Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Teemu Saarinen

Keskustelupuiden visualisointi

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

20.11.2018

Tekijä Otsikko Sivumäärä Aika	Teemu Saarinen Keskustelupuiden visualisointi 33 sivua 20.11.2018
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikan tutkinto-ohjelma
Ammatillinen pääaine	Pelisovellukset
Ohjaajat	Lehtori Miikka Mäki-Uuro Lehtori Jussi Alhorinne

Insinöörityön tavoitteena oli luoda visualisointitapa keskustelupuille, joita voitaisiin myös käyttää pelimekaniikkana nopeuttamaan liikkumista puurakenteisessa keskustelussa. Visualisointi tehtiin osana keskustelupohjaista psykologisia defenssimekanismeja opettavaa peliprojektia Yeea Work Oy:lle.

Työ toteutettiin Unity-pelimoottorilla käyttäen C#-ohjelmointikieltä. Keskustelupuiden rakentamiseen käytettiin Twinery-työkalua, jonka avulla keskustelupuista saatiin selväraken-teisia.

Lopputulos oli toimiva kokonaisuus, joka oli visuaalisesti miellyttävä ja mekaanisesti toimiva. Se nopeutti keskustelupuussa liikkumista ja auttoi hahmottamaan keskustelun kulkua. Visualisointi toimi myös muistiapuna pelaajalle pelin keskustelun tulkinnassa. Mekaniikka ei kuitenkaan integroitunut ympäröivään peliin riittävän selkeästi.

Avainsanat	

Author Title Number of Pages Date	Teemu Saarinen The visualization of conversation trees 33 pages 20 November 2018
Degree	Bachelor of Engineering
Degree Programme	Programme in Information and Communication Technology
Professional Major	Game Applications
Instructors	Miikka Mäki-Uuro, Senior Lecturer Jussi Alhorinne, Senior Lecturer

The goal of the study was to create a way of visualizing a conversation tree, which could also be used as a game mechanic, to speed up navigation within a conversation tree. The visualization was created as a part of a conversation-based game project for Yeea Work Oy which, was meant to teach the player about psychological defense mechanisms.

The visualization was implemented in the Unity game engine using the C# programming language. The conversation trees were built by using a tool called Twinery, which helped in adding a clear structure to the conversation trees.

The final product was a functional whole, which was visually appealing and mechanically functional. It sped up navigation within the tree structures and helped the player perceive the flow of the conversations better. The mechanic didn't integrate to the surrounding game clearly enough.

Keywords	

Sisällys

Lyhenteet

FBM	Fractal Brownian Motion. Tapa saada kohinaan fraktaalimaista toistuvuutta.
HTML	Hypertext Markup Language. Web-sivujen luomiseen käytetty kieli standardi.
XML	Extensible Markup Language. Kielistandardi jonka tarkoitus on olla luettava ihmisille ja koneille.

1 Johdanto

Puumaiset tietorakenteet ovat tärkeä osa tietotekniikkaa. Niitä käytetään paljon varsinkin etsimisen optimoinnissa ja hierarkkisen datan käsittelyssä.

Keskustelupuut ovat pelimekaniikkana hyvin vanha ja paljon käytetty. Ne on toistuvasti todettu toimivaksi ratkaisuksi varsinkin roolipelien kontekstissa. Keskustelupuut korostavat pelaajan valintojen merkitystä ja auttavat pelaajaa syventymään pelimaailmaan. Keskustelupuiden avulla pelaaja voi ottaa pelihahmon omakseen ja luoda tälle halutunlaisen luonteen.

Keskustelupuita ei yleensä näytetä pelaajalle puumaisina rakenteina, vaan ne esiintyvät orgaanisesti osana keskusteluja pelihahmojen välillä. Tässä työssä esitellään peliprojekti, jossa keskustelupuun visualisointi on tärkeää. Pelissä keskustelupuusta tehdään aktiivinen mekaniikka passiivisen sijaan. Tämä toteutetaan antamalla pelaajan nähdä missä kohtaa keskustelupuussa hän kulkee ja antamalla mahdollisuuden siirtyä mihin tahansa muualle keskustelupuun aiheissa, joita pelaajalle on avattu.

Aktiivinen keskustelupuun käyttö vahvistaa pelaajan tehtävää, joka on pelissä keskustelun analysointi ja sen taustatekijöiden arvaaminen ja ymmärtäminen. Peli projektin tarkoituksena on opettaa pelaajalle psykologisien puolustusmekaniikkojen käsitteitä ja toimintaa. Tämä tapahtuu pelissä keskustelun kautta asettelussa, jossa pelaaja on psykiatri ja kyselee tietokoneen ohjaamalta hahmolta, joka on hänen asiakkaansa, kysymyksiä. Näiden kysymysten avulla pelaaja selvittää, mikä potilasta vaivaa oppien samalla psykologiaa.

Tässä työssä käsitellään, mitä keskustelupuut ja puumaiset tietorakenteet yleensä ovat ja miten ne rakentuvat. Tämän jälkeen työssä tarkastellaan useita erilaisia tapoja visualisoida puurakenteita. Viimeiseksi työ käsittelee käytännössä, miten ja miksi keskustelupuita voi visualisoida peleissä käytännön esimerkin avulla. Käytännön esimerkki tehtiin osana peliprojektia, joka toteutettiin Yeea Work Oy:lle.

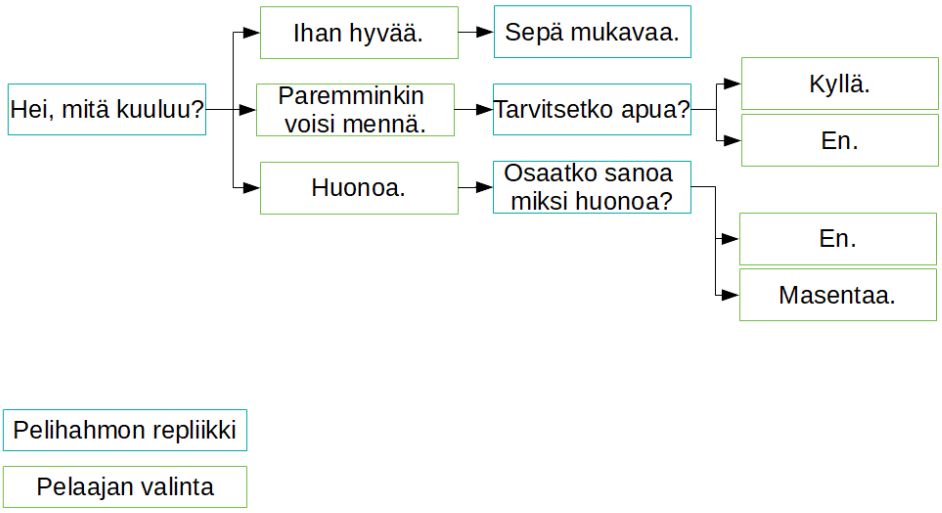
2 Taustaa

Keskustelupuita on käytetty jo yli 50 vuotta erilaisissa tarkoituksissa. Tämä luku käsittelee keskustelupuuta rakenteena ja sen historiaa. Luvussa käydään myös läpi puurakenteita yleisesti.

2.1 Mikä on keskustelupuu

Keskustelupuu on tapa rakentaa keskusteluja peleissä. Se perustuu siihen, että peleissä pelaajalle annetaan mahdollisuus valita, mitä hän haluaa sanoa keskusteluissa, ja minkä seurauksena keskustelu haarautuu pelaajan valintojen mukaan [1]. Keskustelupuiden idea perustuu novelleihin, joissa lukijoille annetaan mahdollisuus valita oma tarinansa. Ensimmäinen tunnettu tämänkaltainen tarina on ”The Garden of Forking Paths”, jonka kirjoitti Jorge Luis Borges vuonna 1941 [2.] Ensimmäinen elektroninen esimerkki keskustelupuusta on ELIZA, joka oli luonnollisen kielen käsittely ohjelma, jonka ideana oli esittää, kuinka pinnallisia keskustelut ihmisten välillä usein ovat [4.]

Pelimekaniikkana keskustelupuita käytetään, jotta saataisiin pelin hahmoille luotua luontevan tuntuisia keskusteluja ja pelimaailmasta saataisiin uskottavampi [kuva 1.] Ne auttavat saamaan pelaajan upottautumaan pelimaailmaan, ja varsinkin roolipeleissä pelaaja voi ottaa pelihahmon täysin omakseen ja luoda mielessään tälle haluamansa luonteen.

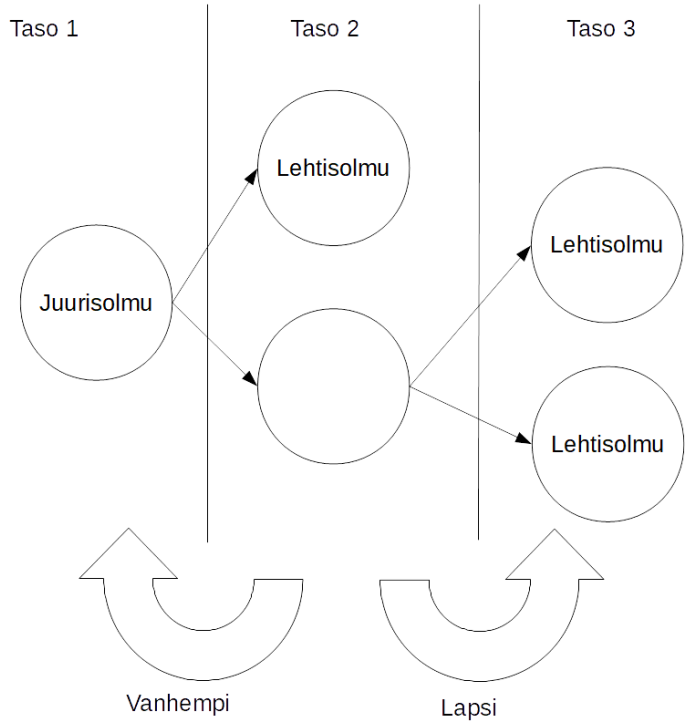


Kuva 1. Esimerkki keskustelupuun käytöstä pelihahmojen välisessä keskustelussa.

Keskustelupuita käytetään lähes kaikissa peligenreissä, joissa on keskustelevia hahmoja. Niitä on käytetty jo pitkään, koska ne ovat yksi parhaaksi todetuista järjestelmistä pelien tarinan kerrontaan ja keskusteluiden luomiseen [1].

2.2 Puurakenteet

Tietorakenteena puu koostuu alkioista, joita kutsutaan solmuiksi. Puu alkaa ensimmäisestä solmusta tai juurisolmusta, josta haarautuu useampia solmuja [kuva 2.] Jokaisella solmulla paitsi juurisolmulla on vanhempi eli solmu, josta ne ovat haarautuneet, ja mahdollisesti lapsisolmuja eli solmuja, jotka haarautuvat kyseisestä solmusta. Jos solmulla ei ole lapsia, sitä kutsutaan lehtisolmuksi [5; 6; 26].



