Leena Karjalainen

FUNGUS: EPÄLINEAARISEN TARINANKERRONNAN TYÖKALU

Digiosaajaksi työelämään -hanke

Opinnäytetyö
Tietojenkäsittely

2018

Kaakkois-Suomen ammattikorkeakoulu

Tekijä/Tekijät	Tutkinto	Aika
Leena Karjalainen	Tradenomi (AMK)	Toukokuu 2018
Opinnäytetyön nimi Fungus: Epälineaarisen tarinankerronnan työkalu Digiosaajaksi työelämään-hanke		45 sivua 2 liitesivua
Toimeksiantaja Kaakkois-Suomen ammattikorkeakoulu, Digiosaajaksi työelämään -hanke, Mika Letonsaari		
Ohjaaja Jukka Selin		

Tiivistelmä

Opinnäytetyössä tutkitaan Fungus-työkalun soveltuvuutta Työelämäpeliin. Opinnäytetyön toimeksiantaja on Digiosaajaksi työelämään -hankeen projektipäällikkö Mika Letonsaari. Opinnäytetyössä tutkittava Fungus on Unity-pelinkehitysympäristöön integroitava työkalu.

Työssä ei keskitytä siihen, kuinka Unitya käytetään, vaan enemmänkin Fungus-työkalun soveltuvuuteen ja sen käyttöönottoon.

Tavoitteena on rakentaa kysymyspatteristosta minipeli käyttäen Fungus-työkalua, niin että se on käytettävissä Työelämäpelissä.

Opinnäytetyön tulevissa luvuissa kerrotaan seuraavanlaisesti: Johdannossa kerrotaan lyhyesti toimeksiannosta ja opinnäytetyöstä. Toisessa luvussa tarkastellaan Funguksen ja Twine-työkaluun eroavaisuuksia ja miksi päädyttiin Fungus-nimiseen interaktiivisen tarinankerronnan työkaluun. Kolmannessa luvussa kerrotaan toimeksianto. Neljännessä luvussa perehdytään työkalun perusperiaatteisiin ennen kuin lähdetään tekemään minipeliä. Viides luku on minipelin luominen, jossa käydään läpi, kuinka monivalintakysely tehdään, miten saadaan minipeli kommunikoimaan Työelämäpelin kanssa. Kuudennessa luvussa kerrotaan päätännössä opinnäytetyön lopputuloksista.

Opinnäytetyössä havaittiin, että Fungus työkalu olisi hyvä erilaisille oppijoille, jotka ovat kiinnostuneita pelisuunnittelusta ja interaktiivisesta vuorovaikutuksesta. Tämä osoittautui, että Twine ja Fungus työkalujen eroavaisuudet olivat huomattavat. Sillä Fungus työkalu osoittautui hyväksi visuaalisen tarinankerronnan työkaluksi.

Opinnäytetyön materiaalia on käytetty Jufo1 -konferenssijulkaisussa.

Asiasanat

Unity,Twine, Fungus, epälineaarinen tarinenkerronta, hanke

South-Eastern Finland University of Applied Sciences

Author (authors)	Degree	Time
Leena Karjalainen	Bachelor of Business Administration	May 2018

Thesis title

Fungus: a non-linear storytelling tool

45 pages
2 pages of appendices

Commissioned by

Mika Letonsaari

Supervisor

Jukka Selin

Abstract

The objective of this thesis was to study the suitability of the nonlinear storytelling tool Fungus for a game about worklife. The study was commissioned by the project called Digi-osaajaksi työelämään. The thesis introduced an integrable tool which would work in Unity.

The Thesis did not focus on how to use Unity. It focused more on the suitability and deployment of the Fungus tool. The objective of thesis was to build a bank of questions to a minigame using, the nonlinear storytelling tool, Fungus, so that it would be available in the game.

The chapters of the thesis can be described as follows: The introduction briefly described the assignment and thesis. The second chapter examined the differences of nonlinear storytelling between the Twine and Fungus tools and why Fungus was chosen to be used as an interactive storytelling tool. The third chapter explained the assignment of the thesis. The fourth chapter introduced the basic principles of the tool before starting to make a mini-game. The fifth chapter went through how to create a mini-game and how to make a multiple choice questionnaire and the end of this chapter thesis explained how to get the mini-game to communicate with the main game. The sixth chapter is the conclusion which explained the final results.

The thesis discovered that the Fungus tool was good for different types of learners, with interest in game design and interactive interaction. It appeared that that the differences between the Twine and Fungus tools were considerable. The Fungus tool proved to be a good visual storytelling tool. The Thesis material has been used in the Jufo1 -conference publication.

Keywords

Unity,Twine, Fungus, nonlinear storytelling, scheme

SISÄLLYS

1 JOHDANTO

Opinnäytetyössä tutkitaan Fungus-työkalun soveltuvuutta Digiosaajaksi työelämään -hankkeen Työelämäpeliin. Tavoitteena on selvittää Fungus-työkalun käytettävyyttä ja sisällön tuottamista.

Toimeksiantajana on Digiosaajaksi työelämään hankkeen projektipäällikkö Mika Letonsaari, joka on käyttänyt aiemmin Twine-työkalua digiprojektin luomiseen.

Opinnäytetyössä perehdytään Fungus-työkaluun, sillä Digiosaajaksi työelämään -hankkeen Työelämäpelissä käytettiin Twineä, joka on myös epälineaarisen tarinankerronnan työkalu ja tehtäväni oli selvittää Funguksen soveltuvuutta samaan käyttötarkoitukseen, jonka Twine on tarjonnut Työelämäpelissä.

Opinnäytetyössä kerrotaan, kuinka Fungusta käytetään, millainen se on käyttöliittymältään ja kuinka rakennetaan toimeksiantajan kysymyspatteristosta minipeli. Työelämäpelissä käytettiin aikaisemmin Twine-työkalua (Twine, 2009). Lisäksi tutkitaan mitkä ovat Twinen ja Funguksen eroavaisuudet.

Funguksen ja Twinen kaltaisten interaktiivisen tarinankerronta työkaluja voitaisiin soveltaa ohjelmointiin ja luovaan kirjoittamiseen lisäksi pelisuunnitteluun. Interaktiivisen tarinankerronta työkalu, kuten Fungus olisi hyvä työkalu erilaisille oppijoille, jotka ovat kiinnostuneita pelisuunnittelusta ja interaktiivisesta vuorovaikutuksesta.

Opinnäytetyössä perehdytään aluksi siihen, mitä epälineaarinen tarinankerronta tarkoittaa yleistasolla, minkä jälkeen tutustutaan kahteen epälineaariseen tarinankerronnan työkaluun. Opinnäytetyössä perehdytään syvemmin Fungus-työkaluun ja toimeksiantoon sekä lopuksi minipelin toteutus Fungus-työkalulla. Päätännössä kerrotaan tarkemmin opinnäytetyön lopputuloksista.

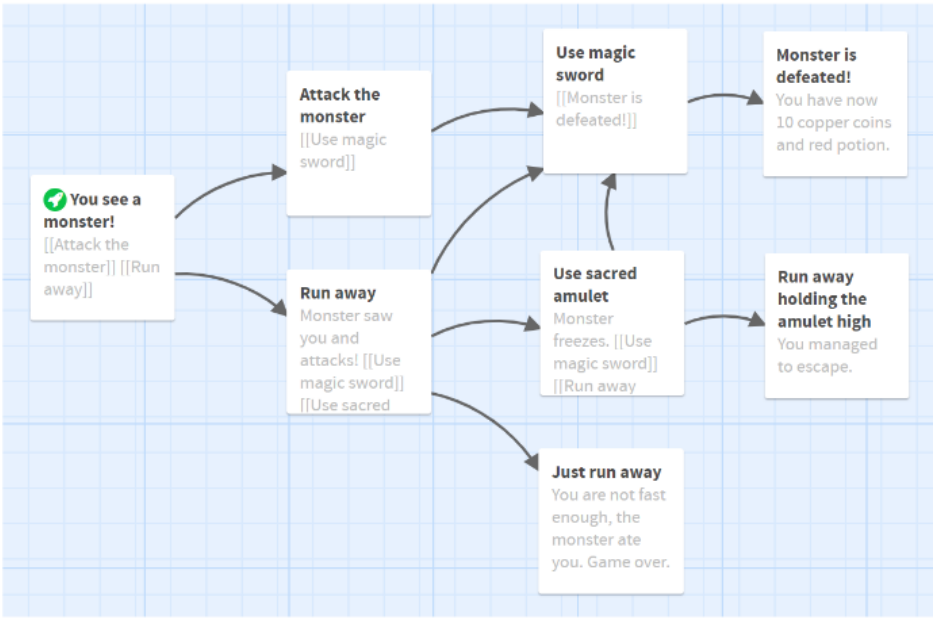
2 EPÄLINEAARISET TARINANKERRONNAN TYÖKALUT

Seuraavaksi kerrotaan mitä epälineaarinen tarinankerronta on, mutta ensin on selvitettävä mitä lineaarinen tarinankerronnalla tarkoitetaan.

Lineaarinen tarina tarkoittaa, että tarinan juoni ei haaraudu. Pelaajalla, lukijalla ja käyttäjällä ei ole mahdollisuutta tapahtumien kulkuun tai tehdä valintoja (kuva 1).

Kuva 1. Lineaarinen tarinan kerronta (Letonsaari ym Selin 2018)

Epälineaarisella tarinalla juoni haarautuu. Pelaajalla, lukijalla ja käyttäjällä on mahdollisuus vaikuttaa tarinan kulkuun (kuva 2).



Kuva 2. Epälineaarinen tarinan kerronta (Letonsaari ym. 2018)

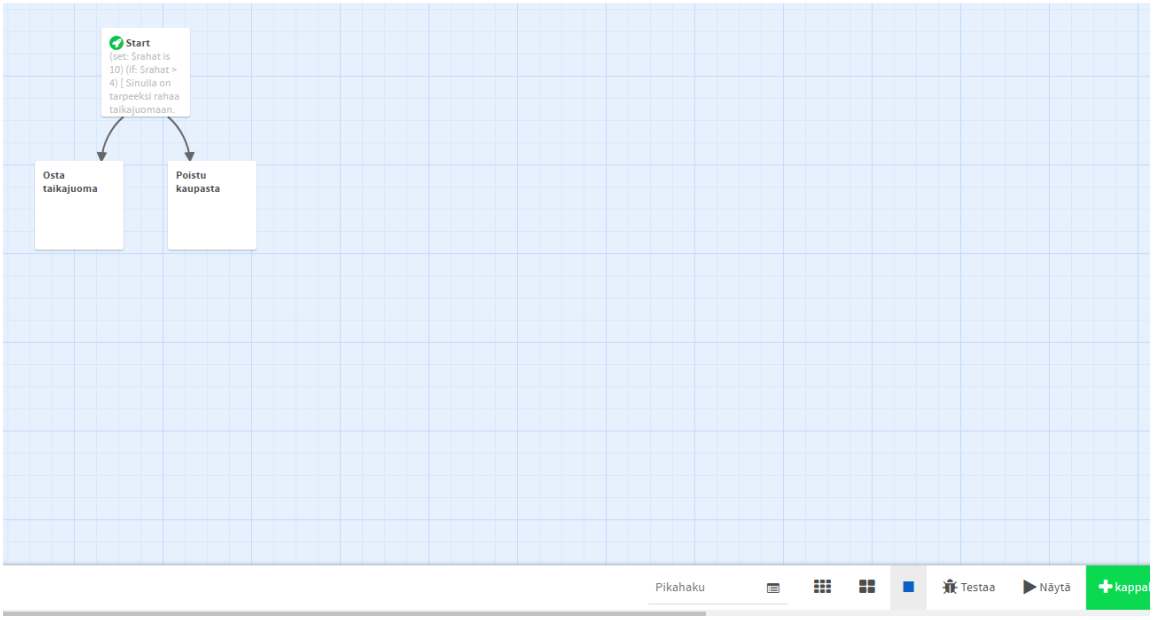
Epälineaarisia tarinankerrointa työkaluja on kattava määrä, mutta opinnäytetyössäni käydään läpi seuraavaksi Twine ja Fungus -työkalut. Twineä käytettiin Digiosaajaksi työelämään -hankkeen yhtenä välineenä. Työssä ei koiteta syrjäyttää Twine työkalua. Opinnäytetyössäni pyrittiin esittelemään epälineaarisen tarinankerronnan työkalu Fungus, jota käyttämällä voidaan tukea erilaisia oppijoita.

Fungus ja Twine työkalujen eroavaisuudet

Digiosaajaksi työelämään -hankkeen Työelämäpeliin käytettiin epälineaarista tarinankerronnan työkalua Twineä. Tässä luvussa käydään lyhyesti läpi Funguksen ja Twinen eroavaisuudet. Lisäksi käsitellään, miksi päädyttiin juuri Fungus-työkaluun.

