LAPIN AMK
Lapland University of Applied Sciences

TIETOTURVAMOBIILIPELIN TOTEUTUS UNITYA HYÖDYNTYÄEN

Kuusiniemi Jussi
Välikangas Ossi

Opinnäytetyö

Tietojenkäsittelyn koulutus
Tradenomi (AMK)

2020

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyö

Tietojenkäsittelyn koulutus
Tradenomi (AMK)

Tekijä	Jussi Kuusiniemi Ossi Välikangas	Vuosi	2021
Ohjaaja(t)	Johanna Vuokila		
Työn nimi	Tietoturvamobiilipelin toteutus Unitya hyödyntäen		
Sivu- ja liitesivumäärä	41		

Tässä opinnäytetyössä oli aiheena toteuttaa mobiilipeli käyttämällä Unity-pelimoottoria. Tämän mobiilipelin tarkoituksena oli tutustuttaa varhaiskasvatusikäisiä lapsia tietoturvaan ja turvalliseen käyttäytymiseen internetissä. Koska pelin kohderyhmänä ovat pienet lapset, niin pelin piti olla helposti ymmärrettävissä ja helposti käyttöön otettava.

Käsittelimme tässä opinnäytetyössä asioita, joita on syytä ottaa huomioon, kun suunnitellaan peliä varhaiskasvatukseen, käyttämäämme pelimoottoria ja työn eri vaiheita. Teimme sovelluksesta proof of concept -version, jolla todensimme kykymme jatkaa pelin kehitystä opinnäytetyön jälkeen. Lopussa pohdimme työmme onnistumista ja varhaiskasvatuksen tilaa suhteessa IT-opetukseen.

Mobiilipelimme on muistipeli, jossa pitää saada oikeita pareja. Pelin tässä vaiheessa pareja on neljä, mutta pelimme on rakennettu niin, että lisäparien lisääminen tulevaisuudessa on vaivatonta. Peli toimii siten, että toisessa kortissa on tietoturvahaitta ja toisessa kortissa on sille korjaus.

Mobiilipeli on toteutettu C#-kielellä, jota käyttämämme Unity-pelimoottori tukee hyvin. C# on Microsoftin kehittämä kieli alun perin .NET-ohjelmistokomponenttikirjastoa varten. Unity on tehokas ja monipuolinen kaiken tasoisille osaajille suunnattu pelimoottori. Unityn käyttö on ilmaista pienille kehittäjille, ja siinä on saatavilla erilaisia ominaisuuksia, joiden avulla pelistä voi muokata monipuolisen ja halutun näköisen.

Avainsanat tietoturva, opetus, varhaiskasvatus, Android

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyö

Degree programme in Business
Information Technology
Bachelor of Business Administration

Author	Jussi Kuusiniemi Ossi Välikangas	Year	2021
Supervisor	Johanna Vuokila		
Subject of thesis	Designing a mobile game about cyber security using Unity		
Number of pages	41		

Our point in our thesis was to develop a mobile game using Unity game engine. Purpose of this mobile game was to familiarize early childhood education aged children about cyber security and safe use of internet. Target group of our game are young children, so the game has to be accessible and easy to understand.

In our thesis, we handle issues that should be taken in to account when a game for early childhood education, game engine that we use and different stages of our work. We made a proof of concept- version of our game, which validated our capabilities of pursuing a fulla game after the thessis. At the end of the thesis we reflected our work and state of early chilhood education in relation to teaching of IT.

Our mobile game is a memory game, where player has to get a matching pair. In this stage of the game, we use four pairs of cards, but our game is built in a way that makes it easy to add cards. Game idea is that in the other card of the pair has a cyber security risk and other pair has a fix for that risk.

We used C# language to create our mobile game, which is natively supported by Unity. C# is originally created for use in .NET framework by Microsoft. Unity is a powerful and versatile game engine intended for users of all levels. Unity is free to use for small developers and it has a lot of features which lets developers edit and create designs they intended.

Key words cyber security, education, early childhood education, Android

SISÄLLYS

KÄYTETYT KÄSITTEET

Proof of Concept tai POC	Idean toteuttamiskelpoisuuden todistaminen.
Scene	Alue, missä työskennellään ja sisältää osan tai kaiken pelissä käytössä olevan sisällön. Scenejä voi olla useampia pelissä.
Class	Voi käyttää koodauksessa luomaan ryhmiä, joilla ryhmän sisällä olevilla objekteilla on sama arvo.
Parent	Hierarkiassa ylempänä oleva objekti tai arvo.
Child	Hierarkiassa parent:n alapuolella oleva objekti tai arvo.
Grid	Auttaa linjaamaan pelin objekteja suoriin linjoihin.

1 JOHDANTO

Digitaalinen maailma ympärillämme kasvaa jatkuvasti, muun muassa internet, sosiaalinen media ja erilaiset pilvipalvelut ovat läsnä lähes jokaisen arjessa. Tämä pakottaa ihmiset huomioimaan tietoturvaa yhä enenevissä määrin. Aikuisten lisäksi myös lapset käyttävät internetiä. Vuoden 2013 lasten mediabarometrin mukaan 93 % lapsista käyttää internetiä ainakin joskus (Suoninen 2013). Tämä tarkoittaa, että yhä nuoremmat altistuvat älylaitteiden käytön nurjalle puolelle, joita ovat esimerkiksi haittaohjelmat, huijaukset, kuten grooming (seksuaalisen hyväksikäytön valmistelu), ja nettikiusaaminen.

Valtioneuvoston toteuttamassa kyselyssä kävi ilmi, että suomalaiset eivät luota omiin tietoturvataitoihinsa tai taidot ovat puutteelliset (Valtionvarainministeriö 2020). Vajavainen ymmärrys tietoturvasta asettaa käyttäjän vaaraan nettiä käytettäessä. Erilaiset tietojenkalastushuijaukset ovat arkipäivää, mikä voi johtaa jopa luottotietojen menetykseen. Tästä syystä lapsille on syytä opettaa tietoturvan tärkeys jo yhtä varhaisessa vaiheessa kuin he alkavat käyttämään älylaitteita. Tämä edistää asian sisäistämistä ja normalisointia myöhemmällä iällä.

Tieto-ja viestintäteknologian osaaminen on yksi laaja-alaisista osaamistavoitteista uusimmassa opetussuunnitelmassa, jota on käytettävä jokaisella vuosiluokalla ja jokaisessa aineessa. (Opintopolku 2021). Kuitenkaan erillistä tieto- ja viestintäteknologian oppiainetta ei alakoulussa ole (Sankila. 2015, 248). Tunneilla tehtävä digiopetus mahdollistaa muun muassa pelien mukaan ottamisen opetukseen, ja tällaisia pelejä on esimerkiksi Minecraft (Tikkanen 2015). Pelatessaan oppimispeliä lapsi oppii ymmärtämään helpommin opittavaa asiaa ja saavuttaa helpommin flow-tilan, joka mahdollistaa tehokkaan oppimisen (Järvilehto 2014, 221).

Opinnäytetyömme tarkoitus on rakentaa mobiilisovellus Android-käyttöjärjestelmälle. Valitsimme käyttöjärjestelmäksi Android-käyttöjärjestelmän, sillä sitä käyttää suurin osa mobiilikäyttäjistä. Tammikuussa 2021 Androidin osuus oli 71,93 % kaikista mobiilikäyttöjärjestelmistä (O'dea 2020). Androidia kehittää tällä hetkellä Open Handset Alliance, joka on 84 teknologiayrityksen

yhteenliittymä (Open handset Alliance 2007). Tekemämme mobiilisovellus on oppimispeli, jonka tarkoitus on auttaa lapsia ymmärtämään tietoturvaa jo varhaisessa kasvun vaiheessa. Mobiilipelimme on suunnattu tukemaan tietoturvan opetusta osana opetussuunnitelmaa.

Peli on toteutettu Unity-pelimoottorin avulla. Unity valikoitui alustaksi sen tehokkuuden takia. Unity on pelimoottori ja ohjelmointiympäristö samassa. Unityn pelimoottori pystyy suoraan tarjoamaan monia ominaisuuksia, joita pelinkehityksessä tarvitsee, kuten fysiikanmallinnuksen. Ohjelmointiympäristö tarjoaa kaikki tarvittavat työkalut Unityssa, mikä sujuvoittaa työntekoa. Myös cross platform -kehitys on mahdollista Unityssa, Tämä mahdollistaa sujuvan kehityksen erilaisille alustoille. (Sinicki 2021.)

Opinnäytetyötämme tekee kaksi henkilöä, joten meidän piti keksiä ratkaisu, jolla voimme toteuttaa tiedostojen vaihdon sujuvasti. Päädyimme lopulta Bitbucketin ja Sourcetreen yhdistelmään. Bitbucket on tarkoitettu erilaisten koodausprojektien versionhallintaan ja mahdollistaa helpon varmuuskopioinnin. Sourcetreen avulla pystyimme hyödyntämään Bitbucketia tehokkaasti. Sourcetree mahdollistaa lataamisen bitbucketista ja bitbuckettiin suoraan omalta tietokoneelta. Sourcetreen avulla pystyimme myös työskentelemään projektin kanssa yhtä aikaa, koska se mahdollistaa sivuhaaran tekemisen projektista, jota voi työstää, eikä tämä sivuhaara vaikuta pääprojektin toimintaan ennen kuin sen siihen takaisin liittää.

Johdannossa käymme läpi pelin kehitystä, tietoturvaa ja syitä tälle opinnäytetyölle. Toisessa luvussa esittelemme työn tutkimusmenetelmää. Kolmas luku on varattu työmme taustoitukselle. neljännessä luvussa käymme läpi pelisuunnittelua ja Unity-pelimoottoria tarkemmin. Viidennessä luvussa tutustumme opinnäytetyömme käytännön osuuteen ja kuinka sen toteutimme. Kuudes luku sisältää opinnäytetyömme jatkoaskeleet, eli kuinka pelin kehitys tulee jatkumaan työmme valmistumisen jälkeen. Lopuksi vielä pohdimme työmme onnistumista.

2 TUTKIMUSKYSYMYKSET- JA MENETELMÄT

Tässä opinnäytetyössä käytetään kvalitatiivista tutkimusotetta. Kvalitatiivisessa eli laadullisessa tutkimuksessa syvennytään esimerkiksi yhteen kohteeseen, josta etsitään tietoa (Jyväskylän yliopisto 2015). Tutkimus keskittyy ko. ilmiön tai ongelman ymmärtämiseen ja ratkaisemiseen. Laadullista tutkimusta ovat esimerkiksi tapaustutkimus, haastattelu ja etnografia. Kvalitatiivinen tutkimus eroaa kvantitatiivisesta tutkimuksesta siten, että kvantitatiivisessa tutkimuksessa keskitytään ilmiön mittaamiseen ja siinä usein käytetään laskennallisia sekä tilastollisia menetelmiä. (Alasuutari 1993, 34.)