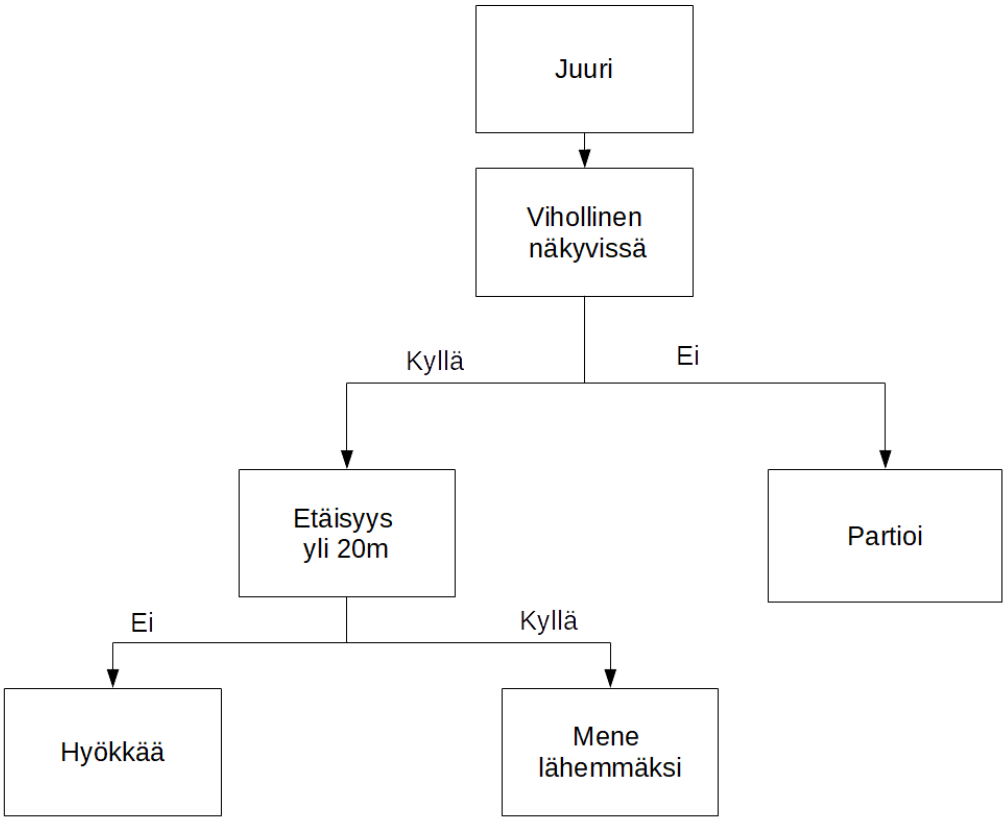
Kuva 2. Esimerkkikuva puurakenteen koostumuksesta

Puurakenteen hierarkian syvyysastetta nimitetään usein tasoksi. Esimerkiksi juurisolmun lapsisolmut ovat tasolla kaksi.

Puurakenne voi olla järjestetty, jos sen lapsisolmut ovat keskenään loogisessa järjestyksessä. Järjestettyjä puurakenteita käytetään laskelmallisista syistä algoritmien tehostamiseen.

2.2.1 Binääripuu

Binääripuu on puurakenne, jossa jokaisella solmulla on enintään kaksi lasta [5]. Binääripuita käytetään esimerkiksi päätöspuina, joissa lopputulos riippuu joukosta ehtoja, joissa on kaksi mahdollista tulosta esimerkiksi kyllä ja ei. Kuvassa 3 on tyypillinen päätöspuu, jossa tekoäly päättää mitä tehdä. Päätöspuissa kaikissa solmuissa pitää olla molemmat vaihtoehdot, eli ei voi olla päätöksiä, joissa on vain yksi vaihtoehto.



Kuva 3. Esimerkki päätöspuusta yksinkertaisessa tekoälyssä

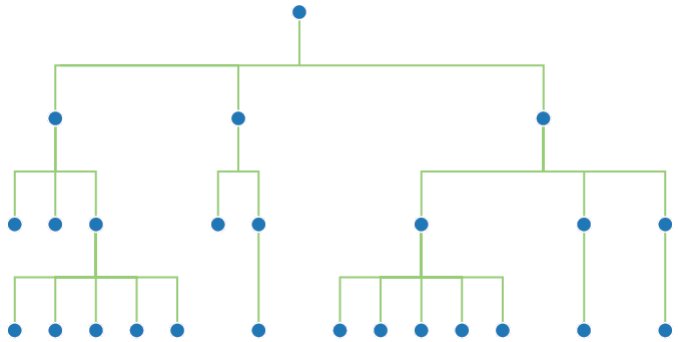
2.3 Erilaiset puiden visualisoinnit

Puurakenteet ovat usein suuria ja vaikeita hahmottaa ilman hyvää visualisointia. Visualisointikeinoja on kehitetty monenlaisia vastaamaan puilla kuvatun datan moninaisuutta. Yhteisenä tekijänä kaikilla puudatan visualisointikeinoilla on se, että ne kuvaavat dataa hierarkiassa.

Datana kuvaukseen käytetään muotojen ja positioiden lisäksi värejä ja kokoja. Suurien datasettien visualisointitarpeisiin on myös alettu tehdä interaktiivisia visualisointeja, jotta koko datasettiä ei tarvitsisi kuvata kerralla, koska ihmisen on vaikea hyväksyä dataa suurissa määrissä kerrallaan.

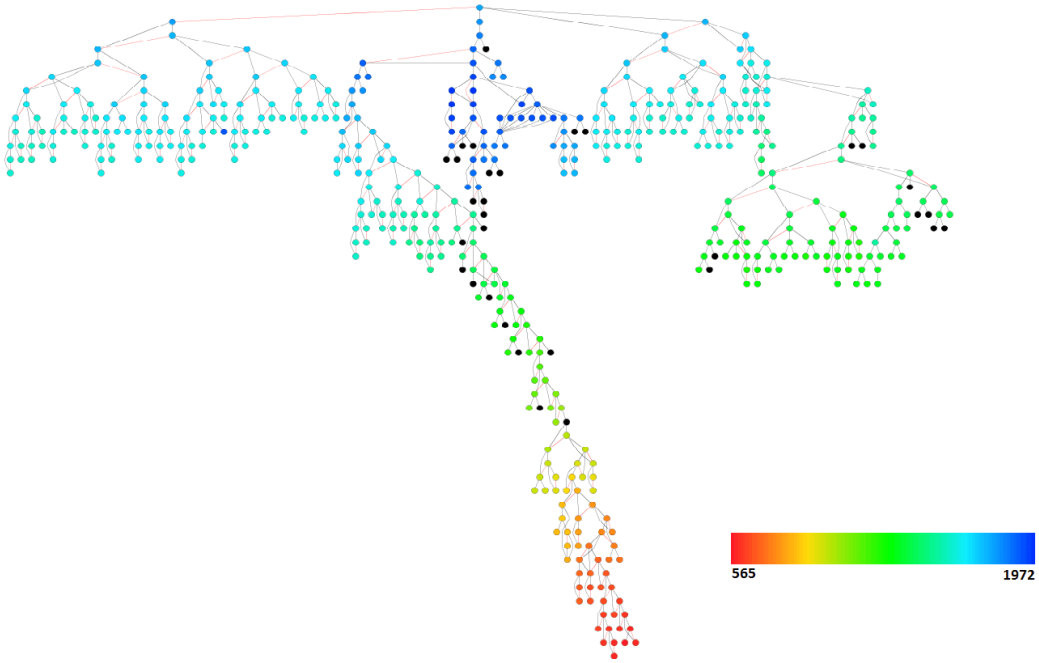
2.3.1 Puudiagrammi

Puudiagrammi on hyvin yksinkertainen tapa esittää puumaista dataa. Sitä käytetään usein esimerkiksi sukupuiden [7; 8; 9] ja päätöspuiden kuvaamiseen. Puudiagrammi on myös yksi vanhimpia tapoja visualisoida puumaista dataa.



Kuva 4. Tyypillinen yksinkertainen puudiagrammi

Puudiagrammi alkaa selvästi yhdestä pisteestä, josta se sitten alkaa haarautua yhteen suuntaan [7]. Kuvassa 4 on diagrammi, joka haarautuu alaspäin. Se kuvataan yleensä solmuilla, joista haarautuu tasaisin kulmin lapsia. Tämä korostaa hierarkiaa ja selventää visualisointia.



Kuva 5. Puudiagrammi on esimerkki suuresta sukupuusta. Väri kuvaa henkilön syntymäaikaa. Kuvaajan yläosassa on uusin sukupolvi ja pohjalla vanhin.

Puudiagrammi on helppo tehdä ja ymmärtää. Se on tehokas visualisointitapa yksinkertaiselle ja suhteellisen pienelle datalle. Jos data on kovin syvää, niin diagrammi levenee paljon [8], ja sen hyödyllisyys ja tehokkuus kärsii [kuva 5.] Puudiagrammin hierarkia on selvä lukijalle, koska kaikki solmut on yhdistetty viivoilla, ja diagrammi etenee yleensä normaaliin lukusuuntaan ylhäältä alas tai vasemmalta oikealle.

Esimerkki käyttötarkoituksia ovat sukupuut, päätöspuut, henkilöstön hallinta organisaatioissa ja evoluution visualisointi [9].

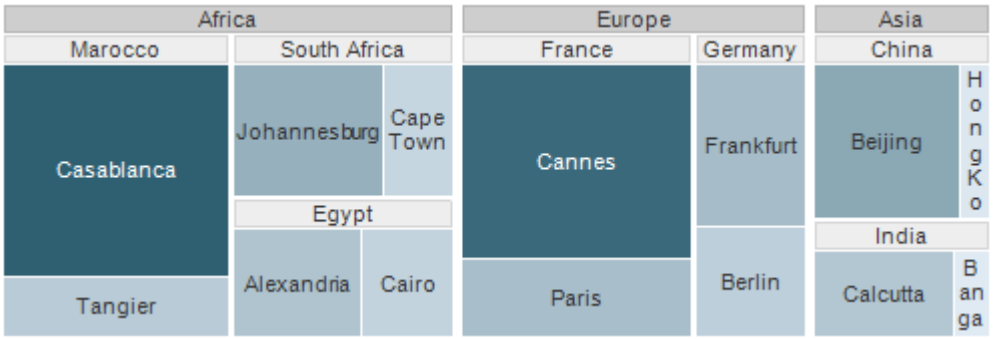
2.3.2 Puukartta

Puukartta visualisoinnin kehitti alun perin Ben Schneiderman vuonna 1990 [3]. Alun perin sen tarkoitus oli kuvata muistin kulutuksen jakaumaa tietokoneen kovalevyllä.

Puukartta on tapa esittää hierarkkista dataa [10; 24; 26]. Puukartassa data esitetään erivärisinä neliöinä, joiden väliset suhteet näkyvät neliöiden koossa ja positiossa. Neliöt,

jotka ovat toisten sisällä, ovat ulomman neliön lapsia. Myös väriä voidaan käyttää kuvaamaan jotain datan piirrettä, jolloin voidaan kuvata uutta ulottuvuutta datasta.

Puukartta on hyvä tapa visualisoida hierarkkista määrällistä dataa kuten väestölukuja [kuva 6.] Se on myös todella tehokas tilankäytössä, koska se ei vaadi yhtään tyhjää tilaa eri solmujen väleihin. Puukartta soveltuu todella hyvin suurien datamäärien kuvaamiseen, koska se on niin tilatehokas, mutta sen lukeminen voi olla tämän takia työlästä.

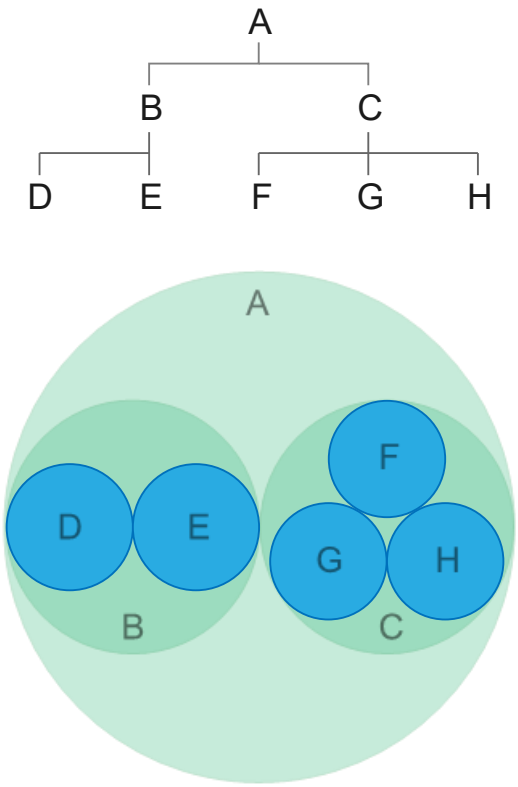


Kuva 6. Esimerkkikuva puukartasta. Värit kuvaavat myyntien määrää, tummempi väri tarkoittaa enemmän myyntejä.

