Haaga-Helia
ammattikorkeakoulu Oy

Päiväkirja opinnäytetyö sisältösuunnittelijan ja pelinkehittäjän työssä

Jari Pentti

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2019

Haaga-Helia ammattikorkeakoulu Oy

Tiivistelmä

28.11.2019

Tekijä(t)
Jari Pentti

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko

Päiväkirja opinnäytetyö sisältösuunnittelijan ja pelinkehittäjän työssä

Sivu- ja liitesivumäärä
83 + 0

Opinnäytetyön otsikko englanniksi
Diary thesis of work as a content designer and as a game designer

Päiväkirjamuotoisessa opinnäytetyössäni raportoin 8 viikkoa työtäni Plehat Oy:ssä ja Yummy Digitalilla. Toimin pelin kehittäjänä Yummy Digitalilla mobiilipelien parissa ja Plehat Oy:ssä sisältösuunnittelijana maisema-arkkitehtuuri ohjelmistojen parissa.

Raportoin jokaisen viikon työtehtävät ja havainnot ja analysoin viikon aikana tehdyt työtehtävät kunkin viikon lopuksi. Opinnäytetyön aluksi kuvaan tilanteen ennen opinnäytetyö jaksoa ja lopussa käyn läpi jakson aikana huomioidut asiat. Työympäristö mobiilipeli puolella on raskaasti kilpailutettu ja vaihtelevia työtehtäviä sisältävä ala. Maisema-arkkitehtuuri yrityksen työympäristössä toimin yhdessä isomman tiimin kanssa ja koordinoin omaa osaani projektista ja toimin vaihtelevissa työtehtävissä. Työtehtävät toimitetaan pääosin Unity ja Unreal pelimoottoreiden sisällä.

Osaamiseni kehittyy todella merkittävästi opinnäytetyön aikana serveri API:en kanssa toimimisessa ja tietokantojen käsittelyssä. Tiedonhakuni ja dokumentointi taitoni saavat ammattimaisempaa otetta luettuani satoja sivuja erilaisia dokumentaatioita. Lisäksi opin käyttämään hyödykseni entistä paremmin erilaisia SDK työkaluja Unity:n sisällä.

Asiasanat
Unity, Unreal, pelimoottori, Virtuaalitodellisuus, Koodaaminen

Sisällys

1 Johdanto

Opinnäytetyö toteutettiin 1.4.2019 – 24.5.2019 välillä. Opinnäytetyö toteutettiin raportoinnilla päivittäisistä työtehtävistä.

Raportoinnin aikana suoritetuissa työtehtävissä tarvitaan laajaa ymmärrystä pelimoottoreista, koodaamisesta, videoeditoinnista, 3D-grafiikan tekemisestä sekä käyttöliittymä suunnittelusta. Pääosin toimintaympäristö rakentuu pelimoottorin sisällä toteutettaviin työtehtäviin, joten pelimoottorin toiminnan ymmärtäminen on keskeisessä osassa työtä. Työtehtävissä käytettiin Unity- ja Unreal-pelimoottoreita ja näiden lisäosia.

Pelimoottorit päivittyvät usein ja niiden olemassa olevat toiminnot muuttuvat jatkuvasti. Tyypillistä on myös vanhojen ominaisuuksien korvaaminen kokonaan eri tavoin toimivilla uusilla järjestelmillä. Työtehtävien ohessa on tärkeää ottaa selvää tulevista sekä jo tapahtuneista muutoksista moottorissa sekä tutustua siihen, kuinka päivitykset vaikuttavat pelimoottorien lisäosien toimintaan. Käytän työssäni ainoastaan internetissä olevaa kirjallisuutta, sillä pelimoottoreiden päivitys tahdin ollessa jopa päivittäinen ei ajantasaista perinteistä kirjallisuutta ole olemassa. Blogit toimivat pelimoottorien osalta tärkeimpänä tietojen päivitys kanavana.

Unity:n nettisivut ja Blogi sisältävät ajantasaiset uutiset ja muutokset Unity pelimoottorin ominaisuuksista. Blogin sisältö vaihtelee tarpeellisista tiedoista parhaisiin käyttötapoihin Unity pelimoottorissa (Unity 2019.).

Unreal Enginen uusista ominaisuuksista ja muutoksista saa parhaiten tietoa heidän omilta sivuiltaan ja Blogista. Blogi sisältää koulutuksia, uutisia muutoksista sekä parhaiden käyttötapojen yhteenvetoja. (Unreal Engine 2019.)

Lisäksi tärkeää on lukea jokaisen lisäosan ja ohjelmiston tuottajan oma dokumentaatio siitä, kuinka ohjelmistot toimivat. Tästä hyvänä esimerkkinä voidaan pitää Substance-ohjelmistojen dokumentaatiota, jota käytän jatkuvasti (Substance 3D (1).).

Joudun käyttämään opinnäytetyössäni osittain englanninkielisiä sanoja, sillä osasta työmenetelmistä ei ole olemassa suomennoksia tai suomennosten käyttö ei anna oikeaa kuvaa tehdystä työstä pelimoottorien kontekstissa.

Työssä menestymiseen tarvitaan 3D-mallintamisen osalta ymmärrys siitä, kuinka pelimoottorit käsittelevät eri tiedostomuodoissa olevia 3D-malleja ja kuinka mallien polygoni

laskenta (polycount) tulee toteuttaa sekä kuinka optimaalinen topologia luodaan objekteille. Pelimoottoreiden osalta tarvitaan vahvaa projektin hierarkian suunnittelemista sekä koodin järjestämistä ymmärrettäviksi toimiviksi osiksi. Ohjelmointikielenä opinnäytetyön projekteissa toimi C# Unity:ssä sekä Blueprint niminen visuaalinen skriptaus kieli Unreal-pelimoottorissa. Ohjelmointikielten käyttäminen on isossa osassa päivittäistä työtäni. Hyvät tiedonhakutavat ja oikeaoppinen olio-ohjelmointi on tarpeen osata työn laadun varmistamiseksi.

Opinnäytetyö toteutettiin kahdessa yrityksessä, joissa työskentelin raportointi aikana. Plehat Oy on vuonna 2017 perustettu digitaaliseen maisema-arkkitehtuuriin erikoistunut yritys. Yritys toimittaa konsultaatioita yrityksille ja yksityishenkilöille maisema-arkkitehtuurin alalla sekä valmistaa ohjelmistotuotteita arkkitehtien ja maisema-arkkitehtien käyttöön. Yrityksen pää ohjelmistotuotteet ovat virtuaalitodellisuudessa toimivia visualisointi ohjelmistoja sekä töiden helpottamiseen tähtääviä ohjelmistoja. Oma työympäristöni on pääosin osana teknologia tiimiä sisältösuunnittelijana. Tuotan sisältöä eri ohjelmistotuotteisiin virtuaalitodellisuudessa koodin ja grafiikan osalta sekä teen tarvittaessa video editointeja sekä muuta koodi työtä C# kielellä.

Yummy Digital Oy on vuonna 2018 perustettu mobiilipelialan yritys, jossa tein fullstack-kehittäjän töitä muutamina päivinä raportointi aikana. Yrityksen tuotteet ovat mobiilialustoilla toimivia hyper kasuaaleja free-to-play pelejä. Toimin graafikkona, UI- suunnittelijana, toimitusjohtajana, koodarina sekä pelisuunnittelijana.

1.1 Keskeiset ammattikäsitteet

Pelimoottori = Perusohjelmisto tietokone tai videopelille. Nykypäivänä pelimoottoreiden käyttö on yleistynyt myös visualisointi ja koodaus käytössä pelien lisäksi.

Unity 3D: Unity:n valmistama C++ kieleen pohjautuva modulaariseen pelintekoon perustuva pelimoottori.

Unreal Engine 4 = Epic Games:in valmistama C++ kieleen pohjautuva pelimoottori.

Polygoni = kolmen tai useamman pisteen muodostama alue tietokonegrafiikassa, joka muodostaa yhtenäisen alueen.

Polycount = Polygonien määrä scenessä tai 3D- mallissa.

Scene = Pelimoottorin yksi karttatiedosto, joka sisältää käyttäjän toimintaympäristön.

Lisäosa = Pelimoottoreissa tai ohjelmistoissa on usein kolmannen osapuolen valmistamia lisäosia, jotka tuovat uusia toiminnallisuuksia.

Topologia = 3D mallin polygoniverkon rakenne.

Visuaalinen skriptaus = Visuaalinen ohjelmointikieli.

Olio ohjelmointi = Object Oriented programming tarkoittaa ohjelmointi tapaa, joka muodostuu kokoelmasta yhteistyössä toimivia olioita perinteinen proseduraalinen tietokoneohjelma sijaan.

Kontrolleri = Virtuaalitodellisuus- tai konsoliohjain.

Blender = Ilmainen 3D- mallinnus ohjelma.

Frame = Pelimoottori piirtää näytölle kuvan useita kertoja sekunnissa riippuen koneen suorituskyvystä. Yksi Frame on siis koneen piirtämä yksi kuva sekunnin aikana.

Layer = Taso, jolle voidaan tallentaa asioita pelimoottorissa. Tapa eristää eri tasot toisistaan.

Prefab = ennalta luotu peliobjekti moottorin sisällä, joka sisältää useita eri osia kuten kuvia, koodia, ääntä.

Mesh = Polygoneista koostuva 3D objekti

Git = versiohallintajärjestelmä.

Terrain Painting = Terrain mesh objektin käsin maalaaminen ja yksityiskohtien lisääminen suoraan objektin tekstuurikarttaan.

Blend = Blend:llä tarkoitetaan kahden tekstuurin toisiinsa sulauttamista jollain tapaa

PBR = Physically based rendering on tekniikka, joka ottaa tosi maailman fysiikan lait huomioon tietokoneen renderöintimoottorissa.

VR = Virtuaalitodellisuus

UE = Unreal Engine 4

2 Lähtötilanteen kuvaus

2.1 Oman nykyisen työn analyysi

Työtehtäviini kuuluu ohjelmisto- ja peli suunnittelu. Luon ohjelmiston toiminnasta yleisen toimintakuvauksen siitä, kuinka käyttäjä tulee käyttämään ohjelmistoa, mitä hyötyjä hän saa ohjelmistosta, miten ohjelmiston tulee rakentua käyttäjän tarpeita palvelemaan sekä mikä on paras tapa toteuttaa projekti. Tämän jälkeen jokaista osaa lähdetään tarkentamaan ja avaamaan enemmän ja osille luodaan omat aikataulut valmistumisen osalta ja ohjelmistoarkkitehtuurista etsitään oma paikka toteutukseen. Osaamiseni liittyy ohjelmisto arkkitehtuurin ymmärtämiseen ja käyttäjän tarpeiden ymmärtämiseen. Käyttöliittymän sujuvuus on myös iso osa suunnitteluprosessia.

Oma osaamiseni on rakentunut vahvalle harrastuspohjalle pelien ja ohjelmistojen valmistamisessa. Työn ohessa hankittu osaaminen perustuu potentiaalisten asiakkaiden tarpeiden tunnistamiseen ja siitä johdettuun ohjelmiston rakenteeseen. Lisäksi työn ohessa tutkin kilpailijoiden ohjelmistoja ja teen niistä lyhyitä raportteja minkälaisia ratkaisuita muut ohjelmistokehittäjät tekevät. Tehtävä vaatii siis kokonaiskuvan teknistä ymmärtämistä sekä käyttäjän tarpeiden ymmärtämistä.

3D mallinnuksen toteutan pääosin Blender- ohjelmistolla, jonka monipuoliset ominaisuudet mahdollistavat helpon toiminnan niin Unreal kuin Unity ympäristöissä. Luon tarpeen vaatiessa uusia 3D malleja sekä siistin olemassa olevien mallien topologioita paremmin optimoiduiksi pelimoottoreiden käyttöön. Osaamiseni liittyä suoraan pelimoottoreiden käyttöön ja niiden teknisten rajoitteiden ymmärtämiseen.

Osaamiseni 3D mallintamisesta on hankittu vahvalla harrastuspohjalla sekä tutustumalla kattaviin online tutoriaaleihin. Lisäksi olen käynyt Haaga-Heliassa Blender mallintamisen kurssit. Tehtävä vaatii ymmärrystä siis pelimoottoreiden toiminnasta, parhaista käytännöistä mallintamisesta, hyvän topologian ymmärtämistä sekä taiteellista näkemystä.

Materiaalien ja tekstuurien valmistaminen projekteissa toteutetaan Substance- ja Photoshop ohjelmistoilla. Valmistan 2D tekstuurikarttoja Unity ja Unreal ohjelmistojen käyttöön ja luon tekstuureista tarvittavat tekstuurikartat, joiden avulla pelimoottori piirtää materiaalin pinnan oikean näköisenä renderöinti moottorissaan. Minun tulee ymmärtää pelimoottoreiden tekninen toteutus ja moottoreiden tarpeet materiaalien luonnin osalta ja osata luoda kaikki tarvittavat tekstuurit. Lisäksi tekstureista tulee pystyä yhdistämään toimivat materiaalit pelimoottorin sisällä. Myös ymmärrys fysikaaliseen mallintamiseen pohjautuvasta

renderöinnistä on tarpeen ja ymmärrys fysiikasta ja matematiikasta sovellettuna tosimaailmaan on pakollista.

Osaamiseni pohjautuu lukuisiin tutoriaaleihin, joita ohjelmistot tarjoavat sekä ohjelmistojen dokumentaatioiden ja parhaiden käytäntöjen tutkimiseen ohjelmiston kehittäjän tietopankeista. Usein tarvittavat taidot ovatkin lähempänä taiteellista näkemystä, kuin täysin oikeaoppista materiaalin esittämistä. Minun tulee ymmärtää siis mikä näyttää pelimoottorin kontekstissa käyttäjän mielestä oikealta, eikä mikä on fysikaalisesti laskettuna oikea tapa esittää materiaali moottorissa.

Koodaus C# kielellä sekä Blueprint visuaalisella ohjelmointikielellä on iso osa päivittäistä työtäni. Luon erilaisia olioita koodissa, joista kukin hallitsee tiettyä osa-aluetta ohjelmiston toiminnasta. Näin saan luotua ohjelmistosta kevyemmän ja toimivamman kokonaisuuden. Olio pohjainen ohjelmointi myös luo mahdollisuuden muuttaa osia ohjelmasta ilman, että koko ohjelman arkkitehtuuria tarvitsee muuttaa. Ohjelmointi osaamisen tulee olla monipuolista, koska työtehtävät vaihtelevat UI järjestelmän koodaamisesta serveri integraatioiden tekemiseen. Tärkein työvälineeni on ollut koodaamisessa erilaisten tutoriaalien lukeminen ja katsominen sekä tiedonhaku internet foorumeilta.