Opinnäytetyössämme hyödynnämme kvalitatiivisen tutkimusmenetelmästä johdettua metodia, konstruktiivista tutkimusmenetelmää. Konstruktiivinen tutkimus lähtee liikkeelle tosielämän ongelmasta, jota lähdetään ratkaisemaan ja jossa päämäärä on valmiina, vaikka keinoa päämäärään pääsyyn ei vielä ole (Lukka 2001). Meidän työmme pääongelmana on, miten saadaan lapset kiinnostumaan tietoturvasta jo varhaisessa kasvuvaiheessa. Tämän lisäksi sovelluksen työstövaiheessa tulee eteen ongelmia, joiden ratkaisut määrittävät tulevan pelin toimivuuden. Ongelman ollessa selvillä aloitetaan suunnitteluvaihe. Suunnitteluvaiheessa haetaan tietoa koskien ongelmaa ja pyritään saamaan mahdollisimman syvä tietämys aiheesta.

Kun tietoa on kerätty tarpeeksi, voidaan alkaa tekemään proof of concept -ratkaisua, eli tuottamaan niin sanottu todiste siitä, että kehitteillä oleva tuote, tässä tapauksessa mobiilipelimme, on mahdollista toteuttaa (Lukka 2001). Tämä on tärkeä vaihe, jossa saadaan selville ratkaisumallin toimivuus ennen kuin se yhdistetään osaksi työtä. Ohjelmointiosiossa tulemme käyttämään poc-ratkaisuja paljon.

Jos suunnitteluvaihe sujuu ilman ongelmia, voidaan toteutus siirtää osaksi työtä. Ohjelmointiosiossa tässä kohdassa voi tulla vielä vastaan uusia ongelmia ratkaisun istuttamisessa osaksi muuta koodia.

Kun ratkaisu on istutettu osaksi työtä, on vuorossa työn kannalta tärkeä vaihe, testaus. Jokainen työhön liitetty ratkaisu tulee testata ja sen sopivuus työhön on

arvioitava. Jos on mahdollista, niin ratkaisun lopputulema tulee myös analysoida. (Lukka 2001.)

Opinnäytetyömme tarkoituksena on vastata kysymykseen, kuinka toteuttaa mobiilipeli tietoturvasta varhaiskasvatuksen käyttöön Unityn avulla. Avustavana alakysymyksenä meillä on, että mitä on otettava huomioon, kun suunnitellaan opettavaa mobiilipeliä varhaiskasvatuksen käyttöön.

3 TIETOTURVA JA OPPIMINEN

3.1 Tietoturva ja digitalisaatio

Ihminen on osa tietoturvaa, pelkästään teknologia ei sitä pysty yksin toteuttamaan (Vestman 2020, 4). Tietoturvasta on tullut digitalisaation yleistymisen myötä osa meidän kaikkien jokapäiväistä elämäämme, ja onkin hyvä ymmärtää tietoturvan perusteet. Käyttämällä internetiä käyttäjä on jo mahdollinen verkkorikoksen kohde. Käyttämällä vahvoja salasanoja ja pitämällä laitteet ajan tasalla digilaitteiden käyttäjä on jo hyvin suojautunut netin vaaroilta (Kyberturvallisuuskeskus 2020).

Perustietämys tietoturvasta on tärkeää ymmärtää. Tietoturva mielletään yleensä tietokoneiden asiaksi, mutta myös mobiili- ja älylaitteisiin on mahdollista suorittaa tietoturvaloukkaus. Tietoturvan taakse kätkeytyy useampi henkilökohtaisia tietoja sisältävä järjestelmä. Tietoturva suojaa henkilötietoja, terveydellisiä tietoja, pankkitietoja ja kaikkea muuta liikennettä ja tietoa, mitä ihmisestä internetistä on saatavilla. Ilman tietoturvaa rikollisilla olisi suora pääsy kaikkeen tietoon, jolla he voisivat esimerkiksi kiristää uhrejaan. (Kyberturvallisuuskeskus 2020.)

3.2 Oppimispelit opetuksessa

Pelin suunnittelu lapsille on lähtökohtaisesti erilaista kuin aikuisille. Siinä missä aikuisilla on jo pitkälle kehittynyt ymmärrys maailmasta, on lapsilla se vielä alkutekijöissään. Lasten keskittymiskyky on rajoittunut, ja he tarvitsevat jatkuvaa palautetta, jotta heidän mielenkiintonsa pysyy yhdessä asiassa. Lapsen mieli on monesti vilkas, ja tämän seurauksena keskittyminen voi olla haastavaa. (Brain balance achievement centers, 2021). Peli oppimisen tukena auttaa opettajaa esittämään oppiaineen kiinnostavassa muodossa ja auttaa opettajaa opetustyössä (Järvilehto 2014, 219).

Varhaiskasvatusiässä olevat lapset kehittyvät kovaa vauhtia, ja tämä on syytä ottaa huomioon peliä suunnitellessa. 4-vuotias on taidoiltaan hyvinkin erilaisella tasolla kuin 6-vuotias. 4-vuotias alkaa vasta ymmärtämään ohjelmissa ja peleissä näkyviä juonikuvioita, kun taas 6-vuotias kykenee jo hahmottamaan hyvin pelin

juonikuviot ja havaitsee toisten ihmisten tunteita ja tarpeita. (Mannerheimin lastensuojeluliitto 2020a; 2020b.)

Opinnäytetyössä meidän tulee huomioida lapsen ikäkehitys ja ymmärryskyky. Yksi ratkaistavista kysymyksistä on se, minkä ikäisille peli ensisijaisesti on tarkoitettu. Pelin kohdeyleisö on esikoululaiset. Viimeistään ensimmäisen luokan oppilaat alkavat käyttämään digilaitteita opetuksen tukena (Opetushallitus 2018). Näin ollen tietoturvan opetus on syytä aloittaa aikaisessa vaiheessa.

Opinnäytetyössämme otamme huomioon lapsen kehitysasteen ja motoriset taidot. Lapselle onnistumisen ja ilon tunne on tärkeä keino, jolla lapset voivat motivoida itseään (Järvilehto 2014, 219). Pelin käyttöliittymä on suunniteltu motivoimaan lapsia.

4 PELIN SUUNNITELMA

4.1 Pelisuunnittelun teoria

Ennen kuin pelin tekemisessä pääsee toteutusvaiheeseen, on suositeltavaa ensin tehdä peliin suunnitelma. Ensimmäiseksi peliin pitää keksiä aihepiiri ja lajityyppi, esimerkiksi toimintapeli. Tämän jälkeen suunnittelussa voi siirtyä kehittelemään pelin käsikirjoitusta, vuorovaikutteisia ominaisuuksia ja mekaniikoita. Jotta pelistä kuitenkin tulisi mieleenpainuva ja ymmärrettävä kokonaisuus, on syytä pitää mielessä pelin ydinidea, se minkä ympärille koko peli perustuu (Manninen 2007, 60). Ydinidea kannattaa kehitellä suunnittelun alussa ja käyttää koko pelin kehittämisvaiheen ajan.

Useimmissa peleissä on jonkinlainen kerronnallinen osuus, suoraviivainen tai sitten avoimempi. Suoraviivainen kerronta etenee tiukasti peliin ennalta määrättyä reittiä. Avoimempi kerronta antaa käsikirjoitukselle vapauksia ja tarina voikin tämän takia tehdä mutkia ja haaroja, jotka sitten jossain pelin vaiheessa palaavat takaisin päätarinan pariin. Kerronnan pääasiallinen tehtävä on koukuttaa pelaaja pelaamaan peliä (Manninen 2007, 62).

Jotta peliä voisi sanoa peliksi, on pelaajan kyettävä vaikuttamaan pelimaailmaan. Tässä astuu mukaan interaktiivisuus, eli kuinka pelaaja kykenee vaikuttamaan pelattavaan peliin tai muihin pelaajiin. Vuorovaikutteisuuteen kuuluu kaikki mitä pelaaja näkee, kuulee tai tekee. Pelaajan toimien seurauksena pelimaailmassa tapahtuu jotain, esimerkiksi pelihahmo hyppää. Pulmatehtävissä on helposti havaittavissa olevaa vuorovaikutusta. Pelaaja yrittää ratkaista käsillä olevaa ongelmaa ja peli antaa palautetta pelaajan tekemistä valinnoista (Manninen 2007, 63).

Kun pelin taustat ovat suunniteltu riittävän pitkälle, niin kehityksessä voi siirtyä pelimekaniikan vaiheeseen. Tämä on osio, missä pelin ohjelmoija on vahvasti mukana. Tässä vaiheessa peliin liitetään pelin sääntöjä, joiden mukaan peli toimii ja toiminnallisuudet, joilla pelaaja hallitsee peliä, kuten edellä mainittu hyppääminen. Projektin tässä vaiheessa alkaa myös näkymään konkreettista jälkeä, kun suunnitelmaa toteutetaan pelattavaan muotoon (Manninen 2007, 61).

4.2 Unity

Teemme pelimme prototyypin Unity-pelimoottorilla. Pelimoottorien tarkoitus on tuoda samaan ohjelmistopakettiin kaikki peliohjelmointiin tarvittava materiaali, kuten äänien lisääminen ja tekstuurien lisääminen hahmomallien päälle, sekä antaa pelinsuunnittelijoille mahdollisuuden keskittyä suunnittelutyöhön (Halpern 2018). Koska peliin tehtyjä muutoksia voi seurata halutessaan reaaliajassa, on kehittäminen suoraviivaista ja tehokasta. Unitysta löytyy myös mahdollisuus kehittää peliä usealle alustalle yhtä aikaa, valmiudet löytyvät muun muassa Androidille iOS:lle ja PC:lle. Mahdollisuus on kehittää myös eri perspektiivistä, kuten 2D ja 3D. 2D sisältää 2 eri ulottuvuutta, eli pituuden ja leveyden. 3D sisältää edellä mainittujen lisäksi syvyysulottuvuuden (BBC 2021).

Unityn saattaminen käyttökuntoon on verrattain helppoa. Asennuspaketin lataus tapahtuu Unityn nettisivujen kautta. Valittavana on mahdollisuus ladata haluttu versio Unitysta, tai Unity Hub:n lataaminen. Unity Hub on eri versioita helpottamaan tarkoitettu hallinnointiohjelma, jolla voi asentaa ja poistaa Unityn versioita, sekä hallinnoida ja luoda omia projekteja. Unityn dokumentointi on viimeisteltyä ja selkolukuista, näin ollen myös sen käyttäminen on sujuvaa. (Unity 2021l). Ohjelman käyttöliittymä on muokattavissa, joten siitä saa tehtyä mieleisensä näköisen.

Unity tukee C#:aa (Unity 2021h). Alun perin C# on kehitetty Microsoftin toimesta .NET- ohjelmistoviitekehystä varten (Microsoft 2021a). Sen myötä C#:sta on tullut standardikieli, josta löytyy valmista dokumentaatiota mittavia määriä. Tämä helpottaa työmme loppuun viemistä. C# on tuttu kieli, jos on joskus työskennellyt C:n, C++:n tai Javascriptin kanssa. Unityn asennuksen yhteydessä voi valita haluaako samalla asentaa Microsoft Visual Studio 2019-ohjelman (Microsoft 2019a). Visual studio on ohjelmointikehitysympäristö, joka toimii saumattomasti yhteistyössä Unityn kanssa. Unitysta voi aukaista suoraan ohjelmointitiedostot Visual Studiossa asettamalla Visual studio käyttöön aina uuden projektin alussa (Microsoft 2021b).