Esimerkki käyttötarkoituksia puukartalle ovat urheilutilastointi, tietokoneen muistin jakauman tutkiminen, väestönlukujen vertailu ja kansantuotteen vertailu [11].

2.3.3 Ympyräpakkaus

Ympyräpakkaus on versio puukartasta, jossa neliöiden sijaan käytetään ympyröitä. Se on parempi esittämään datan hierarkiaa, kuin puukartta, mutta ei ole yhtä tehokas tilan käytössä [kuva 7.]



Kuva 7. Esimerkkikuva ympyräpakkauksesta. Kuva hahmottaa miten puurakenne jaetaan ympyräpakkausmuotoon.

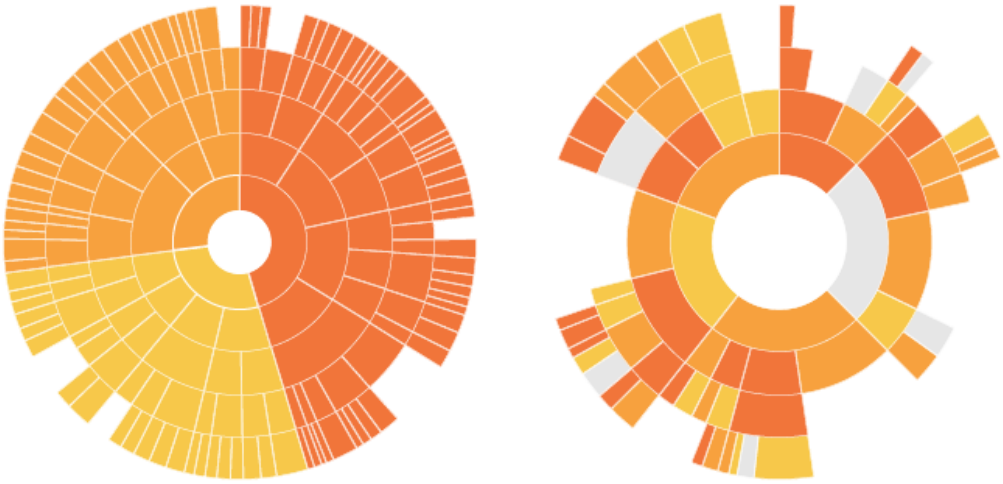
Ympyräpakkaus on hyvä valinta puukartan sijaan, jos hierarkian korostus on tärkeää tai sen jos kartan visuaalinen ilme on tärkeämpää kuin tila tehokkuus. Se voi myös olla parempi vaihtoehto, jos datasta pitää saada nopealla silmäyksellä sen idea irti, koska puukartta on tiheytensä takia vähän vaikealukuisempi.

Esimerkki käyttötarkoituksia ympyräpakkaukselle ovat esitykset, tiedostojärjestelmien kuvaus ja sukupuut [12].

2.3.4 Sunburst-diagrammi

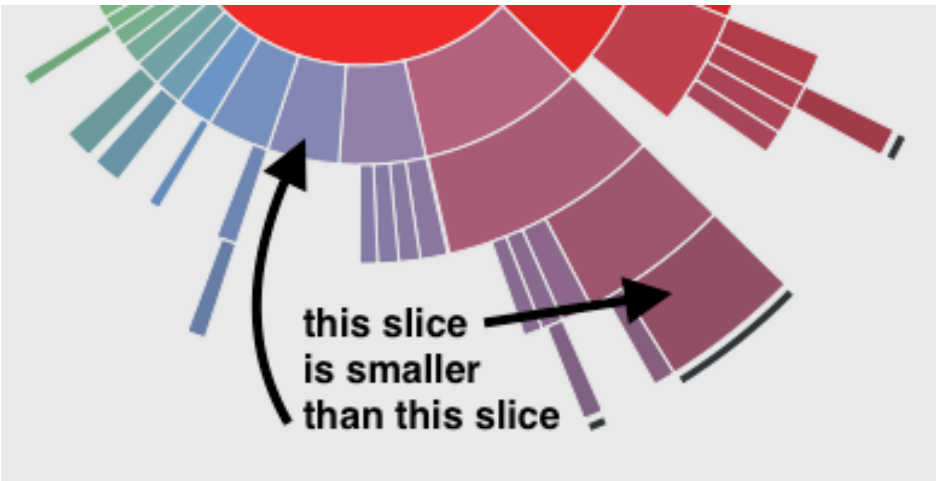
Sunburst-diagrammi on visuaalisesti miellyttävä [kuva 8] ja selkeä visualisointitapa. Se on erityisen toimiva interaktiivisena versiona, jossa voi siirtyä tasolta toiselle, ja nähdä

jokaisen alueen jakauman omana kaavionaan [14; 25]. Sunburst-diagrammissa käytetään voimakkaasti värejä erottelemaan alueita [kuva 8]. Diagrammin hierarkia etenee loogisesti keskeltä ulospäin. Diagrammin voi ajatella monitasoisena pylväsdiagrammina.



Kuva 8. Kuva sunburst-diagrammista. Diagrammin värit vastaavat eri kategorioita.

Sunburst-diagrammin suurin ongelma on se, että osien kokoa ei voi vertailla eri tasojen välillä, vaan pitää verrata kulmaa [13], josta osa koostuu [kuva 9.] Tämän takia interaktiivisuus tai oikean datan valinta on tärkeää tälle diagrammityypille.

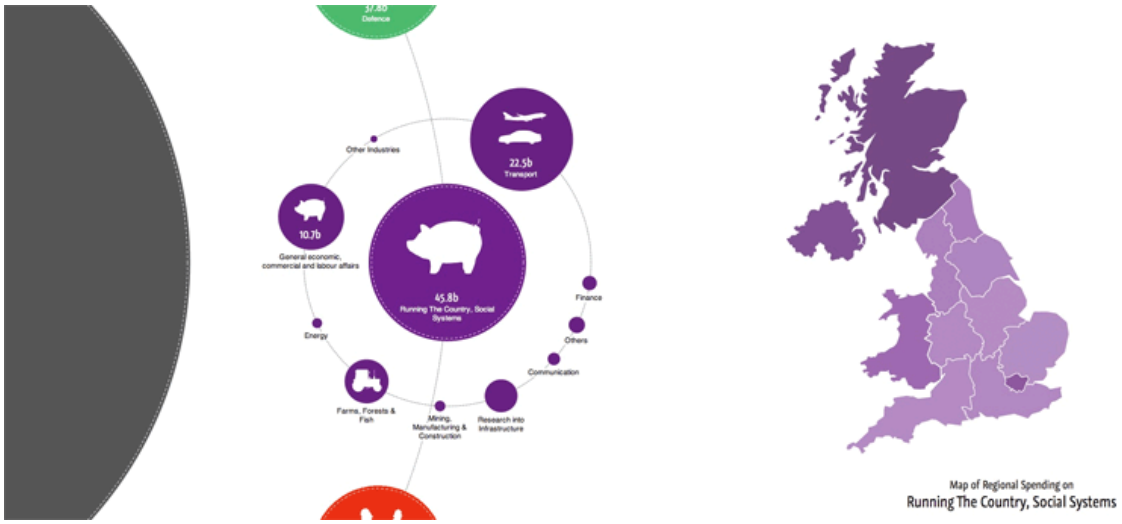


Kuva 9. Kuva joka osoittaa sunburst-diagrammin eri tasojen kokoerojen vertailun vaikeuden.

Esimerkki käyttötarkoituksia sunburst-diagrammille ovat median genrejakauman esitys ja yrityksen vuosituoton esitys.

2.3.5 Kuplapuu

Kuplapuun [engl. Bubble tree] visualisointi on interaktiivinen visualisointitapa, jossa data kuvataan hierarkkisesti niin, että lapsisolmut on aseteltu vanhempansa ympärille, ja jokainen solmu on kooltaan verrannollinen osuuteensa vanhemman määrästä [kuva 10.] Siinä näytetään yleensä valittu datan taso, yksi sitä ylempi ja yksi sitä alempi. Ylempi taso näkyy ruudun reunassa, nykyinen taso näkyy ruudun keskellä, ja sen ympärillä on alempi taso.



Kuva 10. Esimerkki kuplapuusta. Kuvassa esitetään UK:n verorahojen jakautumista eri alueisiin.

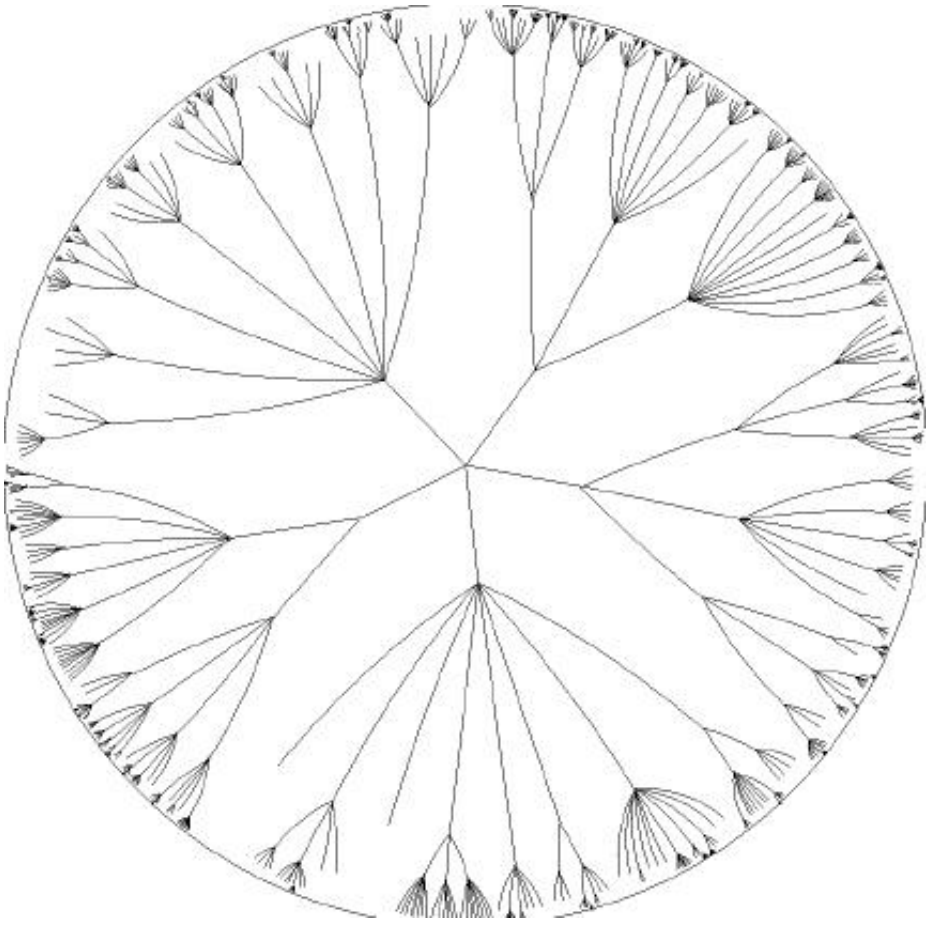
Kuplapuu on suunniteltu juuri määrällistä dataa varten. Alun perin se tehtiin ’Where does my money go?’ -projektiin, jossa kuvatiin UK:n verorahojen kulutusta [15; 22.]