Koodaaminen pelimoottoreissa eroaa hieman muista ympäristöistä, koska pelimoottorit eivät mahdollista usein täysin samoja kirjastoja kuten muut alustat. Tästä esimerkkinä JSON järjestelmän tekeminen Unity-pelimoottorissa. Unityn JSON kirjaston on typistetty versio JSON.NET kirjastosta eikä mahdollista kaikkia samoja toiminnallisuuksia. Tarvitsen työstä selviytyäkseni vankan ymmärryksen pelimoottorin rajoitteista ja toiminnasta sekä vahvan osaamisen. Tiedon haku ja erilaisten olioiden valmistaminen on ensiarvoisen taito työssäni.

Ohjelmistosuunnittelijana olen taitava suoriutuja. Olen saanut ulkopuolisilta tahoilta jatkuvasti palautetta työstäni ja pystyn toimittamaan kokonaisuuksia, jotka vastaavat käyttäjän tarpeita. Päästäkseni asiantuntijaksi tarvitsen kuitenkin vielä harjoitusta, jotta pystyn toimittamaan parempia ohjelmistoja ja ymmärtämään paremmin ohjelmistosuunnittelun yksityiskohtia erityylisissä ohjelmistoissa.

3D-mallintajana olen taitava suoriutuja, joka pystyy toimittamaan annetun työtehtävän itsenäisesti editorista pelimoottoriin järkevällä topologialla toteutettuna. En kuitenkaan koe olevani vielä 3D-artisti, sillä en pysty viemään riittävällä nopeudella monimutkaisia 3D-malleja kokonaan tuotannon läpi ja esimerkiksi animointi puoli ei ole kuulunut työtehtäviini aiemmin.

Pelimoottoreiden käyttöön tarkoitettu materiaalien ja tekstuurien valmistamisessa olen aloitteleva toimija. Pystyn itsenäisesti etsimään tarvittavat työtavat ja suoriutumaan työssäni itsenäisesti erilaisten tutoriaalien ja itse löytämieni ohjeiden mukaan ilman ohjausta. En kuitenkaan pärjää työtehtävässä vielä ilman ulkoisia ohjeita, vaikka ohjeet löydänkin itse.

Koodaamisessa olen taitava suoriutuja, joka pystyy toimittamaan tarvittavat osat ohjelmistosta itsenäisesti tietoa hakemalla. Päästäkseni vielä eteenpäin osaamisessani tulee minun opetella muita ohjelmistokieliä ja laajentaa ymmärrystäni koodin kirjoittamisessa entisestään kurssien ja harjoitusten avulla.

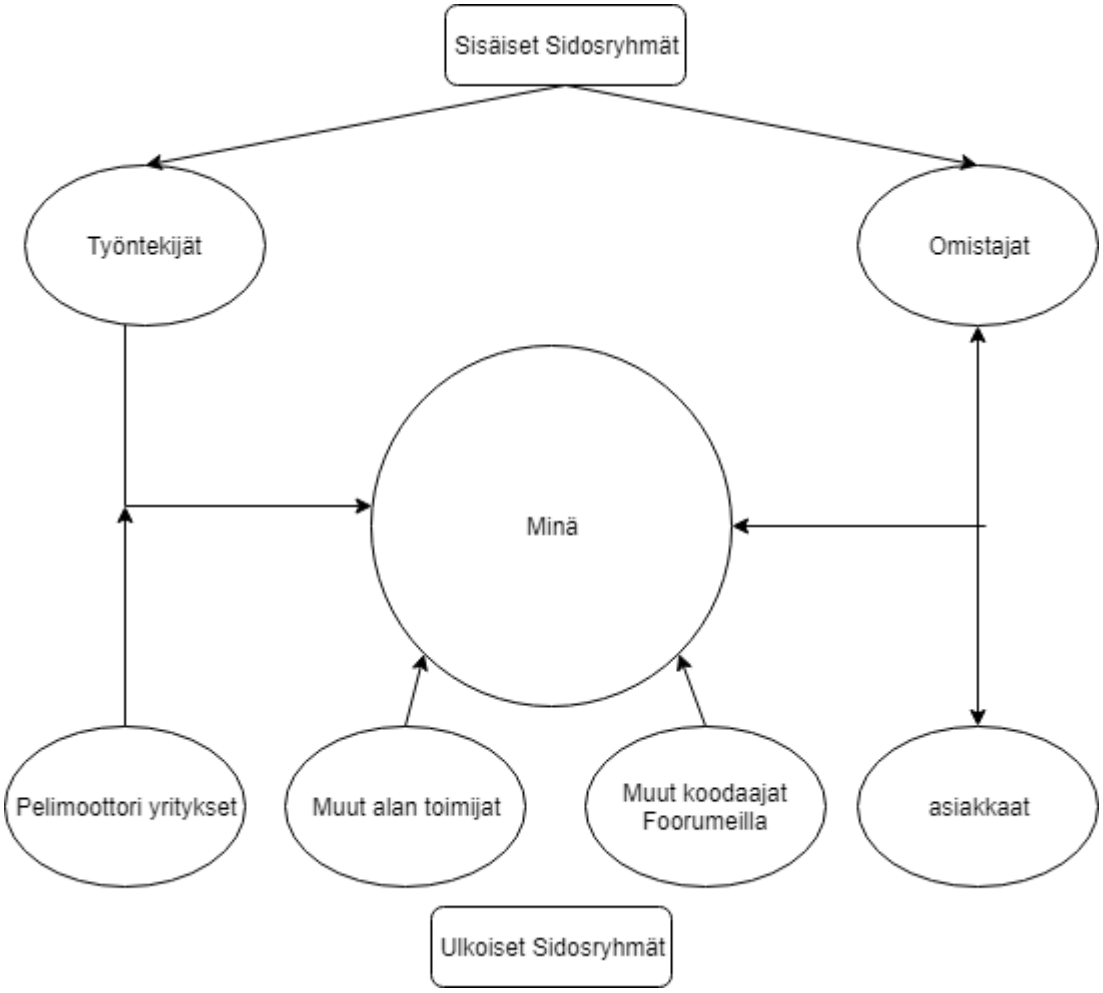
Olen kehittynyt valtavasti eritoten koodin osalta viimeisen puolen vuoden osalta, kun olen saanut vastuuta todella paljon työtehtävissäni. Olen päässyt luomaan serveri integraatioita sekä erilaisia ohjelmiston osia vapaasti ja oppinut mm. JSON järjestelmistä sekä olio pohjaisesta ohjelmoinnista. Tällä hetkellä näen tarvitsevani kuitenkin vielä runsaasti lisää kokemusta, jotta voin olla tyytyväinen osaamiseeni. Iso osa koodaamisen ajasta kuluu edelleen tiedon hakuun siitä, mikä on paras tapa toteuttaa jokin tietty osa-alue koodista. Lisäksi en kirjoita kuin yhtä kieltä sujuvasti minkä näen ehdottomana haasteena tulevaisuudessa ja tähän ratkaisuksi on opetella muita tärkeitä ohjelmistokieliä.

Koen olevani tällä hetkellä vielä urani alkuvaiheessa ja vaikka pystynkin toimittamaan ja rakentamaan pienimuotoisia ohjelmistoja pelimoottoreissa on isompien projektien kokonaisuuksien hahmottamisessa ja hierarkian ymmärtämisessä parantamisen varaa runsaasti. Tämä näkyy mm. siten että opin joka päivä aina jotain uutta ja koen selvästi kehittyväni edelleen ohjelmistojen valmistamisesta.

Jatkossa panostan toiminnassani systemaattisempaan opiskeluun ohjelmistojen suunnittelun osalta. Lisäksi muiden ohjelmistokielien oppiminen vie minut uudelle tasolle osaamisessa tulevaisuudessa.

2.2 Sidosryhmät työpaikalla

Työpaikkani Sisäiset sidosryhmät (kuvio 1) koostuvat muista työntekijöistä ja omistajista. Ulkoiset sidosryhmät koostuvat muista koodaajista internet foorumeilla, pelimoottori yrityksistä, asiakkaista, muista alan toimijoista.



Kuvio 1. Sidosryhmät työpaikalla

Keskeisimmät sidosryhmäni ovat sisäiset sidosryhmät, jotka luovat työtehtävät ja vision työstä sekä yrityksen toimintatavat ja yrityskulttuurin. Keskeisellä paikalla avun saannin kannalta työtehtävissäni ovat sisäisten sidosryhmien lisäksi myös muut koodaajat erinäisillä foorumeilla, joilta voi kysyä apua tarvittaessa ongelmien kanssa.

Asiakkaiden ja omistajien käymät keskustelut vaikuttavat työhöni merkittävästi sillä omistajat määrittelevät työtehtävät asiakkaiden tarpeiden pohjalta. Asiakkaiden mielipiteet ovat siis tärkeässä roolissa ohjelmistotuotteen suunnan kannalta. Tuotteen tulee vastata asiakkaan tarpeeseen yrityksen johdon asettamien tavoitteiden mukaan. Muut alan toimijat ovat hyvä referenssi kohde oman työn arviointiin, vaikka heidän kanssaan ei suoranaista yh-

teistyötä tehdä. Pelimoottoriyritykset vaikuttavat työpaikan toimintaan merkittävästi päivitysten ja uusien ominaisuuksien kautta.

2.3 Vuorovaikutustaidot työpaikalla

Vuorovaikutustilanteet työpaikalla ovat usein hyvin tuttavallisia. Viikoittaiset palaverit, joissa käydään läpi mitä kukakin on tehnyt, mitä ollaan seuraavaksi tekemässä ja mahdolliset avun tarpeen luovat pohjan toimivalle yhteisölle. Haasteita aiheuttaa vuorovaikutuksessa avokonttori, jonka vuoksi lähes kaikki työntekijät käyttävät eristäviä kuulokkeita työrauhan saamiseksi. Työ on tiimityötä, vaikka jokaisella toimijalla on selkeästi oma vastuualueensa. Tästä syystä hyvä kommunikaatio on erittäin tärkeää hyvän lopputuloksen saamiseksi.

Viikkopalaverit ovat usein todella tuttavallisia ja toistaiseksi rakenne on ollut hieman vaihteleva. Näen että järjestelmällisempi palaveri toiminta agendalistojen ja etukäteisen suunnittelun kautta loisi paremman pohjan ja lyhentäisi turhaa ajatuksen virtaa pois palavereista ja näin vapauttaisi työaikaa.

3 Päiväkirjaraportointi

3.1 Seurantaviikko 01

01.04.2019

Maanantain aamupalaverissa käytiin läpi viikon työtehtävät viikolle. Alkuperäisen työlistan mukaan oli tarkoitus jatkaa käyttöliittymän koodin ja suunnittelun kanssa yrityksen kasvi kirjasto tuotteelle. Yllättäen tuli kuitenkin ilmi kiireellisempi työtehtävä virtuaalitodellisuus mallin ja Unreal:in parissa. Toimistolle on tulossa perjantaina käymään vierailijoita, joille tulisi pystyä näyttämään uutta versiota yrityksen virtuaalitodellisuus tuotteesta. Virtuaalitodellisuus tuote on yrityksen toisen tiimin kehittämä tuote ja tehty eri pelimoottorilla kuin mitä yleensä käytän, joten orientoituminen viikoksi eri projektiin sotkee suunniteltuja aikatauluja.

Vieraille esitettävässä tuotteessa on valmiina 3D ympäristö mallinnettuna virtuaalitodellisuudessa kesäaikaan sekä talvi version alku ja päivien ja kuukausien kontrollointi koodi. Tuotteesta puuttuu vielä vuodenaikojen välillä siirtymisen suorittava koodi. Tuotteeseen pitää tämän viikon aikana lisätä toimiva talviversio pelimaailman ympäristöstä ja laittaa nämä toimimaan kontrollereiden kautta.

Alun perin oli ollut ajatuksena luoda jokainen vuodenaika omana kenttänään pelimoottorin sisällä ja vaihtaa kenttien välillä tarpeen vaatiessa visualisointeja. Ensimmäisenä tehtävänä tutustuin, miten karttojen vaihto tapahtuu simulaation aikana ja kuinka paljon viivettä tämä aiheuttaa kokemukseen. Virtuaalitodellisuus kokemuksissa pienikin viive voi aiheuttaa VR- lasien käyttäjälle pahoinvointia, joten usein sulava toimivuus on visuaalista loistoa tärkeämpää. Kartan vaihto osoittautui melko yksinkertaiseksi tehdä Unreal:in omalla Blueprinteillä. Käytin Open Level komentoa, jonka asetin tapahtumaan pelimoottorin kuukauden vaihtuessa.

Jotta pystyin tekemään vuodenajan vaihdon loin uuden Enum-muuttujan nimeltä TimeOfYear johon laitoin 4 parametriä, Winter, Spring, Summer, Autumn. Käytän yleisesti aina isoja alkukirjaimia muuttujissa Blueprinteissä, enkä käytä välejä muuttujien tai funktioiden nimissä, sillä välit saattavat aiheuttaa ongelmia pelimoottoreissa. Toin Enum-muuttujan Player Blueprinttiin ja hain toisesta Blueprintistä sen hetkisen kuukauden, jonka pelaaja on tällä hetkellä laittanut päälle. Jaoin vuoden osiin siten, että kun kuukaudet joulukuusta helmikuuhun ovat päällä Enum Winter on päällä, maaliskuusta toukokuuhun päällä on Spring, kesäkuusta – elokuuhun päällä on Summer ja syyskuusta marraskuuhun

päällä on Autumn. Esiteltävässä versiossa on olemassa vain Winter ja Summer kentät, joten tilapäisesti laitoin myös Autumn ja Spring vuodenajat toimimaan samoin kuin kesäkuukaudet.

Koska järjestelmä on tarkoitettu alun perin toimimaan virtuaalitodellisuudessa käyttäjän kontrollit ovat laitettu toimimaan vain VR ohjaimista. Järjestelmän tekemisen yhteydessä on tarpeen jatkuvasti tarkistaa, toimiiko tehty koodi, ja on todella hankala testata asioita jatkuvasti pukemalla VR laseja päähän ja pois. Tästä syystä tein yksinkertaiset testityökalut järjestelmään. Laitoin numerot 1 ja 2 näppäimistöstä vaihtamaan kuukausia eteen- ja taaksepäin, jotta minun ei tarvitse aina hakea myöskään VR ohjaimia testatakseni ominaisuuksia. Lisäksi asemoin kameran scenessä sellaiseen kohtaan osoittamaan, josta saan kokonaiskuvan koodin toimivuudesta paremmin.

Rakensin järjestelmän siten, että se katsoo jokaisen Framen aikana monesko kuukausi on menossa pelin sisäisen kellon mukaan. Kuukauden X ollessa päällä asettaa koodi myös Enum- muuttujan samaan tilaan. Tällä parametrillä järjestelmä lataa vastaavan kartan päälle, eli tässä tapauksessa joko Winter tai Summer. Ongelmana on, että kaikki koodi pitää olla jokaisen kentän sisällä aina uudestaan ja tämä aiheutti lopulta ongelmia.