Unityyn voi ladata Asset Storesta Assetteja, jotka ovat ohjelmistokirjastoja. Kirjastot sisältävät erilaisia valmiita ratkaisuja, kuten valmiita koodinpätkiä tai

hahmomalleja. Asset Storeen on suora pääsy, joko Unityn käyttöliittymästä tai nettisivun kautta. Eniten ladattu Asset maaliskuussa 2021 on Standard assets -paketti, joka sisältää esimerkiksi hahmohallintaa ja kamerahallintaa. (Unity 2021j). Asset Storesta löytyy myös muita erilaisia kokoelmia, esimerkiksi valmiita äänipaketteja ja tekstuureja. Asset Storen Assetit mahdollistavat sen, että kaikkea ei välttämättä tarvitse tehdä itse. (Unity 2021k.)

Unityn ansaintamalli perustuu kuukausimaksuihin. Saatavilla on myös ilmainen versio, joka on tarkoitettu pienille pelinkehittäjille ja yksittäisille käyttäjille. Kuukausimaksulla, saa käyttöönsä erilaisia lisäominaisuuksia, jotka riippuvat kuukausimaksun määrästä. Halvimmassa paketissa saa käyttöönsä muun muassa diagnostiikkatyökaluja, kun taas kalliimmissa paketeissa on mukana muuan muassa tekninen tuki ja parempi asiakaspalvelun saatavuus. (Unity 2021a.)

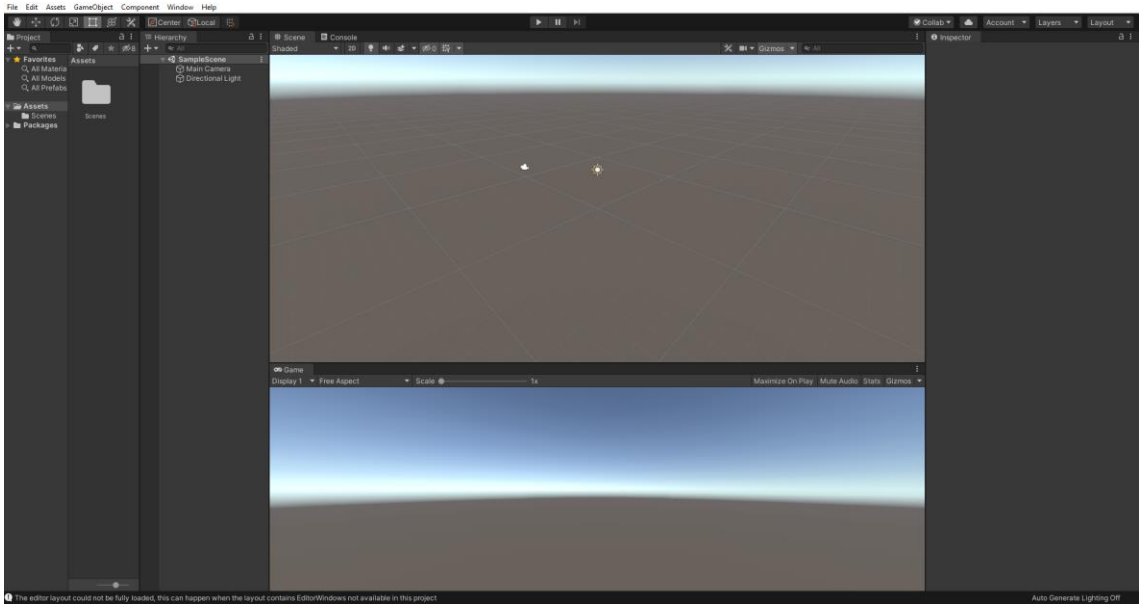
Unitya voi käyttää monenlaisiin peleihin. Suurin ja tunnetuin peli mahtanee olla Activision Blizzardin kehittämä Hearthstone-korttipeli, jolla oli Statistan mukaan n. 100 miljoona pelaajaa marraskuussa 2018 (Statista 2021). Unitya voi käyttää myös pienempiin peleihin ja mobiilipeleihin, kuten Monument valley 2 (Unity 2021g), joka on vanginnut pelaajien sydämet luovalla ongelmanratkaisullaan.

Vaikkakin Unity on pääasiassa tarkoitettu perinteiseen pelinkehitykseen kuten 2D ja 3D -pelit, voi sillä tehdä muutakin. Virtuaalitodellisuus saapui kuluttajien saataville vuonna 2016, kun Oculus ja Sony julkaisivat omat virtuaalisen todellisuuden mahdollistavat laitteensa (Verdict 2020). Unitya voi hyödyntää myös virtuaalipelien suunnittelussa, kuten Coco VR (Unity 2021d). Myös lyhytelokuvien tekeminen onnistuu Unitylla, tästä esimerkkinä voinee pitää Baymax Dreams -lyhytelokuvasarjaa (Unity 2021b).

Muita suosittuja pelimoottoreita on muun muassa Unreal Engine ja Game Maker. Unreal Engine on tarkoitettu kaiken kokoisille pelinkehitysyrityksille, olipa kyseessä sitten yksittäinen pelintekijä tai sitten iso kansainvälinen, satoja pelinkehittäjiä työllistävä yritys (Epic Games 2021). Game Maker on yksinkertaistettu pelimoottori, jonka tarkoituksena on antaa ihmisille mahdollisuus tehdä oma peli vähäisellä tai jopa olemattomalla ohjelmointitaidolla. Tämä mahdollistuu Game Makerin käyttämällä Drag and drop -tekniikalla. Drag and

drop on kehitysmetodi, jossa Game Makerin sisäisestä valikosta poimitaan toiminto, joka halutaan lisätä peliin. Tämän jälkeen toiminnon arvoja voi muokata sopivaksi ja lopulta toimintoja voi ketjuttaa tekemään haluttuja asioita (Yoyogames 2021),

Unityn käyttöliittymässä on erilaisia osia, jolla Unitya voi hallita (Kuva 1). Osien paikkoja ja kokoja voi muuttaa mieleisekseen tai valita mieleisensä valmiiksi tehdyistä järjestyksistä. Inspector-näkymässä hallitaan objekteja ja niiden ominaisuuksia. Myös tehtyjen skriptien hallinta ja implementointi hoidetaan tästä. Game-ruudussa voi kokeilla reaaliajassa meneillään olevaa projektia, eli pelata tekeillä olevaa peliä. Scene-näkymässä voi tarkastella projektia kehittäjän näkökulmasta. Voi myös hallita objektien kokoja ja sijaintia. Projekteissa voi tulla vastaan tilanteita, missä kaikki ei mene ihan putkeen ja tilanteen ratkaisemissa auttaa Console. Virheistä ilmoittavat koodit tulevat Consoleen, josta ne voi tarkastaa ja koodia apuna käyttäen ongelman ratkaista. Consoleen voi myös tulostaa skriptillä tietoja halutessaan. Project-osio sisältää meneillään olevan projektin kaiken sisällön. Pääasiallinen tehtävä Project-osiolla on kansiorakenteen hallinta. Hierarchy-näkymästä näkee avoinna olevan projektin hierarkian, eli missä järjestyksessä pelissä olevat objektit reagoivat toisiinsa.(Unity 2021m.)



Kuva 1. Unityn käyttöliittymä

4.3 Pelimme idea

Pelin idea on opettaa lapsille tietoturvaan liittyviä asiasanoja osana varhaiskasvatusta. Pelaajan täytyy yhdistellä kortteja kahdesta eri sarakkeesta ja yrittää löytää pari. Ideana on, että toisen sarakkeen kortit kuvaavat tietoturvauhkia, ja toisessa sarakkeessa on tapa suojautua näitä uhkia vastaan, esim. Virustorjuntaohjelma ja Virus tuottavat parin.

Tarkoitus on, että lapsi ja varhaiskasvatuksen ohjaaja käyvät yhdessä läpi kortit ja niihin liittyvät tekstit sovelluksesta löytyvästä paikasta. Tämä löytyy aloitusnäytöltä kohdasta ”Kortit”. Korteissa on kerrottu lyhyesti, ja lapsille suunnatulla kielellä, mitä kortin aihe tarkoittaa.

Peli ei toimi tavanomaisen muistipelin tavalla, vaan on enemmänkin yhdistelmämuistipeli. Molemmissa sarakkeissa olevat kortit ovat ulkonäöltään eriäviä. Näin ollen pelaajan täytyy opetella edes vähän teoriaa korttien takana, jotta peliä pystyy pelaamaan. Tämä ei kuitenkaan saa olla liian haastavaa, koska pelin kohdeyleisö on 5–6-vuotiaat lapset.

Pelin tarkoitus ei ole opettaa kaikkea aihealueeseen liittyvää, vaan tarkoitus olisi saada lapset tietoiseksi tietoturvaan liittyvistä uhista ja suojautumisesta näitä uhkia vastaan. Tavoite olisi, että lapselle jää mieleen tietoturvaan liittyvää sanastoa ja näin ollen hänellä on vanhemmalla iällä paremmat lähtökohdat rakentaa tietoturvaverkkoa omassa IT-ympäristössään.

5 PELIN TOTEUTUS

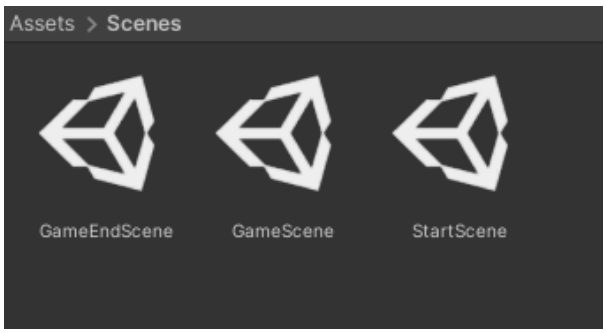
Kun peliä on aloitettu suunnittelemaan, ei kummallakaan ollut kokemusta Unitystä, eikä C# ohjelmointikielestä. Ainoat kokemukset koodaamisesta oli koulussa käyty web-sovelluksen koodaus, joka toteutettiin PHP ohjelmointikielellä.

Grafiikkojen osalta jouduimme heti alussa toteamaan, ettei kummallakaan ole niin hyvää visuaalista silmää, eikä kuvanmuokkaustaitoa, jotta pelistä saataisiin lasten silmää miellyttävää visuaalista kokonaisuutta. Koska toteutus tehdään Unity:llä, emme nähneet tätä ongelmana, vaan tarpeena saada tiimiimme lisähenkilö, joka kykenee toteuttamaan visuaalisia kokonaisuuksia. Unity:n yksi parhaimmista puolista on nopea UI:n suunnittelu ja testaus. Näin ollen pystyimme toteuttamaan pelin toiminnot valmiiksi ja kaikki kuvat voidaan istuttaa jälkeenpäin paikoilleen rikkomatta pelin logiikkaa.