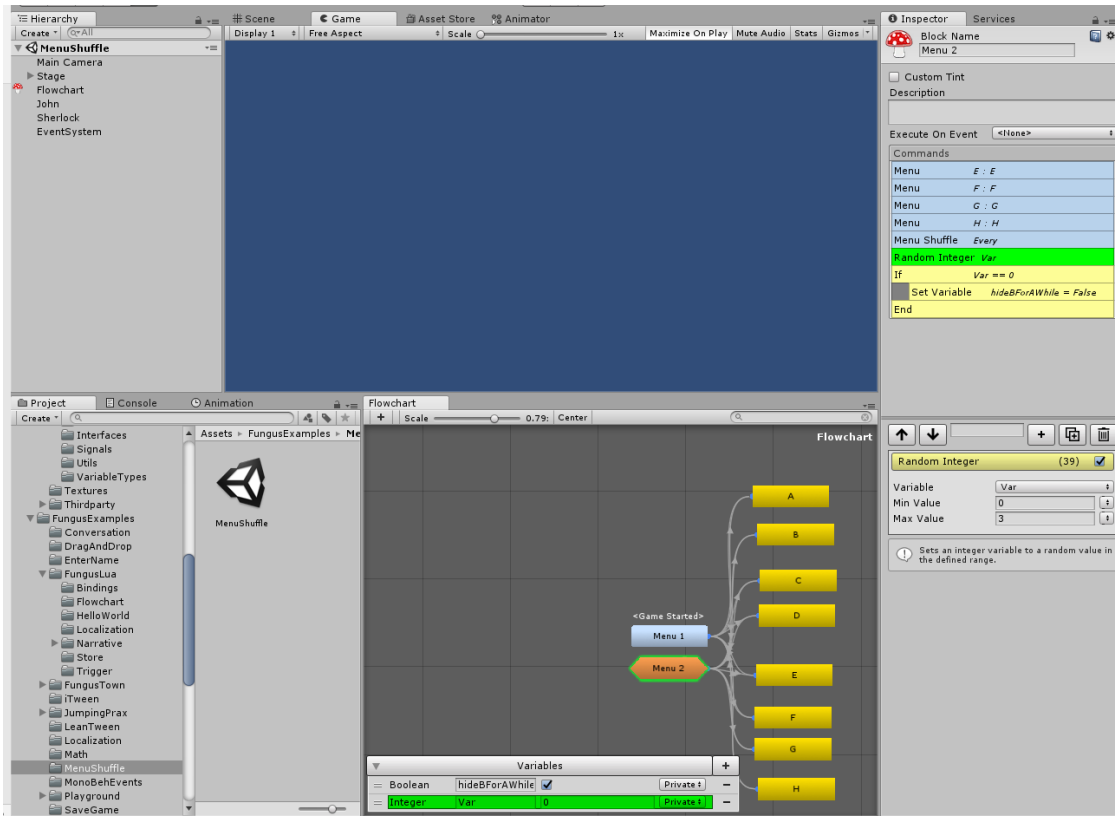
Kehitysalustat

Twine (kuva 3) ja Fungus ovat vuokaaviopohjaisia käyttöliittymältään, joten kummallakin pystyy ketterästi tekemään vuorovaikutteista sisältöä. Työkalujen eroavaisuudet ovat merkittävät. Twine on selainpohjainen, joka on epälineaarinen tarinankerronnan työkalu, ja sillä voidaan tehdä tween-tiedostoja. Twinellä pystytään siis tekemään heti pelattavia projekteja onlinessa. (England 2015).



Kuva 3. Twinellä on vuokaaviopohjainen käyttöliittymä

Funguksen kehitysalusta on Unity-pelimoottorissa. Fungus keskittyy siihen, että vuorovaikutteista sisältöä voidaan tehdä 2d- ja 3d-maailmoissa. Fungus on jo itsessään vuokaaviopohjainen työkalu (kuva 4).

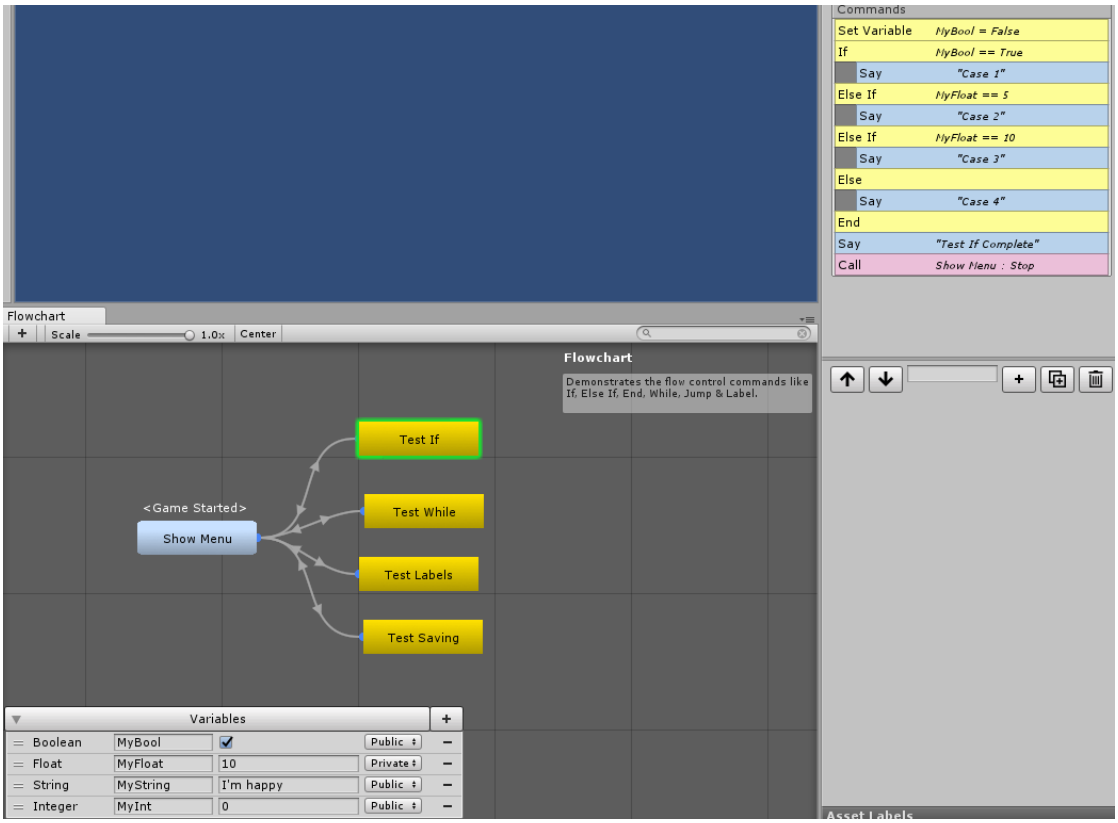


Kuva 4. Fungus työkalun käyttöliittymä

Twine taas tarvitsee Unityssä lisätyökalun Cradle (Github 2017), joka tukee Twine-pohjaisia tarinoita, kun Funguksella taas voidaan luoda suoraan tarinoita Unityssa.

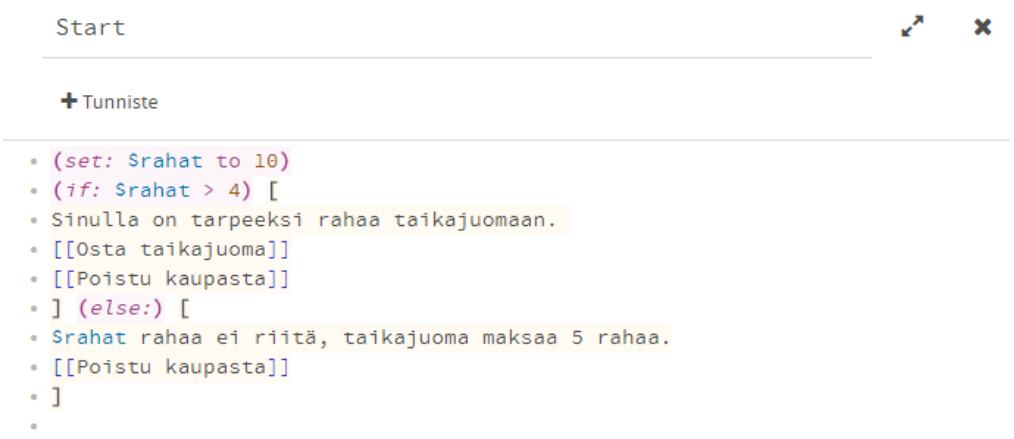
Ohjelmointi

Twinellä ja Funguksella on useita ohjelmointiin liittyviä ominaisuuksia, joissa on kuten muuttujia ja ehtolauseita. Fungus on esimerkkejä ehtolauseista ja muuttujista, jotka tulevat työkalun mukana. Fungus-työkalun mukana tulee esimerkkejä ehtolauseiden ja muuttujien käytöstä: Lataamalla Unityyn Fungus työkalun, työkalun mukana tulee kansio Fungus-esimerkeistä *FungusExamples*, jossa on esimerkkejä muuttujista ja ehtolauseiden toimivuudesta (kuva 5).



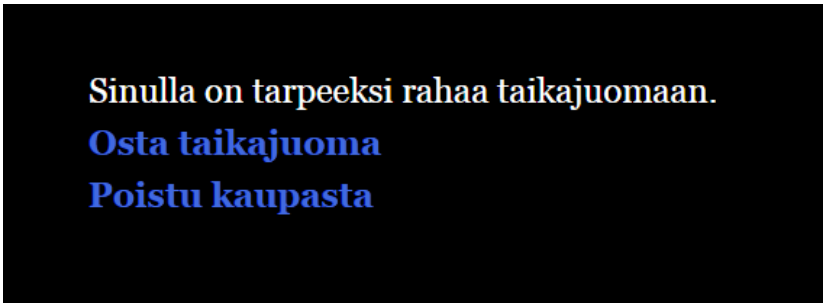
Kuva 5. If-ehtolauseen Funguksessa

Twinessä (kuva 6) muuttujien käyttö ja komennot kirjoitetaan tekstin sekaan, kun taas Funguksella niille on lohkot, joita voi klikata ja raahata sekä vähän muokata (England 2015).

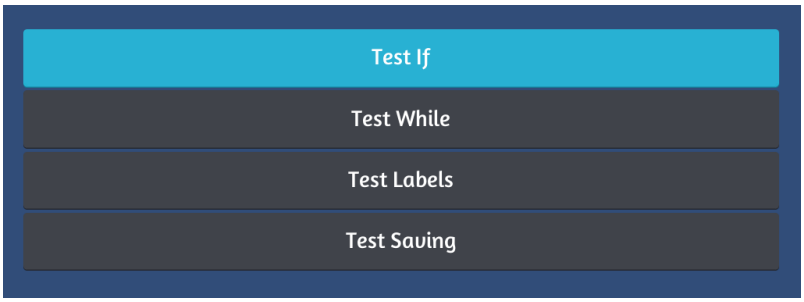


Kuva 6. Twinen Start solmun if-ehtolauseen asettaminen

Funguksen ja Twinen projektien ulkonäköjen muuttamisessa on muutamia huomattavia poikkeavuuksia (kuva 7). Esimerkiksi Twine on selainpohjainen ja omaan projektiin voi siis ladata erilaisia teemoja tai voidaan luoda omia CSS-tyylejä. (England 2015.)



Kuva 7. Twinen ulkonäkö



Kuva 8 Funguksen ulkonäkö

Funguksella on mahdollista muuttaa mitä tahansa pelielementtejä, joita Unity-projektissa esiintyy (kuva 8).

Rajoitteet

Funguksella ja Twinella on myös rajoitteita. Funguksen ikävin puoli on se, että työkalu on keskittynyt ainoastaan Unitylle, kun taas Twinellä pystyy tekemään tekstipohjaisia pelejä verkossa. Twinen kohdalla Unityyn tuodessa tarvitaan ylimääräisiä työkaluja siihen, että Twine-tiedostoja voidaan muuttaa Unity Projekteihin. (England 2015.)

Positiivista on se, että Fungus ja Twine ovat ilmaisia ja avointa lähdekoodia, joten kummatkin työkalut ovat kaikkien saatavilla.

Laitteistovaatimukset

Funguksen kohdalla tarvitaan vain tietokone, joka kykenee käynnistämään Unity 3D -ohjelman ja lataamalla Funguksen Unityyn työkaluksi. (vrt. Unity 2018a). Varsinaisesti mitään kalliita laitteistoa tai ohjelmia ei ole tarpeen, eikä myöskään tarvitse olla suuri ohjelmoijakaan. Twinen kohdalla tarvitaan vain tietokone, jolla pystyy kirjoittamaan ja lukemaan html-tiedostoja. Näiden kahden työkalujen lataamiseen tarvitaan verkkoyhteyttä, jotta käyttäjät voivat itse valita haluamansa työkalun.

Työkaluksi Fungus

Funguksen hyvä moninaisuus on, että peliobjekteille voidaan luoda omia vuokaavioita peliin tai luoda tekstisisältöä responsiivisenpaa muotoon. Funguksella jokainen elementti on muutettavissa ja Funguksella sisältöä voidaan räätälöidä omaan käyttötarkoituksen mukaisesti. Tätä vuokaaviopohjaista työkalua on helppo käyttää 3D- ja 2D-maailmoissa. Fungus-työkalulla voidaan luoda erilaisille Unity objekteille vuokaavioita, jotka voivat olla vuorovaikutuksessa toisiinsa.

3 FUNGUS-TYÖKALU

Seuraavaksi kerrotaan Fungus työkalun kehittäjistä ja mistä oppimateriaalia löytyy. Lopuksi jatketaan Fungus työkalun tutustumiseen ja kuinka työkalu toimii pääpiirteittäin.

Pieni irlantilainen peliyhtiö Snozbot on luonut Fungus-projektin, jossa on kolme perustajaa. Funguksen tärkein kehittäjä ja ylläpitäjä on Chris Gregan, joka on Snozbot perustaja. Snozbot käyttää Fungusta omissa projekteissaan ja pyrkii kehittämään sitä mahdollisimman yksinkertaiseksi muille pelintekijöille. (Snozbot 2018.)

Fungus on vapaata ja avointa lähdekoodia, jolla voi luoda vuorovaikutteisia ja epälineaarisia kerrontoja. Työkalu on integroitavissa Unitylle pelimoottoriin. (Secchi 2016.) Internetsivuilla, http://fungusdocs.snozbot.com/ on laaja valikoima oppimateriaalia, aloittaville ja kokeneille pelinkehittäjille. (Fungus, 2017.)

Fungus on työkalu, jonka avulla voidaan lisätä visuaalista sisältöä ja työkalun voidaan kytkeä erilaisten peliprojektien yhteyteen. Funguksessa on intuitiivinen käyttöliittymä, joka antaa käyttäjälle yksinkertaisen tavan luoda sisältöä ilmaiseksi ja ilman koodausta. Fungus on suosittu kirjailijoiden, kuvittajien, animaattoreiden ja pelisuunnittelijoiden keskuudessa sen lisäksi työkalu on visuaalisten tarinoiden ja interaktiivisten fiktiotekijöiden aktiivisessa käytössä. (Fungus 2017.)

Snozbot (2018) ilmoittaa dokumenteissaan, että interaktiivisen tarinankerronta työkalun pitäisi olla kaikkien saatavilla, joten Fungus on 100-prosenttisesti ilmaista ja avointa. Fungus on ladattavissa ainoastaan Unityn ilmais- ja ammattiversioissa. Fungus v.3.6.1 -dokumenteissa on ilmoitettu, ettei Snozbot peri lisenssimaksua sovelluksista, jotka ovat tehty Funguksella. (Fungus 2017.)