Kuplapuun interaktio perustuu zoomaamiseen valittuihin datapisteisiin, jolloin visualisointi näyttää uuden tason datasta. Tällä tavalla visualisointi kommunikoi hyvin kokonaisuuksia antamatta liian suuria tietomääriä lukijalle kerrallaan.

Kuplapuu ei kuitenkaan sovellu kovin syviin hierarkioihin [15], koska lukijan on helppo hukkua dataan unohtamalla reitin, mistä pääsi nykyiselle tasolleen.

2.3.6 Hyperbolinen puu

Hyperbolinen puu sai inspiraationsa Mautis Cornelis Escherin työstä ’ Circle Limit IV 1960 woodcut in black and ocre’ [16]. Se on verkkomainen puurakenne, jossa solmujen etäisyys ja koko kutistuvat eksponentiaalisesti tasojen syvyyden noustessa. Eksponentiaalinen kutistuminen visualisoinnin reunalla tekee siitä fraktaalimaisen, ja sen takia sillä voi siis visualisoida rajattoman määrän dataa, mutta datana lukeminen vaatisi todella suuressa datasetissä lähennystoimintoa [kuva 11.]



Kuva 11. Esimerkki hyperbolisesta puusta, jonka nollakohta on asetettu puun keskelle.

Hyperbolinen puu on hyvä esittämään suuren määrän dataa ruudulla kerrallaan. Keskustaa lähimpiin tasoihin saa tässä visualisoinnissa liitettyä tekstikuvaukset, mutta lehtiin se ei onnistu, koska ne ovat niin pieniä ja lähellä toisiaan.

3 Implementaation valinta

Peliprojektin tarkoituksena oli tehdä peli, joka opettaa pelaajilleen psykologisista puolustusmekanismeista. Psykologiset puolustusmekanismit ovat alun perin Sigmund Freudin kehittämä teoria [17], jossa ihmisillä on psykologisia strategioita, joilla he selviävät ahdistavista tilanteista. Puolustusmekanismeja on luokiteltu useaan asteeseen sen mukaan, kuinka 'terveitä' ne ovat. Kaikilla ihmisillä esiintyy joitain puolustusmekanismeja. Pelin ideana oli opettaa näistä mekanismeista ja sitä kautta mahdollistaa huomaamaan, mitä näistä itse saattaa käyttää ja ehkä parantaa tapojaan ohjaamalla puolustusmekanismejaan terveempään suuntaan.

Pelin opettavuus oli ajatus tapahtua tarinoiden kautta, koska tarinat ovat tehokas tapa opettaa [18]. Samoin haluttiin myös mahdollisuus herättää pelaajassa erilaisia tunteita peliä pelatessa, koska myös tunteet auttavat oppimista [19].

Koska pelin oli tarkoitus käsitellä läheisesti sosiaalista kanssakäymistä, aloitettiin etsimään inspiraatiota tekstipohjaisista peleistä. Ace Attorney on visuaalinen novelli, jossa pelaaja on asianajaja, jonka tehtävänä on selvittää eri rikoksia vihjeiden perusteella. Salapoliisimaisesta tunteesta pelin pohjana pidettiin, koska jos pelaaja saadaan kiinnostumaan aiheesta ja miettimään tilannetta ja mitä sen taustalla on, hän oppii helpommin aiheesta.

Ace Attorneyn lisäksi valittiin Theme Hospitalin ja The Infectious Madness of Doctor Deckerin innoittamana peliin asettelu, jossa pelaaja on psykiatri, joka haastattelee potilaita, ja yrittää selvittää heidän ongelmiaan. Tällä tavalla pelaaja saadaan ajattelemaan puolustusmekanismeja pakottamatta häntä kuvaamaan tai ajattelemaan suoraan omia ongelmiaan.

Koska pelissä oli tarkoitus olla pitkiä keskusteluja ja sen seurauksena suuria keskustelupuita, joista pelaajan piti päätellä asioita, peli vaatii tavan navigoida keskustelussa aiheesta toiseen helposti ja apua pelaajan muistille. Tähän ei riitä pelkkä takaisin- ja eteenpäin-nappi, koska pitkissä keskusteluissa aiheen vaihtaminen veisi helposti yli 10 painallusta sen sijaan, että sen voisi tehdä kahdella navigaatioavun kanssa. 10 painallusta yhden valinnan tekemiseen on liian paljon, ja se hidastaisi pelin pelaamista liikaa.

AI generoidut kysymykset ja vastaukset annettujen parametrien avulla haluttiin mahdollistaa, jotta näitä suuria keskusteluja ei tarvitsisi kirjoittaa käsin. Tämä mahdollistaisi käytännössä loputtoman pitkät keskustelut yhden asiakkaan kanssa, mikä toisi visualisointiin omat ongelmansa.

Lopullinen peli-idea oli peli, jossa pelaaja pelaa psykiatrina, ja kyselee kysymyksiä potilaalta. Pelaaja saa valita itse avonaisista aiheista minkä tahansa, ja kysyä siitä aiheesta annetuista kysymyksistä minkä tahansa. Kysymykset ja aiheet eivät ikinä sulkeudu, jos niitä ei kysytä, mutta niitä voi tulla lisää. Pelaajan maalina on auttaa potilasta heidän ongelmissaan puhumalla ongelmat läpi ja löytäen täten puolustusmekanismeja, joita hän voi sitten ohjata terveempään suuntaan.

3.1 Visuaalisten elementtien valinta

Navigointi puurakenteessa on hidasta ja hankalaa ilman visuaalista apua. Vaatii siirtymistä, joka on pahimmillaan kaksinkertainen puun syvyyteen verrattuna (nykyisestä aiheesta juureen ja juuresta uuteen aiheeseen). Jo pienessäkin puurakenteessa on helppo eksyä ilman apukeinoja.

Keskustelun kulkua pitäisi pystyä kuvaamaan, koska pelaajalle on vaikeaa hahmottaa normaalissa roolipelityylisessä keskustelussa, mitä on puhunut ja mistä kannattaisi puhua. Rakenteesta olisi hyvä nähdä nopeasti yleisempi konteksti, tarkempi aihe ja mahdollisesti myös vaihtoehtojen määrä ja mahdollisuus edetä puussa.

3.1.1 Kuplapuutoteutus

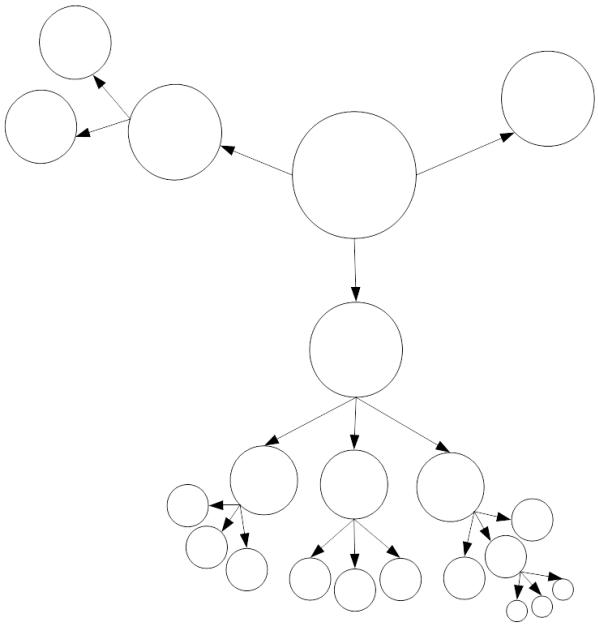
Kuplapuuvisualisoinnilla pystyisi kuvaamaan loputtoman syvän keskustelun fraktaalimaisesta toteutuksestaan johtuen. Tämä ratkaisu ei näytä hyvin pitkiä ketjumaisia suhteita datapisteiden välillä, mutta on hyvä vertaamaan laajoja datasettejä, joissa on vähemmän syvyystasoja.

Koska peliprojektissa karttamaisuus ja navigointiapu oli niin tärkeää, kuplapuutoteutusta ei valittu. Pelin toteutuksessa oli myös tärkeää, että se toimii syvillä rakenteilla, ja kuplapuu soveltuu vain mataliin rakenteisiin hyvin.

Tämän kaltaista toteutusta harkittiin, koska se olisi helppo toteuttaa, ja se mahdollistaa rajattoman syvyyden ilman minkäänlaisia tila ongelmia. Tilaongelma oli suurin ongelma toteutuksen valinnassa. Oli kuitenkin tärkeää, että valittu toteutus myös auttaa pelaajaa keskustelun kulun hahmottamisessa eikä kuplapuu tehnyt sitä

3.1.2 Valittu toteutus

Toinen toteutustapa, mitä ajateltiin, on kuvan 12 mallinen rakenne. Tässä rakenteessa kartta laajenee juuripallon ympärille tasaisesti. Mallissa jokainen taso on skaalattu pienemmäksi edelliseen verrattuna. Täten vältettiin tilaongelma ja säilytettiin silti kaikki solmut näkyvissä. Riittävällä skaalauksella ja etäisyyksillä tällä mallilla on myös mahdollista tehdä rajattoman suuria karttoja, mutta pelin tarkoituksessa kaikkien solmujen pitäisi olla mielellään riittävän suuria, että niistä tunnistaa ikoni kuvan. Tämä takia pelissä skaalausta ja välimatkaa ei voi tehdä kovin suureksi. Malli toimii kuitenkin todella hyvin pienemmisää keskustelupuissa, joten se valittiin pelidemoon käyttöön. Tämä rakenne antaa täyden kuvan keskustelun kulusta, ja koska se on yksinkertainen, se ei myöskään sekoita tai pelota pelaajaa.



Kuva 12. Kuva, joka esittää valitun toteutustavan rakennetta.

Malli muistuttaa paljon sunburst-visualisointia, mutta pylväsdiagrammin sijaan siinä käytetään yhdistettyjä palloja. Se on ympyrämäinen variaatio peruspuudiagrammista.

4 Toteutus

Tässä luvussa käsitellään pelin rakenteen teknistä toteutusta ideasta toteutuksen viimeisteltyyn vaiheeseen. Luvussa esitellään toteutuksen eri vaiheet ja niihin liittyviä mietteitä ja vaatimuksia.

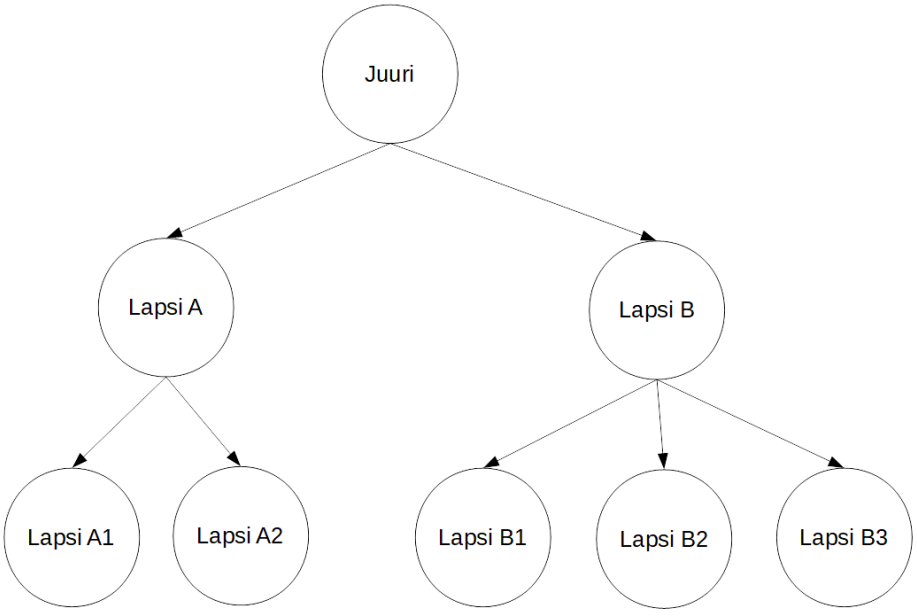
4.1 Pelimoottori

Peliprojekti toteutettiin Unity-pelimoottorilla. Unity on työkalu kolme- ja kaksiulotteisia pelejä ja simulaatioita varten. Se julkaistiin 2005 [20]. Ohjelmointikielenä Unity-pelimoottorin kanssa käytetään C#-ohjelmointikieltä.