02.04.2019

Poissa työpaikalta.

03.04.2019

Päivän tavoite on jatkaa työtä VR ympäristön parissa. Pelimoottori lataa kuukauden vaihtuessa aina uuden kartan, jossa on karttaan ennalta tallennetut asetukset, josta aiheutuu isoja ongelmia. Jokaisessa kartassa käyttäjälle on pelimoottorin asettama katselukulma ja sijainti. Käyttäjä voi tehdä erilaisia komentoja kartalla, jotka vaikuttavat ympäristöön. Tästä syystä kentän vaihtuessa kaikki käyttäjän tekemät muutokset nollautuvat jokaisen kentän omiin perus arvoihin, joten käyttäjän positio pelimoottorin sisällä sekä kellon aika vaihtuvat aina kentän vaihdon yhteydessä takaisin pelimoottorin asettamiin arvoihin. Kaikki muuttujat tulisi siis ensin tallentaa joka Framen aikana, jotta ne voidaan asettaa kentän vaihtuessa käyttäjälle samoiksi kuin aiemmassa kentässä. Lisäksi kentän vaihto aiheutti noin sekunnin pituisen lataus tauon, joka tuntuu pahalta virtuaalitodellisuudessa. Tällaisen järjestelmän toteuttaminen on hieman työlästä sillä en ole kirjoittanut itse kuukausien ja tuntien asettamis koodia alun perin eikä kiireellisen aikataulun vuoksi ole mahdollista perehtyä koodiin täydellisesti. Päätin tehdä asian helpommalla tavalla kuin alun perin oli kaavailtu.

Kentän vaihto olisi ollut mahdollista varmasti tehdä lataamalla muistiin muut kentät valmiiksi sekä tallentamalla oikeat muuttujat, mutta tulin siihen päätökseen, että ajan säästämiseksi luon yhden kentän, joka sisältää kaikki vuodenajat ja Enum:in pohjalta piilotan ei käytössä olevat objektit kuten lumihanget tai puiden lehdet tarpeen mukaan näkyvistä.

Tämä oli helpoin tehdä pelaaja Blueprintissä komennolla ”Set Actor Hidden Ingame”. Jotta en joudu jokaista tuhansista objekteista piilottamaan yksitellen lisäsin kaikkiin objekteihin, joihin haluan vaikuttaa tarpeelliset Tagit. Osa virtuaalimaailman objekteista ovat staattisia ja osa muuttuvia vuodenajan mukaan. Staattiset objektit eivät saaneet Tageja ollenkaan ja muuttuvat objektit saivat joko Winter tai Summer Tagin. Spring ja Autumn tageja ei tarvittu tälle kertaa mutta ne lisätään tulevaisuudessa.

Objekteilla voi olla useita Tageja tai vain yksi tai ei yhtään. Objektit, jolla on Tagit, voidaan lukea muistiin massana käyttämällä Blueprintissä komentoa ”Get Actor With Tag”, komennolla voidaan valita useita objekteja samaan aikaan ja tehdä kaikille haetuille objekteille sama funktio. Tagien käyttö osoittautui erittäin viisaaksi ratkaisuksi, sillä kun olin Tagannyt erilaisia objekteja vuodenajan mukaan, pystyin käsittelemään halutessani kaikkia talvi kentän objekteja tai vain osaa niistä tarpeen mukaan.

Alun perin laitoin talvi kentän objektit piiloon, koska tuotteessa aloitetaan kesäajasta. Asetin talvi objektit näkymättömäksi Start funktion jälkeen, jonka jälkeen peli etenee tekemään muita asioita kuten kosketus kontrollien ja virtuaalilasien orientointia. Koska piilotettavia objekteja oli satoja, aiheutui tästä useiden Framejen pituinen viive, kun kaikki Winter Tagin omaavat objektit tuli hakea muistiin ensimmäisen Framen aikana. Piilottaminen on pakko tehdä varmuudelta aina kun ohjelma käynnistetään, jotta yhtä aikaa ei ole näkyvissä sekä talvi että kesä objekteja. Tämä useiden Framejen pituinen viive osoittautui pahaksi ongelmaksi, kun yhtäkkiä koko järjestelmä lakkasi toimimasta. Aloin virheen testauksen ja viaksi osoittautui se, että Vive virtuaalilasit lataavat ohjelman käynnistyessä itsensä toimimaan ja orientoivat itsensä. Jos tämä ei pääse tapahtumaan ensimmäisen framen aikana heti kun ohjelma käynnistyy, pelimoottori jäätyy. Siirsin talvi objektien piilotuksen heti Vive lasien orientaation jälkeen ja ongelma katosi, samalla myös performanssi moottorissa parantui huomattavasti.

Koska talvi ja kesä kentät piti manuaalisesti yhdistää ja jouduin kopioimaan tietoja kentästä toiseen useita ongelmia ilmeni pelimoottorissa. Korjasin kopioinnin seurauksena hajonneet materiaalit ja Layerit niin, että materiaalit näkyivät taas oikein. Jostain syystä peli-

moottori hävitti materiaaleissa lumi diffuse mapin jokainen kerta. Tästä syystä materiaali muuttui täysin kiiltäväksi.

4.4.2019

Virtuaalitodellisuus ympäristöjen lopullinen yhdistäminen mallin eri vuodenaikojen osalta samaan sceneen ja muutosten esittely esimiehelle huomista vierailua varten sekä laadunvarmistus kuuluivat päivän työtehtäviini.

Tein vuodenaikojen vaihdon loppuun lisäämällä kaikki talvi kentän objektit olemassa olevaan kenttään ja asetin oikeat Tagit niille. Laitoin partikkeli efektillä toimivan lumisateen päälle silloin kun sitä tarvittiin eli Winter Enum:in ollessa päällä Player Blueprintissä, sekä korjasin pieniä visuaalisia bugeja kartasta. Yhdistin muun tiimin tekemät uudet talo meshit sekä materiaalit nykyiseen työ versiooni ja varmistin kaikkien toimivan oikein.

Projektin tila esiteltiin esimiehelle, joka antoi muutamia parannusehdotuksia visuaaliseen ilmeeseen ja huomasi muutamia bugeja, jotka korjasin. Lisäsin ympäristöön hieman värien korjausta ja muutin valaistus efektien vaikutusta suotuisammaksi.

Ensi viikolle lupasin laittaa tämän projektin jatkajille Git versio hallinta järjestelmän käyttöön, sillä tällä hetkellä minkäänlaista versiohallintaa ei ole olemassa. Projektin työstäminen usean ihmisen kanssa yhtä aikaa on todella riskialtista ilman versiohallintaa. Tällä kertaa tiukan aikataulun vuoksi emme kerenneet aloittaa luomalla versiohallintaa, vaan jokaisella käyttäjällä oli oma kopio projektista toisistaan erillään olevilla etälevyillä. Lopulta kaikki versiot yhdistettiin yhdelle koneella kopioimalla oikeat osat projektista toiseen käsin.

Valitettavasti lumisade osoittautui ongelmaksi, sillä toisen henkilön tekemä lumisade Blueprint ei toiminut aivan kuten ajattelin, enkä kerennyt vaikuttamaan sen toimintaan ajan loppuessa. Jouduimme käyttämään vanhaa versiota mutta lopulta sillä ei ollut isoa merkitystä lopputuloksen kannalta.

5.4.2019

Vapaa

06.04.2019

Tavoite päivälle oli viimeistellä mobiilipelin testiversio julkaisijamme testiä varten. Julkaisija oli viimeisessä testissä pyytänyt, että lisäämme seuraavat asiat pelin seuraavaan testiversioon:
Haptinen palaute pelin lentomekaniikkaan, uusia grafiikoita lentämisen aikana sekä pelimaailman muutos niin, että pelaajat pääsevät aiemmin lentämään avaruuteen pelissä.

Käytämme tässä projektissa Unity pelimoottoria pelin kehitykseen. Jaoimme tehtävät työparini kanssa niin, että hän ottaa haptisen palautteen luonnin vastuulleen pää koodarina ja minä siirryn tekemään graafisia muutoksia. Avasin pelin uusimman version Git versio hallinnalla ja aloitin työt muokkaamalla Sprite muotoista taustaa ja skaalaamalla taustaa Prefab objektin sisällä siten, että avaruus alkaa hieman aiemmin. Minulta jäi huomiotta varmistus kerroksen yksi osa tässä kohtaa, jonka jouduin vielä liikuttamaan myöhemmin alemmas, jotta pelin taustat näyttävät yhtenäisiltä. Tein muutokset suoraan olemassa olevaan Prefabiin, mutta olisi ollut järkevämpää luoda kokonaan uusi Prefab objekti, jotta voimme vaihtaa tarpeen vaatiessa pelin taustat takaisin vanhaan versioon.

Siirsin pelimaailman avaruuden planeettojen generointia ylemmäs koodissa ja pilvien Prefabin sisällä spritejä tiiviimmäksi paketiksi. Näin saamme aikaan paremman kokemuksen ylöspäin lentämisestä. Pelissä on mahdollista avata pelaamalla parempia hahmoja, joilla on enemmän bensaa ja erikoistaitoja. Muutosten jälkeen pelaaja pääsee muutamaa hahmoa aikaisemmin avaruuteen, kuten julkaisija toivoi ja sivutuotteena pilvien muutoksista taivas näyttää paremmalta. Vauhdin tuntu lisääntyi muutosten myötä merkittävästi.

Seuraavaksi siirryin tekemään peliin uutta graafista osaa, jossa hahmon käyttämän bensan loppuessa ruudulle piirretään pakokaasupilviä, jotka tupruavat ulos pelaajan rakettirepusta. Aloitin luomalla spritet savu pilville koossa 64x64 pixeliä pitääkseni koon mahdollisimman pienenä. Pyrimme aina mobiilipeleissä mahdollisimman pieneen tiedostokokoon, jotta voimme varmistaa pelin toimivuuden vanhemmillakin laitteilla. Koska kyseessä on partikkeli efektiä varten luodut Spritet png muodossa tein molemmat savupilvet omina kuvinaan. Jos olisin yhdistänyt ne samaan tiedostoon olisi Unity moottorissa tullut ongelmia käyttää niitä myöhemmin.

Loin partikkelit Photoshop CS ohjelmalla Greyscale png kuvana ilman värejä. Värit voidaan lisätä tarpeen vaatiessa suoraan Unity pelimoottorissa pelin aikana. Alpha kanava hoitaa Spriten läpinäkyvyyden, joten on tärkeää muistaa asettaa kuvan tausta Alpha kanavalle. Asettamalla taustan Alpha kanavalle kuvaan voimme tehdä erimuotoisia partikkeleita Unityssä.

Partikkeli kuvien luonnin jälkeen loin partikkeli järjestelmän Unityn sisällä. Partikkelisysteemin piti pystyä piirtämään savupilvi tarpeen vaatiessa. Pilvessä tuli olla erivärisiä tummia savu pilviä, jotka pyörivät ja haihtuvat ilmaan. Savupilvet tulevat kartion muodossa lähtöpisteestään eri aikoihin ja vain silloin kun koodissa niitä kutsutaan. Eli partikkeli efekti piti laittaa pois päältä alkuun ja kutsua vain silloin kun sitä tarvitaan. Partikkeleille annettiin satunnaisgeneroitu arvo alussa, joka määrää partikkelin pyörimisen ja koon jokaiselle partikkelille erikseen. Savupilven tulee näyttää joka kerta hieman erilaiselta sekä luonnolliselta, joten yritys ja erehdys ovat tärkeitä työvälineitä oikean visuaalisen ilmeen saavuttamiseksi.

Kun partikkeli pilvet olivat valmiita, avasin Player Powers ja Player Controller scriptit auki Visual studio 2017:ssa ja kävin kirjoittamassa uudet koodit kohtiin, joissa peli katsoo, onko bensaa jäljellä. Tein kaksi uutta funktiota toiseen koodiin nimillä SmokeStart ja SmokeStop. Nämä funktiot käynnistävät partikkeli efektin ja lopettavat sen ja näitä kahta funktiota kutsutaan toisen Scriptin sisältä, kun bensa on vähissä tai kun bensa on loppunut. Loin koodin siten, että heti kun bensa on jäljellä 20% alkaa pelaajan raketti savuttamaan pieniä pilviä tasaisin väliajoin. Kun pelaajan bensa loppuu kokonaan, myös partikkeli efekti kytkeytyy pois päältä. Tekemättä jäi vielä ääni Scripti, jossa savun lisäksi kuuluisi pulputtavaa kaksitahtimoottorin ääntä. Haluamme tällä kertoa pelaajalle, että bensiini on lähes lopussa ja antaa tästä myös audiovisuaalista vihjettä. Sivutuotteena huomasimme, että oli hauskaa katsoa, jos pelaajalla oli jäänyt bensaa jäljelle, kun hän oli törmännyt maahan ja pelaajan raketti jäi savuttamaan maahan.

Lisäksi testasin haptista palautetta lennon ja törmäämisen aikana ja annoin palautetta ja oman mielipiteeni siitä, miten se kuuluisi laittaa toimimaan.

Erittäin onnistunut työpäivä, jossa asetetut työtehtävät onnistuivat hyvin.

Viikkoanalyysi

Blueprint järjestelmän käyttö eroaa melko paljon C# kielestä, jota Unity käyttää. Unreal:in sisällä koodausta voidaan tehdä myös C++ kielellä, joka on lähellä C# kieltä. Itse en kuitenkaan tällä hetkellä kirjoita C++ joten päätin tehdä visuaalisella scriptaus järjestelmällä tarvittavat koodit.

En ollut vuosiin käyttänyt Blueprint järjestelmää ja kiinnostavaa oli huomata, kuinka paljon se on muuttunut. Osa komennoista oli pudonnut kokonaan pois ja uusia komentoja oli tullut todella paljon. Toisin kuin ohjelmointikielet, jotka eivät juuri muutu, moottorin sisäiset

omat scriptaus järjestelmät näyttävät elävän todella nopealla syklillä. Tästä voidaan päätellä, että oman ammattitaidon ylläpitämiseksi tulisi muistaa lukea eri moottoreiden blog päivitykset ja pitää ammattitaitoa yllä muutenkin moottoreista.

Itse scriptauksessa opin paljon siitä, miten blueprint järjestelmä toimii sillä olin unohtanut ison osan ja samalla opiskelin miten Enum-järjestelmää käytetään Unrealin sisällä. Lähteenä opiskeluun käytin Unrealin omia foorumeita (Unreal Engine Forum 28.4.2014) sekä video tutoriaaleja (Virtual Learning Hub 15.8.2018.).