Fungus-työkalun lataaminen Unityyn on yksinkertaista. Unityn projektin luomisen jälkeen Fungus-työkalun voi ladata *Asset Store* eli kaupan kautta (Unity, 2018c).

3.1 Työkalun käyttöliittymään tutustuminen

Fungus ydinperiaate on se, että työkalu on jo itsessään vuokaaviopohjainen. Tällä työkalulla on helppo luoda pelillistä sisältöä Unity-projekteihin ilman koodausta. (Fungus, 2017.)

Vuokaavio

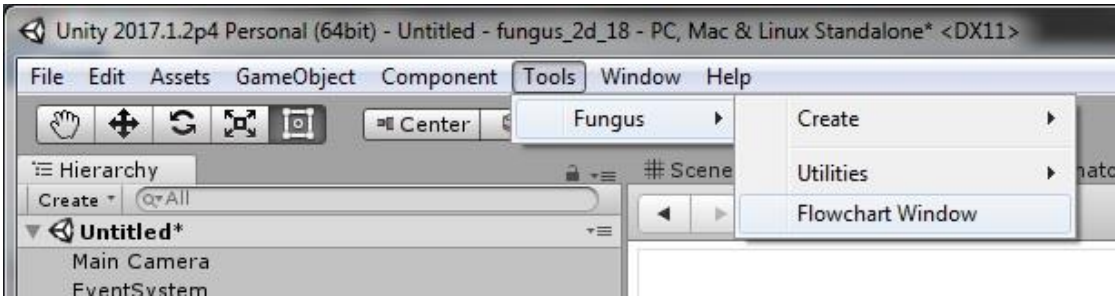
Vuokaavio *Flowchart* on kaavio, joka sisältää yhden tai useamman lohkon. Lohkot taas sisältää yhden tai useamman toteutettavissa olevia komentoja. Unity-kohtauksessa *Scene* voi olla yksi tai useampi vuokaavio, ja vuokaavion komennot voidaan suorittaa samanaikaisesti eri taulukoissa. Monissa eri peleissä riittää kuitenkin yksi vuokaavio ja sen sisältämät lohkot voidaan suorittaa milloin tahansa. (Fungus, 2017.)

Lohko

Lohko *Block* on olennainen käsite Fungus-työkalussa. Lohkot sisällyttävät yhden tai useamman komennon vuokaavion sisällä. Lohkoja on kolmenlaisia; tapahtumalohkot *Event block*, jotka ovat väriltänsä sinisiä lohkoja ja niiden tehtävänä on suorittaa laukaistava toiminto toiseen tapahtumaan. Haaroituslohko *Branching block* on oranssin värisiä ja niiden tehtävänä on hallita kahta tai useampaa muuta lohkoa. Standardoitu lohko *Standard block* on keltainen lohko, joka ei ole tapahtuma ja se voi ohittaa yhden lohkon. (Fungus, 2017.)

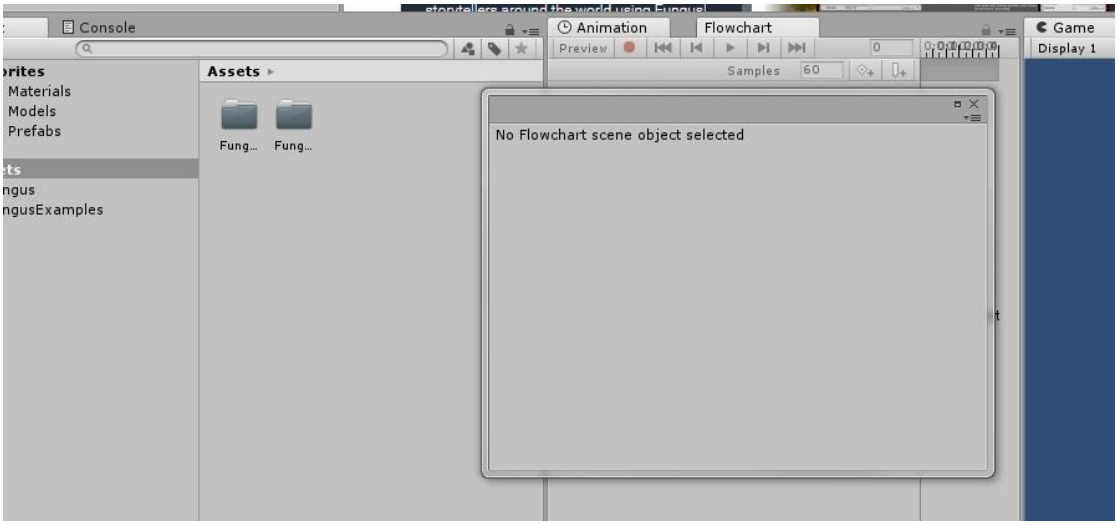
3.2 Fungukseen perehtyminen

Kun valitaan Työkalut *Tools,* löydetään valikon Sieni *Fungus* ja Vuokaavio ikkunan *Flowchart Window* (kuva 9).



Kuva 9. Vuokaavion ikkunan avaaminen (Fungus 2017.)

Vuokaavio ikkuna tulee irrallisena, joten se kannattaa siirtää välilehdeksi, vaikka animaatio ikkunan viereen (kuva 10).

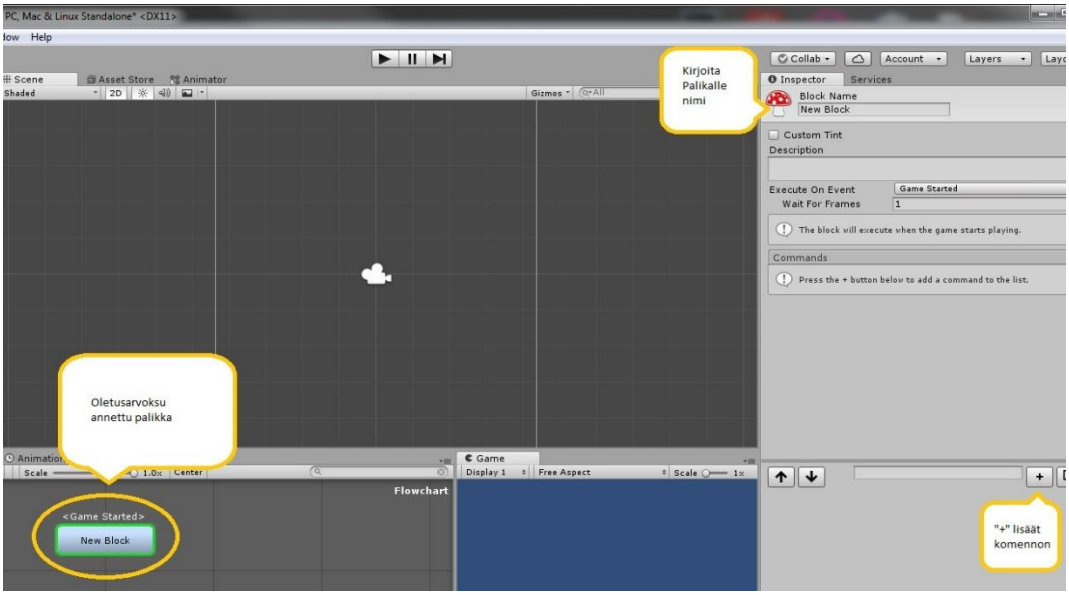


Kuva 10. Vuokaavion ikkunan liittäminen välilehdeksi (Fungus 2017.).

Seuraavaksi samasta työkaluvalikosta luodaan, *Create,* vuokaavio *Flowchart*. (kuva 11).

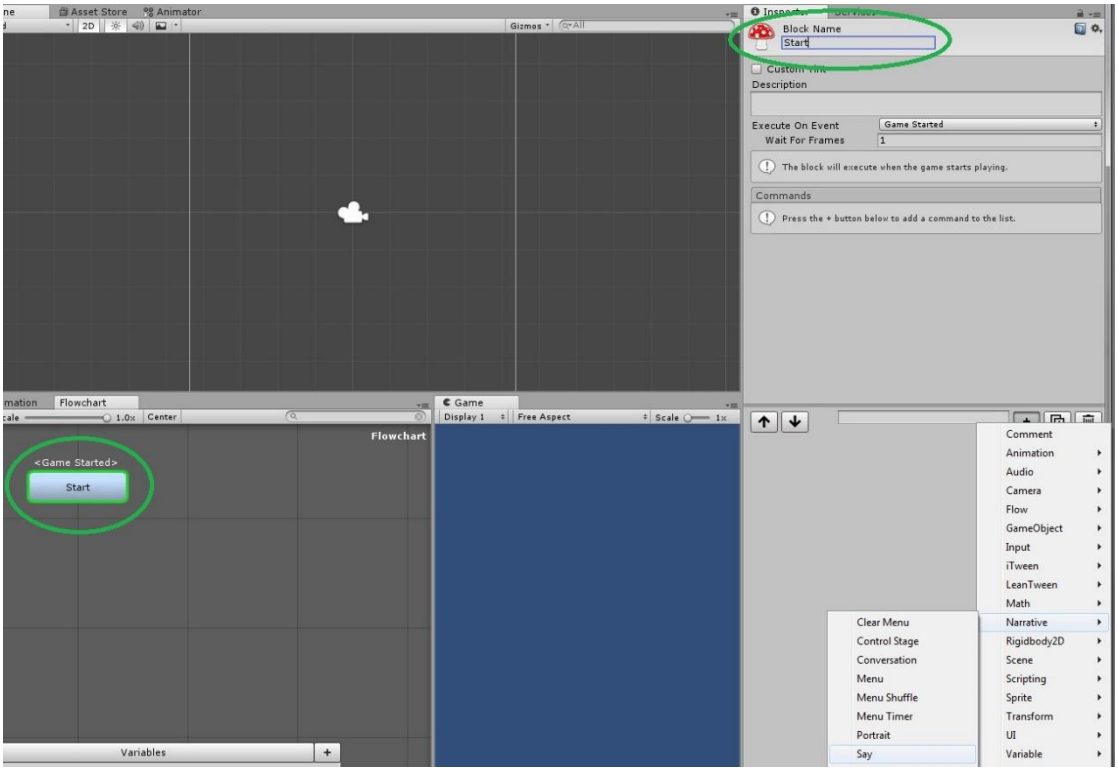
Kuva 11. Vuokaavion luominen (Fungus 2017.).

Kun vuokaavio on luotu, niin vuokaavioikkunaan ilmestyy lohko *Block*. Fungus luo sieni-ikonin kuvaamaan Fungus-työkalun elementtejä. Kuvassa 10 tulee oikealle lohkon nimi *Block name*, jonka voi nimetä Start-lohkoksi. Tällä samalla tarkastaja *Inspector* ikkunassa on alhaalla plusmerkki. Siitä pystyy lisäämään vuokaavion lohkon sisäiset komennot. (Kuva 12.)



Kuva 12. Fungus luo sieni-ikoneita kuvaamaan oletusarvoksi Fungus elementtejä (Fungus-docs 2018)

Vuokaavion tarkastajaikkunaan ja lohkoon tulee automaattisesti muutoksia, kun lohko on nimetty aloitukseksi *Start*, jolloin vuokaavion lohkon nimi muuttuu samannimiseksi *Start*-lohkoksi (kuva 13). Seuraavaksi lohkolle haetaan sisäiset komennot plusmerkillä, josta aukaistaan valikko ja valitaan kerrontakomentojen, *Narrative,* kohdasta Sano-komento, *Say*.

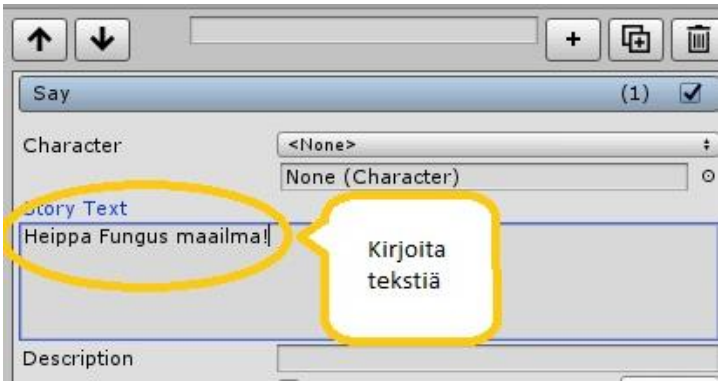


Kuva 13. lohkon nimen muuttaminen ja kerrontakomennon Sano-komento asettaminen (Fungusdocs 2018.)

Komennot, *Commands,* on lista, johon tulevat kaikki halutut komennot esimerkiksi Say-komento eli Sano-komento (kuva 14).

Kuva 14. Komentolistalla näkyy Say -komento

Valitsemalla Sano-komennon aktiivisesti, jolloin aukeaa uusi ikkuna (kuva 15). Sano-komennon sisällä on Tarina-tekstikenttä, *Story text*. Tähän tekstikenttään voidaan kirjoittaa haluttua tekstiä, esimerkiksi: ”Heippa Fungus-maailma!”.



Kuva 15. tekstin syöttäminen tekstikenttään.

Kuva 16. Fungus dialogin Sano-komennon ulostulo.

Kun sano -käsky on haettu ja Unity-projekti käynnistetään aktiivisesti, saadaan haluttu vuoropuhelu, johon tulostuu, ”Heippa Fungus maailma!” -dialogi ikkuna (kuva 16).

4 TOIMEKSIANTO

Esitutkimuksessa määritellään Digiosaajaksi työelämään -hankkeen kehittämistarpeet. Seuraavaksi tehdään Digiosaajaksi työelämään -hankkeeseen digityö Työelämäpelin sisälle minipeliä. Toimeksiantajana on Mika Letonsaari, joka on ollut hankkeessa osallisena digityö Työelämäpelin projektipäällikkö.

Tarkoituksena on helpottaa hankkeeseen osallistuneiden osallistumista, sillä lähtötilanteena on erinomaisesti rakennettu peli, josta puuttuu osallistuneiden liittojen ja yritysten sisältö, joka voitaisiin pelillistää. Toimeksiantaja on antanut minulle Työelämäpelin kysymyspatteriston, joka on kerätty osoitteesta: www.tyoelamapeli.fi –pelistä. (Työelämäpeli s.a, Tki-blogi 2018.)