Unityssä pelit ja simulaatiot koostuvat tasoista (engl. scene), jotka toimivat konteksteina peliin tai simulaatioon rakennetulle logiikalle. Tason sisällä logiikka koostuu peliobjekteista (engl. GameObject), jotka toimivat pohjana kaikelle simulaation maailman sisällä tapahtuvalle. Peliobjekteihin voi moottorissa liittää komponentteja, jotka tuovat niihin erilaisia toimintoja. Yleisiä komponentteja ovat esimerkiksi BoxCollider ja RigidBody, jotka ovat fysiikkakomponentteja. Moottorilla tehdyt pelit koostuvat siis tasoista, joihin on aseteltu peliobjekteja, joihin on tehty erilaisia komponentteja.

4.2 Idea

Teknisen toteutuksen idea oli rakentaa juurisolmusta alkava verkostomainen rakenne [kuva 13], joka voisi teoriassa jatkua ja laajentua loputtomiin asti. Tämä tukisi fraktaalimaisia visualisointeja parhaiten, koska rakenne olisi toistuvaa, mutta niin laajaa, että se ei mahtuisi järkevän kokoisena ruudulle. Rakenteen oli tarkoitus olla hyvin skaalautuva ja nopeasti toteutettavissa.

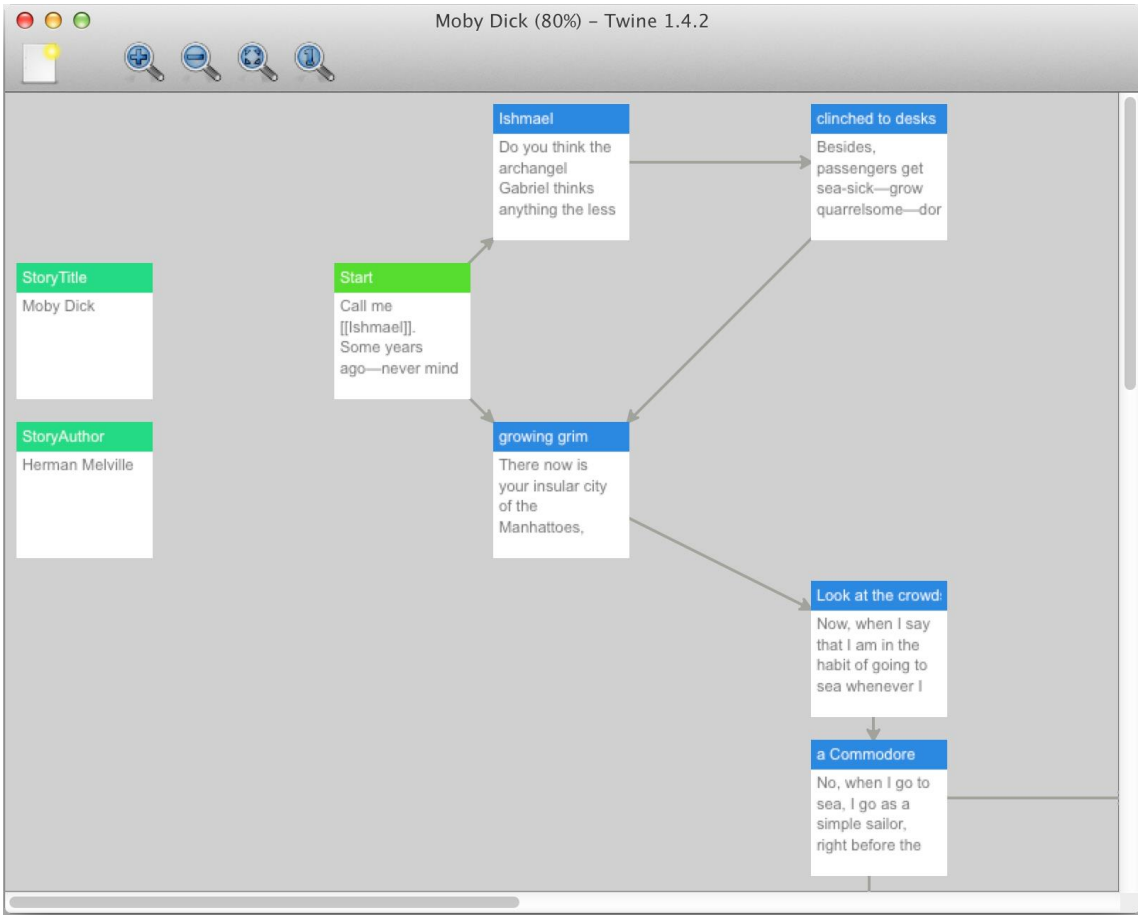


Kuva 13. Kuva puudiagrammin rakenteesta.

Keskustelupuu haarautui pelaajan valintojen mukaan eri aiheisiin, joihin pelaaja pystyi palaamaan aina, jos hänelle tuli mieleen, että joku aikaisempi aihe voisi olla relevantti hänen päättelemälleen hypoteesille. Tämä aiheutti ongelman keskustelussa sujuvassa navigoinnissa suurehkoissa keskustelupuissa. Tarkoituksena oli mahdollistaa vapaa liikkuminen keskustelun eri aiheiden välillä. Esimerkiksi jos oli kulkenut kuvan 13 solmusta Lapsi A1 solmuun Lapsi B3, se olisi vaatinut neljä klikkausta/siirtymistä ilman jonkinlaista mekaniikkaa, jolla ylitetään suorat yhteydet keskustelun aiheiden välillä. Tämä ongelma korostui pelin tapauksessa, koska juuresta lähti useampi pääaihe, joihin pelaaja saattoi haluta siirtyä missä vaiheessa keskustelua tahansa, ja keskustelun haarat olivat ainakin viisi tasoa syviä jo demoversiossakin.

Keskusteludatan rakenne on toteutettu haarautuvana puurakenteena [kuva 13], jossa haarat eivät ikinä yhdisty eivätkä jakaudu yli viiteen haaraan yhdestä solmusta. Normaalisti solmut jakautuvat nollasta kolmeen alasolmuun. Yli kolmeen solmuun jakautuminen useassa kohdassa aiheuttaisi vaikeuksia visuaalisessa toteutuksessa, koska se veisi niin paljon tilaa.

4.3 Twinery-työkalun käyttö



Kuva 14. Kuva Twinery-sovelluksen tarinan rakenteesta.

Twinery on avoimen lähdekoodin käsikirjoitustyökalu [21], jolla voi nopeasti luoda keskustelupuita. Se sisältää työkalut koodin, kuvien ja yksinkertaisen logiikan lisäämiseen tarinaan. Twinery otettiin käyttöön käsikirjoituksen nopeuttamiseksi ja siihen selvän loogisen rakenteen lisäämiseksi, jota voitaisiin käyttää pelin logiikassa. Twineryllä tehdyt keskustelupuut oli tarkoitettu pelin demovaiheen ratkaisuksi, joka korvattaisiin myöhemmin AI-generoiduilla repliikeillä tai chatbotilla.

Twineryssä tarinat luodaan ‘kappaleista’ [engl. passage], jotka linkitetään toisiinsa [kuva 14]. Tarina alkaa aina aluksi merkitystä kappaleesta ja jatkuu pelaajan valintojen mu-

kaista reittiä. Koska tarinoihin voi lisätä myös muuttujia ja logiikkaa, koko demon tarinankulun logiikka saatiin luotua Twineryssä, joten käsikirjoittaja pystyi testaamaan keskustelujaan itse.

4.4 Tietorakenne

Keskustelupuiden tekemiseen valittiin Twinery-työkalu, koska se oli avointa lähdekoodia ja yksinkertainen käyttää. Twinerystä voi tallentaa tarinoita vain HTML-muodossa, joten tarvittiin tapa erotella tarvitut osat HTML-koodin seasta. Tähän tarkoitukseen käytettiin ’HTML agility pack’ -nimistä kirjastoa HMTL-koodin parsimiseen. Tämän lisäksi käytettiin C# kielen natiivia XML -parsimiskirjastoa ja säännöllisten lausekkeiden [engl. regular expression] kirjastoa. HTML- ja XML-kirjastoilla saatiin eroteltua tärkeät osat tarinan rakenteesta, ja säännöllisten lausekkeiden hauilla saatiin haettua eri solmujen väliset linkit ja Twineryssä tehdyt muuttujat.

4.5 Parsiminen

Twinery-tiedostosta parsittiin jokaiselle solmulle otsikon, nimen, kuvan, linkit muihin solmuihin, lukkomuuttujat, muuttujan vaihdosoperaatiot, avainsanat ja solmun tyypin. Otsikot ja kuvat merkittiin Twineryyn yksinkertaisesti kirjoittamalla muuttujan nimen ja yhtäsuurikuin-merkin ja arvon nimen esimerkiksi 'Heading=Family'. Twinery asetti jokaiselle solmulle oman ID-numeron ja vaati jokaiselle solmulle uniikin nimen, jotta se toimisi. Linkit parsittiin suoraan Twineryn merkintätavasta, jossa hakasulkujen sisään merkitään ensin linkin teksti ja sitten '|' merkin jälkeen linkitettävän solmun nimi [esimerkkikoodi 1]. Twinery-tiedostolle tehtiin ensin alustavan muokkauksen, jossa poistettiin HTML-notaatio ja linkit tehtiin selvemmiksi. HTML Agility Pack -kirjastolla parsittiin tekstistä kaikki kappaleet erikseen käsittelyyn ja XML-parserilla parsittiin tarkemmat alueet tekstin sisältä.

```
<tw-passagedata pid="5" name="Reason for coming" tags="" position="1202,1400"
size="100,100">
Icon=(Question)
Heading =(Reason for seeking therapy)

[[1. So what brings you to me, Henry?|A1 Reason for coming]]
[[2. It says here in my file you&#39;ve been depressed for some time now?|A2
Reason for coming]]
```

```
(if: $partyanimal is true)[[[3. So do you like to party Henry?|A3 Reason for
coming]]]
(if: $feelinglow_opened is true)[[[Feeling low|Feeling Low /Reason for com-
ing]]]
</tw-passagedata>
```

Esimerkkikoodi 1. Esimerkki parsimattomasta HTML koodi formaatista, mitä Twinery käyttää tallentamiseen.

Säännöllisten lausekkeiden parsimista käytettiin, koska tekstissä oli paljon toistuvia malleja, jotka eivät kuitenkaan olleet HTML tai XML notaatioiden tyylisiä. Esimerkiksi linkkien rakenne ja muuttujat olivat tällaisia kohtia. Tyypillinen linkin rakenne ‘[[Linkin nimi | Kohteen nimi]]’ parsittiin säännöllisten lausekkeiden mallilla ”[[(\d+\..*)|(.*?)]]” olettaen, että kyseessä oli kysymys tai ”[[(/*[a-z]+.*)|(.*?)]]”, jos kyseessä oli avainsana. Ensimmäinen malli olettaa, että linkin nimi alkaa numerolla jonka jälkeen tulee piste, sen jälkeen määrittämätön määrä merkkejä, joita seuraa ‘|’ merkki, minkä jälkeen tulee mitä tahansa merkkejä, jotka loppuvat kahteen hakasulkuun. Kysymysmerkki tarkoittaa, että merkkejä lasketaan mukaan mahdollisimman vähän, jos mahdollisuuksia on useampi. Avainsanoja tunnistava malli olettaa, että linkin nimi koostuu pelkästään kirjaimista ja loppuu samalla tavalla kuin kysymysmalli.

```
<tw-passagedata pid="5" name="Reason for coming">
Icon=(Question)
Heading=(Reason for seeking therapy)

[[1. So what brings you to me, Henry?|9]]
[[2. It says here in my file you've been depressed for some time now?|10]]
(if: $partyanimal is true)[[[3. So do you like to party Henry?|147]]]

(if: $feelinglow_opened is true)[[[Feeling low|11]]]
</tw-passagedata>
```

Esimerkkikoodi 2. Parsittu välimuoto yhdestä keskustelun kappaleesta.