Unrealissa kentästä toiseen kopiointi aiheuttaa useiden riippuvuus suhteiden hajoamisen koska kentissä ei ole käytetty Prefabeja vaan jokainen kenttä sisältää omat puunsa yksilöinä. Prefab järjestelmää hyödyntämällä voisimme tehdä muutokset suoraan kaikkien kenttien välillä ja kopioida asioita kentästä toiseen ilman, että riippuvuussuhteet hajoavat. Suosittelen jatkossa tiimiä luomaan prefabeja objekteista ja rakentamaan järjestelmän alun perin tämän pohjalta ongelmien välttämiseksi.

Tiimityössä näkyy selkeästi se, ettei kaikilla tiimin jäsenillä ole aikaisempaa kokemusta pelimoottoreista. Hortonomimme Marika saa todella näyttävää jälkeä aikaan, mutta koska pelimoottorikokemus puuttuu aiheuttaa se isoja ongelmia tuotteen teknisessä laadussa. Tämän kommunikointi jatkossa tulee olemaan erityisen tärkeää tulevien projektien onnistumista silmällä pitäen. Samoin versiohallinnan puute on todella riskialtista, sillä isoissa projekteissa on aina mahdollista inhimillisiin virheisiin.

Versiohallinta on aivan ehdoton osa mitä tahansa pelimoottorilla tapahtuvaa kehitystyötä. Versionhallinnan hyödyiksi näen ensisijaisesti juuri inhimillisten virheiden poistamisen ja toisekseen työn helpottamisen. Tällä hetkellä projektin parissa työskentelevät Unrealissa joutuivat tekemään toisistaan irrallaan työtä pelimoottorin parissa ja on todella tarkkaa, ettei kukaan muokkaa toisen käytössä olevia moottorin osia samaan aikaan. Jos muokkausta tapahtuisi päällekkäin lopulta toisen käyttäjän työ katoaisi, kun tiedot yhdistetään. Versionhallinta eroaa myös Unreal ja Unity järjestelmissä. Molemmissa on tuki moottorin sisäiselle versiohallinnalle. Unreal hoitaa asian siten, että käyttäjä saa luoda joko oman järjestelmänsä ja voi integroida sen suoraan moottorin sisäisellä työkalulla projektiinsa tai voi alkaa käyttämään Unreal:in tarjoamaa versiohallintaa. Unity ei anna suoraa mahdollisuutta käyttää omaa versiohallintaa ja jopa tuntuu, ettei moottori halua tukea ulkoisia versiohallintoja. Unityn oma Collaborate versiohallinta on kuitenkin vain yksinkertaistettu Git järjestelmä, josta puuttuu todella merkittäviä ominaisuuksia verrattuna täyteen Git järjestelmään ja on kaiken lisäksi maksullinen ja äärimmäisen hidas.

Git järjestelmän suurimmaksi eduksi lasketaan juuri se, että se mahdollistaa usean käyttäjän yhtäaikaisen projektin käyttämisen ja muokkaamisen ja LFS lisäosalla Git taipuu erinomaisesti myös peliprojekteihin. Git on avoimeen lähdekoodiin perustuva järjestelmä ja erittäin ketterä ja siksi onkin erikoista, että Unityn oma Collaborate eroaa todella merkittävästi alkuperäisestä Git järjestelmästä lähes kaikin osin. Collaborate toimii kyllä kuten kuuluukin, mutta sen käyttäminen tuntuu todella kankealta.

Kaikinpuolin viikko onnistui hyvin ja orientoituminen toisen moottorin sisällä olevaan tuotteeseen onnistui. Sain aikaan tarvittavat toimenpiteet, jotta projekti voi edetä seuraavalla viikolla eteenpäin ja paineen alla toimiminen onnistui tiimin kanssa hyvin.

Olen varsin tyytyväinen todella kiireellä tehtyyn uuteen koodiin ja siihen, kuinka nopeasti pystyin analysoimaan aiemmin toisen henkilön toimesta kirjoitettua koodia. En ole graafiseen lopputulokseen täysin tyytyväinen, mutta toimiva lopputulos ajankäyttöön nähden oli paras mahdollinen lopputulos tälle viikkoa.

3.2 Seurantaviikko 02

Viikon työtavoitteet olivat jatkaa Plantabib ohjelmiston kehittämistä User Interface ja serveri integraatioiden osalta sekä parantaa graafisesti Unityssä olevaa virtuaalimallia. Virtuaalimallin isoimmat haasteet liittyvät performanssi kyvyn rajoissa tapahtuvaan graafiseen kasvojen kohotukseen.

Graafiset parannukset tulisi pystyä tekemään siten, ettei Frame Per Second (FPS) tipu enempää nykyisestä. Tavoite oli myös tutkia viikon aikana tekstuuri ja materiaali vaihtoehtoja ja olemassa olevia kirjastoja niin Terrain meshin kuin mallinnettujen meshien ja muiden objektien osalta.

Torstaina oli sovittu tekninen palaveri lahteen serverin integraation ymmärtämiseksi Plantabib:in osalta ja serverin rajapintojen hahmotteluun. Tavoite on luoda ymmärrys mitä rajapintoja tarvitaan, kuinka ne tuotetaan ja missä aikataulussa ne toteutetaan serverin ja ohjelmiston osalta. Iso osa viikosta oli budjetoitu tutkimiseen ja mallin graafisen ilmeen ymmärtämiseen.

08.04.2019

Maanantai palaverissa käytiin viime viikko läpi ja maanantain tekniikka palaverissa katsottiin edellisen viikon toteutuminen tekniikan osalta. Unreal versioon oltiin tyytyväisiä edellisellä viikolla ja se kelpasi sellaisenaan esittelyyn. Puhetta oli tulevasta PlehatCon tapahtumasta ja siihen liittyvistä työtehtävistä.

Tällä hetkellä työ resursseja siirrettiin pois Plantabib ohjelmiston kehittämisestä, jotta Unity Enginen virtuaalimalli tuotetta voidaan parantaa pikaisesti graafisen ilmeen osalta. Plehat esittelee tuotetta pian asiakkaille, joten graafinen ulkomuoto pitää saada paremmaksi esittelyä varten. Myynnin edistämiseksi toimivan tuotteen tulee myös näyttää hyvältä toimivuuden lisäksi.

Aloitin työn tutustumalla, kuinka virtuaalitodellisuus malli on luotu graafisen ilmeen osalta ja kuinka sitä voitaisiin parantaa. Tutustuin mallin eri Shadereihin ja terrain teknologiaan. Terrain teknologiassa on käytetty CTS terrain Shaderia, joka vaikuttaa olevan yksi suosituimpia maksullisia työkaluja. Löysin useita parannuskohteita mm. kasveista, joita mallissa on käytetty sekä Terrain Shadereistä ja tekstuureiden käytöstä. Terrain Shader on toimiva mutta on mahdollista, että perinteinen Mesh objekti voisi olla parempi graafisen kyvyn osalta mutta työläämpi valmistaa.

Tutkin miten Unity eroaa Unreal Enginen Terrain ja Vegetation työkalujen osalta ja tutkin miten Unityn Vegetation työkalujen käyttö onnistuu osana tuotetta vertailemalla eri vaihtoehtoja (Procedural Worlds 2019.).

09.04.2019

Päivän tavoite oli jatkaa maanantain tutkimuksia virtuaalitodellisuus mallissa ja kartoittaa teknologiaa lisää.

Jatkoin edellisen päivän dokumentaatioiden lukemista ja tutustuin tarkemmin CTS shaderin ja Vegetation Studion toimintaan Unityn sisällä. Opin kuinka Unityn lisäosien käyttö toimii yhteistyössä muiden lisäosien kanssa ja kuinka CTS Shader on rakennettu.

Lounaan jälkeen aloin tutkimaan kuinka löydettyjä parannuskohteita voidaan alkaa viemään käytäntöön. Isoimmat parannusten osat tekstuureissa löytyivät siitä, että tekstuurit eivät ole samaa tasoa toistensa kanssa, koska ne eivät tule samoista lähteistä alun perin. Tämä voidaan korjata halutessamme käyttämällä samanlaista tekstuurien ja materiaalien valmistustapaa. Toisistaan hieman eri tavoin valmistetut tekstuurit eivät sovi koskaan

saumattomasti yhteen graafisen tyylinsä perusteella ja aiheuttavat käyttäjälle hämmennystä ja rikkovat visuaalista tyyliä.

Lisäksi selvitin, miten tekstuurien päälle on blendattu Color Map tekstuuri Terrain Meshissä. Color Map on heijastettu CTS Terrain Shaderissä Terrain Meshin päälle luomaan vaihtelevuutta ja tosimaailman tyyliä. Color Map on tehty tosi maailmaan oikeista ortograafisista kuvista ja sen resoluutio on huomattavasti matalampi kuin tuotteen muiden tekstuurien. Ongelma resoluutiossa näkyy pienilläkin väriarvoilla. Color Map:in tarkoitus on saada vaihtelevuutta aikaan mallin materiaaleihin, ja se onnistuukin tässä vaikkakin resoluution kustannuksella. Tekstuurin epätarkkuudesta johtuen mallin ulkonäkö kärsii merkittävästi. Tuotteessa käytetään Color Mappia tällä hetkellä siksi, koska käsin Terrain Painting:in tekeminen jokaisen yksityiskohdan osalta on todella aikaa vievää.

Toinen vaihtoehto kokonaan käsin maalaamiselle on tehdä Color Map:in päälle Color Mask ja näin saada reunoista tarkempia. Tämä ei kuitenkaan poista kaikkia materiaalin ongelmia ja epätarkkuuksia eikä ole juuri nopeampaa kuin käsin maalaaminen. Ongelmia aiheuttaa myös se, ettei CTS Shader tue parametrisiä materiaaleja suoraan.

Puiden tutkimisesta selvisi eilen se, että puut on tehty Speedtree 7 ohjelmistolla ja uuden Speedtree 8 ohjelmiston puut ovat laadultaan parempia niin resoluution kuin tekniikkansa osalta. Tavoite on muuttaa mallissa olevat kasvit Speedtree 7 ohjelmistosta Speedtree 8 version vastaaviin. Uudella ohjelmistolla tehdyt kasvit parantavat huomattavasti tuotteen yleisilmettä.

Päivän aikana osallistuin myös konsultaation antamiseen virtuaalimallin sisällä tapahtuvaan virtuaalikasvien käsittely työkalun valmistamista varten.

10.04.2019

Latasin aamulla Unityn oman showcase mallin Book of the Dead (Unity Book of the Dead 29.6.2019) ja asensin sen työkoneelleni. Tavoite oli katsoa kuinka Book of the Dead demossa on luotu luonnonympäristöjen Color Correction, Environment Fog sekä valaistus efektit. Lisäksi luin projektin dokumentaation ja tutkin kuinka demon Photogrammetria on toteutettu. Parasta olisi, jos pystyisimme tekemään samantasoisen visuaalisen ilmeen omaan ohjelmistoomme.

Päädyin edellisen päivän tutustumisen jälkeen työskentelemään tekstuurien parissa ja etsimään parasta mahdollista työkalua Terrain Mesh ja Mesh objektien tektsturointia var-

ten. Tarpeet työkalulle ovat, että saisimme parametrisesti hallittavia materiaaleja luotua Unityn standard shaderiin käyttöön. Emme voi käyttää HDPR Shadereitä joita Unity jo tukee. HDPR Shaderit ovat laadultaan todella korkeatasoisia, mutta niitä ei ole optimoituja virtuaalitodellisuutta varten vielä.

Illalla pidimme 2 tunnin mittaisen palaverin QR koodi virtuaalitodellisuus järjestelmän tekemisestä ja suunnittelimme ohjelmiston valmistusta ja puhuimme aikataulut valmiiksi kesää varten. Samalla suunniteltiin kahta muuta prototyyppi ohjelmistoa.

11.04.2019

Matkustin Lahteen palaveriin aamusta ja palasin illalla toimistolle. Tapasimme serverin API rajapinnan suunnittelusta vastaavan tahon Lahdessa ja keskustelimme kuinka rajapinta tulisi valmistaa teknisesti, jotta se on yksinkertainen päivittää tulevaisuudessa laajemmaksi. Keskustelimme eri portaista, kuinka projektia viedään alun serveri integraatiosta kohti lopullista tuotetta. Kävimme läpi tarpeet Plantabib ohjelmistoa varten ja kuinka teknisesti valmistetaan rajapinta ohjelmiston ja serverin väliin, Sovimme työtehtävistä ja aikataulusta ja esittelimme mobiilin virtuaalitodellisuus mallimme.

Toimistolla jatkoin käyden läpi virtuaalitodellisuus mallin teksturointi työkalujen vaihtoehtoja ja dokumentaatiota läpi. Tutkin kuinka parhaiten voimme valmistaa parametrisen järjestelmän tekstureiden käyttöä varten ja miten saamme eniten irti erilaisista ohjelmistoista. Samalla tutkin kuinka Book Of The Dead malli oli toteutettu materiaalien osalta ja minkälaisia ongelmia ja ratkaisuja tekijä tiimi oli päättänyt käyttää ja hain lisää tietoa parametrisistä materiaali järjestelmistä.

Kävimme läpi palaverissa mitä lahdessa oli keskusteltu ja aikataulut Plantabib ohjelmiston kehitykselle ja missä vaiheessa projektin eri vaiheet tulevat kehitykseen.

12.04.2019

Päivän tavoite oli teksturointi työkaluihin tutustumisessa ja dokumentoinnin ja PBR shadereiden tieteen läpikäyntiä sekä opiskelua kuinka ohjelma toimii käytännössä.

Päädyimme tekstuurien valmistuksen osalta Substance ohjelmistoihin, joilla voimme luoda sekä parametrisiä tekstuureja, hankkia valmiita materiaaleja Unityyn sekä parametristen materiaalien lisäksi exportata tarpeellisia tekstuureja yhtenäisemmän virtuaalitodellisuus tuotteen rakentamista varten.

Aloitin asentamalla Substance paketin ja ostamalla lisenssin itselleni. Asensin paketista Designer ja Painter ohjelmistot ja aloitin näihin tutustumisen. Ohjelmat mahdollistavat erittäin tehokkaan työskentelytavan parametristen materiaalien valmistukseen ja tukevat lähes kaikkia pelimoottoreita suoraan. Ohjelmat ovat Node pohjaisia, kuten Unrealin Visuaalinen ohjelmointikieli, joten käyttäminen vaikuttaa tutulta heti alussa. Katsoin useita ohjevideoita valmistajan sivuilta ohjelman käyttöönottoon liittyen ja tutkin dokumentaation avulla mistä kannattaa lähteä liikkeelle. Lisäksi luin tieteellisiä artikkeleita, miten ohjelma käyttää PBR materiaaleja tehdäkseen todellisuutta jäljittelevät materiaalit pelimoottoreiden käyttöön.