Digiosaajaksi työelämään hanke oli Euroopan sosiaalirahaston *ESR* -rahoittama hanke. Hankkeen tavoitteena oli parantaa aikuisten tieto- ja viestintäteknisiä taitoja, *TVT-taidot*, sellaisten henkilöiden osalta, joilla perustaidot olivat heikot, ja sitä kautta parantaa Etelä-Savon työllisyystilannetta.

Mikkelin ammattikorkeakoulun nyk. Kaakkois-Suomen ammattikorkeakoulu (XAMK) oli toteuttamassa hankkeessa monipelattavan Työelämäpelin, joka tarjoaa vaihtoehtoisen tavan oppia työelämän TVT-taitoja, työelämän pelisääntöjä, alaistaitoja ja vuorovaikutusta työelämässä. Työelämäpeli toi uutuusarvoa kohderyhmän tarpeisiin vastaamisessa. Peliä kehitettiin jatkuvana käyttäjälähtöisenä prosessina ja kohderyhmät osallistuivat vahvasti sen kehittämiseen.

Varsinaisen pelin tuloksena syntyi. ns. jatkokehitysalusta, työkalu, joka huomioi erilaiset oppijat ja tulevaisuuden kehitystarpeet. (Eura s.a)

Päämääränä on tutustua Unityn pelimoottoriin integroitavaan Fungus nimiseen työkaluun (Fungus, 2017). Opinnäytetyön kannalta tutkin Fungus-työkalun soveltuvuutta sekä voiko työkalua käyttää ketterästi sisällön tekemiseen ja pelejä luomiseen. Tarkoituksena on perehtyä syvemmin tähän Unityyn integroitavaan Fungus-työkaluun. Tässä vaiheessa käyttäjällä kannatta olla ja ymmärrystä, kuinka Unity-pelimoottori asennetaan ja kuinka sitä käytetään (Unity 2018b).

Liitteessä 1 on esitelty vaatimusmäärittely opinnäytetyölle, josta löytyy toiminnalliset vaatimukset, ei-toiminnalliset vaatimukset ja rajoitteet.

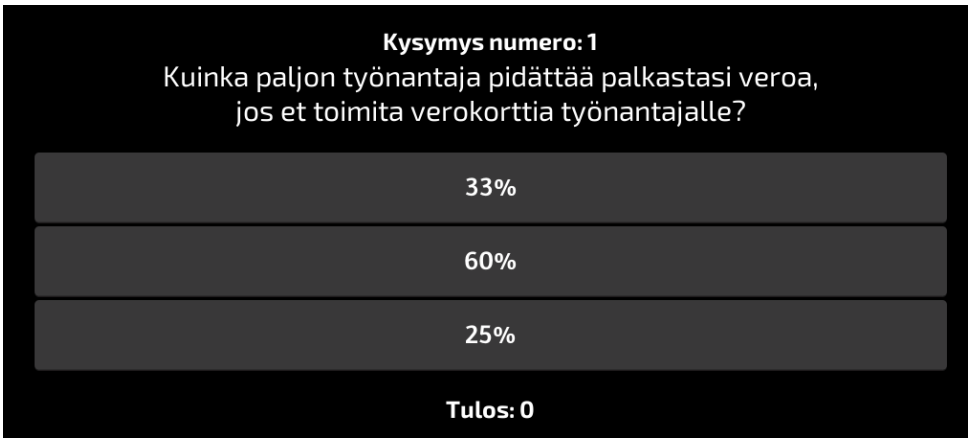
5 MINIPELI

Hankkeessa toteutetun pelin lähdekoodi on ladattavissa https://bitbucket.org/digityo/tyoelamapeli (Tki-blogi 2018). Tämä linkki ohjeistaa käyttäjän suoraan sivulle, josta voi ladata lähdekoodin itselleen ja voi halutessaan rakentaa oman versionsa Työelämäpelistä (kuva 17). Työelämäpeli käyttää bitbuckettia, sillä bitbucket tarjoaa issue traking -järjestelmää eli tätä kautta käyttäjä voi ilmoittaa pelistä löytyneistä bugeista ja virheistä sekä esittää omia parannusehdotuksiaan. (Bitbucket, 2018.)

Kuva 17. Työelämäpelin näkymä

5.1 Minipelin rakentaminen

Seuraavaksi kerrotaan, että kuinka minipeli on toteutettu toimeksiantajan kysymyspatteristosta. Kysymyspatteristosta on tarkoitus rakentaa kolmen vastausvaihtoehtoinen minipeli, jossa on pisteytysjärjestelmä. (Kuva 18.)

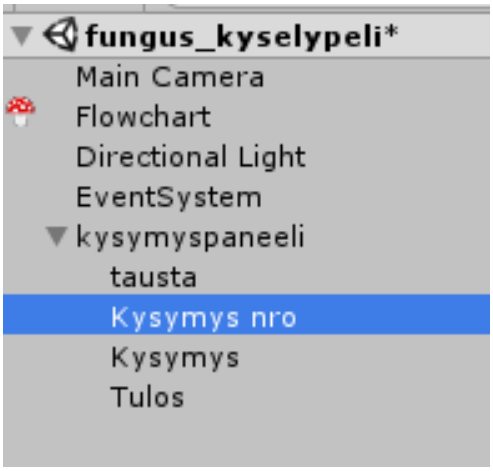


Kuva 18. Kysymyspatteristosta rakennettu minipeli

Vastausvaihtoehtoja on kolme kappaletta, joista kaksi vastausvaihtoehdoista on väärin. Oikeasta vastausvaihtoehdosta saa neljä lisäpistettä ja väärin vastauksista menettää kaksi pistettä. Kysymyksiä on kymmenen, joten korkein tulos on neljäkymmentä pistettä.

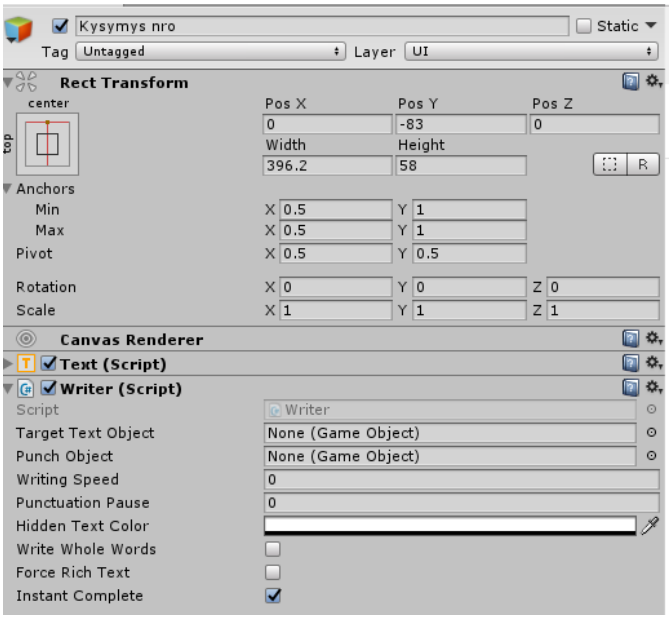
UI-elementit

Unityssa luodaan Canvas-objekti, jonka nimeksi on asetettu kysymyspaneeli. Tämän objektin sisälle laitetaan seuraavanlaiset objektit: tausta, kysymys nro, kysymys ja tulos (kuva 19).



Kuva 19. Teksti -objektit: Kysymys nro, tulos ja Kysymys -objektit

Näiden jokaisen objektien tarkastajien alle tulee kirjoittaja, *writer*-komponentteja. Tässä *writer*-komponentissa on muutamia asetuksia, joita on otettava huomioon. Nämä ovat kirjoitusnopeus *writing speed* ja välimerkin pysäytys *Punctuation Pause.* (Kuva 20.)

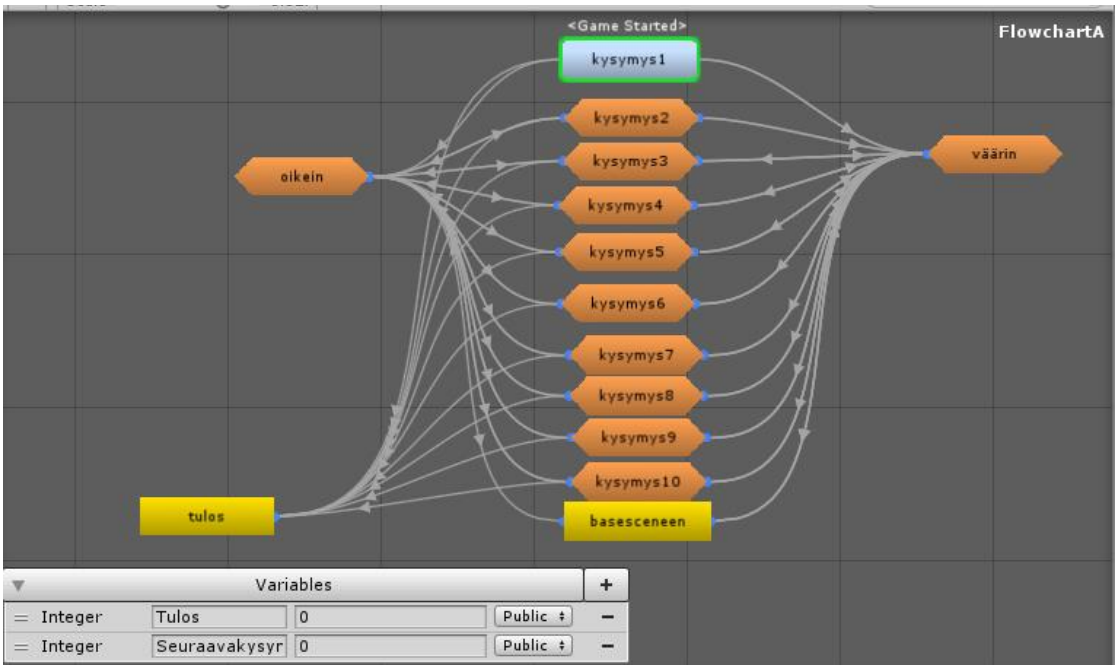


Kuva 20. Tarkastajaikkuna, writer-komponentti

Kirjoitusnopeuden ja välimerkin pysäytyksen arvot on pidettävä nollassa ja aktivoimalla, *Instant Complete*, jotta saadaan tekstin heti valmiiksi näkyviin. Näillä asetuksilla jokainen objekti, joka on tekstiä, pysyy automaattisesti esillä. Tämän vuoksi lisäanimaatiota ei tarvita.

5.2 Lohkot ja sisäiset komennot

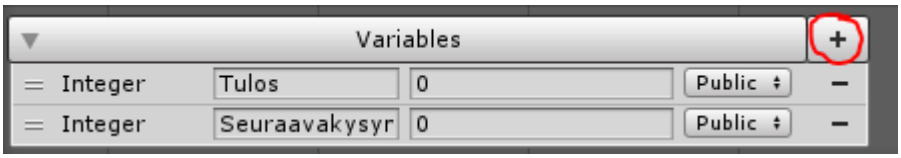
Seuraavaksi rakennetaan Funguksella kysymyspeliä. Vuokaavion ensimmäinen kysymys on pelin aloittaja, *Game Starter*. Tämän lohkon sisälle asetetaan *Menu*-komentoja, jotka johtavat, joko oikein- tai väärin-lohkolle ja kirjataan pisteytys eli tuloksen omaksi lohkoksi. Tarvitaan tulos- ja seuraava kysymysmuuttujat pisteytyksen tulostamiseen ja siirtymiseen seuraavaan kysymykseen. (Kuva 21.) Seuraavaksi kuvataan yksityiskohtaisemmin jokaisen lohkon sisäisistä komennoista.



Kuva 21. Vuokaavioon on luotu kysymys1 -, oikein – ja väärin lohkot.

Muuttujat

Minipeli tarvitsee muuttujia edetäkseen seuraavin kysymyksiin ja pisteiden keräämiseen. Eli tarvitaan vuokaavioon tulos-lohkon, sen lisäksi tarvitaan kaksi kokonaisluvun *Integer* muuttujia, jotka ovat nimeltään tulos ja seuraava kysymys.

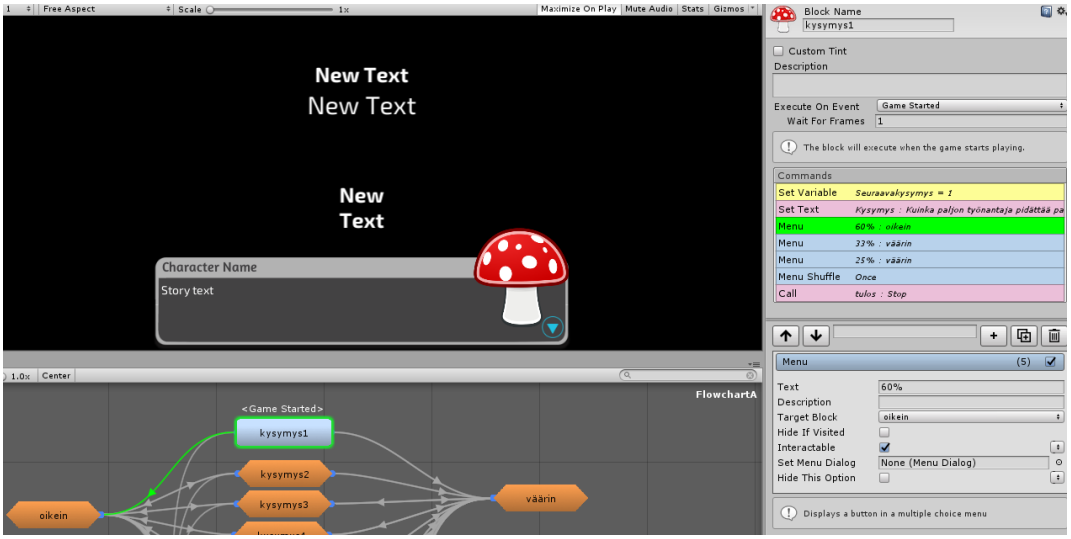


Kuva 22. kokonaisluvun muuttujat.

Nämä muuttujat ovat pisteytyksen ja kysymysten jatkuvuuden keskeisiä muuttujia. Muuttujat, *Variables*-ikkunasta plusmerkillä lisätään uusia muuttujia (kuva 22).

