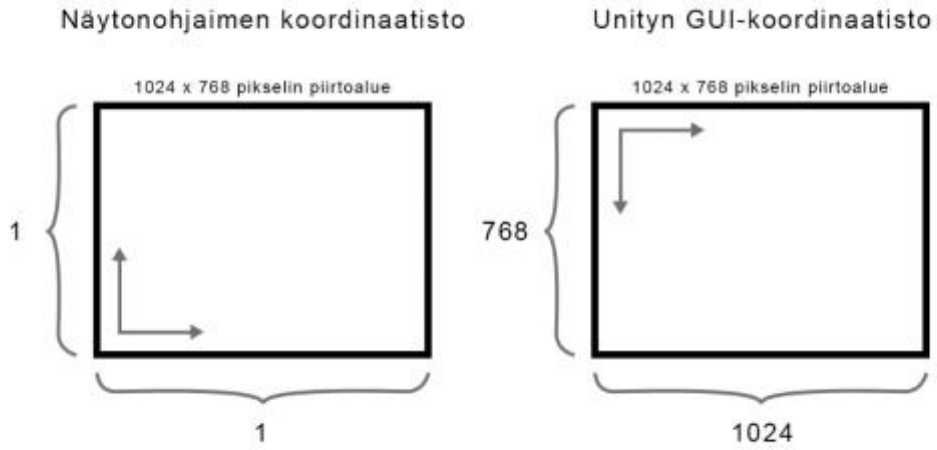
saadaan Bézier-käyrän normaali pyörittämällä sitä 90°. Käyrän pisteen ja normaalin avulla onkin yksinkertaista laskea mihin nelikulmioiden kulmat tulee sijoittaa.

3.4.3 3D-piirtäminen

GL-komennot ovat matalan tason piirtokomentoja, jotka antavat paljon valtaa mutta tasapainoksi myös paljon vastuuta. Normaalisti Unityn kaltaiset 3D-työympäristöt kantavat tämän vastuun ohjelmoijan puolesta ja häivyttävät näytönohjainten laitteistoon liittyviä erikoisuuksia. GL-luokkaa käytettäessä täytyy kuitenkin ottaa joitain niistä huomioon.

Yksi huomioitava seikka on, että matalan tason piirtokomennoissa Unityn normaali 3D-avaruus katoaa. Näyttökortin mielestä piste (0, 0, 0) sijaitsee piirtoalueen vasemmassa alareunassa ja (1, 1, 0) kuuluu piirtoalueen oikeaan yläreunaan. Kaikki pisteet täytyy lähettää näytönohjaimelle tässä muodossa.

Samalla Unity GUI-systeemi toimii koordinaatistossa, jossa (0, 0, 0) on piirtoalueen vasemmassa *yläreunassa* ja piirtoalueen oikeassa alareunassa on piste (w, h, 0), jossa w = piirtoalueen leveys pikseleinä ja h = piirtoalueen korkeus pikseleinä. Kuva 8 selventää koordinaatistojen eroa. Kehittäjän mielenterveyden kannalta paras ratkaisu on piilottaa tämä pieni poikkeavuus. Sijoitetaan kaikki solmuja yhdistävien lankojen piirtely omaan luokkaansa, jossa koordinaatistomuutokset on helppo automatisoida.



Kuva 8. Näytönohjaimen ja Unityn GUI-systeemin käyttämien koordinaatistojen erot samankokoisella piirtoalueella.

Tämänhetkisessä sovelluksessa piirtäminen toimii siten, että viivojen vektorit syötetään GUI-systeemin mukaisessa ”pikselikoordinaatistossa”. Piirtojärjestelmä muuntaa kaiken näkymättömästi sopivalla transformaatiomatriisilla näytönohjaimelle sopivaksi. Tällä tavalla koordinaatistojen erot eivät hankaloita käyttöliittymän kehittämistä.

3.5 Suurennos

Selkeiden viivojen lisäksi kokonaiskuvan hahmottamista helpottaa huomattavasti kyky vetää virtuaalinen kamera kauemmaksi solmuista. Käytännössä se tarkoittaa, että solmut ja niiden yhteydet täytyy pystyä piirtämään pienempänä. Tähänkään Unity ei tarjoa mitään valmista ratkaisua, mutta temppu on yllättävän yksinkertainen.

3D-grafiikassa kaikilla kameroilla on oma transformaatiomatriisi. Matriisi määrittää kaiken kameran näkökenttään liittyvän. Kaikki kameran ”näkemät” vektorit on kerrottu kameran matriisilla. Unityn GUI-systeemi paljastaa transformaatiomatriisin, jota käytetään GUI-elementtien piirtämiseen. Zoom-efektin saa siis aikaan muokkaamalla tätä matriisia. Unity onneksi tarjoaa matriisien käsittelyyn helppokäyttöisiä funktioita.