PBR materiaalit ovat fysikaalisesti oikeaa materiaalia jäljitteleviä tekstuureja, joissa pinnan muodot ja materiaali vaikuttavat sen ominaisuuksiin. Opiskelin kuinka matemaattisesti lasketaan eri taitekertoimia ja mistä eri materiaalien ominaisuudet johtuvat. Dokumentaatio oli erittäin hyvää oppia siitä, kuinka materiaaleja valmistetaan oikeaoppisesti. Huomioon tulee ottaa pinnan muoto, sen läpikuultavuus, sen sähköä johtavuus, materiaalin taitekerroin sekä pinnan mikrovaihtelut.

Viikkoanalyysi

Viikon aikana opin todella paljon yrityksen Virtuaalimallin osalta Unity:ssä sekä kuinka serveri integraatiota lähdetään toteuttamaan. Erityisen kiinnostavaa oli oppia virtuaalimallin toteutuksesta CTS Shaderin osalta ja kuulla tiimin tekemiä ratkaisuita. Oli kiinnostavaa nähdä miten Unity version osalta vuodenaikojen vaihtokoodi oli tehty ja verrata siihen, kuinka itse tein sen Unreal:ssa. Tässä tulee hyvin esiin pelimoottoreiden isoimmat erot.

Unreal ei juuri sisällä lisäosia vaan se on itsessään toimiva valmis kokonaisuus. Terrain Shaderit ja kasvillisuus työkalut ovat suoraan sisään rakennettuja ominaisuuksia moottorissa, kun taas Unity:n omat vastaavat ovat todella puutteellisia. Tästä syystä Unity:n käyttäjät usein vaihtavat pelimoottorin versiot toimivampiin kolmansien osapuolien valmistamiin järjestelmiin, joissa ominaisuuksia ja valintoja on enemmän. Vaikka kolmannen osapuolen versiot voivat Unity:ssä olla parempia kuin Unreal:in mukana tulevat ominaisuudet on jokaisen lisäosan käyttäminen ongelmallista projektin koon kasvaessa eivätkä kaikki lisäosat toimi muiden lisäosien kanssa moitteetta. Tästä syystä näen Unrealin kokonaisratkaisua tarjoavan järjestelmän olevan Terrain shader:ien kohdalla järkevämpi tapa toteuttaa asia, kun ylimääräistä säätämistä tulee vähemmän. Toisaalta Unityn kolmannen osapuolen lisäosia on saatavilla tuhansittain, joten tarjonta on laajempi kuin Unrealissa monen muun lisäosan osalta.

Edellisellä viikolla tekemäni nopea vuodenajan vaihtokoodi yrityksen toisessa virtuaalitodellisuus tuotteessa on tehty hyvin samalla tavalla kuin tämä loppujen lopuksi. Eroa oli lähinnä siinä, kuinka Unityn järjestelmä taipuu monipuolisempaan vuoden ajan muutokseen sekä eri vuosikymmenten vaihtoon tällä hetkellä. Nämä ominaisuudet on mahdollista tehdä myös Unreal versioon, mutta toistaiseksi niitä ei ole tarvittu. Tiimin kanssa käydessämme läpi asioita huomasin myös, että joissain kohdissa graafisen ilmeen kanssa oli tehty oikaisuja ajan säästämiseksi Unity versiossa.

Ajanhallinta on ylipäätään usein haastavaa projekteissa, joissa isompi tiimi ratkoo ongelmia. Mielestäni tärkeintä tiimin toimivuuden ja ketteryyden varmistamisessa onkin selkeä projektin johtaminen, joka pystyy tekemään päätöksiä tiimin ehdottamien ratkaisuiden pohjalta. Tällä hetkellä Unity moottorissa kaikki koodi on yhdessä scriptissä, joten koodin muuttaminen on hankalaa ja kankeaa ja muutoksia joudutaan miettimään todella monelta kannalta. Isojen muutosten tekeminen koko järjestelmän toimintaan on hankalaa.

Oma työskentelytapani tuntuu eroavan hieman muusta tiimin työskentelystä. Pidän tehokkaana työtapana miettiä muutamia vaihtoehtoja ja lähteä välittömästi rakentamaan melko pienistä olioista koostuvia järjestelmiä ja iteroimaan eri vaihtoehtoja. Hyötynä tässä tavassa on, että yhden olion kokonaan korvaaminen ja muutokset eivät vaikuta muuhun projektiin ja uusien ominaisuuksien tuonti järjestelmään on helppoa. Haittapuolena tässä tavassa on, että aikaa voi mennä hukkaan iteroidessa uusia ominaisuuksia ja ratkaisumalleja testatessa. Oma mielipiteeni on kuitenkin, että pienistä osista koostuva järjestelmä on hyvä, kunhan projektin dokumentaatiota pidetään ajan tasalla. Dokumentaation tekeminen on erityisen tärkeää, jotta muu tiimi ymmärtää miten järjestelmä toimii.

Perinteisen ei olio pohjaisen järjestelmän tekemisen ongelmana on, ettei se jätä liikkumavaraa tuotteen kehityksessä riittävästi uusille ajatuksille ja järjestelmille. Koska yrityksen ohjelmistotuotteet ovat vielä alkuvaiheessa kehityskaarta ja potentiaalisten asiakkaiden tarpeet voivat vaihdella todella paljon näen, että olio pohjainen kehitys sopisi projektiin paremmin tulevaisuudessa.

Olio pohjainen ohjelmointi ja ketterien menetelmien hyötykäyttö on mielestäni ensiarvoisen tärkeää pelimoottorien kanssa toimiessa. Pelimoottoreiden jatkuvat päivitykset ja uusien ominaisuuksien lisääminen voivat aiheuttaa yllättäviä ongelmia koodissa ja jos kaikki koodi suoritetaan samassa scriptissä on moottorin päivityksen yhteydessä riskinä, että koko järjestelmä hajoaa.

Ketterien menetelmien käyttöön otossa vaikuttaa myös yrityksen kulttuuri, jossa mielestäni näkyy, että tiimin muut jäsenet ovat maisema-arkkitehteja. Tiimillä on todella vankka osaaminen ohjelmistojen teosta ja ymmärryksestä pelimoottoreista mutta uusia ominaisuuksia kokeillessa käydään todella tarkkaan etukäteen läpi mitä ongelmia ja mahdollisuuksia on olemassa. Asioiden tarkka läpikäynti on varmasti hyödyllinen työkalu ohjelmisto kehityksessä, kuitenkin itse näen, että nopealla iteroinnilla ja testauksella saadaan tuloksia nopeammin aikaan. Oma kokemukseni on osoittanut, ettei kaikkeen voi kuitenkaan varautua etukäteen projekteissa ja hyvät ideat voivat osoittautua käytännössä huonoiksi ja tästä syystä on parempi epäonnistua nopeasti.

Viikon aikana teimme päätöksen hankkia Substance ohjelmistot ja lähdimme testaamaan niiden käyttöä graafisen ilmeen parantamisessa Unity tuotteessa. Substance vaikuttaa helppokäyttöiseltä ja loogiselta kokonaisuudelta, josta saamme todennäköisesti runsaasti säästöä ajankäyttöön ja laadukkaita materiaaleja tuotteeseemme. Haittapuolena järjestelmässä on, että Unity:n Substance lisäosa on vielä testivaiheessa, joten sen toiminta voi muuttua todella paljon tulevaisuudessa.

Substance järjestelmän suurimmat hyödyt sain mielestäni kuitenkin tutkiessani PBR materiaalien dokumentaatiota ja kuinka pelimoottorit laskevat oikeiden matemaattisten mallien avulla tekstuurit. Oli todella tärkeää oppia oikeat laskukaavat ja ymmärtää syvemmin materiaalien toimintaa, jotta pystyn rakentamaan Virtuaalitodellisuus tuotteeseemme oikeanlaiset materiaalit. Ilman fysiikan lakien ymmärtämistä taitekertoimissa ja sähkönjohtavuudessa ei ole mahdollista rakentaa luonnollisen näköisiä virtuaalitodellisuus ympäristöjä, jotka käyttäytyvät oikeaoppisesti. Tästä syystä näen, että oman kehitykseni kannalta on tulevaisuudessa entistä tärkeämpää, että tutustun myös tieteelliseen puoleen enemmän ja pintaa syvemmältä. (Substance 3D 2019 (2).)

Serverirajapinnan rakentamiseen liittyvät asiat sujuivat hyvin ja saimme hyvää teknistä tukea siitä, kuinka Plantabib järjestelmän serveri integraatiota lähdetään rakentamaan. Tiimityötaidot vaikuttavat olevan myös tässä projektissa avain asemassa, sillä aikataulu näyttää kiristyvän loppua kohti tulevan Plehatcon tapahtuman myötä.

3.3 Seurantaviikko 03

Viikon työ tavoitteisiin kuuluu grafiikoiden parantamista Virtuaalitodellisuus tuotteeseen Unity pelimoottorissa tulevaa Plehatcon tapahtumaa varten. Tavoite on lähteä luomaan ensimmäistä isoa parannusta tuotteeseen graafisesti tekstuurien ja materiaalien kautta. Isojen graafisten muutosten takaraja on kahden viikon päässä perjantaissa, jolloin Ple-

hatcon tapahtuma järjestetään Otaniemessä. Lisäksi kasvien päivittäminen on tarkoitus aloittaa Speedtree 7 versiosta Speedtree 8 versioihin ja tutkia mitä etuja Speedtree 8 antaa kasvien malleihin Unityssä sekä voiko Unreal versioon tehtyjä kasveja hyödyntää sellaisenaan Unityssä ja kuinka ne siirretään pelimoottorista toiseen. Substance ohjelman käyttöönotto ja opettelu on myös viikon tehtävälistalla.

15.4.2019

Maanantain aamupalaverissa käytiin läpi viime viikon tapahtumat sekä suunniteltiin kuluvaa viikkoa. Lisäksi käytiin keskustelua ja suunniteltiin tulevaa Plehatcon tapahtumaa ja sen työtehtäviä. Tekniikka ryhmän palaverissa käytiin läpi viikon työtehtävät ja edellisen viikon kehitys ohjelmistojen osalta. Plantabib on nyt jäähyllä Plehatcon tapahtumaan asti, joka on tällä hetkellä priorisoitu tärkeämmäksi.

Työ jatkui Unityn virtuaalimallin graafisen parantamisen osalta palaverin jälkeen. Asensin Speedtree 8 ohjelman työ koneelleni. Ohjelmasta on olemassa Unity ja Unreal versiot ja asensin molemmat koneelleni, jotta voin testata versioita ja niiden eroja. Aloitin tutustumalla ohjelmistoon lukemalla dokumentaatiota (Speedtree 2019.) ja selvitin miten ohjelmat toimivat avaamalla valmiita puita ja tutustumalla niiden hierarkiaan.

Aloitin vertaamalla aiemmin valmiiksi tehtyjä Unreal version puita sekä ostettuja malleja keskenään sekä vertaamaan niitä Unity Speedtree 7 version puihin. Ajan säästämiseksi pyrimme tiimissä hankkimaan puita, jotka olisivat suoraan valmiita Unity:n käytettäväksi Speedtreen verkkokaupasta. Lisäksi kartoitin mitä kasveja yritykselle on aiemmin ostettuna tai valmistettuna itse ja mitä kasveja vielä puuttuu. Loin listauksen molempien pelimoottoreiden tarpeista yhteiseen käyttöön tiimille tästä. Selvitykseni perusteella meille on mahdollista hankkia helposti muutamia kasveja ulkopuolisista lähteistä, mutta osa joudutaan tekemään itse.

Unity pelimoottorin virtuaalimallin puut on tehty Speedtree 7 järjestelmällä, joka on osoittautunut järkeväksi ja kustannustehokkaaksi tavaksi luoda hyvin toimivia puita tuotteeseen. Speedtree 7:ssä on joitakin rajoitteita muiden kasvien mallien teossa, koska järjestelmä on tehty puita varten eikä näin ollen sovellu muiden kasvityyppien mallintamiseen. Yrityksen toiseen Unreal pohjaiseen Virtuaalitodellisuus tuotteeseen mallit on tehty käyttämällä Speedtree 8 ohjelmistoa. Speedtree 8 antaa huomattavan graafisen kasvojenkohotuksen verrattuna Speedtree 7:ään ja kasvien realismin tuntu on parempi. Myös paremmat tekstuurit, jotka tulevat ohjelman uuden version myötä sekä muiden kasvien luonti mahdollisuudet ovat selviä edistysaskelia. Unreal versio Speedtree 8:sta on ollut markki-

noilla hetken pidempään kuin Unity versio ja tästä syystä on pidemmälle kehitetty ja bugittomampi versio.

Koska Speedtree 8 mahdollistaa uskottavamman ympäristön ja kasvien suunnittelun ja näin parantaa visuaalista ilmettä on tarpeen tutkia, onko mahdollista siirtyä Unityssäkin Speedtree 8 versioon vaikka version tuotekehitys on vielä kesken. Lisäksi oli tarpeellista tutkia, saataisiinko Speedtree 8 Unreal version puut siirrettyä Unity:n Speedtree 8:aan helposti ja mikä puiden suorituskyky on pelimoottorissa Unityn puolella.

Hortonomimme oli tehnyt Speedtree 8:n Unreal versioon itse kasveja ja nämä puut olisi tarkoitus siirtää Unity Virtuaalitodellisuus tuotteeseen. Speedtree mallinnustyökalu on kuitenkin tehty niin, ettei samaa mallinnettua tuotetta voi käyttää sekä Unity että Unreal tuotteille, vaan molempiin tarvitaan omat lisenssit eivätkä tiedostot ole ristiin toimivia. Koska mallit eivät ole siirrettävissä ohjelman versioista toiseen aiheuttaa tämä tarpeettomia ongelmia. Joko puiden kaikki parametrit tulisi siirtää käsin toiseen editoriin ja tarkastaa toimiiko algoritmi täysin samoin molempien versioiden osalta tai pitäisi pystyä luomaan jokin välitila, josta tiedot saadaan ladattua helposti editorista toiseen.

Jotta kaikkea työtä ei tarvitse tehdä uudestaan puiden kanssa ajatuksena oli, että lataan Speedtree 8:n Unreal mallit työkaluun auki j oka tallentaa puiden parametrisen mallin template tiedostona koneelleni. Tämän template tiedoston koittaisin avata Unity versiolla ja tarkastella hajoaako mallista mikään. Parametrien tallennus ja lataus vaikutti aluksi toimivan moitteetta ja puiden siirto näytti onnistuvan editorista toiseen aluksi ongelmitta. Ongelmia aiheutti se, että editorit on koodattu hieman eri tavoin ja parametrien käsittely tapahtuu eri järjestyksessä. Tästä syystä puut eivät näyttäneet identtisiltä mikä varsinaisesti ei ole ongelma, mutta hortonomimme mukaan Unity:n version kasvit eivät enää vastannut todellisia puita ulkomuodoltaan samoin kuin Unreal version kasvit.