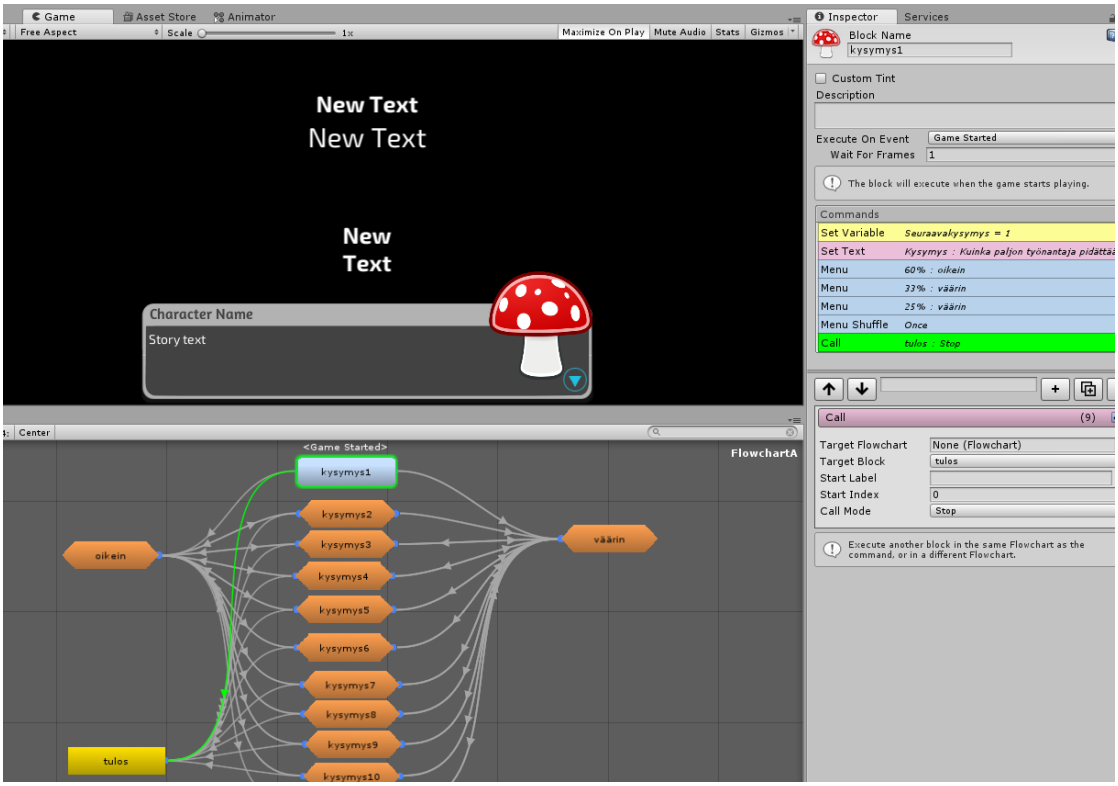
Kysymys -lohko

Pelin aloittajaksi luodaan *Game Started,* kysymys 1-lohko, johon rakennetaan seuraavanlaiset komennot: Asetetaan muuttujan seuraava kysymykseen arvoksi 1 ja asettamalla tekstin, *Set Text*, johon kirjoitetaan kysymys. Asetetaan kolme *Menu*-komentoa, jotka ovat kolme vastausvaihtoehtoa ja lisäksi suunnataan *Menu*-komennot tuleviin lohkoihin, *Target block,* oikein- tai väärin-lohkoihin. Eli väärät vastaukset menevät väärin-lohkoon ja oikea vastaus menee oikein-lohkoon. Tarvitaan *Menu shuffle* -komento sekoittamaan vastausten järjestystä. *Target Block* -kohdasta voidaan antaa polku haluamalle lohkolle (kuva 23).



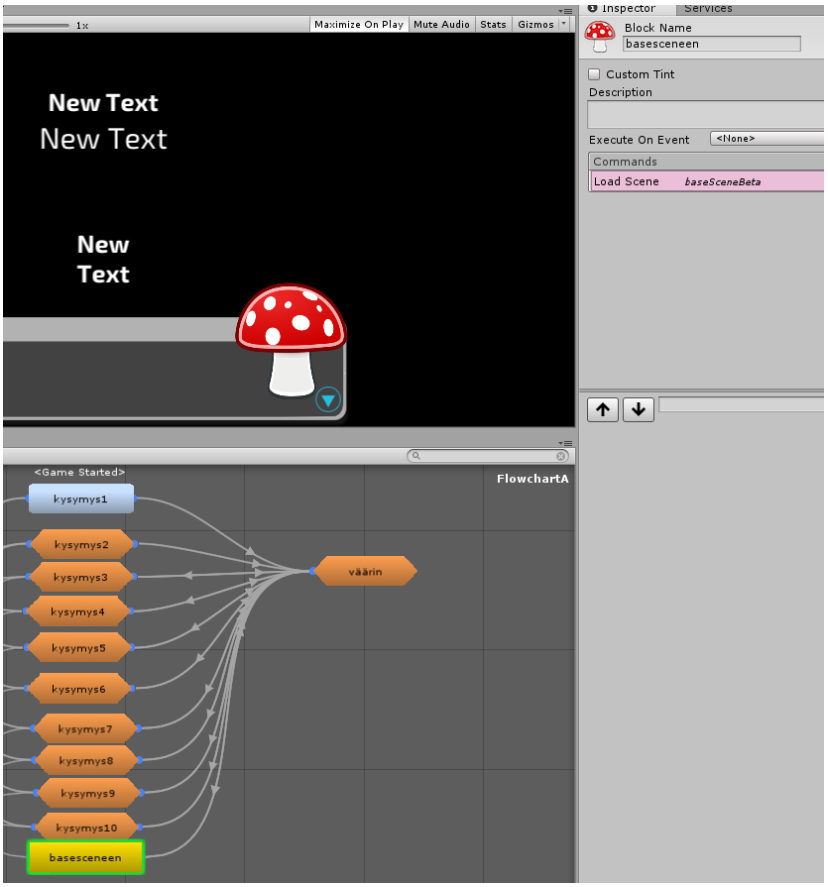
Kuva 23. kuvassa on esitetty kysymys 1-lohkon komentolista ja polku oikein-lohkolle

Call-komennolla kutsutaan toiseen lohkoon, joko komennon tai toiseen vuokaavioon. Tässä projektissa *Call*-komento suorittaa tulos-lohkoon (kuva 24). On myös huomioitava, että ensimmäinen kysymys on laitettu aloittaa tapahtuman.



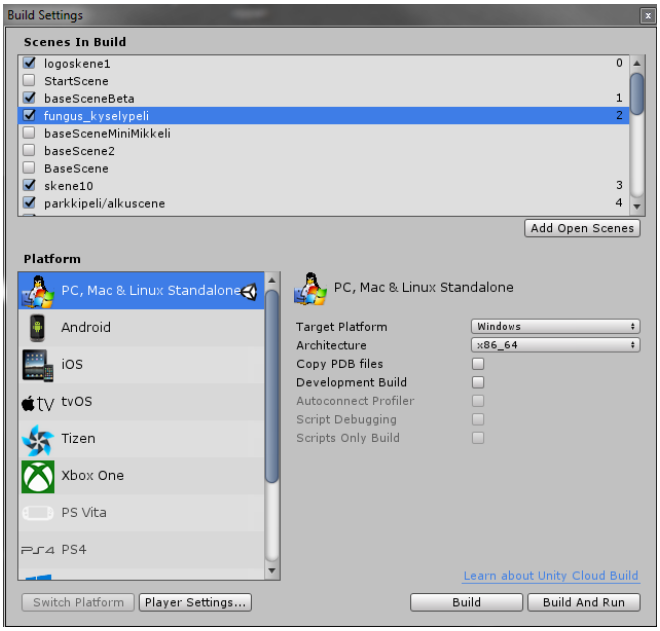
Kuva 24. Kutsu komento kysymys1-lohkosta, tulos -lohkoon

Lohkot, jotka ovat kysymys lohkoja eli kysymykset 1-10, on tehty samalla tavalla kuin kysymys 1 -lohko. tosin Basesceneen-lohko on uuden kohtauksen lataaminen *Load Scene.* (Kuva 25.)



Kuva 25. Basesceneen -lohkon komento Load scene

Kohtauksen lataamisen saa toimimaan, kun käydään Unityn tiedostot, *File,* koonti asetuksista, *Build settings,* ja raahataan kohtaus Basescenebeta koontiin sekä myös minipeli eli fungus_kyselypeli niminen kohtaus. Tällä toiminnolla Fungus osaa lukea omasta Basesceneen-lohkosta kohtauksen lataamisen komennon ja aktivoi seuraavan kohtauksen. (Kuva 26.)



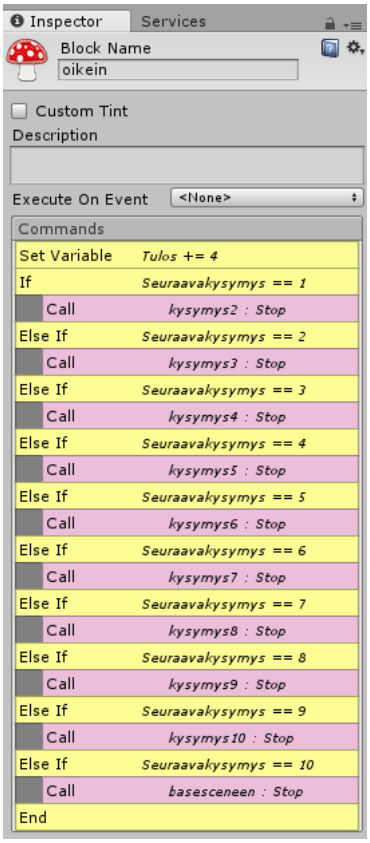
Kuva 26. Tapahtumien koonti asetuksiin on hyvä laittaa tarvittavat tapatumat pääpeliä ja minipeliä varten.

Jos tämä unohdetaan tehdä, Fungus ei löydä ladattavaa kohtausta omasta vuokaaviostaan, sillä vuokaavion lohkossa esiintyvässä kohtauksen lataaminen epäonnistuu.

Oikein- ja Väärin-lohko

Seuraavaksi tullaan tarkastelemaan Oikein- ja Väärin-lohkojen komentolistaan. Nämä kaksi listaa ovat lähes samanlaisia, mutta oikein-lohkossa lisätään Tulos-muuttujaan 4 (kuva 27), kun taas väärin-lohkossa vähennetään muuttujasta 2.

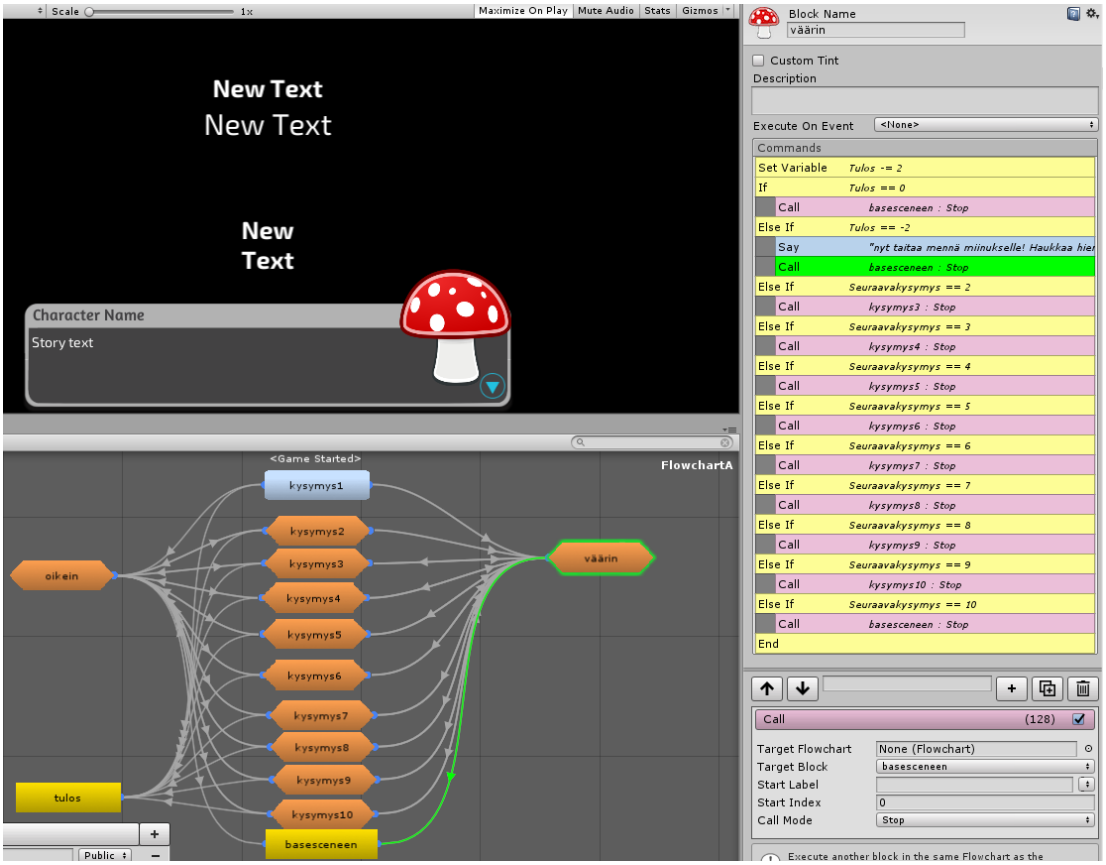
Oikein-lohkolla on ehtolauseet. Eli ensimmäisen kysymyslohkon muuttujan arvo seuraava kysymykseen on 1, jos vastataan oikein, niin tuloksen muuttujaan lisään 4. Kutsutaan seuraava kysymyslohko eli kysymys 2, joka muuttujan arvo on 2. Jos siihen vastataan oikein, niin lisätään tulos muuttujaan 4, eli tulos on yhteensä 8. Ja tätä jatketaan niin kauan kun on asetettu ehtolauseita ja muuttujia saman verran mitä on kysymyksiä eli kymmenen kysymystä. Viimeisen kysymyksen kohdalla laitetaan polku Basesceneen -lohkoon, jotta uuden kohtauksen lataaminen aktivoituisi pelin päätyttyä. (Kuva 27.)