Samalla tiedosto muokattiin ihmisluettavampaan välimuotoon [esimerkkikoodi 2.] Linkit olivat uudessa tiedostossa muodossa '[[Linkin nimi | linkitettävän solmun ID numero]]'. Välimuodon parsimisessa luotiin ohjelmassa käytettävä tietorakenne. Solmut luotiin siinä järjestyksessä, missä ne olivat parsittavassa tiedostossa. Samalla ylläpidettiin vanhemmuus sanakirjaa, jonka avulla lisättiin linkit vanhempiin sen jälkeen, kun linkattavat solmut oli luotu. Kaikki solmut lisättiin sanakirjaan, joka tallennettiin tiedostoon, jotta se voitaisiin ladata pelin alussa käytettäväksi.

4.6 Kartan tietorakenne

Keskustelupuun datan lukemista varten oli HTMLParser-luokka, joka luki Twineryn luomasta HTML-tiedostosta tarvittavat tiedot, ja loi niiden perusteella kaksi sanakirjaa: muuttujasanakirjan, johon oli tallennettu string-muotoinen nimi muuttujalle ja boolean-arvo muuttujan arvolle, ja kappalesanakirjan, jossa avaimena oli kappaleen nimi ja arvona PassageNode-luokka, johon oli erikseen tallennettu kaikki kappaleen tarvittava tieto. PassageNode-luokka oli objekti, jonka tarkoitus oli pitää sisällään tallennettavassa muodossa yhden kappaleen tiedot. Kappaleista tallennettiin muistiin nimi, id-numero, kappaleen teksti, linkit lapsikappaleisiin ja vanhempaan ja ikonin numero. Tallennus tehtiin binäärimuodossa, joten sanakirjoja ei voitu tallentaa suoraan sellaisinaan, vaan ne piti konvertoida listamuotoon, jotta ne pystyttiin sarjallistaa C#-kielen BinaryFormatter-luokan avulla. Tämä tarkoitti myös, että data piti muuntaa takaisin sanakirjamuotoon ladatessa. Lataamisen hoiti ConversationTree luokka, jonka tarkoitus oli hallinnoida puurakennetta kokonaisuudessaan.

Näiden luokkien lisäksi peliin tehtiin myös StoryController-luokka, jonka tehtävänä oli käyttää ConversationTree-luokan dataa pelin keskustelujen toteuttamiseen. StoryController-luokka siis luki dataa ja esitti sitä pelaajalle. Se myös hoiti tarinan kulun loogisen ohjauksen, esimerkiksi lukittujen kappaleiden kohdalla.

Visuaalisen toteutuksen puolella ConversationTree-luokkaa luki NodeMapManager-luokka, jonka tehtävänä oli luoda visuaalinen rakenne keskustelupuulle, ja hallinnoida sen toimintoja. NodeMapManager-luokka hallinnoi sitä, mitkä osat puusta näkyivät pelaajalle ja minkä muotoinen visualisointi yleisesti oli. Se myös hallinnoi puun solmujen liikettä ja [esimerkkikoodi 3].

```
void Update()
      {
            foreach(GameObject o in _nodePool)
            {
                  float noise = Mathf.PerlinNoise(o.transform.position.x,
                   o.transform.position.y ) * 2 -1;
                  float of = Mathf.Sin(Time.realtimeSinceStartup) * noise;
                  Vector2 offset = new Vector2(of,of);
                  o.transform.position +=
                              new Vector3(offset.x,offset.y, 0) * 0.005f;
            }
            if(_convoTree)
                  foreach(GameObject go in _nodePool)
```

```
                {
                      Node n = go.GetComponent<Node>();
                      if(n.m_pNode.m_visited == false &&
                      _selectedNode.m_pNode.m_id != n.m_pNode.m_id)
                            n.Highlight(1);
                      else if(_selectedNode.m_pNode.m_id != n.m_pNode.m_id)
                            n.Highlight(0);
                }
    }
```

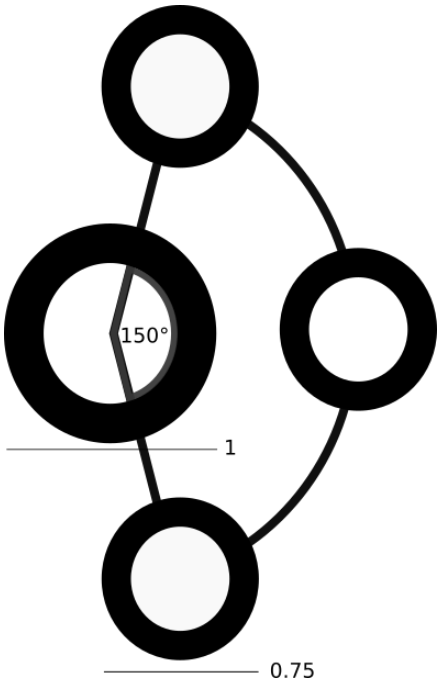
Esimerkkikoodi 3. Kartan päivitysfunktio, joka hallinnoi solmujen liikettä

Kartan yksittäisiä solmuja varten oli Node-luokka. Sen tehtävänä oli hallinnoida solmun ulkonäköä ja toimintoja. Node-luokka piti sisällään solmun tilan tiedot ja funktiot sille, mitä tapahtuu, kun solmua painetaan. Solmua painettaessa tarina siirtyi solmun kohdalle StoryController-luokassa, solmun ikonin ympärille laitettiin korostustehoste ja soitettiin ääni.

4.7 Kartan rakentaminen pelissä

Kartan rakentaminen pelissä alkoi tallennetun rakenteen lataamisesta muistista. Ladattu rakenne passattiin rekursiiviseen funktioon, joka loi juuresta käsin jokaisen solmun yksi kerrallaan ja yhdisti solmun viivalla vanhempaansa.

Toisen tason solmujen position laskettiin jakamalla ympyrä tasaisiin lohkoihin, joista jokaiseen asetettiin yksi solmu. Sovittiin, että toisen tason solmuja olisi enintään viisi, että tila riittäisi varmasti. Tästä jaosta pääteltiin kaikille syvemmän tason lapsille, mihin suuntaan niiden pitäisi jakaantua verrattuna vanhempaansa. Tämä tehtiin yksinkertaisella vektorilaskulla (nykyisen solmun positio – vanhempisolmun positio). Päätettiin antaa kaikille toisen tason jälkeisille lapsille 150 asteen kulman, joka jaettiin tasan niiden kesken. Jokaista lasta skaalattiin pienemmäksi vanhempaansa verrattuna (1/1.35) kertoimella [kuva 15.] Lapsisolmut olivat siis kooltaan lähes 75 % vanhempansa koosta.



Kuva 15. Esimerkkikuva solmun asettelusta puun visualisoinnissa toisen tason jälkeen. Kuvassa oletetaan, että laajenemissuunta on suoraan oikealle.

Etäisyys edellisestä solmusta laskettiin kaavalla:

(suunta * oksan pituuskerroin * solmunleveys) / 1.33^rekursionsyvyys

Näin saatiin etäisyys kutistumaan luonnollisen näköisesti solmujen koon mukaan.

4.8 Ongelmat ja ratkaisut

Koska keskustelun ohjaaminen tarvitsi paljon erilaisia loogisia sääntöjä, kappaleiden kirjoittamisesta tuli liian monimutkaista ja virhealtista käsikirjoittajalle. Yksinkertaisempi järjestelmä olisi säästänyt paljon aikaa. Säännöllisten lausekkeiden mallien virheiden löytäminen ja poistaminen on hyvin hankalaa, koska se vaatii suurien tekstimäärien analysointia ja jatkuvaa testausta.

Parsimisessa huomattiin, että kirjoitusvirheet tulivat jatkuvaksi ongelmaksi. Uuden keskustelupuun lisäämisprosessia tai vanhan keskustelupuun editointiprosessia olisi auttanut, jos projektin aikana olisi tehty työkalu, joka olisi luonut solmun tekstin oikeaan muotoon sen sijaan, että pelin käsikirjoittajan täytyi muistaa paljon erilaisia merkintäsääntöjä.

Suomen kielen parsimisessa huomattiin, että HTML käsittelee ääkkösiä eri tavalla kuin englannille syntyperäisiä kirjaimia. Myös erikoismerkit kuten &-merkki merkitään HTML-notaatiossa eri tavalla. Nämä merkkiongelmat tulivat esille parsimisessa. Ongelma korjattiin korvaamalla HTML-parsimisen jälkeen merkit oikeilla vastineillaan.

5 Visualisointi

Pelin ideana oli sujuva keskustelu, jossa pelaaja voisi miettiä keskustelun taustalla olevia tekijöitä. Tämän mahdollistamiseksi keskustelussa navigoinnin pitäisi olla helppoa ja nopeaa. Pelaaja tarvitsee myös tehtäväänsä muistiapua, jota ei mielellään tarvitsisi selailla pitkään, koska se rikkoisi pelikokemusta ja työtäisi pelaajia pois pelin parista. Myös peliin uudelleen palaaminen vaikeutuisi ilman muistiapuja, koska olisi helppo unohtaa koko keskustelu, minkä on käynyt ja ajatuksensa siitä, mikä tarkoittaisi, että peli pitäisi aloittaa alusta. Näitä ongelmia ratkomaan tehtiin karttamaisen visualisoinnin keskustelulle, joka toimi myös UI-elementtinä navigointiin ja muistiapuna keskustelun kulusta.

Peleissä visuaalisuus on todella suuressa asemassa. Tämänkin takia kartan piti olla mekaanisen toimivuuden lisäksi visuaalisesti miellyttävä.

5.1 Visuaalisten elementtien tarkoitus

Visualisoinnilla haluttiin antaa pelaajalle heti ensisilmäyksellä kuva siitä, mikä keskustelun aihe on missäkin kohtaa, ja idean siitä, mistä keskustelu oli johtunut. Esimerkiksi keskusteltaessa perhesuhteista pelaaja näkisi heti, että alku aihe liittyy perheeseen ja sillä olisi selkeitä ala-aiheita, jotka kertoisivat nopeasti katsomalla, mihin ne liittyvät. Tätä varten luotiin ikoneja [kuva 15], jotka kuvaavat pelissä käsiteltäviä aiheita. Näiden ikonien avulla saatiin kommunikoitua aihe selvästi ja nopeasti pelaajalle.

Kuva 16. Esimerkkejä pelissä käytetyistä ikoneista. Kaikki kuvan ikonit ovat ihmissuhteisiin liittyvistä aiheista.

5.2 Solmu

Solmujen haluttiin olevan ympyrämäisiä. Niiden keskellä olisi ikoni ja ympärillä olisi tehoste ympyrä. Ensimmäinen toteutusidea tästä oli, tehdä sävytin, jolle annetaan kaikki ikonit järjestetyssä kuvassa, taustaväri-muuttuja ja ikonin numero.