Ongelmana on lisäksi, että puiden materiaalit ja tekstuurit eivät siirry template tiedoston mukana. Tämä tarkoittaa joka tapauksessa merkittävää lisätyötä, kun materiaalit laitetaan kaikille kasvuvaiheille ja vuodenaika malleille käsin. Jokaisella kasvilla on useita kymmeniä tekstuurikarttoja sekä kasvuvaiheita ja jokaisella kasvilla on useita kymmeniä asetuksia, jotka tulisi myös laittaa käsin vastaamaan Unrealin mallia. Lisäksi tutkiessani mahdollisuutta siirtää puita Unreal:sta Unity version editoriin löytyi lisää ongelmia, kun materiaalien laatu ei ollut riittävä. Tekstuurikarttojen heikko laatu ja puutteellisesti tehty varjojen poisto diffuusi tekstuureista aiheuttaa visuaalista ongelmaa lopullisessa tuotteessa dynaamisen valaistuksen osalta.

Isoin ongelma on kuitenkin puiden yksityiskohtien tasossa. Tutkiessani puumalleja Unreal version editorissa huomasin polygon määrien olevan aivan liian suuria. Tämä johtui mm. siitä, että puiden pienet oksat oli mallinnettu jokainen omana objektinaan kuvatiedoston sijasta ja jokainen lehti oli myös oma objektinsa. Ikävä kyllä hortonomimme oli mallintanut puut eri tavalla kuin pelimoottoreiden puut tulisi valmistaa. Pelimoottorimalleissa tähtäämme noin 20 000 - 30 000 polygoniin puuta kohden. Mallinnetuissa puissa oli pahimmillaan jopa 650 000 polygonia yhtä puuta kohti.

EI ole väärin mallintaa kasveja korkealla polygon määrällä jos niiden ei tarvitse toimia reaaliaikaisessa virtuaalitodellisuus ympäristössä, mutta meille nämä puut eivät ole ideaaleja käyttää. Jos yhdessä scenessä on 100 puuta näkyvillä yhtä aikaa, piirtää pelimoottori ruudun yleensä noin 60 kertaa sekunnissa. Tämä tarkoittaisi, että sadan puun kanssa pelimoottori piirtää 650 000 polygonia x 100 60 kertaa jokaisen sekuntin aikana. Tämä aiheuttaa äärimmäistä stressiä koneen suorituskyvylle. Puut eivät siis ole käytettävissä ilman todella isoa muokkausta. Tutkittuani asiaa totesin, että on helpompaa tehdä kokonaan alusta asti puut uudestaan kuin muokata vanhoja.

Onneksemme 1 neljästä tarvittavasta kasvista oli mahdollinen helposti luoda Unity pelimoottoriin, sillä se oli muokattu ostetusta versiosta. Tämän puut polygoni tasot ja tekstuurien laatu olivat riittävällä tasolla käytettäväksi suoraan pelimoottorissa koska ne tulivat pääosin suoraan ostetusta mallista. Tämän kasvimallin avulla oli mahdollista selvittää, minkälaisia graafisia hyötyjä saadaan Speedtree 8:lla aikaan verrattuna Speedtree 7:n kasvimalleihin. Toin puun Unity pelimoottoriin ja vertailimme sen visuaalista ilmettä vanhoihin malleihin VR ympäristössä ja havaitsimme laadullisen eron olevan valtava. Puut kannattaa tehdä Speedtree 8:lla visuaalisen ilmeen parantamiseksi. Tästä ainoasta toimivasta puusta puuttui vielä kasvu versiot eri mitoissa ja vuodenajoissa, jotka tulee valmistaa.

16.4.2019

Tavoite on siirtyä puiden tutkimisesta Substance ohjelmien käyttöönottoon tänään ja luoda ensimmäinen materiaali käytettäväksi Substance Designer ohjelmalla ja tuoda tuo materiaali Unity:n käyttöön.

Aloitimme mahdollisimman yksinkertaisista staattisista objekteista ja kanaalin varressa olevat kivet sopivat hyvin tarkoitukseen. Yksi tiimin jäsenistä aloitti mallintamalla kivet sekä UV kartoittamalla ne. Samanaikaisesti annoin konsultointia tähän tehtävään ja samalla

etsin tarvittavaa lähdemateriaalia 3D mallin alueesta tutkien kuva- ja videomateriaalia paikan maaperästä ja kivistä, jotta voin luoda todelliseen maailmaan pohjautuvat tekstuurit ja materiaalit Unityyn. Tavoite on saada yksi materiaali alusta loppuun valmiiksi Unityyn käyttöön.

Ensimmäinen työtehtävä tekstuurien mallintamisessa on aina etsiä sopivaa lähdemateriaalia, josta voidaan alkaa tekemään oikeanlaista tekstuuria. Löysin useita valokuvia ympäristöstä, joista pystyin päättelemään kuinka kivet ovat muodostuneet ja minkälainen pinta ja materiaali kivissä on kyseessä. Kanaalin kivet ovat harmaata ja beigeä graniittia, jotka ovat karkeaa ja todella huokoista materiaalia ja melko tasaväristä materiaalia. Värien osalta tulee luoda materiaali, jossa on vaihtelevuutta sekä selkeät väri rajat. Kivet ovat halkaistu lähes kaikki samansuuntaisesti kanaalin vartta varten ja selkeät halkeamisurat ovat tyypillisiä alueen kiville. Lisäksi kivissä on jonkin verran pinnan korkeuden vaihteluita ja eri sedimentti kerroksia, joissa graniitin väri vaihtelee. Alimmissa kivissä, jotka ovat veden kanssa kosketuksissa on tarpeen olla myös vihreää levää sekä kosteudesta johtuvaa pinnan kulumista, kiiltoa ja sileyttä. Työtavat olen ottanut itselleni seuraamalla alan parhaita ja kiinnostavimpia tekijöitä sosiaalisessa mediassa vuosia.

Aloitin tämän kartoituksen jälkeen mallintamaan kivien tekstuuri karttoja Substance Designer ohjelmistossa käyttäen apuna muutamia tutoriaaleja ja ohjeita, joita etsin saadakseni paremman käsityksen Substance Designerin parhaista työtavoista. Yksi hyvä esimerkki erinomaisista työtavoista löytyi tutoriaalin kautta (3dEx 7.12.2018.).

Kivissä tulee olla selkeä sivuttaissuuntainen kuluma ilme, joka tulee Blend Nodea käyttämällä lisätä karkeaan kiven materiaaliin sekä väreihin. Ensimmäisenä aloitin tekemään kiven Height mappia käyttämällä Noise tekstuureja sekä Linear Warp Nodea sekä Blend ja Level Nodea. Blend Nodella yhdistin eri Height Nodeja toisiinsa ja yhdistin niitä toisiinsa luoden näin epätasaisen värikartan, joka on mahdollista toistaa loputtomiin, eli niin sanottu reunaton tekstuuri (Seamless texture). Epätasainen värikartta mahdollistaa minulle mahdollisuuden luoda materiaaliin eri värejä eri kohtiin sekä eri karkeus ja korkeus tasoja kiven pinnalle maskien avulla. Käytin useita erilaisia Noise tekstuureja ja useita Blend Nodeja yhdessä luodakseni valokuvan tapaisen karkean graniitti pinnan. Tärkeää on luoda tekstuuri, josta ihmissilmä ei näe rajoja eikä huomaa toistuvaa kuviota.

Ensimmäisenä loin siis tekstuurin, joka näyttää kiveltä Substance Designerissä ja on lineaarinen harmaasävy värikartta. Tästä värikartasta Level Nodea apuna käyttäen nostin vaaleita värejä kirkkaammiksi näin poistaen tummia yksityiskohtia. Tästä vaaleammasta harmaasävykartasta johdin Height värikartan, joka kertoo materiaalille kuinka ylhäällä ja

alhaalla pintojen tasot ovat toisiinsa nähden ja kuinka materiaali piirretään pelimoottorissa. Käytän tätä karttaa pohjana kaikille muille värikartoille.

Height kartasta loin seuraavaksi Normal Nodella Normal kartan, jonka ohjasin Blend ja Levels Nodejen läpi saadakseni haluamani Normal kartan aikaiseksi. Normal map kertoo pelimoottorille miltä pinnan muoto näyttää ja on samantyylinen kuin Height kartta mutta monimutkaisempi ja mahdollistaa paremmat yksityiskohdat.

Height kartasta tein myös Roughness mapin, mutta koska Unity haluaa Roughness mapin käänteisenä eli Smoothness mappina, käytin Invert ja Level Nodeja saadakseni kiveen oikeanlaiset sileät kohdat, joihin katsoin valokuvista mallia. Roughness map kertoo pelimoottorille missä kohdissa kivi kiiltää ja onko kohta materiaalissa Metallic vai Di-electric materiaalia. Metallic kartta on kivillä ja muilla sähköä johtamattomilla materiaaleilla aina musta lineaarinen harmaasävy kartta. Musta ja tumma väri tarkoittaa, ettei materiaalissa mikään kohta johda sähköä ja valkoinen ja vaalea väri tarkoittaa, että kartan kohta johtaa sähköä.

Seuraavaksi loin Ambient Oclusion kartan Level Nodia hyödyntäen ja nostaen harmaasävykuvan valkoista kanavaa siten, että kuvaan jäi selkeästi näkyviin kohtia joihin valo ei pääse pinnan muotojen vuoksi. Näin ollen tekstuuri karttaan jää vain hieman mustaa, joka osoittaa missä kohtaa varjot mikro ja makro pinnoissa tulee piirtää. Ambient Oclusion kartta luo realismia pintoihin piirtäen varjoisia kohtia joihin valo ei pääse osumaan koskaan.

Lisäksi tuli vielä valmistaa värikartta kivestä, yhdistin Blend Nodeilla useita Noise karttoja toisiinsa sekä tasaisia värikarttoja käyttäen erilaisia sattumanvaraisia väriteemoja. Loin näitä yhdistämällä Height mapin kanssa luonnollisen värisen graniitti kartan.

Päivä onnistui todella hyvin ja opin Substance Designer ohjelmasta todella paljon ja oli mahtavaa viedä tieteellisten artikkeleiden tieto käytäntöön ohjelman sisällä.

17.4.2019

Tavoite oli saada kivimateriaali siirrettyä Unityyn ja opetella Unityn Substance lisäosan käyttöä ja kuinka Substance toimii CTS terrain Shaderin uuden version kanssa ja mitä hyötyjä voidaan saada ohjelmasta muuten Unityssä.

Substance ja Substancen Unity lisäosa opettelu alkoi asentamalla loput lisäosat koneeseen. Asensin Substance lisäosan koneelle ja ensimmäisenä tein harjoitukseksi netistä ladatulla materiaalilla testejä. Hain Virtuaalitodellisuusmallin yhteen taloon tiili tekstuurin Substance lisäosalla Unityn sisällä ja aloitin muokkaamaan sitä, jotta se sopii paremmin tosimaailman referenssikuviin. Unityn sisällä Substance lisäosassa on muutamia käteviä etuja normaali materiaaleihin, mutta myös huonoja puolia tutkittuani lisäosaa.

Käytämme Unityssä version hallintana Unityn Collab järjestelmää, joka aiheuttaa Substance lisäosain kanssa ongelmia. Tämä johtuu siitä, että lisäosa on vielä testiversio ja kehitys vaiheessa. Collab tuntuu hävittävän Substance lisäosain asennuksen aika-ajoin ilman näkyvää syytä ja silloin materiaalit hajoavat. Tämä aiheuttaa todella ärsyttävän ongelman materiaaleissa, mutta onneksi vika korjaantuu asentamalla Substance lisäosan uudestaan Assetstoresta. Lisäksi toinen outo puoli on, että Substance ei anna vaihtaa materiaalin tekstuurikarttoja myöhemmin, joten se materiaali jonka tuot Unityyn pysyy sellaisena tekstuuri karttojen osalta. Onneksi itse valmistetuissa materiaaleissa tämä on todennäköisesti kierrettävissä järkevällä parametrien käytöllä.

Hyviä puolia Substance lisäosasta löytyy myös todella paljon. Lisäosan avulla voimme tehdä helposti samasta materiaalista Unityn sisällä useita muokkauksia materiaaliin ja luoda näin toisen instanssin siitä Unityn sisällä. Esimerkiksi jos meillä on tiiliseinän tekstuuri emme voi vaihtaa materiaalin tekstuuri karttoja, mutta voimme muuttaa parametrejä, jotka materiaaliin sisään on luotu. Esimerkiksi voimme tehdä kaksi instanssia tiili materiaalista, joissa on isot halkeilleet vanhat tiilet vanhalla laastilla ja punaisilla tiilillä. Toisia parametrejä käyttämällä voimme luoda toisen instanssin materiaalista, jossa on uudet kiiltävät siniset tiilet. Molemmat kartat käyttävät samaa materiaalia, mutta isot muokkaukset eivät vaadi lisää työtä vaan ainoastaan pientä lisätyötä materiaalin parametrien kanssa. Normaalisti tekstuurista tehtäisiin kokonaan kaksi täysin erilaista materiaalia kaikkine tekstuuri karttoineen ja tämä olisi isompi työ.

Tiili testin aikana huomasin, että mallin talojen UV mapit ovat väärin tehtyjä eivätkä näytä oikealta, joten tämä tulee korjata 3D ohjelmistossa. Delegoimme tämän toiselle työntekijälle ja annoin hänelle ohjeet mitä korjauksia pitää tehdä.
Toimistolle tuli iltapäivällä vierailijoita käymään ja tutustumaan toimintaamme. Maisema-arkkitehtiopiskelijat oli kutsuttu tutustumaan tuotteisiimme ja toimintaamme sekä katsomaan miten ja mitä teemme yrityksessä. Esittelin tuotetta vierailijoille noin 2.5 tunnin ajan ja vastailin heidän kysymyksiinsä. Kerroin tuotteen käyttötarkoituksista ja mahdollisista tulevaisuuden näkymistä alalla ja yleisesti siitä minkälaisia hyötyjä ala voi saada tämän tyylisistä tuotteista. Kerroin kuinka pelimoottoreita voidaan käyttää jo nyt arkkitehtuurin ja

maisema- arkkitehtuurin alalla ja mitä hyötyjä Virtuaalitodellisuudesta on. Opiskelijoiden kysymykset koskivat enimmäkseen teknistä toteutusta, mallinnuksen tarkkuustasoa ja realismia. Esittelin sekä Unreal että Unity virtuaalimallit ja laitoin ihmisiä testaamaan tuotteita ja haastoin ihmisiä kysymään ja kyseenalaistamaan tuotteen hyötyjä, jotta sain paremman kuvan siitä, miten opiskelijat näkevät tulevaisuuden alalla oppimansa kautta.