GUI-elementtien lisäksi myös omalla menetelmällä piirretyt yhteyslangat täytyy skaalata muuttamalla koordinaatistot kääntävää matriisia. Lähentämisen ja panoroinnin selkeyttämiseksi kaiken taustalle lisättiin vielä ruudukko, joka kutistuu ja liikkuu muiden elementtien mukana.

3.6 Käyttöliittymä

Sovellusta käytetään sijoittamalla tyhjälle kankaalle tarinanpalasia eli solmuja ja yhdistelemällä niitä langoilla tai viivoilla. Jokaisessa solmussa on pistokkeet edellisille ja seuraaville solmuille. Kappalesolmujen edellinen-pistokkeeseen voi yhdistää mielivaltaisen määrän solmuja, mutta seuraavaa merkitsevään pistokkeeseen vain yhden.

Myös valintasolmuihin voi vetää langan useammasta kuin yhdestä solmusta, mutta valintasolmulla voi olla myös monta seuraavaa solmua. Tarkoituksena on tietenkin, että pelaaja valitsee mihin näistä solmuista siirrytään seuraavaksi ja näin ohjaa tarinaa.

Solmu ja lanka -metafora valittiin käyttöliittymälle, koska sen muodostamat puurakenteet visualisoivat haarautuvaa tarinaa luonnollisella tavalla. Tämä esitystapa on myös suhteellisen yleinen ratkaisu samankaltaisissa sovelluksissa. Tämä voi auttaa käyttäjää ymmärtämään ohjelmaa paremmin, jos vastaavat sovellukset ovat hänelle tuttuja.

3.7 Tallennus

Työkalulla kirjoitetut tarinat täytyy voida tallentaa, jotta työ ei mene hukkaan, kun sovellus suljetaan. Sovelluksen täytyy myös osata palauttaa tarinan osaset takaisin muokattaviksi tallennuksen luomasta tiedostosta. Nämä ovat melko normaaleja odotuksia.

Tallentamisen kannalta onnekkaasti sovellus muotoutui hierarkkisiksi niin, että melkein kaikki sisällön kannalta tärkeät oliot ovat säilössä jonkin toisen olion sisällä. Lisäksi ylimpänä hierarkiassa on yksi olio, jonka viittauksista on mahdollista päästä käsiksi kaikkiin muihin.

Ratkaisu tallentamiseen oli käyttää sovelluksen rakennetta hyväksi. Kaikilla tallennettaville luokille määritellään kaksi funktiota, joista toinen kirjoittaa annettuun tiedostovirtaan kuvauksen oliosta XML-muotoisena ja toinen muodostaa vastaavan olion sisään annetusta XML-rakenteesta.

Kun käyttäjä painaa tallenna, hierarkian yliolio avaa tiedostovirran ja antaa sen vuorollaan jokaiselle lapsioliolleen, jokainen näistä lapsista antaa virran eteenpäin omille lapsilleen. Tämä jatkuu, kunnes oliopuun lehti tulee vastaan. Lopputuloksena saadaan XML-dokumenttiin sisäkkäisiä elementtejä, joiden suhteista voidaan päätellä palautuksen yhteydessä avainasioita.

3.8 Fungus-vienti

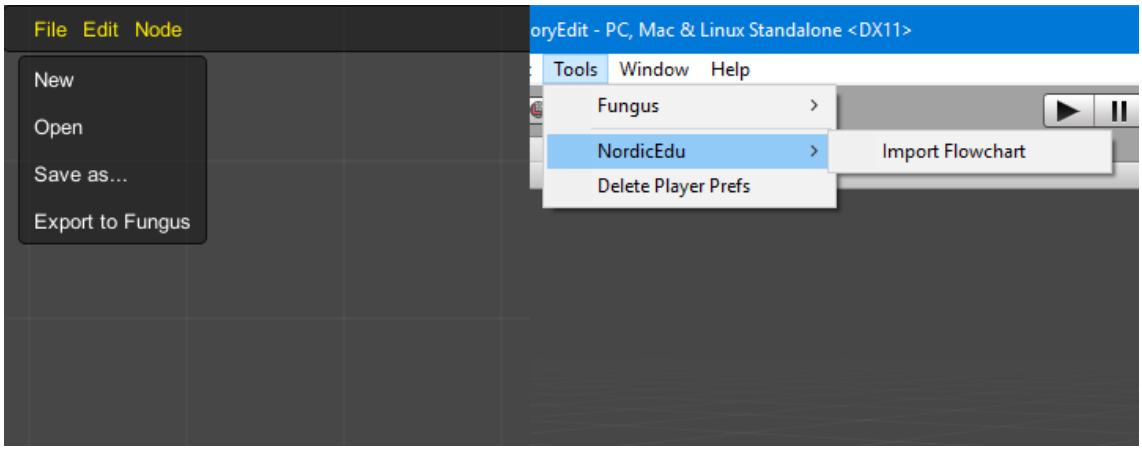
Fungus on ilmainen avoimen lähdekoodin lisäosa Unitylle, joka mahdollistaa haarautuvien tarinoiden luomisen nopeasti Unityn editorissa. Se on hyvä työkalu, ja sillä on paljon päällekkäisyyksiä tämän työn sovelluksen kanssa, mutta Fungusta ei ole mahdollista käyttää ilman Unityn editoria.