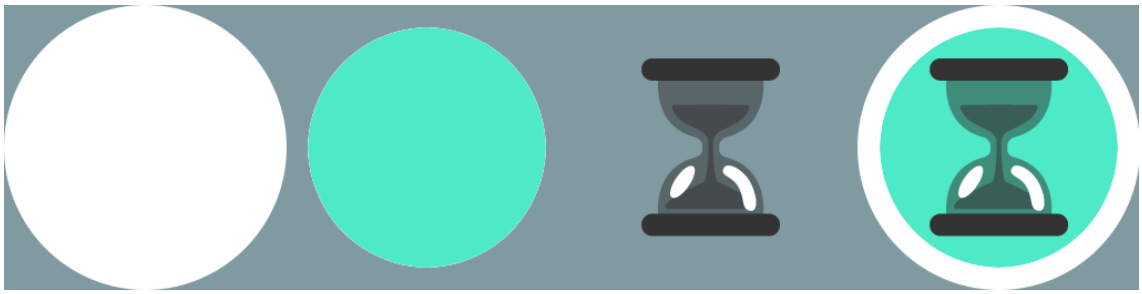
Sävytin laskee ensin ympyrämäisen alueen, ja sitten korostusympyrän alueen sen ympärille. Tämän jälkeen sävytin lisää tehostevärin solmun taustaksi ja viimeiseksi lisää tehosteympyrän solmun ympärille, jos sille on tarvetta. Tässä sävyttimessä on melko paljon laskuja [esimerkkikoodi 4], ja kaikki sävyttimet eivät toimi eri laitteilla samalla tavalla. Sävyttimen käytössä oli myös muita ongelmia ja sen kehitystyö oli hidasta. Näistä syistä solmun visualisointitapa suunniteltiin uudestaan yksinkertaisemmaksi.

```
fixed4 frag (v2f i) : SV_Target
{

      fixed4 icon = tex2D(_IconSheet, i.uv2);
      icon.rgb = (1 - icon.rgb) * _IconColor;
      icon.a = _IconColor.a * icon.a;

      //calculating node circle
      fixed4  node = circle(i.uv, _NodeSize, _NodeSharpness);
      node *= _NodeColor;

      //calculating highlight circle
      fixed4 hl = circle(i.uv, _NodeSize + (_HighlightSize *
      (0.5 - _NodeSize)), _HighlightSharpness);
      hl *= _HighlightColor * _HighlightIntensity * _HighLight;
      hl -= max(node, 1) * node.a;
      hl.a = hl.a * (1-node.a);

      //Isolating node area
```

```
        float nv = 1 - step(node.a, 0);

        //Calculating flat highlight
        float f = 1 - smoothstep(.49, .51, i.uv.x);
        fixed4 ff = fixed4(f, f, f, 1) + .3;
        node += ff * 0.09 * nv;

        ff *= (1-nv);
        hl *= ff;

        icon.a *= nv;

        //adding higlight
        fixed4 result = lerp(node, hl, hl.a-node.a);
        //adding icon on top
        fixed4 res = lerp(result, icon, icon.a);
        res.a -= icon.a * (1 - step(IconIsAlpha, 0));
        return res;
}
```

Esimerkkikoodi 4. Sävytin, jossa luodaan solmun ulkonäkö.

Uudessa toteutustavassa solmu koostettiin useasta eri osasta [kuva 17], jotka olivat pelilogiikan sisällä kaikki omia objektejaan, jotka oli vain aseteltu päällekkäin. Tämä toteutustapa oli paljon sävytintapaa yksinkertaisempi, ja sitä oli nopeampi iteroida.



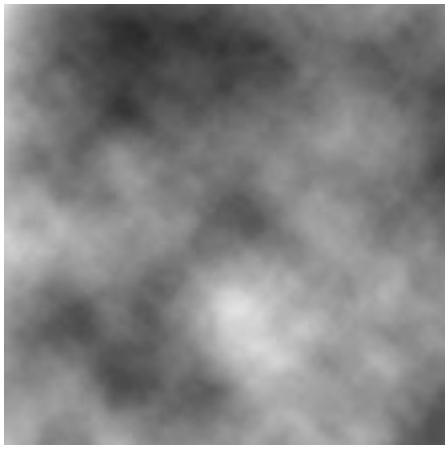
Kuva 17. Solmun visuaalisen rakenteen koostaminen.

Yksi solmu koostuu taustapallosta [kuva 17], jota käytetään korostamaan solmua, jos se on valittu tai sen lapsisolmuissa on jotain uutta. Tämän pallon väriä muuttamalla saadaan aikaan erilaisia huomiotehosteita. Taustan päällä on solmun normaalitausta [kuva 17.] Tämän väri on liitetty aiheeseen, joten vaikka karttaa katsoessa ei näkisi koko ketjua, voi silti päätellä, mihin pääaiheeseen solmu liittyy. Viimeiseksi solmussa on ikoni [kuva 17], jonka tarkoitus on viestiä pelaajalle kyseisein solmun tarkka aihe. Esimerkiksi kuvan 17 ikonin ylin aihe voisi olla työ, ja tarkempi aihe ajankäyttö työssä. Näitä elementtejä yhdistämällä saatiin luotua pisteitä karttaan, jotka kertovat oman osansa tarinasta yksinkertaistettuna. Jokaiseen solmuin oli kanssa liitetty yhdysviiva, joka osoitti, mistä aiheesta solmu oli tullut. Visuaalisten elementtien lisäksi noodeihin lisättiin liikettä, ’Perlin

Noise'-funktion avulla. Tällä tavoin noodeihin saatiin rauhallinen luonnollisen omainen keinuva liike, joka auttaa karttaruudun tunnelmaa ja vähentää sen monotonisuutta.

Väreiksi korostukselle valittiin vihreän, jos pelaaja oli valinnut kyseisen solmun ja sinisen, jos solmun alle oli tullut uusia kysymyksiä tai solmu oli kokonaan uusi pelaajalle. Myös punaista väriä mietittiin osoittamaan nykyistä positiota, koska se on yleisesti käytetty siihen tarkoitukseen, mutta punaisella on peleissä usein negatiivisia konnotaatioita, joten sitä päätettiin olla käyttämättä.

Kartan taustaksi valittiin yksinkertainen sinertävän väri, jonka päällä kulkee hitaasti pilvimäistä usvaa [kuva 18].



Kuva 18. Esimerkkikuva arvokohinasta.

Taustan pilvet toteutettiin yksinkertaisella kohina sävyttimellä [esimerkkikoodi 5], jossa vain lisättiin generoitua arvokohinaa [23] taustaväriin. Arvokohina [kuva 18] oli toteutettu näennäissatunnaisuusfunktiolla, joka olettaa 4 arvoa joka pikselillä, ja palauttaa niiden välillä interpoloidun arvon.

```
fixed4 frag (v2f i) : SV_Target
{
      fixed4 col = tex2D(_MainTex, i.uv);
      float c = gnoise(i.uv * 2. + float2(_CosTime[0],  _SinTime[1]));
      c /= 5;
      fixed4 cloud = fixed4(c, c, c, 1);
      return col + cloud;
}
```

Esimerkkikoodi 5. Fragmentti sävytin, joka ottaa päätekstuurin värin ja lisää siihen arvokohinaa.

Taustan tarkoitus oli nostaa esiin kartan solmut, ja auttaa luomaan tunnelmaa karttaruudussa. Pilvet olivat pieni tehoste auttamaan tunnelmanluomisessa ja myös lisäämään liikettä karttaruutuun.

5.3 Ongelmat

Visualisoinnin rakenteen suunnittelussa ongelmaksi nousi varsinkin tilan tarve isoissa puurakenteissa, ja solmujen sommittelu niin, että ne eivät mene toistensa päälle. Koska visualisoinnin oli tarkoitus toimia myös muistin apuna ja navigointivälineenä, siinä oli hyvä olla mahdollisimman paljon keskustelun tasoja näkyvissä yhdellä kertaa.

Ikoneiden suunnittelussa ongelmaksi nousi yleisesti ymmärrettävien ikonien valinta, koska kuvat voi useimmiten tulkita monella tavalla, ja pelin logiikalle oli tärkeää, että ikoneiden avulla pystyisi ymmärtämään mistä aiheista oli kyse milläkin hetkellä.

6 Mobiililaiteen tuomat ongelmat

Peli tehtiin mobiilialustoille. Tämä toi mukanaan omia haasteitaan kartta elementin puolelta. Mobiilikäyttäjillä on valmiit konventiot siihen, miten he odottavat asioiden toimivan. Esimerkiksi pyyhkäisy ja zoomaus ovat yleisiä normeiksi päätyneitä ohjaus menetelmiä mobiilialustalla.

Ruudun pieni koko on myös ongelmallista, koska kartassa täytyy skaalata noodeja joka kerroksessa pienemmällä, mutta silti pitäisi pystyä erottamaan solmun sisäisen ikonin kuva. Tämän takia ikoneita yritettiin kehittää mahdollisimman yksinkertaisiksi ja selkeiksi.

Koska normaali mobiilikäyttäjä käyttää yleensä puhelintaan pystyasennossa, päätettiin että pelikin on potretti kuvasuhteessa. Tämä tarkoitti sitä, että ruutu on hyvin kapea ja korkea, kun perinteinen tietokoneruutu on leveämpi kuin korkea. Tästä syystä oli tärkeää, että karttaa pystyi liikuttamaan ja zoomaamaan.

Potretti kuvasuhteella olisi ollut mahdollista käyttää perinteistä puudiagrammia, vaihtamalla kasvusuunnan oikealle sen sijaan, että se kasvaisi alas. Perinteisessä mallissa kuitenkin joutuisi tekemään paljon sivuttaista ja pystyliikettä syvissä puurakenteissa, koska puu levenee sitä nopeammin mitä syvemmälle tasot jatkuvat.

Koska kartan oli tarkoitus olla mahdollista olla teoriassa loputtoman suuri ja ainakin niin suuri, että se ei leveyssuunnassa mahtuisi ruudulle potretti kuvasuhteella, sitä varten täytyi mahdollistaa vapaa liikkuminen kartalla. Tämä toteutettiin liikuttamalla kameraa pelaajan vetäessä sormea ruudulla. Koska tämä mahdollisti kartan kadottamisen, tehtiin nappi, jolla kartta palasi takaisin keskelle ruutua.

Mobiililaitteiden tehovaatimusten takia, ei voitu käyttää jälkikäsittely tehosteita pelin ulkonäön viimeistelyyn. Vinjetti jälkikäsittelytehoste olisi auttanut keskittämään pelaajan katsetta ruudun keskelle. Valaistus tehosteilla olisi ollut mahdollista korostaa ja kirkastaa huomion kiinnitykseen tarkoitettuja tehosteita. Solmujen reunat olisivat näyttäneet paremmilta, jos olisi voitu käyttää reunojen pehmennys tehosteita.

7 Lopputulos

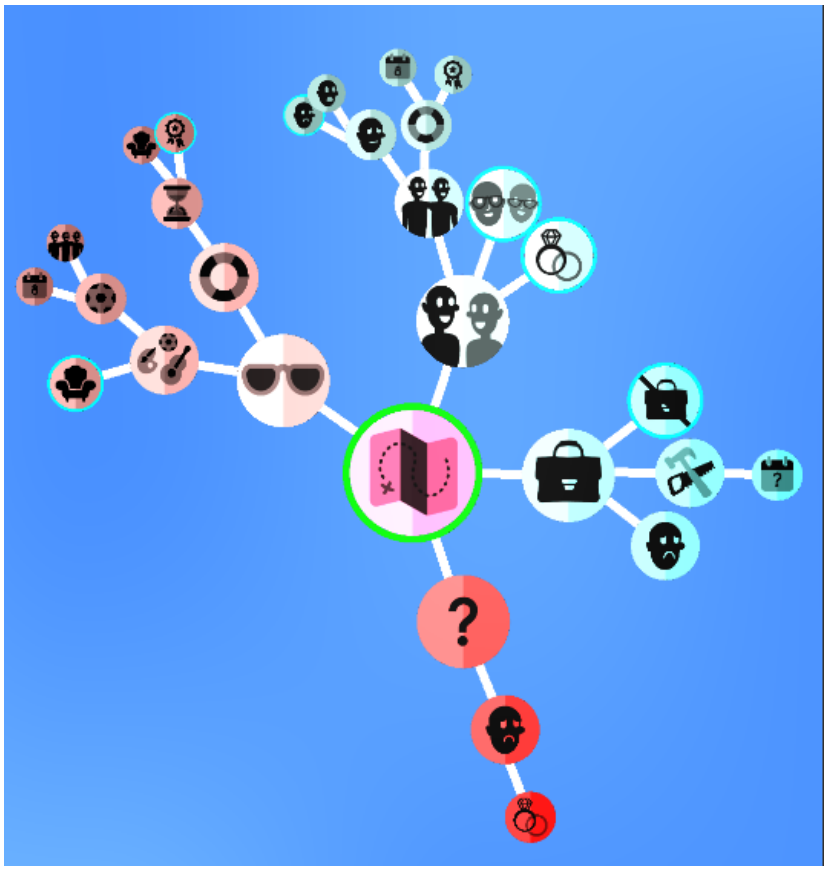
Tässä luvussa käsitellään työn lopputulosta ja arvioidaan sen laatua. Ensin luvussa käsitellään tehtyä toteutusta, ja sen jälkeen miten sitä olisi voitu parantaa.