Päivä onnistui todella hyvin ja opiskelijoiden kysymykset olivat hyödyllinen työväline ohjelmiston suunnittelun ja jatkon kannalta.

18.4.2019

Tavoitteeni päivälle on saattaa valmiiksi virtuaalitodellisuus tuotteen ensimmäinen graafisen parannuksen osa Unity Virtuaalitodellisuus mallissa.

Aamulla viimeistelin kivien Meshin ja materiaalien testauksen ja Substancen perustutkimukset. Seuraavaksi aloitin tuomaan Unity pelimoottoriin Speedtree 8:sta ensimmäisiä toimivia malleja. Päädyimme yksinkertaistamisen vuoksi hakemaan yhden aiemmin ostetun mallin Speedtreen sivuilta ja avasin sen Speedtree 8 for Unity ohjelmaan. Malli oli polygoni määrältään riittävä ja tavoite oli viedä se Unityyn.

Tuonti Unityyn onnistui helposti, koska ostettu puumalli on tehty Unityyn sopivaksi Speedtreen toimesta. Tämä siis eroaa merkittävästi itse tehdyistä Speedtree 8 for Unreal puista siten, että toisessa editorissa tehdyt puut eivät siirry Unityyn ilman todella merkittävää lisätyötä.

Tämän jälkeen toin kanaalin varteen mallinnetun kivimuuri Meshin Unityyn ja muokkasin sen maaston muotoihin sopivaksi ja asettelin kivet paikalleen. Tästä tuli huomattavasti parempi visuaalinen ilme malliin. Tekemättä Meshiin jäi vielä LOD kartat. Lodit ovat Meshin eri tarkkuus tasoja, jotka kytketään päälle eri etäisyyksillä, sillä ei ole optimaalista piirtää Meshin kaikkia polygoneja koko aika. Mikäli Mesh objekti on kaukana pelaajasta ei kaikkia yksityiskohtia kannata näyttää, sillä se vaatii enemmän laskentatehoa ja lisäksi pelaaja voi nähdä paljon enemmän objekteja yhdellä kerralla. Tavoite on, ettei käyttäjä huomaa LOD grouppien vaihtumista korkeasta tarkkuudesta siirryttäessä matalampaan tarkkuuteen.

Seuraavaksi toin Unityyn tekemäni materiaalin Substance Designeristä importtina. Tuotuani materiaalin asetin materiaalin parametreistä sopivat UV tiling koordinaatit silmämääräisesti, jotta kiven materiaali on katsojasta nähden riittävällä tarkkuus ja pienuus tasolla

uskottavan kiven kuvaamiseksi. Lisäsin materiaalin kiviin, jotka toin aiemmin Unityyn ja säädin materiaalin kiillot ja muut parametrit kuntoon. Kivistä ja materiaaleista tuli melko hyvän näköiset lopulta. Lisäksi testailin, miten voidaan tehdä märkiä kiviä suoraan materiaalin parametrejä käyttämällä ja saimme hyviä tuloksia testeistä aikaan. Kun materiaalin Metallic mapin arvoa nostettiin mustasta harmaata kohti alkoivat kivet kiiltää kuten fysikaalisesti niiden kuuluisikin, muutin Smoothnes arvoa tasaisemmaksi ja vähensin Normal mapin voimaa ja sain aikaan lähes realistisen näköisen märän kivimateriaalin, jonka asetin alimmalle kivi riville, jotka ovat vesirajassa Virtuaalitodellisuus mallissa.

Päivän työt onnistuivat omasta mielestäni yli odotuksien, vaikka tekeminen onkin todella hidasta alkuun. Ohjelmien käyttö onnistui vaivatta ja iterointi mallien osalta osoittautui mahdolliseksi nopealla aikataululla.

20.4.2019

Aloitimme päivän mobiilipelien tekoa koskevalla palaverilla ja alkuvuoden katsauksella sekä yhtiökokouksella. Yhtiökokouksessa käsiteltiin aamulla mm. kuinka tällä viikolla tehdyt testit ovat edenneet ja tarkastettiin yrityksen suuntaa ja tulevia tuotteita loppuvuodelle. Samalla sovittiin päivän työtavoitteista ja toiminnasta tälle päivää.

Aloitimme päivän työparini kanssa rahoitushakemuksen kuvien valmistamisella. Teimme suunnitelman kuvien sisällöstä sekä siitä, mitä kuvilla halutaan viestittää. Tein rahoitushakemusta varten kaksi kuvaa, joilla näytetään, kuinka suunnittelemamme Virtuaalitodellisuus kontrolleri toimii. Toisessa kuvassa nähdään tosimaailmassa oleva henkilö, joka pitelee kontrolleria toimistossa ”First Person” kuvakulmasta. Toisessa kuvassa näkyy saman henkilön ”First Person” kädet, mutta käsien päälle on projisoitu ”Grid” tekstuuri osoittamaan kuvan tilanteen olevan virtuaalimaailmassa. Lisäksi käsiin on laitettu heijastusta ja sinistä pintaa osoittamaan virtuaalimaailmaa. Lisäksi kontrolleri on hieman erilainen kuin tosimaailmassa ja taustalla näkyy virtuaalimaailmaa kuvaava tausta.

Tavoite oli saada päivän aikana tehtyä mobiilipeliimme uudet valikkografiikat sekä ”Upgrade” mekaniikka. Upgrade mekaniikka jäi kollegalleni koodattavaksi ja annoin testaus apua ja näkökulmia päivän mittaan. Muuten keskityin grafiikoiden piirtämiseen ja muokkaamiseen valikoihin. Loin päivän aikana uudet versiot kaikista valikko grafiikoista vanhojen pohjalta. Tavoite oli pitää visuaalinen tyyli samanlaisena läpi kaikkien kuvien ja luoda käsin piirretyn näköinen tyyli.

Piirsin kaikkiin grafiikoihin mustan ääriviivan Wacom piirtonäytön avulla, jotta saamme ikonit nousemaan paremmin esiin taustasta. Tein grafiikat vanhojen päälle ja tallensin ne PNG muodossa Unityyn. Koska kaikki grafiikat ja napit ovat samassa tekstuuri tiedostossa, erotin jokaisen ikonin omaksi Spritekseen kuvasta käyttämällä Sprite Editoria. Sprite Editorissa asensin tekstuurin tilan “Multiple” tilaan, jotta voin erotella samasta kuvatiedostosta useita osia ja irrotin kuvasta jokaisen Spriten omaksi objektiksi pelimoottorin sisällä nimeten objektit selkeästi samalla. Jos tekstuurissa oli esimerkiksi symboli Pause nimesin sen nimellä UI_Button_Pause ja näin loin kaikille objekteille selkeän hierarkian Unityn sisällä. Tämä tehdään siksi, että näitä Spritejä tarvitaan jatkuvasti ja on helpompaa, kun kaikki asiat on nimetty samalla tyylillä kaikissa paikoissa.

Viikkoanalyysi

Viikon työtehtävät onnistuivat varsin hyvin ja opin todella paljon Substance ja Speedtree ohjelmien käytöstä. Selkeä hyöty oli dokumentaatioiden tarkasta lukemisesta ja aiemman viikon tieteellisten shader ja PBR tutkimusten teosta koska pystyin nyt viemään teorian suoraan käytännön tasolle. Substance ohjelman opettelu tulee olemaan tulevaisuudessa erittäin merkittävässä roolissa pelimoottorien kanssa toimimisessa.

Substance ohjelmistojen suurimmaksi hyödyksi osoittautui laaja tieteellinen pohja, jonka dokumentaatio tarjoaa käyttäjälleen. On äärimmäisen tärkeää ammatilliselle kehitykselleni, että ymmärrän kuinka pelimoottorien ja rendaus engineiden laskukaavat toimivat ja mihin tosimaailman fysikaalisiin tietoihin ne perustuvat. Heijastuskulmien ja heijastusten määrän laskeminen vaikuttaa suoraan materiaalien ulkonäköön sekä toimintaan. Itselleni isoimmat uudet hyödyt tulivat juuri sähköä johtavien ja johtamattomien materiaalien toiminnan opettelusta. Myös erilaisten projisointien ja 3D skannauksien tutkiminen oli erittäin hyödyllistä projekteja miettien. Lisäinfoa löysin lisäksi artikkelista (jmonkeyengine 2018.) jossa käydään läpi, kuinka Cubemapping toimii, sekä selitetään lähes samat asiat hieman eri tavalla.

LOD järjestelmän käyttäminen Virtuaalitodellisuus malleissa Unity ja Unreal pelimoottoreissa on hieman erilaista keskenään. Unity:n sisällä LOD:it luodaan suoraan 3D mallinnus ohjelmassa, joten se vaatii selkeästi lisäosaamista mallintajan taholta ja melko paljon työtä. Esimerkiksi Blender 3D Mallinnus ohjelmassa ei ole selkeää tapaa tehdä mallien laadun pienennystä. Tehtyjä 3D malleja voidaan plugineja käyttämällä muokata matalampaan polygoni tasoon, mutta tulokset voivat olla melko huonoja. Mallit tulevat usein pluginien käytön seurauksena epämuodostuneina. Unrealin erot näkyvät jälleen kerran siinä, että engineen on rakennettu sisään LOD järjestelmä joka auto generoi tarvittavat LOD-

mallit luotettavasti tarvittaessa. LOD järjestelmän suurin hyöty on aukeilla alueilla, joita Virtuaalitodellisuus tuotteemme sisältävät runsaasti. Järjestelmä on kehitetty niin, että pelimoottori mittaa käyttäjän etäisyyttä mallissa eri objekteihin ja vaihtaa 3D mallit lennosta matala polygonisempiin versioihin. Kun pelimoottorin ei tarvitse laskea jokaisen Framen aikana niin montaa polygonin piirtämistä rasittaa se vähemmän grafiikkaprosessoria ja saamme korkeamman frameraten aikaan.

Yhtiökokousten pitämisen merkitys kasvaa jatkuvasti, mitä pitemmälle yritystoimintamme kehittyy. Aikaisemmin pidimme yhtiökokouksia lähinnä siksi, että niitä lain mukaan tulee pitää. Nyt olemme saaneet varattua hyvin ajan yhtiökokoukselle ja koska käsiteltäviä asioita on jatkuvasti enemmän kuin aiemmin yhtiökokousten merkitys on suurentunut. Kokouksista pitää saada tulevaisuudessa vielä enemmän irti, joten olisi hyvä tehdä kokonainen vuoden perus agendojen läpikäynti seuraavassa kokouksessa ja päättää hyvissä ajoin etukäteen seuraavien kokousten päivät ja agendalistat. Tulevia yhtiökokous aiheita yrityksemme osalta ovat osakassopimuksen teko ja allekirjoitus, eli kuinka jaamme vastuut ja työtehtävät yrityksen sisällä, toimitilojen hankinta sekä pelin julkaisu. Vuoden aikana olemme saaneet mielestäni toiminnan kivasti käyntiin ja allekirjoitukseen tuleva julkaisu sopimus olisi mahtava lisä yrityksen tarinaan. Neuvottelut ja testit ovat edistyneet tähän asti hyvin mutta tekemistä riittää. Isoimmat oppimisen kohteet ovat tulleet ihan perus valikkojärjestelmien kanssa. Valikkoja voi tehdä monella tapaa mutta vain osa niistä on järkeviä ja toimivia tapoja.

Valikkojen suunnittelussa tärkeintä on pitää mielessä se, että sen kuuluu helpottaa käyttäjän toimintaa. Värisuunnittelu ja selkeästi esiin pomppaavat rakenteet helpottavat käyttäjän navigointia ja litteät valikko rakenteet ovat tärkeitä onnistuneen lopputuloksen kannalta. Aiemmin valikoiden ongelmana oli, että niiden käyttö oli hankalaa. Meillä oli useampia erityylisiä valikoita, jotka sisälsivät samat informaatiot kuin tälläkin hetkellä mutta useiden eri valikkotyylien takia pelaajan ei ollut helppo löytää haluamansa. Yksinkertaisimme päävalikon ja laitoimme tärkeimmät asiat korostus väreillä. Tällaisia asioita ovat aloita pelaaminen nappi, sekä kun pelaaja saa uusimman hahmon auki sen valikon nappi animoituu ja se vaihtaa väriään. Samalla laitoimme skaalauksen kuntoon kaikilla laitteilla niin, että pelaajan laitteesta riippumatta tyyli säilyy samana. Värien muutosten ja paremman designin avulla saimme pelistä huomattavasti helpommin lähestyttävän.

3.4 Seurantaviikko 04

Viikon tavoitteena on jatkaa Unity:n Virtuaalitodellisuus mallin parissa töitä ja tehdä merkittäviä parannuksia graafiseen tyyliin ja luoda kasvit mallin sisään uudella Speedtree 8

ohjelmalla sekä käyttää Substance ohjelmistoa luodaksemme materiaalit ja tekstuurit kasveihin ja muihin malleihin sekä selvittää mihin kasvit tulee laittaa mallissa realistisen ympäristön luomiseksi.

23.4.2019

Tiistai alkoi aamupalaverilla, jossa kävimme läpi edellisen viikon työtehtäviä ja edistymisiä sekä keskustelimme ensiviikon perjantain Plehatcon tapahtumasta ja työtehtävistä tapahtuman aikana sekä aikatauluista ja niiden rajoitteista. Keskustelun teemana oli selvittää, kuinka tuotteet saadaan esittelykuntoon varmasti ennen tapahtumaa molempien Virtuaalitodellisuus tuotteiden osalta sekä kaiken muun osalta mitä yrityksessä esitellään ulkopuolelle.

Aloitimme viikon kartoittamalla mitä virtuaalitodellisuus mallista vielä puuttuu Unityn osalta, jotta graafinen ilme on valmis esiteltäväksi. Tutkin CTS Shaderin toimintaa sekä Vegetation studio PRO lisäosaa Unity:n sisällä saadakseni paremman kuvan ohjelmistojen toiminnasta.