Kuva 27. oikein-lohkon komentolista

Väärin-lohkon komentolista on erilaisempi, sillä tuloksesta vähennetään, jos vastataan kysymykseen väärin. Eli Siinä missä oikein lohkon komentolistassa lisättiin, niin nyt vähennetään. Kun pelin tulos muuttujan arvo on 0, ja jos vastataan väärin, niin tulos muuttujasta vähennetään 2 (kuva 28).

Kuva 28. Väärin-lohkon komentolista

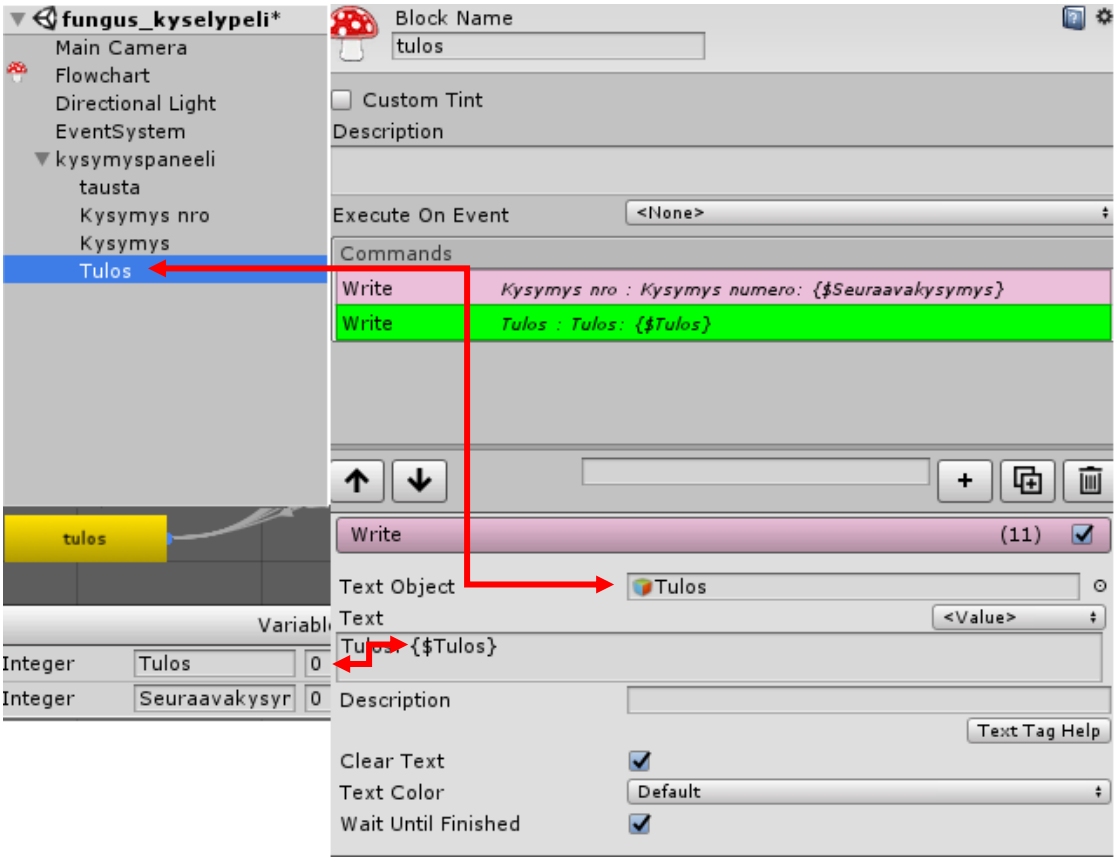


Kuva 29. Väärin-lohkon muuttujien arvot, jos arvo on -2 niin kutsutaan basescene-lohko aktivoimaan basescenebeta kohtauksen

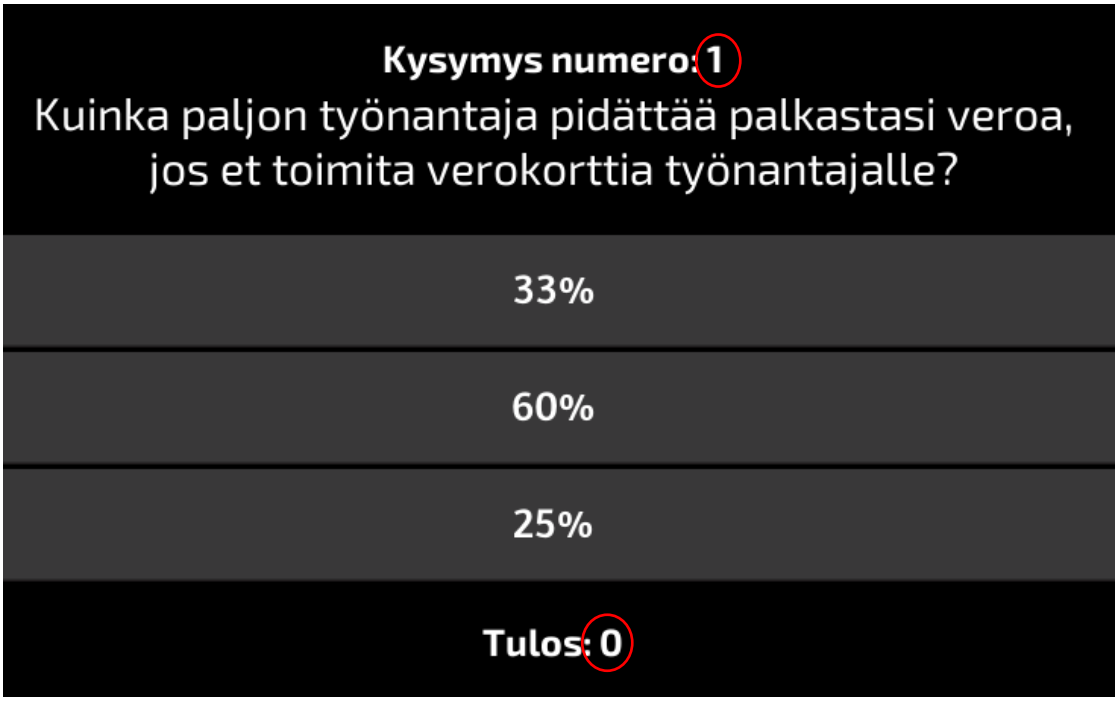
Jos ensimmäisessä kysymyksen vastataan väärin, niin silloin tulee Sano-komento, joka ilmoittaa, ” Nyt taitaa mennä miinukselle! Haukkaa hieman happea ja tule paremmalla ajalla uudestaan!”. Tämän jälkeen komentolistassa suoritetaan polku Basesceneen-lohkoon, joka lataa Basescenebeta-kohtauksen eli Työelämäpelin (kuva 29).

Tulos -lohko

Kun projektissa on rakennettu tarvittavat UI-elementit, perehdytään syvemmin Tulos-lohkoon. Seuraavaksi asetetaan vuokaavion Tulos-muuttuja näkyviin tekstielementissä kirjoittamalla tulos-lohkon kirjoita, *Write*-komentoon, *{$Muuttujan-Nimi},* jotta saadaan tulos-muuttujan arvo näkyviin UI-elementissä. Tämä UI-elementti on tekstiobjekti, joka linkitetään samaan *Write*-komentoon. (Kuva 30.)



Kuva 30. Ui-elementin ja muuttujan arvon linkittäminen

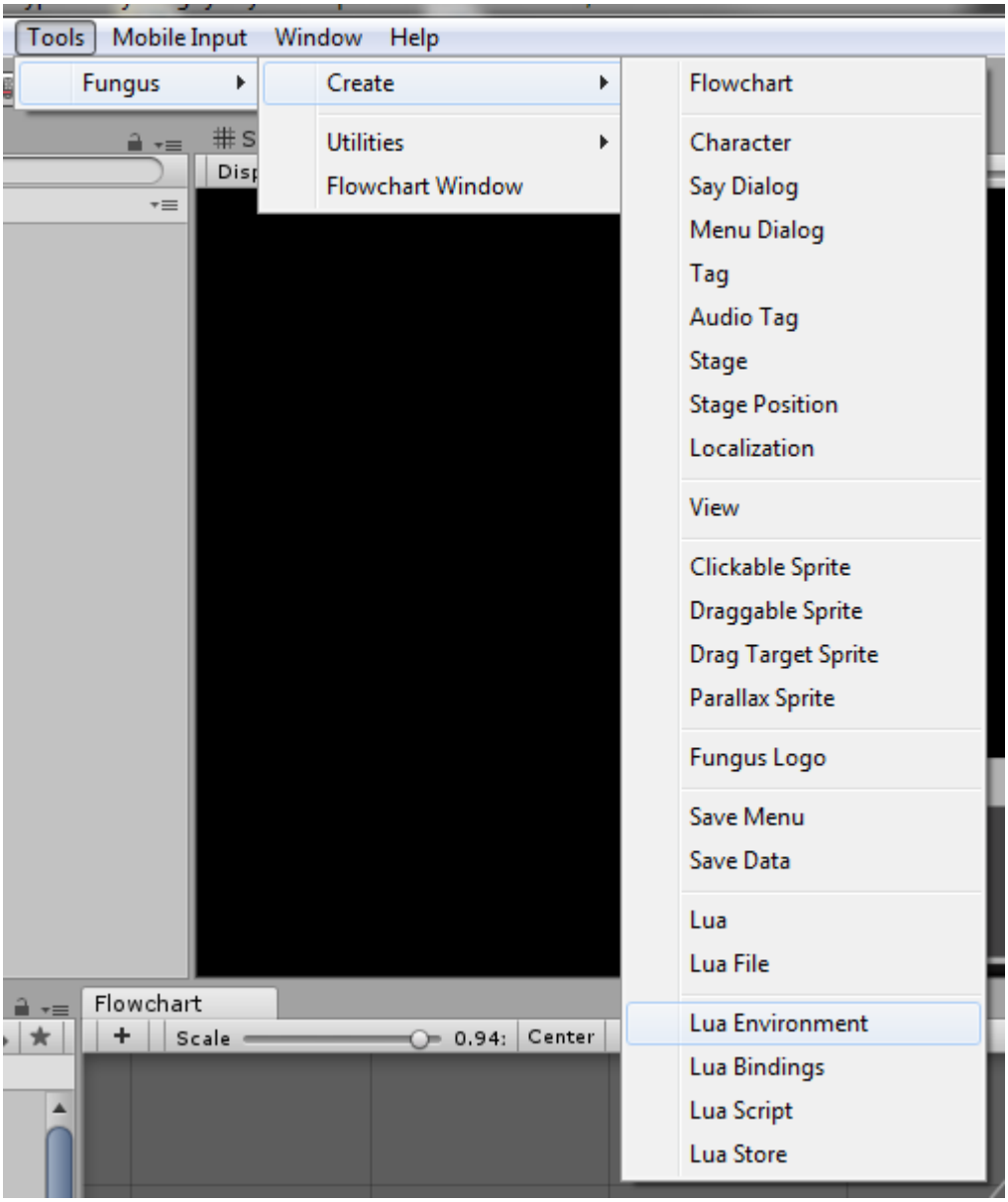


Kuva 31. Integer muuttujien tulostus minipelissä. Muuttujien arvot ovat ympyröity punaisella värillä.

Tämä toiminto pyrkii näyttämään Tulos-muuttujan arvon 0 pelinäkymässä. Kysymysnumeron näkyminen pelinäkymässä onnistuu (kuva 31), kun samaan Tulos-llohkoon laitetaan toinen *Write*-komento, linkitetään kysymys nro -tekstiobjekti ja tekstialueeseen kirjoitetaan tarvittava *{$MuuttujanNimi}.* (Fungusdocs 2017.)

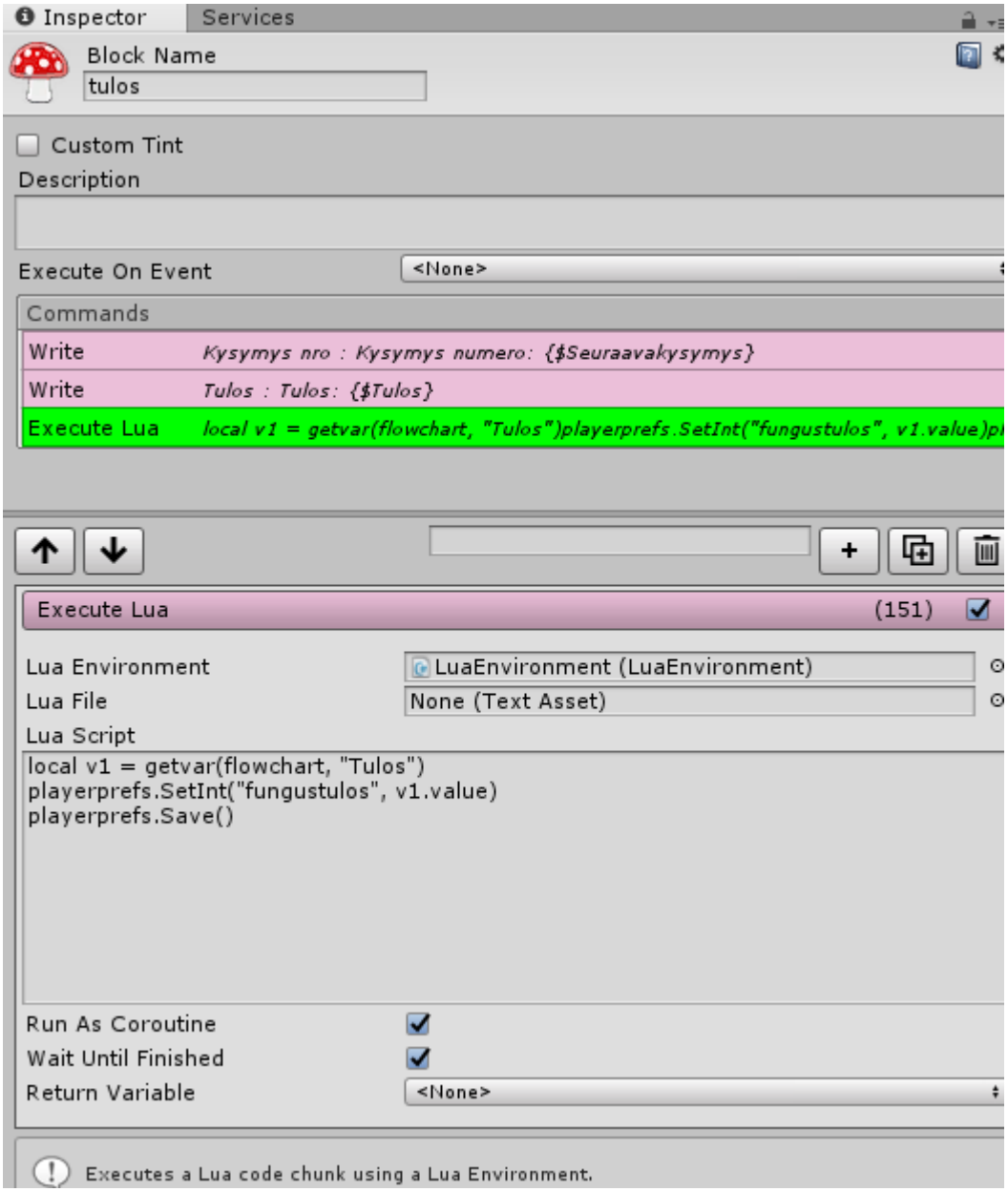
Pisteytyksen välittäminen Työelämäpeliin

Tuloksen välittäminen Työelämäpeliin voidaan kirjoittaa Playerprefsin kautta Lua-scriptillä. Valitse seuraavasti: työkalut, *Tools,* Luo, *Create* ja Lua Enviroment (kuva 32).