Koska molemmat lähtivät toteuttamaan projektia lähes nollatason koodaamisosaamisella, ensimmäinen vaihe oli tutustua Unityyn ja siinä käytettävään C# ohjelmointikieleen. Udemy on webissä toimiva palvelu, josta voi ostaa valmiita kurssikokonaisuuksia eri koodauskielistä, elämäntapavalmennuksesta ja monesta muusta aihealueesta. Monen Unity kurssin joukosta valikoitui ”Complete C# Unity Game Developer 2D”-kurssi, jonka avulla sai käsityksen, miten Unity ja C# toimivat keskenään. Unity Forum on myös paikka, johono aloittelevan pelinkehittäjän on syytä tutustua.

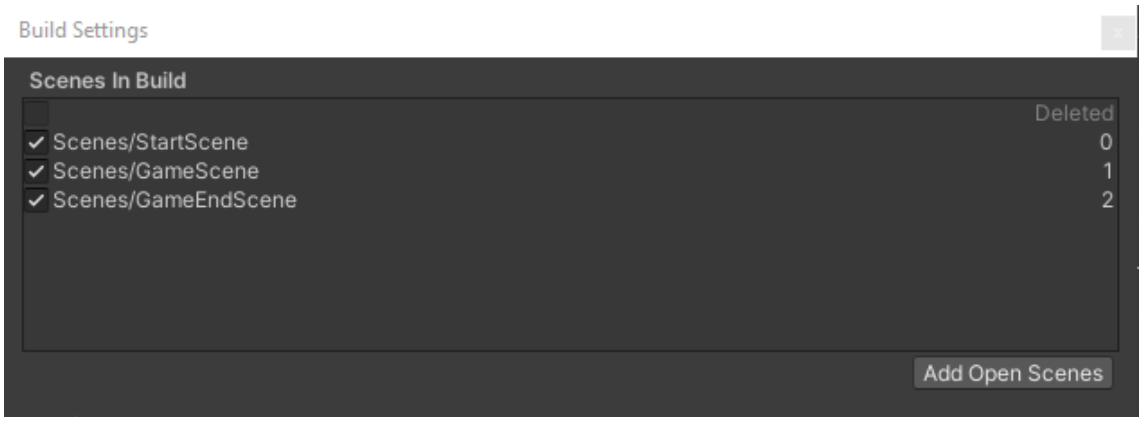
5.1 Navigointi

Ensimmäinen toiminnallisuus, jonka toteutimme peliin, oli navigointi eri scenejen välillä. Scenet ovat Unity:ssä eri näkymiä, jotka sisältävät kaiken toiminnallisuuden, kuten UI-elementit ja GameObjectit. Meidän sovelluksemme sisältää tällä hetkellä kolme sceneä (Kuva 2).



Kuva 2. scenet unityn näkymässä

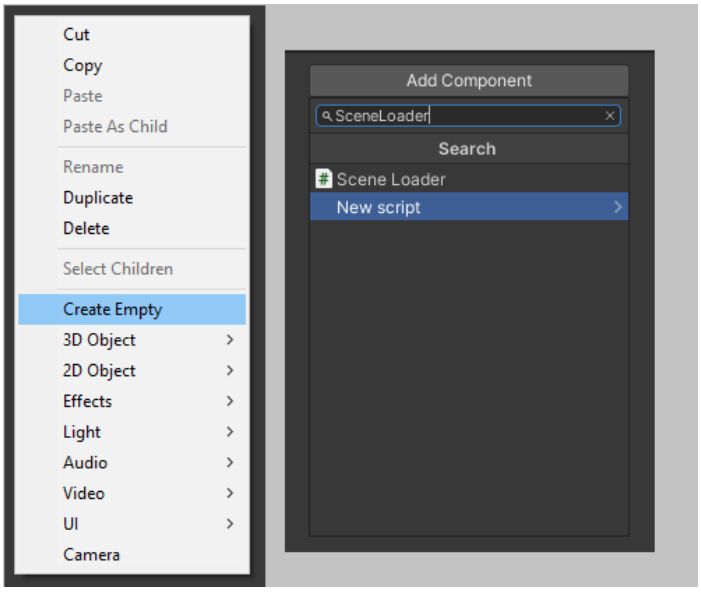
Unity:ssä scenejen järjestystä hallitaan Build Settings-valikossa, joka löytyy File-valikosta. Scenejen järjestystä muokataan yksinkertaisesti drag and drop menetelmällä. Aina kun scene lisätään Build Settingsiin, niille generoituu index-numero. Index-numero määräytyy Scenejen järjestyksen mukaan (Kuva 3).



Kuva 3. Build settings-näkymä

Unityn SceneManager:lla pystytään hallitsemaan scenejä joko index-numeron mukaan tai nimen mukaan. Meidän toteutuksemme käyttää index-numeroa navigoinnissa.

Jotta SceneManager:a päästään hallitsemaan, täytyy ensin luoda uusi script-tiedosto ja liitää se GameObject:iin. Tämä tapahtuu helpoiten luomalla uusi GameObject, ja luoda sinne uusi script-tiedosto Inspector paneelista (Kuva 4).



Kuva 4. Scriptin lisäys objektiin Inspector-paneelissa.

Tällä tavalla toimiessa script-tiedosto liitetään automaattisesti luotuun GameObjectiin. Luotu script-tiedosto ilmestyy myös Unity:n assets-paneeliin. Tässä tapauksessa assets-paneeliin ilmestyy SceneLoader.cs niminen tiedosto.

Tuplaklikkaamalla tiedostoa se aukeaa Visual Studio koodieditorissa, tai siinä editorissa, jonka on käynyt etukäteen määrittämässä. Scenejen hallinta C#:lla ei vaadi montaa riviä koodia, mutta yksi asia on tärkeä muistaa tehdä. Scriptin alussa kerrotaan, mitä namespace-luokkakirjastoja skripti käyttää. SceneManager käyttää UnityEngine.SceneManagement luokkakirjastoa (Kuva 5). Ilman kuvan 5, rivillä 4 olevaa koodinpätkää ei SceneManager toimi.

```
3    using UnityEngine;
4    using UnityEngine.SceneManagement;
5
```

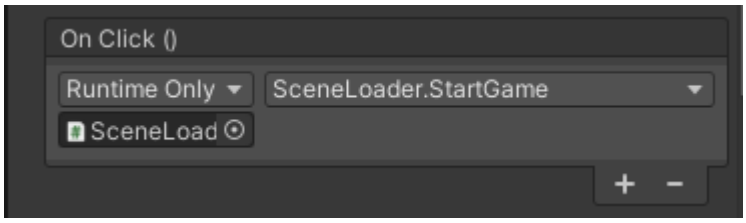
Kuva 5. namespace- luokkakirjasto

Meillä on käytössä kolme skenaariota, joissa tarvitaan SceneLoader GameObjectia. Ensimmäinen on, miten StartScene:stä päästään GameScene-näkymään. Tätä varten luodaan SceneLoader.cs-tiedostoon uusi julkinen funktio ja nimetään se tarkoituksenmukaisesti. Tämän jälkeen kerrotaan, mikä scene ladataan, kun kyseistä funktiota kutsutaan (Kuva 6).

```
25      public void StartGame()
26      {
27          SceneManager.LoadScene(1);
28      }
29  }
30
```

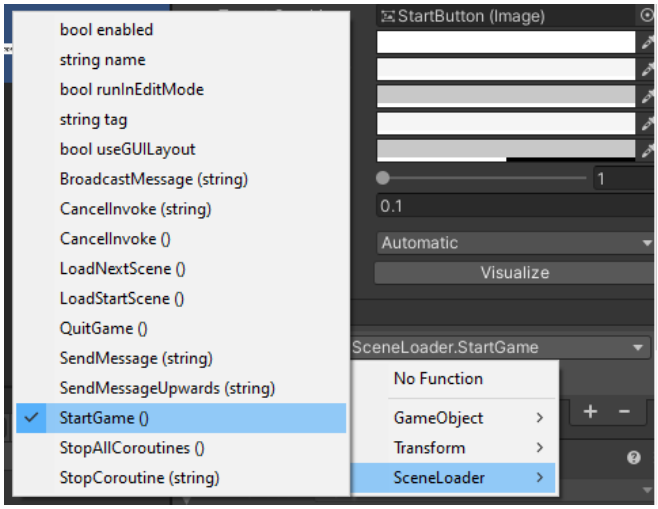
Kuva 6. Scenen lataus- skripti

Kun script-tiedosto on saatu valmiiksi, täytyy luotu toiminnallisuus käydä ottamassa käyttöön Unityssa. SceneLoader.cs-funktiota kutsutaan, kun nappia painetaan. Funktion saa käyttöön menemällä napin Inspector-valikkoon ja vetämällä luotu GameObject OnClick() kohtaan (Kuva 7).



Kuva 7. On click-valikko Inspector-paneelissa

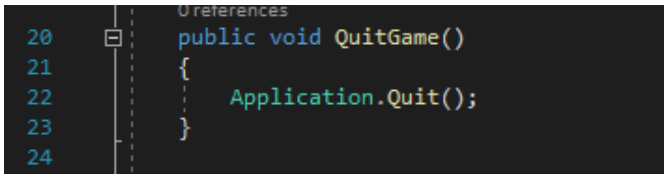
Tämän jälkeen juuri luotu funktio löytyy OnClick- asetuksen alasvetovalikosta (Kuva 8).



Kuva 8. OnClick-alasveteovalikko

Toinen skenaario on, miten peli sammutetaan. Tämä on tärkeä toteuttaa varsinkin siinä tapauksessa, jos peliä haluaa testata standalone-buildina tai Android debugger:lla. Kaikki edellä mainitut vaiheet toteutetaan samalla tavalla,

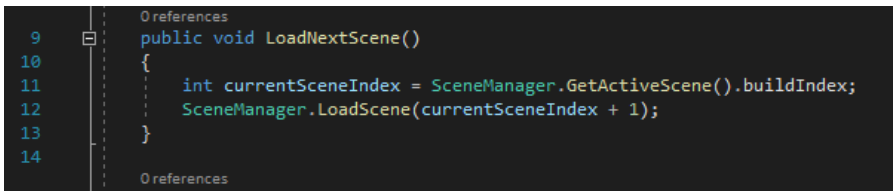
kuin StartGame() funktionkin kanssa. SceneLoader.cs tiedostoon luodaan uusi funktio, joka ei ohjaakaan peliä uuteen sceneen vaan sammuttaa sovelluksen (Kuva 9).



Kuva 9. Sovelluksen sammutus- skripti

Kolmas skenaario poikkeaa toiminnaltaan hieman jo mainituista. Tässä tapauksessa, kun kaikki parit on löydetty, niin ohjelma kutsuu automaattisesti funktiota, joka ohjaa pelin seuraavaan sceneen. Meidän tapauksessamme peli ohjautuu GameScenestä GameEndScene:een. Tämänkin olisi voinut toteuttaa samalla kaavalla, kuin ensimmäisessä skenaariossa on kuvattu. Eli olisi kirjoitettu scenen Index-numero sulkuihin ja kaikki olisi toiminut. Halusimme kuitenkin toteuttaa tämän hieman eri tavalla.