7.1 Miten meni

Kartasta tuli toimiva kokonaisuus, mutta sen integrointi peliin mekaniikkana ei onnistunut hyvin. Kartta nopeutti navigointia keskustelussa ja näytti, missä päin keskustelua mentiin, mutta ikonien merkitys ja kartan käyttö ei ollut uudelle pelaajalle itsestään selvää.

Visualisoinnin tyypin valinta onnistui mielestäni hyvin. Se saatiin hyväntuntuiseksi käyttää mobiiliympäristössä, ja muoto oli selkeä. Visualisoinnissa solmut olivat tarpeeksi erillään, että niitä pystyi valitsemaan yksitellen ja navigaation apuna käytetty korostusväri tehoste näkyi hyvin.

Mielestäni kartan ikonit onnistuivat kertomaan tarinoita itsekseen [kuva 18], jos pelaaja ymmärsi katsoessaan karttaa ajatella, että tarina alkaa keskeltä suurimmasta kohdasta ja jatkui ulospäin.



Kuva 19. Kuva viimeistellystä kartasta yhdessä pelin keskustelupuista.

Kartan visuaalinen ilme oli miellyttävä ja vaikuttava. Kartan elementtien liike tuntui luonnolliselta, ja taustan pilvisyys loi rauhallista tunnelmaa.

Kartan solmujen värien sitomista aiheeseen ei ehditty toteuttaa. Se olisi ollut tärkeä tapa sitoa useiden keskustelupuiden välein aiheita toisiinsa. Esimerkiksi jos työ aihe olisi sidottu punaiseen väriin, pelaajalle voitaisiin aina viestiä, että punaiset asiat ovat töihin liittyviä kartassa, ja näin kartan tarinankerrontaan olisi saatu uusi ulottuvuus. Nykyinen

toteutus jakoi värispektrin tasaisesti toisen tason solmujen välille, mikä on omalla tavallaan miellyttävää katsojalle, koska kontrastit eri haarojen välein eivät ole suuria, mutta kontrastilla olisi voitu saada voimakkaampia viestejä aikaan.

7.2 Mitä voisi parantaa

Kartan tarinan kulun olisi voinut hahmottaa pelaajalle paremmin. Kaikille pelaajille ei ollut itsestään selvää, että tarinaketju alkaa suurimmasta ja loppuu pienimpään kohtaan. Tämän olisi voinut tehdä esimerkiksi lisäämällä yhdysviivoihin nuolen tai virtaavan visuaalisen tehosteen yhdysviivoihin kulkemaan tarinan kulun suuntaan.

Karttamekaniikan integraation olisi voinut tehdä paremmin. Kartan toimintaa olisi voinut selittää pelin alussa selkeämmin. Pelin aikana olisi voinut antaa visuaalisia merkkejä, että nyt liikuttiin kartassa tai että nyt karttaa olisi hyvä katsoa. Yksi tapa antaa näitä merkkejä olisi ollut näyttää asiakkaan vastatessa aihe ikoneja ruudulla, ja animoida ne menemään karttaan.

Uusia aiheita kartassa olisi voinut animoida niin, että ne olisivat selvemmin huomattavissa pelaajalle. Näin ne olisivat vetäneet paremmin pelaajan huomion. Tämä olisi kuitenkin tuonut pelaajalle sellaisen tunteen, että hänen täytyisi käydä kaikki aiheet läpi, mikä ei ollut kartan tarkoitus.

Taustan pilvet olisi voineet käyttää FBM-sävytintä paremman pilvi tehosteen aikaansaamiseksi tai vaihtoehtoisesti yksinkertaisia pilvikuvia, jotka sopisivat pelin yksinkertaiseen tyyliin. Taustan väriä ja tunnelmaa olisi myös voinut sitoa pelin asiakkaan mielentilaan, jolloin kartta olisi tuntunut enemmän osalta pelimaailmaa.

Karttaan olisi ehkä voinut liittää avustus toiminnon tai ohjeen, jonka avulla pelaaja saisi selitykset kartan ikoneille ja väreille katsellessaan sitä. Tämä olisi auttanut selittämään värien tarkoituksia, jos ne olisi sidottu aiheisiin ja myös mahdollistanut ikonien selittämisen niin, että kaikilla pelaajilla olisi sama lähtökohta karttaa tulkittaessa.

Kartan skaalautuvuutta voisi parantaa laskemalla tarkat arvot sille, miten paljon etäisyyttä ja skaalausta tarvitsee, että solmut mahtuvat ruudulle menemättä toistensa päälle.

Solmujen skaalaukseen olisi voinut myös lisätä tavan kuvata informaatiota, esimerkiksi kertomalla solmun alla olevan kysymysten määrän sen suhteellisen koon avulla.

Kartan liikuttamiseen olisi voinut lisätä fysiikkalaskuja, joilla solmut reagoivat liikesuuntaansa. Tämä olisi tehnyt kartan liikuttamisesta paremman tuntuista. Mobiililaitteilla tämä olisi kuitenkin voinut aiheuttaa performanssiongelmia varsinkin suurilla keskustelupuilla, koska mobiililaitteilla on selvästi rajoitetummat laskutehot pöytätietokoneisiin verrattuna, ja tämä olisi lisännyt vähintään yhden laskun jokaiselle solmulle liikuttaessa karttaa.

Lähteet

1. Alexander Freed. Branching Conversation Systems and the Working Writer. Gamasutra. <https://www.gamasutra.com/blogs/AlexanderFreed/20140902/224609/Branching_Conversation_Systems_and_the_Working_Writer_Part_1_Introduction.php>. Luettu 9.10.2018.

2. Dialogue tree. Verkkoainesto. Wikipedia. <https://en.wikipedia.org/wiki/Dialogue_tree>. Luettu 8.10.2018.

3. Ahomaa Ossi. 2001. Puukartat. Verkkoaineisto. Helsingin Yliopisto. <https://www.cs.helsinki.fi/u/erkio/klsem01/ahomaa.pdf>. Luettu 13.10.2018.

4. Eliza. Verkkoainesto. Wikipedia. <https://en.wikipedia.org/wiki/ELIZA>. Luettu 10.10.2018.

5. Puut. Verkkoaineisto. Tampereen yliopisto. <http://www.sis.uta.fi/~tira/lectures/tira4.pdf>. Luettu 11.10.2018.

6. Everything you need to know about tree data structures. Verkkoaineisto. <https://medium.freecodecamp.org/all-you-need-to-know-about-tree-data-structures-bceacb85490c>. Luettu 13.10.2018.

7. Tree diagram. Verkkoaineisto. The Center for Human Rights and Global Justice and Tandon School of Engineering at New York University. <http://visualizingrights.org/kit/charts/tree-diagram.html>. Luettu 9.10.2018.

8. Zhukov Leonid. 2014. Visualizing Family Trees. Verkkoaineisto. Ancestry <https://blogs.ancestry.com/ancestry/2014/01/17/visualizing-family-trees/>. Luettu 14.10.2018.

9. Tree diagram. Verkkoaineisto. The Data Visualisation Catalogue. <https://datavizcatalogue.com/methods/tree_diagram.html>. Luettu 8.10.2018.

10. What is a Treemap? Verkkoaineisto. TIBCO Spotfire. <https://docs.tibco.com/pub/spotfire/6.5.0/doc/html/tree/tree_what_is_a_treemap.htm>. Luettu 9.10.2018.

11. Treemap. Verkkoaineisto. The Data Visualisation Catalogue. <https://datavizcatalogue.com/methods/treemap.html>. Luettu 16.10.2018.

12. Circle packing. Verkkoaineisto. The Data Visualisation Catalogue. <https://datavizcatalogue.com/methods/circle_packing.html>. Luettu 16.10.2018.

13. Gregg Brendan. 2017. Flame Graphs vs Tree Maps vs Sunburst. Verkkoaineisto. <http://www.brendangregg.com/blog/2017-02-06/flamegraphs-vs-tree-maps-vs-sunburst.html>. Luettu 14.10.2018.

14. Plisson Fabien. 2017. Creating a D3js Sunburst plot – tutorial. Verkkoaineisto. <https://www.fabienplisson.com/d3-sunburst-tutorial/>. Luettu 14.10.2018.

15. 2011. Visualize Hierarchical Data with a Bubble Tree. Verkkoaineisto. The Why Axis. <http://thewhyaxis.info/bubble-tree/>. Luettu 14.10.2018.

16. Hyperbolic trees. Verkkoaineisto. InfoVis:Wiki. <https://infovis-wiki.net/wiki/Hyperbolic_trees>. Luettu 15.10.2018.

17. McLeod Saul. 2017. Defence Mechanisms. Verkkoaineisto. SimplyPsychology. <https://www.simplypsychology.org/defense-mechanisms.html>. Luettu 12.10.2018.

18. Walker Sherelle. 2012. Using Stories to Teach: How Narrative Structure Helps Students Learn. Verkkoaineisto. Fast ForWord <https://www.scilearn.com/blog/using-stories-to-teach>. Luettu 9.10.2018.

19. Hoffman Erin. 2015. Precision of Emotion: A New Kind of "Fun" Approach in Educational Games. Verkkoaineisto. Game Developers Conference. <https://www.youtube.com/watch?v=FP-LNRtwpb8>. Luettu 10.10.2018.

20. Unity (game engine). Verkkoaineisto. Wikipedia. <https://en.wikipedia.org/wiki/Unity_(game_engine)>. Luettu 4.11.2018.

21. Twinery. Verkkoaineisto. <twinery.org>. Luettu 11.10.2018.

22. Aisch Gregor. 2011. Bubble Tree Library. Verkkoaineisto. <http://old.driven-by-data.net/about/interactive-bubbletree/#/0 >. Luettu 8.10.2018.

23. Value noise. Wikipedia. <https://en.wikipedia.org/wiki/Value_noise>. Luettu 13.10.2018.

24. Sondag Max. Speckman Bettina. Verbeek Kevin. 2017. Stable Treemaps via Local Moves. Verkkoaineisto. IEEE <https://ieeexplore.ieee.org/document/8019841>. Luettu 11.11.2018.

25. Li-Wei Gong. Yi Chen. Xin-Yue Zhang. Yue-Hong Sue. 2013. A Hierarchical Data Visualization Algorithm: Self-Adapting Sunburst Algorithm. Verkkoaineisto. IEEE. <https://ieeexplore.ieee.org/document/6689415>. Luettu 11.11.2018.

26. Isenberg Petra. Visualizing trees and graphs. Inria <https://aviz.fr/wiki/uploads/TeachingVA2017/Lecture14-TreesAndGraphs.pdf>. Luettu 11.11.2018.