Kuva 32. Lua Enviormentin löytäminen

Seuraavaksi Tulos-lohkoon lisätään Luan toteutuksen komento, *Execute Lua*.



Kuva 33. Tulos muuttujan haku Lua Scriptiin

Lua Script -kohtaan kirjoitetaan kolme riviä koodia. Ensimmäinen rivi hakee Luaan muuttujan Tulos, vuokaaviosta *Flowchart.* Toinen Rivi tallentaa Unityn PlayerPrefs-nimiseen ”fungustulos” -muutujaan ja kolmas rivi tallentaa Playerprefsiin niin, että muuttuja on muualla käytettävissä. Tässä tässä tapauksessa, siis viedään muuttujan arvon pääpeliin. (Kuva 33.)

5.3 Minipelin Työelämäpelissä

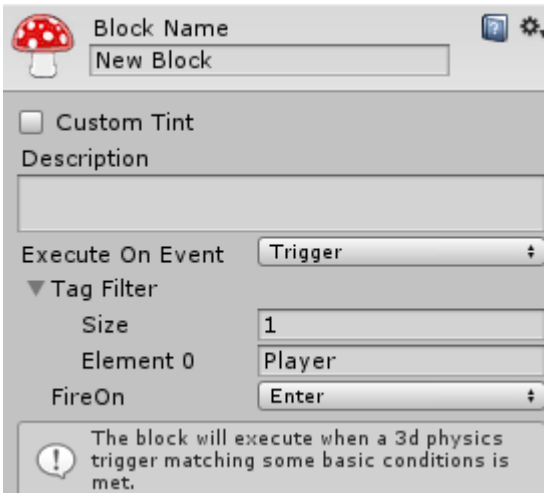
Seuraavassa osiossa siirrymme toiseen kohtaukseen, joka on Basescene-beta. Osiossa kirjoitetaan hieman koodia, jotta saadaan pisteytys näkymään pääpelissä minipelin suorittamisesta.

kuva 34. Työelämäpeliin lisääminen kyselypeli_trigger peliobjektiin vuokaavion ja törmäys komponentin lisääminen tarkastajaikkunaan

Tämän jälkeen luodaan kyselypeli_trigger-niminen peliobjekti. Kyselypeli_trigger-peliobjektiin lisätään *Flowchart*-komponentti, joka on vuokaavio. Tämä vuokaaviokomponentti tekee Kyselypeli_trigger-objektille vuokaavion. Tähän vuokaavion tehdään toimintoja, kun pelihahmo ns. pelaaja, *Player*, törmää tähän kyseiseen objektiin, joka laukaisee minipelin. Eli tämän lisäksi tarkastajaikkunassa, *Inspector,* lisätään törmäyskomponentti, joka on tässä yhteydessä pallonmuotoinen, ja *is trigger* aktivoidaan päälle. Nämä lisäykset näkyvät tarkastajaikkunassa. (Kuva 34.)

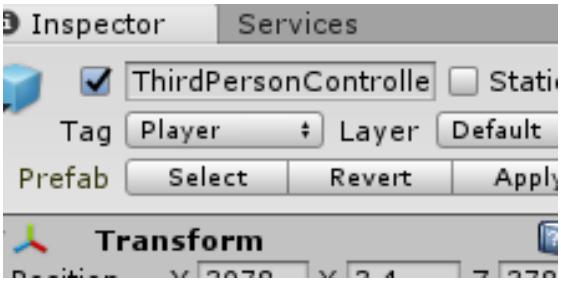
Seuraavaksi käydään läpi vuokaavion New Block -lohkon asetukset. Kysely-peli_trigger-nimisessä peliobjektissa aloittavan lohkon tapahtuman suorittaminen, *Execute on event,* asetetaan törmäykseksi, *Trigger*. Tämän toiminnon löytää Monobehaviour-kohdasta. *Trigger* ja *Trigger2D* eroavaisuudet ovat siinä, että Trigger on yhteensopiva 3D-fysiikan kanssa yhteensopiva. (Fungus 2017.)

Avaamalla *Tag filte r*-pudotusvalikko, nähdään koko, *Size,* jonka arvoksi tulee 1, jolloin saadaan ensimmäinen elementti *Element0*, johon kirjoitamme *Player* (kuva 35).



Kuva 35. Kysely_triggerin aloituslohkon tapahtuman muuttaminen törmäykseksi ja merkki-suodattimen arvojen asettaminen.

Seuraavaksi otetaan pelaaja-objekti ja tarkistamme, että tämä kyseinen pelaaja-objekti on se, joka on itse pelihahmo. Sen merkki, *Tag,* on Player. Jos tätä toimintoa ei tehdä, kyselypeli_trigger-objektin Elementti 0, johon on kirjoitettu *Player*, ei aktivoidu, jolloin pelihahmo jatkaa törmäysalueesta eteenpäin ja minipeli ei aukea.



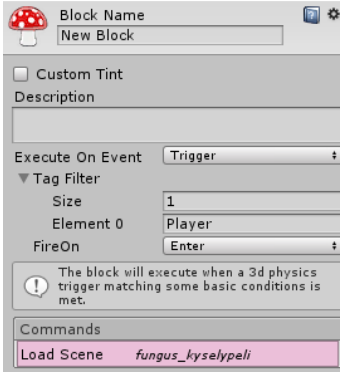
Kuva 36. Työelämpelin ThirdpersonController Tag -merkki

Tällä toiminnolla saadaan laukaistua pallomainen törmäysalue, kun on merkattu Thirdpersoncontroller peli objektiin merkki *Tag* pelaajaksi *Player* (kuva 36, Secchi 2016).

Seuraavaksi siirrytään kyselypeli_trigger-objektin lohkoon. Komentolistalle lisätään lataa kohtaus, *Load Scene* (kuva 37).

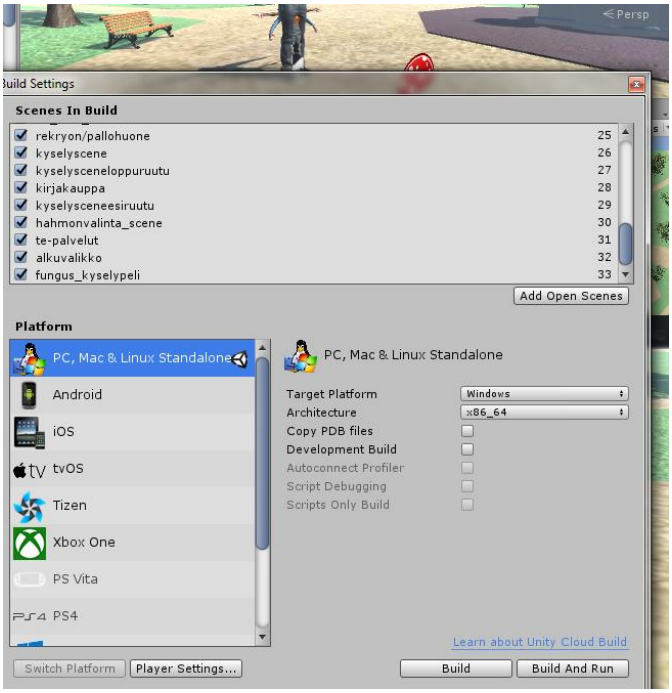
Kuva 37. lataa kohtaus- komento.

Kun komentolistassa on ladattava kohtauksen-komento, seuraavaksi kirjoitetaan seuraavaksi uuden kohtauksen nimi eli minipelin nimi fungus_kyselypeli (kuva 38).



Kuva 38. Minipelin kohtauksen asettaminen lataa kohtaus-komentoon

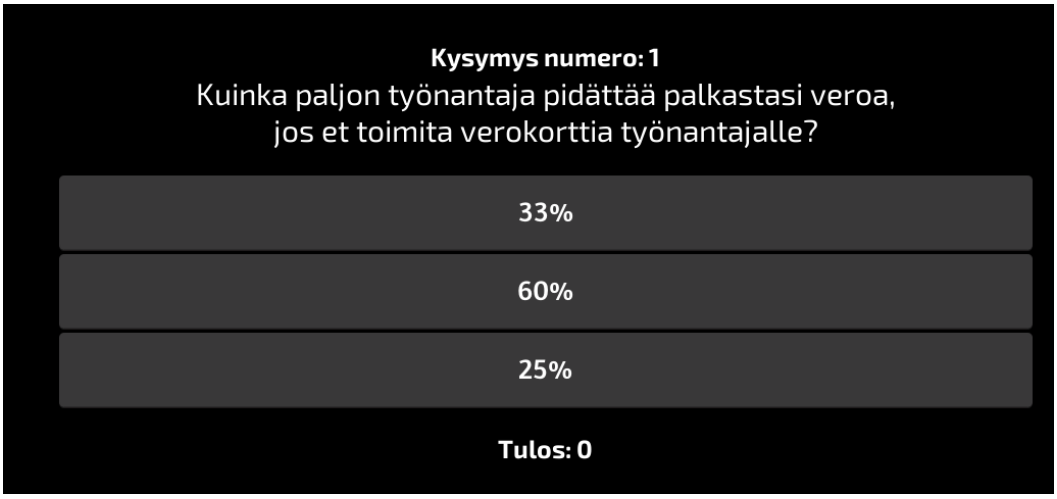
Kohtauksen lataaminen onnistuu, kun mennään koontiasetuksiin, *Build settings.* On hyvä tarkistaa, että tarvittavat kohtaukset, *scenes,* ovat mukana eli minipeli fungus_kyselypeli on koonnissa mukana. Tänne siirretään jokainen pelikohtaus, jonka halutaan Työelämäpelissä näkyvän. Siirtämällä fungus_kyselypeli kohtauksen, *Scene,* tähän listaan ja tällä tavalla saadaan minipeli aktivoitumaan sekä näkymään (kuva 39).



Kuva 39. Koonti asetukset ja minipelin tarkastelu sekä liittäminen

Kun koontiasetuksen ikkuna on suljettu, niin seuraavaksi kokeillaan pelin toimivuutta. Kun käynnistetään peliprojektin pelinäkymässä, *game view*, pitäisi tulla näkyviin Mikkelin kirkkopuisto, pelihahmo ja kärpässieni. Kärpässieni kuvaa törmäysalueen sijaintia. (Kuva 40.)

Kuva 40. Työelämäpelin lopullinen pelinäkymä

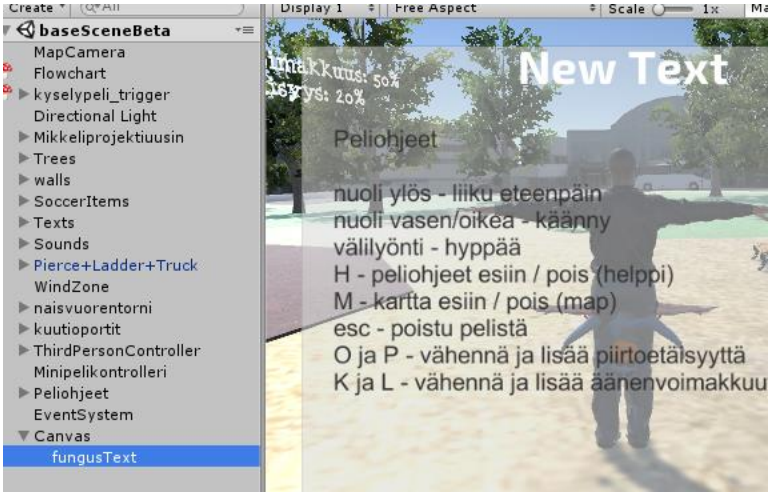


Kuva 41. minipeli käynnistyy, kun törmätään pääpelissä sieneen

Kun pelihahmo törmää kärpässieneen, niin pallomaisen törmäysalueen vuokaavion lohkon sisäinen toiminto aktivoituu. Tämä aktivoituminen käynnistää minipelin, tämä tarkoittaa sitä, että pelissä edetään seuraavaan kohtaukseen. (Kuva 41.)