Koska pelin lopetus scenen voi olettaa tulevan aina peli-scenen jälkeen, voidaan siirtyminen toteuttaa hakemalla käynnissä olevan scenen index-numero ja yksinkertaisesti kasvattaa sitä yhdellä (Kuva 10). Luvussa 5.5 kerrotaan, miten ohjelma käyttää tätä funktiota.



Kuva 10. Siirtymisskripti pelinäkymästä loppunäkymään

Soveltamalla näitä kolmea funktiota saamme hoidettua kaiken navigaation, mitä pelissämme tulee tulevaisuudessa tarvitsemaan.

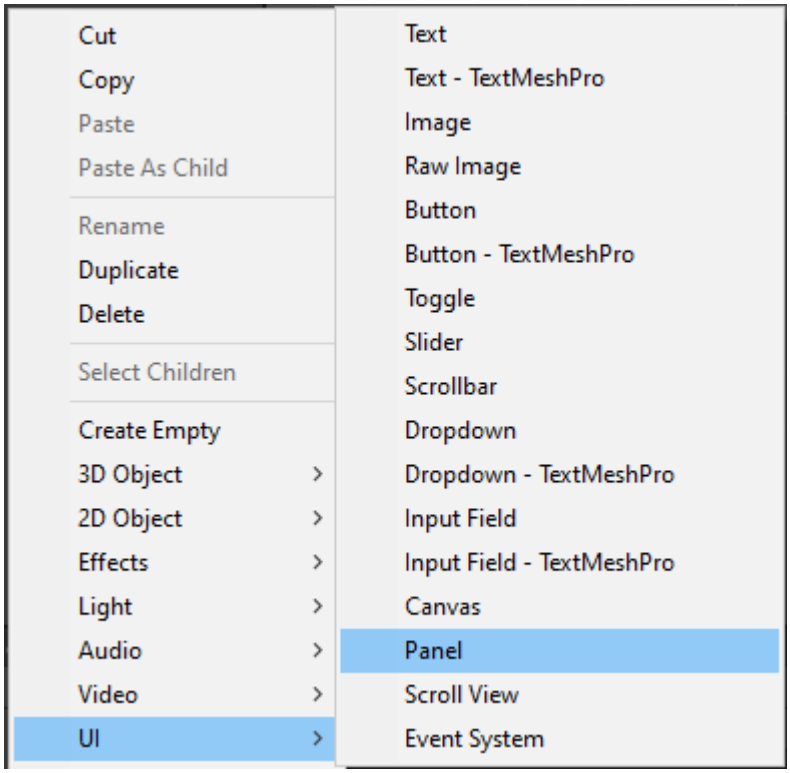
5.2 Nappien tulostus

Pelinäkymän toteutukseen testasimme useampaa eri tapaa, miten kortit saadaan tulostettua näytölle. Ensimmäiset kokeilut toteutettiin manuaalisesti suoraan Unityn kautta. Tällä tavalla on helppo testata ideoita käytännössä, mutta pitkällä

tähtäimellä tämä käy työlääksi. Pelissä tulee olemaan useampi vaikeustaso. Manuaalisesti toteutettuna tämä tarkoittaa sitä, että jokaiselle vaikeustasolle joudutaan tekemään asemointi ja sommittelu erikseen. Näin ollen halusimme toteuttaa korttien tulostuksen koodilla, jotta skaalautuvuus tulevaisuuden tarpeisiin onnistuu vaivattomammin.

Meillä oli myös kaksi vaihtoehtoa kortin UI-objektiksi. Kortissa on yleensä kaksi puolta, etu- ja takakansi. Tämä voidaan toteuttaa joko luomalla UI-kuva, jolle annetaan edellä mainitut kaksi arvoa, tai meidän tapamme luoda UI-nappi, jolle annetaan edellä mainitut arvot.

UI-elementin luonti Unityssa on helppoa. Hiiren oikealla napilla painetaan scenen Hierarchy-paneelia ja valikko aukeaa näkyviin. Valikosta valitaan UI, jonka alta löytyy kaikki valmiit elementit (Kuva 11).



Kuva 11. UI:n luomisvalikko Unityssa

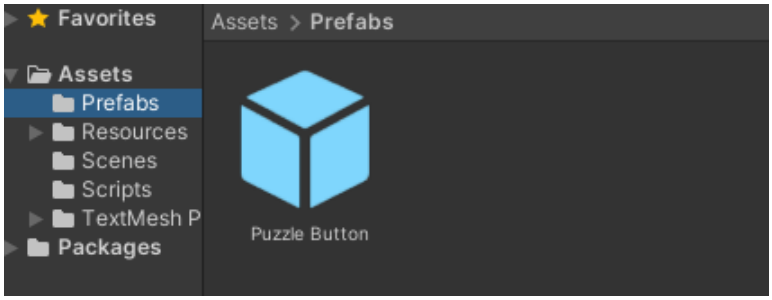
Ennen koodin lisäämistä täytyy Unityn puolella tehdä muutama esivalmistelu. GameScene on tällä hetkellä vielä tyhjä, joten luomme sinne UI-elementin nimeltä Canvas. Canvas on peliobjekti, joka sisältää canvas-elementin, jonka sisälle kaikki UI-elementit on hyvä luoda (unity 2021c). Canvas luodaan

automaattisesti, jos yrittää luoda UI-elementin suoraan näytölle. Canvas kannattaa tässä vaiheessa jo asemoida kameranäkymän mukaan. Tämä tapahtuu Unity:n Inspector paneelissa.

Canvas elementin alle luodaan uusi UI-elementti nimeltä UI-Panel. Paneelin tehtävä on myös pitää elementit järjestyksessä. Lisäksi paneeliin voidaan lisätä esimerkiksi taustakuva, joka meilläkin tulee peliin jatkokehityksessä.

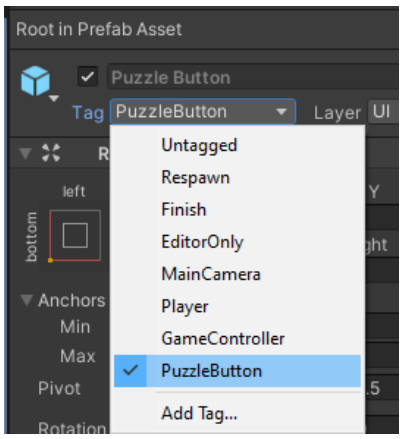
Panel-elementin alle luodaan UI Button-elementti. Tämä tapahtuu vielä tässä vaiheessa manuaalisesti. Nappia ei kuitenkaan ole tarkoitus jättää tässä vaiheessa UI-elementiksi, vaan napista tehdään uudelleen käytettävä peliobjekti.

Uudelleen käytettävät peliobjektit tunnetaan Unity:ssä nimellä ”Prefabs”. Uuden prefabs-objektin luonti tapahtuu helposti Unityn käyttöliittymässä. Ensin luodaan hierarkiaan uusi kansio nimeltä ”Prefabs”. Tämän jälkeen hiirellä raahataan vasta luotu elementti, meidän tapauksessamme UI Button, Prefabs-kansioon. Tämän jälkeen napin voi poistaa canvasilta ja sen voi palauttaa raahaamalla takaisin canvasille. Kuva 12 on otettu Unityn Project -välilehdeltä, Prefabs-kansion sisällöstä, jossa napistamme luoma prefabs on tallennettuna (Kuva 12).



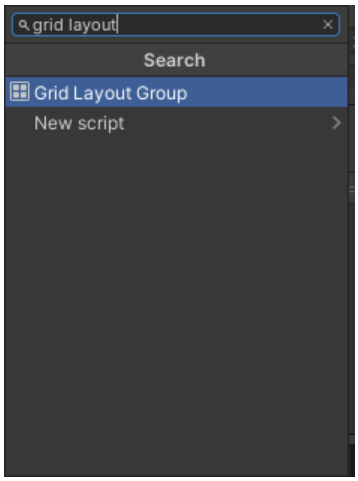
Kuva 12. Prefabs-kansio

Luotuun napin prefabs:iin täytyy vielä lisätä tag, jotta pääsemme hallitsemaan luotuja nappeja toisessa skriptissä. Tag lisätään nappiin Unity:n Inspector-valikossa (Kuva 13).



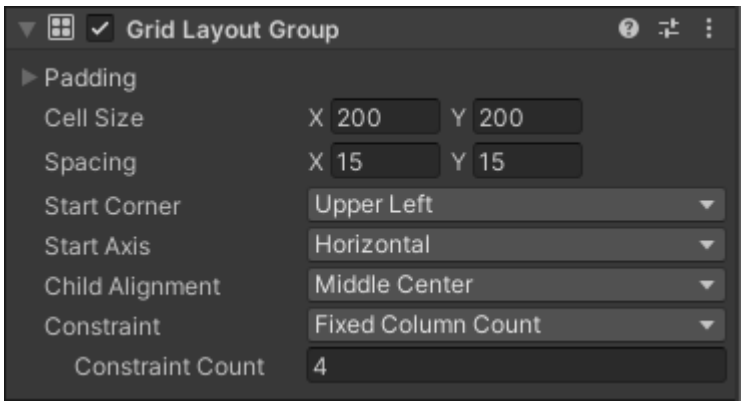
Kuva 13. Tag:n lisäys nappiin.

Jotta saisimme napit tulostettua riviin, ilman etukäteen tehtyä nappien järjestystä, lisäsimme Panel-elementtiin komponentin nimeltä ”Grid Layout Group” (Kuva 14).



Kuva 14. Grid layout- valikko

Grid Layout Group on Unity:n sisäänrakennettu komponentti, joka järjestää elementin child gridiin (Unity 2021e). Grid Layout Group:ssa pystyy muokkaamaan mm. tulostettavan elementin kokoa ja määrittämän, kuinka suuret välit elementeille tulostetaan. Kuvassa 15 on näkymä Grid Layout Group:n säätömahdollisuuksista. Kuvassa näkyy, että olemme tähän asti muokanneet neljää asetusta. Napin koko on muutettu (Cell Size), nappien välejä on kasvatettu Y- sekä X-akselilla, mistä kohdasta ruutua tulostus lähtee (Child Alignment), sekä kuinka monta nappia tulostuu aina riville.



Kuva 15. Grid layout- asetukset.

Tämän jälkeen peli on valmisteltu Unity:n päässä valmiiksi. Tässä vaiheessa peli ei tee vielä mitään käynnistyessään. Tätä varten luomme uuden tyhjän GameObjektin nimeltä GameController, johon lisäämme script -tiedostot, joita peli tarvitsee toimiakseen.