Tärkeä osa tämän työn tavoitteita oli, että työkalun voisi antaa käyttöön myös NordicEdun asiakkaille. Tähän liittyi myös tavoite, että näiden asiakkaiden ei tarvitsisi

asentaa Unityn editoria työkoneilleen tai opetella käyttämään sitä, koska Unity on suunniteltu paljon monipuolisempaan kehitystyöhön kuin pelkkään tarinankirjoittamiseen.

Tässä työssä Fungusta käytetään editorin pelikomponenttina, eli editorilla luotu sisältö saatettiin pelattavaan muotoon Funguksen avulla. Työn sovellus ei ota varsinaisesti kantaa sitä käyttävän pelin toteutukseen, koska se on sisältötyökalu, joten tarvittiin erillinen ohjelma työkalulla kirjoitettujen tarinoiden testaamiseen. Fungus sopi tähän täydellisesti, mutta ei toimi suoraan sovelluksen kanssa.

Jotta työkalun tuotoksia voisi testata Funguksella, kehitettiin sovellukseen vientitoiminto (englanniksi *export*), jolla työkalun käyttämä tietorakenne muutetaan XML-pohjaiseen välimuotoon. Tämän välimuodon voi sitten muuttaa Unityn editorissa Fungus-rakenteeksi työn osana kehitetyllä tuontilisäosalla (englanniksi *import*) (Kuva 9).



Kuva 9. Vasemmalla Fungus-välimuotoon vienti sovelluksen valikossa. Oikealla välimuodon tuontilisäosa Unity-editorissa.

4 LOPPUTULOS JA JATKOKEHITYS

Sovelluksesta tuli suunnitellun mukainen ja toteutuksen aikataulu toteutui. Kehityksen loppupäässä suunnitelman asteelle jäi kuitenkin monia ideoita, joita pohditaan tässä.

Nykyisessä versiossa tarinan haarautuminen tapahtuu aina pelaajan valitseman vastauksen pohjalta, mutta vahvan immersion luomiseksi tarinan täytyy voida haarautua myös pelaajan tekojen seurauksena. Tällaisen toiminnon toteuttamiseksi olisi hyödyllistä, jos keskusteluun pystyisi lisäämään muuhun pelidataan perustuvia ehtoja haarautumiselle.

Samaa toiminnallisuutta voi toki simuloida nykyisellään tekemällä useita keskustelutiedostoja ja valitsemalla niiden väliltä jonkin pelimuuttujan perusteella. Ehdollisuuksien rakentaminen varsinaiseen keskusteluun on kuitenkin elegantimpi ratkaisu.

Monet NordicEdun peleistä julkaistaan monikielisinä, yleensä suomeksi, ruotsiksi ja englanniksi. Koska käännökset ovat pelin samassa versiossa, tarvitaan systeemi, jolla pelin tekstit voidaan vaihtaa käännettyihin. Tällainen järjestelmä on mahdollista tehdä kokonaan erikseen tarinasysteemeistä, mutta silloin käännösten muokkaamisesta tulee nopeasti hyvin työlästä. Se olisi muutenkin intuitiivisesti huonolta tuntuva ratkaisu.

Tarinasysteemi käsittelee niin paljon tekstisisältöä, että järkevintä olisi rakentaa lokalisaatioratkaisu osaksi sitä. Erikielisten sisältöjen sisällyttämiseen on kaksi strategiaa. Ne voidaan tallettaa lähelle rakennetta tai rakenteen ulkopuolelle.

Käännösdatan säilyttäminen lähellä rakennetta tarkoittaa, että jokaiseen tekstisisältöä omistavaan solmuolioon talletetaan myös kaikki käännökset samasta sisällöstä. Tällaisen solmuolion tekstisisältöä näytettäessä sisältö haetaan vain oikeasta kohdasta solmun sisäisestä käännössanakirjasta.