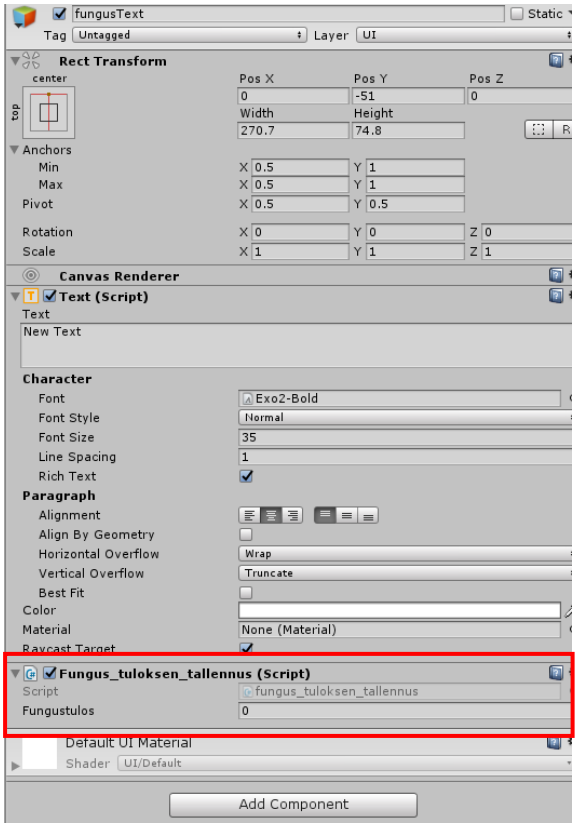
Pisteytyksen vastaanottaminen

Työelämäpelin Basescenebetassa luodaan sopiva objekti esimerkiksi teksti-objekti nimeltään fungusText (kuva 42).



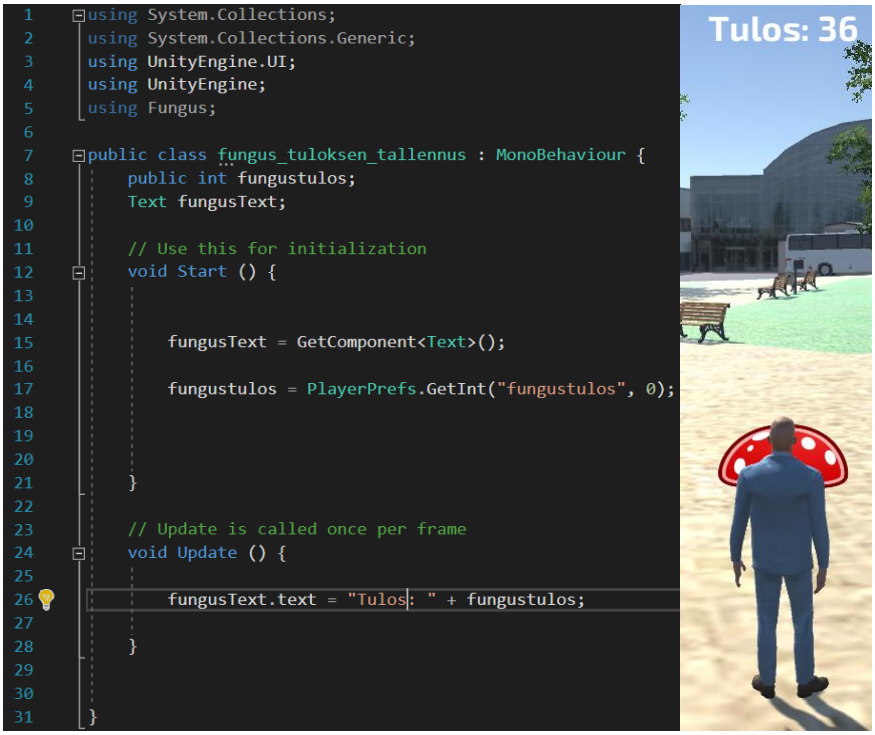
Kuva 42. Tekstiobjekti fungusText:n sijainti

Tekstiobjektiin lisätään ohjelmakoodia, jotta saadaan minipelin pisteytys näkymään Työelämäpelissä. Tämä ohjelmakoodi on nimetty Fungus_tuloksen_tallennus. (Kuva 43.)



Kuva 43. Ohjelmakoodin lisääminen

Seuraavaksi kirjoitetaan tekstiobjekti ohjelmakoodiin Start() -funktioon ja Update() -funktioon fungustulos (kuva 44).



Kuva 44 Kun minipelissä on käyty kerran, tulos ilmestyy pelin yläreunaan

Näillä koodin riveillä saadaan aikaiseksi se, että minipelin pisteytys koostuu Työelämäpeli-näkymässä (kuva 44).

6 PÄÄTÄNTÖ

Olen tyytyväinen opinnäytetyöhöni ja opin paljon Fungus-työkalun soveltuvuudesta. Työelämäpelin minipeliä tehdessäni havaitsin, että sisällöntuottamisessa ilmeni puuteellisuutta, mihin kannattaa kiinnittää huomiota, kun vastaisuudessa tehdään samankaltaisia hankkeita. Jos olisi kiinnitetty huomiota enemmän pelin opetuksellisen sisältöön, niin peliä olisi pystytty viimeistelemään kokonaisuudessa paremmin. Peliprojekti omistaa valmiin idean ja selkeän konseptin sekä hyvät mahdolliset tavoitteet. On erittäin tärkeää ottaa huomioon, onko tavoite opettavaa ja kuinka sitouttaa hankkeen osallistujia. (vrt. Nordicedu, 2018a.)

Hankkeen kohderyhmä oli erittäin hyvin määritelty alusta alkaen ja se on pelin suunnittelun kannalta hyvä asia. Kun tavoitteet ja tarpeet ovat tiedossa, pitäisi muistaa panostaa peliin tulevaan sisältöön. Työelämäpeli oli erinomainen pohja, josta uupui opettavainen sisältö. Jos pelin rakenteesta olisi alusta alkaen luotu dynaaminen kaavio, kuten UML-kaavio (kts. liite 2), niin resursseja olisi voitu suunnata paremmin, jolloin olisi saatu tarinallisempaa juonta ja rakennetulle pelihahmolle juonen lisäksi tavoitteet. Hankkeeseen olisi kaivattu osallistuneiden panosta pelin opetuksellisen sisällön tuottamiseen. (vrt. Nordicedu, 2018a.)

Jos olisi kiinnitetty huomiota enemmän pelin opetuksellisen sisältöön, peli pystyttäisiin viimeistelemään. Näin saataisiin kokonaisvaltiainen pelin käsikirjoitus tehtyä, joka sisältäisi pelillisiä ja tarinallisia osia. (vrt. Nordicedu, 2018b.) Twine-työkalun lisäksi on löydetty Fungus -työkalu, joka oli myös epälineaarinen tarinankerrontatyökalu. Kummankin työkalun käytettävyys ja soveltuvuus Työelämäpelissä oli mutkatonta ja helppoa.

Opinnäytetyössäni tarkoitukseni oli madaltaa kynnystä ohjelmoinnissa ja pelien innovoimisessa. Fungus-työkalu on oivallinen väline Unityssa, jos on kiinnostunut pelien tekemisestä ja haluaa ymmärtää ohjelmoinnin perusteita. Olin tyytyväinen työtulokseeni, sillä sain Fungus-työkalun toimimaan 3D ja 2D -maailmassa ja kyselypelini toimii moitteitta.

Työelämäpeli olisi hyvä alusta oppimateriaalin lisäämiselle ja se olisi hyvä oppimisympäristö ohjelmoinnin tueksi heille, jotka haluavat oppia uusia taitoja ja haluavat uuden tulokulman interaktiivisesta vuorovaikutuksesta.

Opin työssäni sisällön tuottamisesta ja hyödyntämään Fungus -työkalua. Suosittelisin Fungus-työkalua niille, jotka haluavat Unityyn monipuolisen epälineaarisen työkalun opeteltavaksi, sillä sisällön tuottamisessa se on oivallinen. Itse käyttäjänä pääsin muokkaamaan haluamiani pelielementtejä sen mukaiseen tarkoitukseen mitä työni tarvitsi.

Opinnäytetyön materiaalia käytettiin Jufo1 -konferenssi julkaisussa *Nonlinear Storytelling Method and tools for Low-Thresrhold Game Development* tutkijatapaamisessa. Tämä ITK-konferenssi, eli interaktiivinen tekniikan koulutus järjestettiin Hämeenlinnassa 11.4-13.4.2018 ja tapahtuman nimi oli Digiajan löytöretkeilijät.

Erityisesti haluan kiittää toimeksiantajaa, Mika Letonsaarta opinnäytetyön tukemisesta.

LÄHTEET

Github. 2017. Cradle. WWW-dokumentti. Päivitetty: 27.11.2017. Saatavissa: https://github.com/daterre/Cradle [Viitattu 7.4.2018]

Bitbucket. 2018. Työelämäpeli. WWW-dokumentti. Saatavissa: https://bitbucket.org/digityo/tyoelamapeli [Viitattu 3.4.2018]

Fungus. 2017. Fungus v3.6.1. Documentation. WWW-dokumentti. Päivitetty: 28.8.2017. Saatavissa: http://fungusdocs.snozbot.com/ [Viitattu 27.3.2018]

England, L. 2015. What is Twine (For developers). WWW-dokumentti. Päivitetty: 11.3.2015. Saatavissa: http://www.lizengland.com/blog/2015/03/what-is-twine-for-developers/ [viitattu 27.3.2018]

Letonsaari, M., Karjalainen, L., Selin, J. 2018. Nonlinear storytelling method and tools for low-thershold game development. Interaktiivinen tekniikka koulutuksessa -conference in Hämeenlinna the programme for ITE researcher meeting 11.4 –13.4.2018.

Nordicedu. 2018a. Peliprojektin askeleet. WWW-dokumentti. Saatavissa: http://nordicedu.com/askeleet [viitattu 6.3.2018].

Nordicedu. 2018b. Viisi yleisintä hyöty- tai opetuspelin kompastuskiveä. WWW-Dokumentti. Saatavissa: http://nordicedu.com/viisi-yleisinta-hyoty-tai-opetuspelin-kompastuskivea [viitattu 6.3.2018].

Secchi, M. 2016. Interacting with a GameObject in Fungus. WWW-dokumentti. 26.11.2016. Saatavissa: http://www.marcosecchi.it/2016/11/26/interacting-with-a-gameobject-in-fungus/?lang=en [viitattu 16.2.2018].

Tki-blogi. 2018. Työelämäpeli. WWW-dokumentti. 24.01.2018. Saatavissa https://tkiblogi.wordpress.com/2018/01/24/tyoelamapeli/ [viitattu 1.2.2018].

Twine. 2009. Twine. WWW-dokumentti. 2009. Saatavissa: http://twinery.org/ [viitattu 29.1.2018].

Työelämäpeli. s.a. WWW-dokumentti. Saatavissa: https://tyoelamapeli.fi/ [viitattu 7.4.2018]

Unity. 2018a. System requirements. WWW-dokumentti. Saatavissa: https://unity3d.com/unity/system-requirements [viitattu: 7.4.2018]

Unity. 2018b. Unity. WWW-dokumentti. 2018. Saatavissa: https://unity3d.com/ [viitattu 29.1.2018].

Unity. 2018c. Unity Asset Store. WWW-dokumentti. 2018. Saatavissa: https://assetstore.unity.com/packages/templates/systems/fungus-34184 [viitattu 29.1.2018].

Youtube. 2018. 09 Fungus Flow. WWW-dokumentti. Saatavissa: https://www.youtube.com/watch?v=vrLNeFsoCyw [viitattu 14.3.2018]

LIITE 1
Vaatimusmäärittely

Toiminnalliset vaatimukset

Vaatimus : 1	**Vaatimuksen tyyppi:** Toiminnallinen	**Toiminto/käyttötapaus**
Kuvaus: Digiosaajaksi työelämään, digityö Työelämäpeliin minipeli		
Vaatimuksen perustelu: Toimeksiantaja näkee minipelin pelattavuuden digityö työelämä pelissä		
Prioriteetti: 1		
Hyväksymiskriteeri: Minipeli käynnistyy digityö Työelämäpelissä		
Lähde: Toimeksiantajan vaatimus		

Vaatimus : 2	**Vaatimuksen tyyppi:** Toiminnallinen	**Toiminto/käyttötapaus**
Kuvaus: Minipeli käyttää toimeksiantajan kysymyspatteristoa		
Vaatimuksen perustelu: Tämä on edellytys minipelin sisällölle ja edistää hankkeen jatkokehitystä sekä kehitystarpeita		
Prioriteetti: 2		
Hyväksymiskriteeri: Kysymyspatteristoa käytetään digityö Työelämäpelin sisällön tuottamiseksi		
Lähde: Toimeksiantajan vaatimus		

Ei-toiminnalliset vaatimukset

Vaatimus : 3	**Vaatimuksen tyyppi:** Ei-toiminnallinen	**Toiminto/käyttötapaus**
Kuvaus: Pelinmoottorina on Unity		
Vaatimuksen perustelu: Digityö Työelämäpeli on rakennettu Unity pelimoottorilla.		
Prioriteetti: 2		
Hyväksymiskriteeri: Minipeli tehdään Unitylla		
Lähde: Toimeksiantajan vaatimus		

Vaatimus : 4	**Vaatimuksen tyyppi:** Ei-Toiminnallinen	**Toiminto/käyttötapaus**
Kuvaus: Unityssä käytetään integroitavaa Fungus -työkalua		
Vaatimuksen perustelu: Opiskelija perehtyy itsenäisesti Fungus -työkalun soveltuvuuteen ja minipeli on rakennettu käyttämällä tätä työkalua.		
Prioriteetti: 2		
Hyväksymiskriteeri: Fungus työkalun soveltuvuus		
Lähde: Toimeksiantajan vaatimus		

Rajoitteet

Vaatimus : 5	**Vaatimuksen tyyppi:** Rajoite	**Toiminto/käyttötapaus**
Kuvaus: Minipelin käyttöliittymä tulisi olla yksinkertainen ja selkeä		
Vaatimuksen perustelu: Visuaalinen toteutus vastaa toimeksiantajan vaatimuksia		

Prioriteetti: 2
Hyväksymiskriteeri: Kysymyspatteristoa käytetään digityö Työelämäpelin sisällön tuottamiseksi
Lähde: Visuaalisuus vastaa toimeksiantajan vaatimuksia.

Vaatimus : 6	**Vaatimuksen tyyppi:** Rajoitteet	**Toiminto/käyttötapaus**
Kuvaus: Thirdperson -objekti juoksee törmäys alueeseen ja minipeli käynnistyy		
Vaatimuksen perustelu: Digityö Työelämä pelin, peli scenestä pääsee minipeli sceneen		
Prioriteetti: 2		
Hyväksymiskriteeri: Mitataan 2D ja 3D toimimuutta		
Lähde: Toimeksiantajan vaatimus		

LIITE 2

UML -kaavio Työelämäpelistä