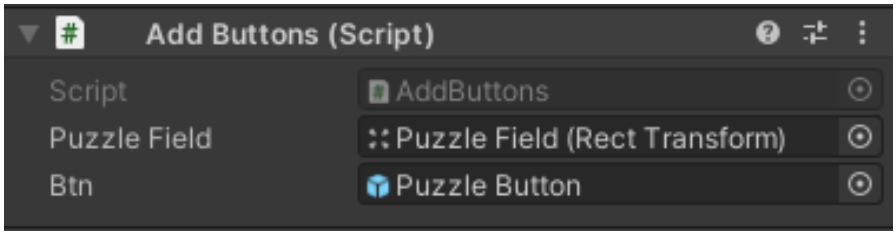
Ensimmäinen Script -tiedosto on nimeltään AddButtons.cs, jonka tehtävä on vain tulostaa haluttu määrä nappeja näytölle. Tulostuksen yhteydessä napeille annetaan myös nimi index-numeron mukaan, sekä ne asetetaan puzzleField Transform muuttujan child-elementiksi. Kuvassa 16 on näytetty AddButtons.cs tiedoston sisältö (Kuva 16).

```
7       [SerializeField]
8       private Transform puzzleField;
9
10      [SerializeField]
11      private GameObject btn;
12
            Unity Message | 0 references
13      private void Awake()
14      {
15          for (int i = 0; i < 8; i++)
16          {
17              GameObject button = Instantiate(btn);
18              button.name = "" + i;
19              button.transform.SetParent(puzzleField, false);
20          }
21      }
22  }
23
```

Kuva 16. AddButtons.cs- skripti

Skriptin alussa luodaan kaksi yksityistä (private) muuttujaa. Muuttujien yhteyteen lisätään vielä [SerializeField]-teksti, jotta ohjelma ottaa käynnistyessään yksityiseksi merkityt muuttujat huomioon (Unity 2021i).

Muuttujien määrittämät kentät tulevat näkyviin myös Unityn päässä siinä GameObject:ssa, johon kyseinen skriptitiedosto on yhdistetty (Kuva 17).



Kuva 17. Unityn näkymä AddButtons- skriptistä

PuzzleField-muuttujan arvo on aikaisemmin luodun UI Panel-elementin arvo. Tämän saa otettua käyttöön Unity:n käyttöliittymässä vetämällä Panel-elementti hiirellä luotuun puzzleField-muuttujaan. Btn-muuttuja on määritetty olemaan GameObject, joka meidän tapauksessamme on aikaisemmin luotu Prefabs-napista. Napin prefabs-elementin voi myös raahata hiirellä Btn-muuttujaan.

Nappien tulostus tapahtuu for-loopissa, joka ajetaan Awake()-funktiossa. Awake()-funktiota kutsutaan ennen sovelluksen käynnistystä. Eli kun peli käynnistyy, niin sillä on jo tiedossa mitä elementtejä tulostetaan ja kuinka monta niitä tulostetaan. Tässä vaiheessa ohjelmalla ei ole vielä tiedossa, mitä kuvia nappeihin tulostetaan tai mitä tapahtuu, kun nappia painetaan.

5.3 GameController

Pelilogiikka on rakennettu omaan skriptitiedostoon. Nimesimme skriptin GameController.cs ja liitimme sen samaan GameObject:iin, kuin AddButtons.cs.

GameController.cs-tiedoston rakentaminen alkaa luokkakirjastojen viittauksilla. GameController hallitsee UI-objekteja, joten pääsy Unityn UI- luokkiin tarvitaan. Tämä tapahtuu lisäämällä alkuun ”using UnityEngine.UI”. Käytämme myös listoja nappien kuvien tulostuksessa, joten tarvitsemme myös pääsyn näihin luokkiin. Tämä tapahtuu lisäämällä alkuun ”using System.Collections.Generic”. Kuvassa 18 on nähtävissä kaikki luokkakirjastot, joita GameController käyttää.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
```

Kuva 18. GameControllerin luokkakirjastot.

Kaikkia funktioita, joita käytetään pelin asemoimisessa, kutsutaan Start()-funktiossa. Start()-funktiota kutsutaan sovelluksen käynnistyessä ensimmäisenä (Unity 2021f). Kuva 19 näyttää kaikki funktiot, joita pelimme käynnistyessään kutsuu.

```
Unity Message | 0 references
void Start()
{
    GetButtons();
    AddListeners();
    AddGameCards();
    Shuffle(gameCards);
    gameGuesses = gameCards.Count / 2;
}
```

Kuva 19. Pelin kutstuttavat funktiot

GetButtons()-funktio hakee AddButtons.cs-skriptissä luodut napit ja käyttää tähän hakuun nappeihin lisättyä tag:a. Tämän jälkeen napit lisätään ’btns’ muuttujaan ja niille annetaan taustakuva (Kuva 20). Muuttujat sijaitsevat skriptin alussa, ennen ensimmäistä funktiota. Niitä ei tarvitse funktion sisällä enää esittää.

```
1 reference
void GetButtons()
{
    //Haetaan napit tagin avulla.
    GameObject[] objects = GameObject.FindGameObjectsWithTag("PuzzleButton");

    //Lisätään napit 'btns' variableen ja näytetään bgImage
    for (int i = 0; i < objects.Length; i++)
    {
        btns.Add(objects[i].GetComponent<Button>());
        btns[i].image.sprite = bgImage;
    }
}
```

Kuva 20. Taustakuvan lisäys nappeihin

AddListeners()-funktio tunnistaa, kun nappia painetaan. Tämä sama toiminto on toteutettu manuaalisesti SceneLoader:n kanssa, mutta pelinäkymässä tätä ei voida manuaalisesti toteuttaa, koska napit luodaan vasta pelin käynnistämisen

jälkeen. Kun nappia painetaan, AddListeners() kutsuu PickACard()-funktiota (Kuva 21).

```
1 reference
void AddListeners()
{
    //Nappeihin lisätään Listener, joka tunnistaa kun korttia painetaan.
    //OnClick kutsuu PickACard() funktiota.
    foreach (Button btn in btns)
    {
        btn.onClick.AddListener(() => PickACard());
    }
}
```

Kuva 21. Addlistener:n kutsu

Tässä vaiheessa pelin napit sisältävät vain taustakuvan. AddGameCards() funktiolla haetaan korttien kuvat ja lisätään ’gameCards’ muuttujaan. Koska pelissämme on kahden tyyppisiä kortteja, antivirus ja virus, joudutaan kortit hakemaan kahdesta eri muuttujasta. Tämän toteutimme käyttämällä kahta eri sisäistä muuttujaa (index ja tmp) ja for-loopia (Kuva 22).

```
1 reference
void AddGameCards()
{
    int looper = btns.Count;
    int index = 0;
    int tmp = 0;

    //Pelikortit haetaan ja lisätään gameCards variableen.
    //Antivirus haetaan index mukaan ja Virus haetaan tmp mukaan
    for (int i = 0; i < looper; i++)
    {
        if (index < looper / 2)
        {
            gameCards.Add(antivirus[index]);
            index++;
        } else
        {
            gameCards.Add(virus[tmp]);
            tmp++;
        }
    }
}
```

Kuva 22. Antivirus- ja viruskorttien haku

Antivirus ja virus-muuttujat on luotu skriptin alussa. Ilman Awake()-funktiossa tapahtuvaa kuvien noutoa, ei AddGameCard()-funktio löydä kyseisistä muuttujista arvoja. Korttien kuvat ladataan ’Resources’ kansiosta ja haku tapahtuu kansiopolun mukaan (Kuva23).

```
Unity Message | 0 references
private void Awake()
{
    //Haetaan korttien kuvat resurssikansiosta
    antivirus = Resources.LoadAll<Sprite>("Sprites/Cards/Antivirus");
    virus = Resources.LoadAll<Sprite>("Sprites/Cards/Virus");

}
```

Kuva 23. Korttien kuvien lataus

Shuffle()-funktion tehtävä on sekoittaa kortit satunnaiseen järjestykseen pelinäytöllä. Ilman Shuffle()-funktiota tulostus tapahtuu niin, että ensin antivirus-muuttujan neljä ensimmäistä arvoa tulostuu, jonka jälkeen virus-muuttujan neljä ensimmäistä arvoa tulostuu. Näin ollen parien löytö olisi liian helppoa, eikä pelinäkymä muuttuisi pelikertojen välillä. Shuffle()-funktio suluissa on määritetty argumentti ja nimetty lista nimellä ’list’ (Kuva 24). For-loop näyttää hieman sekavalta, mutta yksinkertaistettuna se hakee listan Sprite:jä, eli kuvia, pyörittää for loop:n kuvien lukumäärän mukaan ja sekoittaa syntyneen listan numeroita.

```
1 reference
void Shuffle(List<Sprite> list)
{
    for(int i = 0; i < list.Count; i++)
    {
        Sprite temp = list[i];
        int randomIndex = Random.Range(i, list.Count);
        list[i] = list[randomIndex];
        list[randomIndex] = temp;
    }
}
```

Kuva 24. Korttien sekoitus

Start()-funktiossa on vielä lisätty Shuffle()-funktion sulkuihin listan nimi, jota funktio sekoittaa. Meidän tapauksessamme haluamme sekoittaa pelikorttien järjestyksen ja pelikortit löytyvät gameCards-muuttujasta (Kuva 25).

```
40      void Start()
41      {
42          GetButtons();
43          AddListeners();
44          AddgameCards();
45          Shuffle(gameCards);
46          gameGuesses = gameCards.Count / 2;
47      }
48
```

Kuva 25. Sekoitettujen korttien järjestyksen tallentaminen

PickACard()-funktio on ensimmäinen funktio, joka hallitsee pelilogiikkaa. Pelilogiikkaa varten tarvitsee skriptin alkuun luoda uusia muuttujia. Osaa muuttujista käytetään korttien tunnistamiseen ja osaa käytetään pisteiden laskussa. Kuvassa 26 on nähtävillä pelilogiikan toteuttamiseen tarvittavat muuttujat.

```
private bool firstGuess, secondGuess;

private int countGuesses;
private int countCorrectGuesses;
private int gameGuesses;

private int firstGuessIndex, secondGuessIndex;

private string firstGuessCard, secondGuessCard;
```

Kuva 26. Muuttujia

PickACard()-funktiossa käytetään ’!’ ehtoa, joka muuttaa ehdon jälkeisen osan vastakohdaksi. FirstGuess on boolean-muuttuja, joka kantaa arvoa ’false’ tai ’true’. Boolean-arvo on oletusarvoisesti ’false’(Microsoft 2019b). Näin ollen funktio alkaa kysymällä ”Onko firstGuess ’true’”, jonka jälkeen firstGuess-muuttujan arvo muutetaan true-arvoksi (Kuva 26). Tällä tavalla toinen napin painallus ei enää täytä ehtoa ja hypätään seuraavaan ehtoon.

Ehdon täyttyessä haetaan painetun objektin nimi, muutetaan se Parse-methodilla numeromuotoon ja tallennetaan muuttujaan (Kuva 27).