Toinen vaihtoehto on säilyttää tekstidataa erillään rakenteesta. Se tarkoittaa, että solmuoliot kuvaavat tarinan rakennetta ja omistavat jonkinlaisen tunnisteen, mutta kaikki varsinainen tekstisisältö pidetään erikseen esimerkiksi tietokannassa. Tekstisisällön näyttämiseksi tietokannasta haetaan oikea arvo solmun tunnisteen ja valitun kielen perusteella.

Nykyisellään sovellus kehitettiin melko yksinkertaisia tarinoita silmällä pitäen. Sovellukseen voisi helposti lisätä ainakin mahdollisuuden syöttää hahmodataa ja sitten määrittää tekstisolmuille puhujahahmo. Tällaista metadataa voisi pelisovelluksessa tulkita esimerkiksi näyttämällä puhuvan hahmon nimi ja kuva puhelaatikon vieressä.

Suunnitellut toiminnot valittiin myös epäsuotuisasti täysin hypoteettista peliprojektia varten. Sovelluksen kehittäminen yhdessä oikean tarinapelin kanssa olisi pakottanut työkaluprojektin keskittymään tärkeimpiin toimintoihin ja valmistanut sovelluksen paremmin tuleviin projekteihin sopivaksi.

Kehitystyön alkupäässä oli tavoitteena pitää toiminnot ja käyttöliittymä täysin erillään. Työn aikana tavoitteet kuitenkin kirkastuivat eikä näiden osien erittely ollut lopulta kovin tärkeää. Alkuperäinen tavoite turhaan monimutkaisti tämän ensimmäisen version valmiiksi saamista.

Lopputulos on toimiva, mutta puutteellinen sovellus. Parasta olisi ajatella sitä prototyyppinä, jossa ideoita on testattu toimiviksi tai huonoiksi. Sovellusta voi käyttää jo nykyisellään, mutta parasta olisi kehittää yksi tai kaksi uutta versiota oikeiden asiakasprojektien yhteydessä. Oikea tuotantokäyttö on paras tapa löytää työkalun puutteet ja vahvuudet.

LÄHTEET

Chat Mapper 2017a. Chat Mapper Features. Viitattu 17.4.2017 http://www.chatmapper.com/features/

Chat Mapper 2017b. Downloads and Pricing. Viitattu 17.4.2017 http://www.chatmapper.com/pricing/

Ellison, B. 2008. Defining Dialogue Systems. Viitattu 11.3.2017 http://www.gamasutra.com/view/feature/3719/defining_dialogue_systems.php

Mateas, M. ja Stern, A. Façade. 2005.

Mass Effect. 2007. Bioware. Electronic Arts.

Montfort, N. 2007. Riddle machines: The history and nature of interactive fiction. Hoboken: John Wiley & Sons.

Petit, C. 2013. Power to the People: The Text Adventures of Twine. Viitattu 20.3.2017 https://www.gamespot.com/articles/power-to-the-people-the-text-adventures-of-twine/1100-6402665/

Ren'Py 2017. Why Ren'Py? Viitattu 17.4.2017 https://www.renpy.org/why.html

Roberts, M. 2007. Choosing a Conversation System. Viitattu 11.3.2017 http://www.tads.org/howto/convbkg.htm

Starship Titanic. 1998. The Digital Village. Simon & Schuster Interactive, R&P Electronic Media

Unity 2017a. Script Reference: MonoBehaviour.OnGUI(). Viitattu 11.4.2017 https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnGUI.html

Unity 2017b. Script Reference: GUI.Window(). Viitattu 11.4.2017 https://docs.unity3d.com/ScriptReference/GUI.Window.html

Unity 2017c. Script Reference: MonoBehaviour. Viitattu 11.4.2017 https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

Unity 2017d. Script Reference: GL. Viitattu 11.4.2017 https://docs.unity3d.com/ScriptReference/GL.htm

The Walking Dead. 2012. Telltale Games. Telltale Games.

Weizenbaum, J. 1976. Computer power and human reason: From judgment to calculation. New York: W.H. Freeman and Company.

Wilcox, B. 2011. Beyond Façade: Pattern Matching for Natural Language Applications. Viitattu 23.4.2017 http://www.gamasutra.com/view/feature/134675/beyond_façade_pattern_matching_.php

The Wolf Among Us. 2013. Telltale Games. Telltale Games.

Yoyo Games 2017. Dialog Editor 2.0. Viitattu 17.4.2017 https://marketplace.yoyogames.com/assets/2339/dialog-editor-2-0