```
1 reference
public void PickACard()
{
    //Jos ei ole ensimmäinen arvaus
    if(!firstGuess)
    {
        firstGuess = true;
        //Haetaan painetun napin nimi ja muutetaan se integeriksi.
        firstGuessIndex = int.Parse(UnityEngine.EventSystems.EventSystem.current.currentSelectedGameObject.name);
        //firstGuessCard muuttujaan lisätään pelikortin nimi
        firstGuessCard = gameCards[firstGuessIndex].name;
        //Napin kuva tulostuu gameCards muuttujasta firstGuessIndexin mukaan.
        btns[firstGuessIndex].image.sprite = gameCards[firstGuessIndex];

    } else if (!secondGuess)
    {
        //Tekee saman kuin firstGuess
        secondGuess = true;
        secondGuessIndex = int.Parse(UnityEngine.EventSystems.EventSystem.current.currentSelectedGameObject.name);

        secondGuessCard = gameCards[secondGuessIndex].name;

        btns[secondGuessIndex].image.sprite = gameCards[secondGuessIndex];
        //countGuess muuttujan arvoa kasvatetaan yhdellä.
        countGuesses++;

        StartCoroutine(CheckIfTheCardsMatch());
    }
}
```

Kuva 27. arvauksen tallennus muuttujaan

PickACard()-funktion toinen ehto tekee saman, kuin ensimmäinenkin, mutta eri muuttujilla. Ehdon lopussa tapahtuu kuitenkin countGuesses-muuttujan arvon kasvattaminen, sekä kutsutaan funktiota, joka tarkistaa onko valitut kortit pareja.

CheckIfTheCardsMatch()-funktiota kutsutaan, kun PickACard()-funktion molemmat ehdot on käyty läpi. Parien tarkistukseen käytämme korttien nimiä, jotka olemme tallentaneet muuttujiin edellisessä funktiossa. Tässä vaiheessa on hyvä huomioida, että Resources-kansiosta löytyvissä Antivirus- ja Virus-kansioissa olevat kuvat ovat nimetty yhtenevästi. Vaikka korttien kuvat ovat eriäviä, niin niiden nimet voivat olla yhteneviä. Olemme käyttäneet tätä hyväksi parien tarkistuksessa.

Ehtolauseessa tarkistetaan, onko firstGuessCard-muuttujan arvo sama, kuin secondGuessCard-muuttujan arvo. Jos ehto täyttyy, napit jäädytetään, ettei niitä voi enää painaa. Kortit myös piilotetaan näytöltä. Tämän jälkeen kutsutaan funktiota, joka tarkistaa päättyikö peli. Jos ehto ei täyty, nappeihin tulostetaan taustakuva (Kuva28).

Funktion lopussa vielä asetetaan boolean-muuttujien, firstGuess ja secondGuess, arvot ’false’ksi (Kuva 28). Tämä on tärkeä vaihe, jotta peli pystyy jatkumaan.

```
1 reference
IEnumerator CheckIfTheCardsMatch()
{
    yield return new WaitForSeconds(1f);

    //Tarkistetaan onko löytynyt pari.
    //Onko muuttujien arvot samat.
    if (firstGuessCard == secondGuessCard)
    {
        yield return new WaitForSeconds(.5f);

        //Parin löydyttyä näitä kortteja ei voi enää klikata
        btns[firstGuessIndex].interactable = false;
        btns[secondGuessIndex].interactable = false;

        //Kortit muutetaan näkymättömiksi
        btns[firstGuessIndex].image.color = new Color(0, 0, 0, 0);
        btns[secondGuessIndex].image.color = new Color(0, 0, 0, 0);

        //Kutsutaan funktiota, joka tarkistaa päättyikö peli.
        CheckIfTheGameIsFinished();
    } else
    {
        yield return new WaitForSeconds(.5f);

        //Paria ei löytynyt. Palautetaan taustakuva kortteihin.
        btns[firstGuessIndex].image.sprite = bgImage;
        btns[secondGuessIndex].image.sprite = bgImage;
    }

    yield return new WaitForSeconds(.5f);

    //Boolean arvot palautetaan arvoon false.
    firstGuess = secondGuess = false;

}
```

Kuva 28. Muuttujien arvon muuttaminen false-tilaan

CheckIfTheGameIsFinished()-funktion tarkoitus on päättää peli, kun kaikki parit on löydetty. Start()-funktiossa on määritetty, kuinka monta paria pelissä on ja sen arvo on tallennettu muuttujaan nimeltä ’gameGuesses’ (Kuva 25).

CheckIfTheGameIsFinished()-funktiota kutsutaan aina, kun pari on löytynyt. Näin ollen voimme jokaisen kutsun yhteydessä kasvattaa countCorrectGuesses-muuttujan arvoa ja verrata tätä gameGuesses-muuttujan arvoon (Kuva 29). Jos ehto täyttyy, kirjoitetaan lokiin, kuinka monta arvausta tarvittiin pelin läpi viemiseen, sekä ohjataan peli seuraavaan sceneen.

```
1 reference
void CheckIfTheGameIsFinished()
{
    countCorrectGuesses++;
    //Tarkistetaan onko kortteja vielä pelattavana vai onko peli päättynyt.
    if (countCorrectGuesses == gameGuesses)
    {
        //Kirjoitetaan lokiin tulokset näin aluksi.
        Debug.Log("Game Finished");
        Debug.Log("It took you " + countGuesses + " many guess(es) to finish the game");

        //Siirtää ehdon täyttyessä pelin seuraavaan sceneen.
        //Käyttää UnityEngine.SceneManagement.
        int currentSceneIndex = SceneManager.GetActiveScene().buildIndex;
        SceneManager.LoadScene(currentSceneIndex + 1);
    }
}
```

Kuva 29. Arvausten määrän tallentaminen muuttujaan ja tulostus

6 JATKOKEHITYS

Pelin nykyisessä versiossa on itse tehdyt yksinkertaiset graafiset kuviot, jotta pelin kulku on mahdollista hahmottaa. Jatkokehityksen aikana grafiikat päivittyvät näyttävämmäksi ja lasten maailmaan sopivammaksi, kuten värikkääksi maailmaksi ja huomiota herättäviksi visuaalisiksi elementeiksi.

6.1 Logiikkamuutokset

Tällä hetkellä peliin on luotu vasta elementit, joilla voidaan lähteä jatkokehitystä suorittamaan. Logiikan osalta suurimmat muutostarpeet ovat korttien tulostuksessa.

Tällä hetkellä kortit tulostuvat takakansi ensin ja jäävät tähän asetelmaan. Korteilla on kuitenkin pelin käynnistyessä jo tiedossa korttien kuvapuolikin. Tämä täytyy koodin puolella hoitaa niin, että kortin takakansi tulostuu ensin, kuten se tekee nytkin, mutta ennen kuin pelaaja pääsee valitsemaan kortteja, korttien tulisi kääntyä kuvapuoli näkyville. Tähän tulemme käyttämään visuaalista efektiä, joka näyttää kortin pyöräytyksen.

Kortit myös tulostuvat sekaisin kahteen sarakkeeseen. Alkuperäinen ajatus kuitenkin on, että virukset ja antivirukset tulostuisivat molemmat omiin sarakkeisiinsa. Logiikan muutos pitää hoitaa niin, että ensin tulostuvat kortit antivirus-hakemistosta, jonka jälkeen korttien index-numero tallennetaan muuttujaan, ja virus-hakemistosta tulostetaan vastaavalla index-numerolla olevat kortit. Näin on helpompi hallita eri sarakkeisiin tulostuvia elementtejä.

6.2 Käyttäjätili

Peliin on tulossa käyttäjäprofiilit. Profiilin lisäyksen voi joko toteuttaa suoraan hakemalla Unityn Asset Storesta valmiin paketin, jonka muokkaamme omaan peliimme sopivaksi tai sen voi alusta asti itse koodata.

Käyttäjäprofiiliin voi myöhemmin kehityksessä lisätä graafisia ja pelillisiä elementtejä. Pelaaja voi tehdä itselleen hahmon, jolla voi personoida itseään. Online-Käyttäjätilin kautta tulee mahdolliseksi myös kaveri-ominaisuus, jonka

avulla pelaajat voivat vertailla pisteitään helposti ja näyttää luomaansa hahmoa muille.

Pistejärjestelmänä on paikallisesti tallennettu top10-taulukko, josta näkee oman pistemääränsä ja puumerkin, jonka voi lisätä halutessaan. Top10-taulukossa voi valita näkymään kaikkien pelanneiden top10 tai vain omien pelattujen pelien top 10.

Pelin pisteytysjärjestelmä tulee pohjautumaan käytettyihin arvauksiin. Neljän parin pelissä paras pistetulos on 4 arvausta, kun taas huonoin on ääretön määrä arvauksia.

6.3 Valikot

Peliin tullaan lisäämään taustamusiikki, joka alkaa soimaan sovelluksen käynnistyttyä. Myös nappien painamisesta tulee kuulumaan klikkaus. Äänien hallintaa tehdään asetukset voimakkuuden säätöön, sekä äänet kokonaan pois-valinta.

Peliin tulee vaikeusasteet. Vaikeusasteet määrittävät korttien määrän pelilaudalla, näin ollen pareja tulee korkeammilla vaikeusasteilla enemmän. Tämän voi toteuttaa, joko tekemällä jokaisen vaikeustason omaan sceneen, tai muuttamalla nappien tulostuksen logiikkaa niin, että nappien määrä tulostuu pelaajan valinnan mukaan. Vaikeusasteen voi valita erikseen ennen jokaista uutta peliä.

Päävalikkoon on tulossa Kortit-nappi, josta pääsee tarkastelemaan kaikkia mukana olevia kortteja. Tällä tavoin kortteja pystyy opettelemaan pelaamatta peliä. Ensiksi korteista näkyy vain nimet, mutta kun niitä painaa, näkyville tulee kortin tarkempi kuvaus.

6.4 Pelin julkaisu

Lopullinen päämäärämme on saada peli ladattavaksi Google play-kauppaan. Ensimmäiseksi peliä on kuitenkin syytä testata loppukäyttäjällä, eli esikoululaisilla. Tätä varten peli pitää siirtää mobiililaitteelle ja käydä

esittelemässä peliä päiväkodeissa. Testikierroksen jälkeen voi tehdä peliin tarvittavat muutokset ja siirtää peli julkiseen tarjontaan.

Jotta pelin saa Google play-kauppaan, pitää julkaisijan luoda kehittäjätili, jonka jälkeen Unityssa pakattu peli lisätään Google play Console-palveluun. Tämän jälkeen palvelu vaatii erilaisia tietoja pelistä, kuten nimen, lyhyen kuvauksen ja kuvakaappauksia pelistä. Seuraavaksi pitää valita, että haluaako pelistä julkaista Alpha, Beta vai julkaisuvalmiin version. Tähän vaikuttaa se, että missä vaiheessa pelinkehitystä peli on, Alpha-vaihe on hyvin aikaisessa vaiheessa oleva peli. Tämän jälkeen tulee asetuksia, missä pitää valita muun muassa ikäraja-asetuksia, sisältääkö peli mainoksia ja onko peli lapsille suunnattu. Kaikkien vaiheiden jälkeen pelin voi laittaa tarkastukseen, jonka Google suorittaa. Tämän jälkeen peli tulee Google Play-kauppaan ladattavaksi. (Google 2021.)

Jatkokehitysvaiheessa olemme siirtyneet pois opinnäytetyö-vaiheesta ja voimme keskittyä pelin suunnitteluun ja päivittämiseen. Edellä luetellut ominaisuudet tulevat olemaan valmiita vuoden 2021 loppuun mennessä.

7 POHDINTA

Opinnäytetyömme tarkoituksena on vastata kysymykseen, kuinka toteuttaa mobiilipeli tietoturvasta varhaiskasvatuksen käyttöön Unityn avulla. Avustavana alakysymyksenä meillä on, että mitä on otettava huomioon, kun suunnitellaan opettavaa mobiilipeliä varhaiskasvatuksen käyttöön. Mielestämme projekti onnistui hyvin ja kykenimme vastaamaan meidän tutkimuskysymykseemme. Pelin Proof of Concept-versio on toivotulla tasolla ja työtä on helppo lähteä jatkamaan ja tuomaan lisää toiminnallisuuksia mukaan. Alakysymyksen kautta meille tuli paljon uutta tietoa varhaiskasvatuksesta ja pelaamisen yhdistämisestä, jota pystyimme hyödyntämään työssämme.

Unityn käyttäminen oli helpompaa kuin aluksi kuvittelimme. Käyttöliittymän ensimmäistä kertaa nähdessään voi helposti tulla riittämättömyyden tunne. Kun itse ohjelmaa pääsee käyttämään ja opettelemaan toimintoja, tuli aika nopeaa selväksi, että ohjelman käyttäminen on sujuvaa ja tehokasta.

Saimme tehtyä pelistä proof of conecpt-version josta on hyvä lähteä jatkokehittämään tuotetta. Taustatietoa tutkiessamme huomasimme, että digitalisaatio ja sen opetus suomalaisessa koulussa on vielä lasten kengissä, huolimatta siitä, että tällä hetkellä sovellettavassa opetussuunnitelmassa digiosaamista painotetaan jokaisessa oppiaineessa. Informaatioteknologiaa opettavan tunnin puuttuminen varhaiskasvatuksesta ja alakoulusta oli hämmentävä löytö, sitä kun tarvitaan nykyään kaikkialla. Myös opettajien puutteelliset tiedot ja taidot tieto- ja viestintäteknologiasta heikentävät mahdollisuuksia opettaa lapsille ja nuorille tarvittavia it-taitoja. Tällöin oppilaalle voi jäädä pinnallinen käsitys IT:stä ja myös tietoturvasta.

Osio, joka jäi meillä vajaaksi, oli pelin suunnittelu. Emme riittävästi paneutuneet pelisuunnittelun teoriaan. Jos olisimme ottaneet pelisuunnittelun teoriaa enemmän mukaan työhömme, olisimme kyenneet työskentelemään tehokkaammin ja monipuolisemmin. Myös työskentely olisi ollut suoraviivaisempaa ja olisimme voineet keskittyä pelin rakentamiseen paremmin.

Johtuen vaillinaisista visuaalisista taidoistamme, pelin ulkonäössä on vielä kehitettävää. Pelin asiasisältöön olemme tyytyväisiä. Uskomme, että

tekemästämme proof of concept-mallista saa tarvittavan ajatuksen, siitä kuinka pelimme toimii ja tästä on sujuvaa jatkaa pelin kehitystä. Olemme saanut koottua peliin rungon, joten pelistä on myös mahdollista lähteä työstämään erilaisia versioita eri oppiasteille, alakoulusta aina aikuiskoulutukseen.

LÄHTEET

Alasuutari, P. 1993. Laadullinen tutkimus. 3. Uudistettu painos. Tampere:Vastapaino.

BBC 2021. What is the difference between a 2D shape and a 3D object? Viitattu 31.3.2021 https://www.bbc.co.uk/bitesize/topics/zbtp34j/articles/zjjkpg8.

Brain balance achievement centers 2021. Normal attention span expectations by age. Viitattu 28.04.2021 https://www.brainbalancecenters.com/blog/normal-attention-span-expectations-by-age.

Epic Games 2021. Frequently asked questions. Viitattu 29.03.2021 https://www.unrealengine.com/en-US/faq.

Google 2021. Create and set up your app. Viitattu 14.04.2021 https://support.google.com/googleplay/android-developer/answer/9859152?hl=en.

Halpern, J. 2018 The What and why of game engine. Medium. Viitattu 29.03.2021 https://medium.com/@jaredehalpern/the-what-and-why-of-game-engines-f2b89a46d01f.

Jyväskylän yliopisto 2015. Laadullinen tutkimus. Viitattu 31.03.2021 https://koppa.jyu.fi/avoimet/hum/menetelmapolkuja/menetelmapolku/tutkimusstrategiat/laadullinen-tutkimus.

Järvilehto, L. 2014. Opi pelaamalla. Teoksessa H. Ruuska & M.Löytönen & A. Rutanen (toim.) Laatua! Oppimateriaalit muuttuvassa tietoympäristössä. Porvoo: Bookwell, 219-227.

Kyberturvallisuuskeskus 2020. Näin pidät huolta tietoturvasta kotona ja työpaikalla. Viitattu 19.09.2020 https://www.kyberturvallisuuskeskus.fi/fi/ajankohtaista/ohjeet-ja-oppaat/nain-pidat-huolta-tietoturvasta-kotona-ja-tyopaikalla.

Mannerheimin lastensuojeluliitto 2020a. 4-5-vuotias ja median käyttö. Viitattu 30.08.2020 https://www.mll.fi/vanhemmille/lapsen-kasvu-ja-kehitys/4-5-v/4-5-vuotias-ja-median-kaytto/.

Mannerheimin lastensuojeluliitto 2020b. 5-6-vuotias ja median käyttö. Viitattu 30.08.2020 https://www.mll.fi/vanhemmille/lapsen-kasvu-ja-kehitys/5-6-v/5-6-vuotias-ja-median-kaytto/.

Lukka, K. 2001. Kari Lukka: Konstruktiivinen tutkimusote. Metodix. Viitattu 27.08.2020 https://metodix.fi/2014/05/19/lukka-konstruktiivinen-tutkimusote/.

Manninen, T. 2007. Pelisuunnittelun käsikirja – Ideasta eteenpäin. Pello: Rajalla.

Microsoft 2021a. A tour of the C# language. Viitattu 29.03.2021 https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/.

Microsoft 2021b. Get started with Visual Studio and Unity. Viitattu 28.04.2021 https://docs.microsoft.com/en-us/visualstudio/gamedev/unity/get-started/getting-started-with-visual-studio-tools-for-unity?pivots=windows,

Microsoft 2019b. Default values of C# typers (C# reference). Viitattu 15.04.2021 https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/default-values.

Microsoft 2019a. Welcome to the Visual Studio IDE. Viitattu 29.03.2021 https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?utm_source=vscom-download&view=vs-2019.

O'Dea, S. 2020. Mobile operating systems' market share worldwide from January 2012 to July 2020. Viitattu 03.03.2021 https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/.

Open Handset Alliance 2007. FAQ. Viitattu 27.08.2020 https://www.openhandsetalliance.com/oha_faq.html.

Opetushallitus 2018. Mitä opetussuunnitelman perusteissa sanotaan itseohjautuvuudesta, digitalisaatiosta ja ilmiöoppimisesta?. Viitattu 29.03.2021 https://www.oph.fi/fi/uutiset/2018/mita-opetussuunnitelman-perusteissa-sanotaan-itseohjautuvuudesta-digitalisaatiosta-ja.

Opintopolku 2021. Perusopetuksen opetussuunnitelman perusteet 2014. Viitattu 02.02.2021 https://eperusteet.opintopolku.fi/#/fi/perusopetus/419550/tekstikappale/428611.

Sankila, T. 2015. Oppimista muuttava teknologia. Teoksessa H. Ruuska & M.Löytönen & A. Rutanen (toim.) Laatua! Oppimateriaalit muuttuvassa tietoympäristössä. Porvoo: Bookwell, 219-227.

Sinicki, A. 2021. What is Unity? Everything you need to know. Android Authority 20.3.2021. Viitattu 28.4.2021 https://www.androidauthority.com/what-is-unity-1131558/.

Statista 2021. Number of worldwide Hearthstone players as of November 2018. Viitattu 29.03.2021 https://www.statista.com/statistics/323239/number-gamers-hearthstone-heroes-warcraft-worldwide/

Suoninen, A. 2013. Lasten mediabarometri 2013. Helsinki:Unigrafia. Viitattu 25.08.2020 http://www.nuorisotutkimusseura.fi/images/julkaisuja/lastenmediabarometri2013.pdf.

Tikkanen. T 2015. Polte päälle. Opettaja 1/2015, 16.

Unity 2021a. Store. Viitattu 29.03.2021 https://store.unity.com/.

Unity 2021b. baymax Dreams. 29.03.2021 https://unity.com/madewith/baymax-dreams.

Unity 2021c. Canvas. Viitattu 14.04.2021 https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/UICanvas.html.

Unity 2021d. Coco VR. Viitattu. 29.03.2021 https://unity.com/madewith/coco-vr.

Unity 2021e. Grid Layout Group. Viitattu 14.04.2021 https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/script-GridLayoutGroup.html.

Unity 2021f. MonoBehaviour.Start(). Viitattu 15.04.2021 https://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html.

Unity 2021g. Monument Valley 2. Viitattu 31.3.2021 https://unity.com/madewith/monument-valley-2.

Unity 2021h. Scripting in unity for experienced programmers. Viitattu 29.03.2021 https://unity.com/how-to/programming-unity.

Unity 2021i. SerializedField. Viitattu 14.04.2021 https://docs.unity3d.com/ScriptReference/SerializeField.html.

Unity 2021j. Top free assets. Viitattu 31.3.2021 https://assetstore.unity.com/top-assets/top-free.

Unity 2021k. Unity Asset Store. Viitattu 29.03.2021 https://assetstore.unity.com/.

Unity 2021l. Unity user manual. Viitattu 29.03.2021 https://docs.unity3d.com/Manual/index.html.

Unity 2021m. Unity manual. Viitattu 28.04.2021 https://docs.unity3d.com/Manual/UsingTheEditor.html.

Valtionvarainministeriö 2020. Digikartoitus. Viitattu 17.9.2020 https://valtioneuvosto.fi/-/10623/suomalaisten-digitaidot-ovat-suurimmaksi-osaksi-hyvalla-tasolla-digitaitokartoitus-nosti-esiin-myos-huolenaiheita.

Verdict 2020. History of virtal reality: timeline. Viitattu 31.3.2021 https://www.verdict.co.uk/history-virtual-reality-timeline/.

Vestman, T. 2020. Kriittinen analyysi neutralisoimisteorian soveltamisesta tietojärjestelmätieteessä. Jyväskylän yliopisto. Viitattu 27.08.2020 https://jyx.jyu.fi/bitstream/handle/123456789/69048/978-951-39-8174-7_vaitos05062020.pdf?sequence=1&isAllowed=y.

Yoyogames 2021. Game maker features. Viitattu 29.03.2021 https://www.yoyogames.com/en/gamemaker/features.