CTS Shader hoitaa siis Landscape Meshin osalta materiaalien hallinnan ja tekstuurien piirrot. Virtuaalitodellisuus mallien osalta on muutamia yleisiä tapoja luoda pelimoottorissa maa Meshit jonka päällä käyttäjä toimii. Kokemuksen liikkumisesta täytyisi tuntua realistiselta sillä muuten pelaaja saattaa voida huonosti. Ensimmäinen tapa on luoda perinteinen Mesh objekti, joka maalataan tekstuuri ohjelmasta kuten Substance Painterissa ennalta ja maan muodot luodaan 3D mallinnusohjelmassa juuri sellaisiksi kuin halutaan. Tämän tyylin hyödyt ovat siinä, että voidaan tehdä todella tarkkaa työtä ja miettiä miten mikäkin kivi tai kallionhalkeama laitetaan paikalleen. Realismi on todella tarkkaa, jos tekijällä on aikaa ja hän jaksaa nähdä vaivaa mallin tekemiseen ja maalaamiseen. Huonona puolena voidaan pitää työtavan hitautta ja raskautta. Toinen iso ongelma on, perinteisen Meshin ollessa kooltaan iso ei LOD järjestelmää voida käyttää helposti, jolloin pelimoottori renderöi täysikokoisen mallin, vaikka siitä on näkyvissä vain osa ja renderöinti tapahtuu aina parhaalla tarkkuus tasolla. Tämä lisää piirto käskyjen määrää merkittävästi ja hidastaa ohjelman toimivuutta polygonimäärän ollessa jatkuvasti korkea. Lisäksi huonona puolena voidaan pitää muutosten teon vaikeutta. Mikäli muutoksia joudutaan tekemään malliin, tulee ne tehdä ulkopuolisessa ohjelmassa samoin kuin tekstuurien ja materiaalien muutokset.

Toinen tapa tehdä pelimaailman Mesh objekti on niin sanottu Landscape Mesh suoraan Unityssä. Tämä Mesh luodaan käyttäen Unity:n omaa työkalua ja tämä yhdistettynä CTS Terrain Shaderiin voidaan luoda Landscape Meshin pintaan kasveja ja maalata eri mate-

riaaleja minne halutaan. Hyötynä on, että kaikki tehdään Unityn sisällä, jolloin ulkopuolisia ohjelmia ei tarvitse käyttää ja kaikki muutokset tapahtuvat Unityn sisällä. Myös ohjelman suorituskyky nousee, kun Landscape Meshistä voidaan renderöidä vain näkyvät osat, jolloin voimme tehdä LOD:it oikein. CTS Terrain Shader mahdollistaa myös erilaisten suorien tekstuuri karttojen heijastamisen mallin geometriaan. Näin saamme nopeasti luotua käveltävän pinnan virtuaalitodellisuus malliin, joka tuntuu luonnolliselta käyttäjälle. Huonona puolena on, ettei tämä tapa ole yhtä visuaalisesti vaikuttava, kuin alusta asti käsin tehty versio eikä materiaalien maalaus ole yhtä hyvän näköistä kuin erillisissä ammatti ohjelmissa. Ajan säästö on kuitenkin todella merkittävä ja siksi alusta asti tässä projektissa on käytetty Landscape Mesh tyyliä.

Vegetation Studio Pro toimii Landscape Meshin kanssa niin, että voimme luoda instansoituja kasvi Meshejä todella isoja määriä. Jos maastoon halutaan oikean maailman tuntua tulee maastossa olla pieniä yksityiskohtia, kuten varpuja ruohoja pieniä kasveja ja kiviä. Jos käytämme projektissa yhtä aikaa esimerkiksi kahta miljoonaa ruohonkortta yhteen kohtaan mallin aluetta, aiheuttaa se merkittävää suorituskyvyn laskua välittömästi. Jokaisessa ruohonkorressa voi olla 10 polygonia ja kun ruohonkorsia koko kenttään tulee kymmeniä miljoonia ja muita kasveja sama määrä ei pelimoottori selviä laskennasta. Instansoitu kasvisto toimii kuitenkin toisella tavalla. Vegetation studio Pron Instansoitu kasvisto piirretään yhdellä piirto käskyllä kahden miljoonan erillisen piirto käskyn sijaan ja se on optimoitu todella hyvin Unityn tarpeisiin. Näin voimme laittaa pientä kasvustoa ja kasveja paikalleen malliin ilman merkittävää tehon laskua. Kasvit eivät siis ole omia objektejaan eikä niiden valaistus ole oikeaoppisesti laskettu jokaiselle korrelle erikseen. Mutta koska hyödyt ovat riittävän suuret Instansoidut Meshit ovat loistava työkalu realismin luontiin.

Opittuani riittävästi CTS Shaderin ja Vegetation Studio Pron toiminnasta siirryin parantamaan Landscape Meshin päälle heijastettuja karttoja ja maalasin niihin paremmat Alpha maskit, jotta saamme paremmin otettua mallissa ne kohdat huomioon, joissa materiaalit vaihtuvat toisiin. Paranneltuani tätä kautta hieman graafista ilmettä lähdin tuomaan uusia puita Unreal mallista Unityyn ja tutkimaan Speedtree 8:n jo valmiiksi tehtyjä puita ja kuinka Unrealista voidaan tuoda puu Unityyn.

Speedtree 8 Unreal versiossa voidaan tehdä niin sanottu template tiedosto jo valmistuneesta puusta. Tämä template tiedosto on siirrettävissä Speedtree Unity versioon ja näin voimme luoda saman puun uudestaan Unityyn sopivaksi. Ongelmia tulee kuitenkin, kun template tiedoston mukana ei tule eri vuosimalleja eikä materiaaleja. Kaikki vuodenajat puu malleihin joudutaan kuitenkin tekemään uudestaan käsin ja sama koskee puiden ma-

teriaaleja. Tämä on erittäin hidasta joten on melko turha lähteä muokkaamaan vanhoja puita.

24.4.2019

Tavoite keskiviikolle oli aloittaa puiden tekoa ja saada aikaan kaikista malleista 3 eri kasvuvaihetta ja kaikista kasvuvaiheista 4 vuodenaika mallia. Ensimmäiseksi tuli kuitenkin katsoa paras tapa tuoda Unreal puut Unityyn ja tarkastaa mitä puita on valmiina ja mitä vielä tarvitaan.

Aloitin päivän tutkimalla mitä puita Unity virtuaalitodellisuus mallista löytyy nyt Speedtree 7 versioina ja kartoitin mitä ongelmia niiden käyttöön liittyy tällä hetkellä mallissa. Kartoitukseni osalta päädyin siihen, että puiden isoin ongelma on, että ne eivät vastaa realistista kuvaa siitä, miltä tosimaailman alueella näyttää. Puut ovat liian tuuheita eivätkä visuaalisesti riittävän korkeatasoisia malleja. Myös materiaalit ovat huomattavasti heikompi laatuisia kuin näkemäni Speedtree 8:n muutamat esimerkit.

Listasin nämä huomioni ja aloitin kartoituksen siitä mitä tarvitaan. Valitettavasti en tajunnut laittaa ylös mikä on prioriteetiltaan tärkein kasvi hankkia. Tämä olisi ollut fiksua, koska olisimme voineet varmistaa, että varmasti kaikki tulee ajoissa valmiiksi ja tärkeimmät kasvit valmistuvat ensin.

Aloitimme hortonomin kansa yhteistyön ja annoin ohjausta ja apua siitä, kuinka kasveja luodaan Speedtree 8:ssa oikeaopppisesti Unity pelimoottoriin käyttöön. Huomasin kartoitukseni aikana, että lähes kaikki jo tehdyt mallit Unreal versiota varten sisälsivät liian korkean polygon määrän. Tavoite on yleisesti pidetty, että yksi puu Unityssä sisältää noin 20 000 - 30 000 polygonia keskimäärin, jotta pelimoottori pystyy pyörittämään virtuaalitodellisuus mallia riittävällä FPS:llä. Aiemmin tehdyissä Unreal malleissa oli noin 650 000 polygonia puuta kohti tavoitellun 20 000 polygonin sijaan. Unreal Virtuaalitodellisuus malli ei sisällä yhtä paljon muita toiminnallisuuksia kuin Unity versio ja pyörii siksi kevyemmin raskailla malleilla.

Tutkittuamme aiemmin tehtyjä puita päädyimme ratkaisuun, ettei ole mahdollista muokata puista versioita, jotka vastaisivat toivottua polygoni määrää helposti, vaan on helpompaa luoda puut kokonaan alusta asti uudestaan Speedtree 8 Unity editorissa.

Neuvoin kuinka polygon määrä voidaan pitää pienempänä ja minkälaisia muutoksia mallinnus prosessiin tulee tehdä, jotta polygon määrä ei karkaa ylöspäin halutusta. Isoin

muokkaus on se, että aiemmin tehdyissä puissa jokainen lehti oli yksi polygoni. Näin jos puussa on 300 000 lehteä tarkoittaa tämä myös 300 000 polygonia. Tämä voidaan kiertää luomalla oksia, jotka ovat kuvia oksista ja kuva sisältää jo valmiiksi useita lehtiä. Näin laatu ei juuri kärsi silmin havaittavasti, mutta polygoni määrät saadaan tiputettua 300 000:sta 10 000:een.

Lisäksi tarkkuustasoa tiputettiin kaikissa osissa puissa ja tämä vaikutus negatoitaisiin tekemällä mahdollisimman hyvät tekstuurikartat ja niiden avulla mahdollisimman hyvät materiaalit puihin ja lehtiin.

Aloitin itse tekemään kartoitusta siitä mihin puut tulevat Virtuaalitodellisuusmallissa ja mitä muuta voidaan vielä tarvita graafisesti parhaan lopputuloksen saavuttamiseen. Puita tarvittiin lopulta 14 erilaista mallia eri vuodenaikoina käyttöön Unity moottorin virtuaalitodellisuus malliin.

25.4.2019

Tavoite oli saada ensimmäiset puut valmiiksi mallinnettua ja tehdä niihin vuodenajat sekä kartoitus mihin kaikki asetetaan mallin sisällä sekä eri vuodenaikojen materiaalien valmistusta lehtien ja rungon materiaalien osalta puihin.

Ensimmäisenä materiaalin teossa aloitin referenssikuvien haulla. Viime kesänä oli käyty kuvaamassa todellista paikkaa ja eri puita. Näistä kuvista sain löydettyä hyvät referenssit sekä melko hyvät kuvat. Kuvissa oli kuitenkin kovia varjoja sekä liikaa kiiltoa. Tämä kiilto piti ensin poistaa Photoshopissa.

Aloitin materiaalien teon kasveihin Substance Designer ohjelmalla yhteen puista ja tavoite oli saada aikaan graafi, jonka avulla voitaisiin tehdä kaikki muutkin puut helposti. Loin Substance Designer ohjelmassa Shader Graafin, jonka avulla voimme luoda käytännössä kaikista puista erilliset tekstuuri kartat. Tein lehti graafin, joka ottaa sisään toisesta inputista väri RGB kartan ja toisen mustavalkoisen lineaarisen värikartan. Värikartat ovat siis lehtien skannauksia tai kuvia, mutta mieluiten skannauksia korkeiden Gloss osien välttämiseksi. Jos kasveja ja niiden osia kuvataan kameralla skannaamisen siasta, on ongelmana, että kuviin tulee varjoja ja valoisia osia. Heijastukset originaali kuvissa ja varjot vaikeuttavat tekstuurien luontia merkittävästi. Tasainen värikartta kuuluu luoda siksi, että voimme käsitellä sitä pelimoottorissa haluamallamme tavalla. Epätasainen värikartta ei anna tätä mahdollisuutta, sillä jos värikartassa on jo valmiiksi varjoja ja pelimoottorissa

haluamme, että valo tulee toisesta suunnasta näyttää se väärältä. Varjo ei siis voi olla valon puolella.

Koska käytössämme olleet kuvat eivät olleet oikeaoppisesti skannattuja tekstuureja vaan ainoastaan valokuvia jouduin käsipelillä maalaamaan lehtien varjot piiloon ja samalla poistamaan kiillon kuvista. Kiiltojen ongelma on käänteisesti sama kuin varjojen kanssa. Käsin maalaaminen Wacom piirtopöydällä on tehokas tapa poistaa varjoja ja kiiltoja kuvista, mutta tarpeeton työvaihe sekä melko hidasta. Jatkossa teemme lehtien materiaalit kasveihin skannaamalla ja runkojen kuvaamiseen tulee käyttää useita diffuusi valoja, jotka sirottavat valon mahdollisimman tasaiseksi kuvaus hetkellä.

Sain melko hyvin poistettua epätasaisuuksia tekstuurien kuvista ja aloitin luomaan lehti graafin tekstuuri karttoja. Koska tekstuurit eivät mene suoraan Unityyn vaan Speedtree 8 ohjelman kautta tarkoittaa se samalla, että tarvitsin luoda aivan erityylisiä karttoja kuin aiemmin oli ajatuksissa. Normaalisti tehdään diffuusi/albedo, normaali, height, roughness/smoothness, Ambient oclusion ja oclusion kartat mutta koska Speedtree tarvitsi lisäksi vielä Subsurface scattering ja subsurface scattering amount mapit sekä muutamia tarkentavia karttoja vaati se graafilta hieman lisää työtä.

Otin Speedtreen kaupasta valmiiksi tehdyn puun tekstuuri kartat auki ja siirsin ne Substance Designer ohjelmaan ja käytin niitä väri ja ulkonäkö referenssinä itselleni, mutta en voinut käyttää niitä suoraan puun osien tekoon. Luodessani uusia väritekstuurikarttoja koitin katsoa miten Speedtree oli luonut kartan ja luoda itse parametrisen järjestelmän, jonka avulla voisin luoda saman näköisiä värikarttoja, jotka olisivat mahdollisimman saman tyylisiä. Jatkossa minun tulee tutustua paremmin, kuinka Speedtree 8 käyttää karttoja ja miten PBR materiaalit lasketaan ohjelman sisällä, mutta ajan säästämiseksi tein silmämääräisesti samantyyliset värikartat. Tämä lähestymistapa ongelmaan vaikutti tuottavan melko hyvän lopputuloksen ja saimme siistejä lehtiä ja runkoja aikaisiksi melko hyvin.

Tekemäni graafin kautta pystyn siis nyt tuottamaan uusia kasveja todella nopealla syklillä, se mikä normaalisti Photoshopissa vie tunnin vie nyt 5 minuuttia Substance Designerin kautta. Tämä muutos on ehdottomasti ohjelman kuukausimaksun arvoinen.

Hortonomimme sai tehtyä ensimmäisen kasvin rungon valmiiksi yhden vuodenajan lehtineen ja polygoni määrä oli lähelle oikeaa mitä halusimme ja hän siirtyi tekemään seuraavaa mallia samasta kasvista eri koossa. Itse siirryin parantamaan optimointia kyseiseen malliin poistamalla juurista useita tuhansia polygoneja, joita pelimoottori ei kuitenkaan piirrä ja optimoin myös rungon polygoni määriä. Aloitin Substance designerissa tekemieni

eri karttojen asettamisen malliin, jotta saisimme tehtyä tieteellisesti oikean näköisiä puita Unityyn pian valmiiksi.

26.4.2019

Tavoite on saada kaikki puumallit valmiiksi ja tehtyä niihin vuodenajat tämän päivän aikana.

Hortonomimme oli saanut tehtyä kaksi neljästä tehtävästä kasvista lähes valmiiksi ja kaikkiin yhden vuodenajan ja kolme eri kasvuvaihetta kustakin. Tehtäväkseni jäi viimeistellä edellisen päivän ensimmäinen puu ja tästä opitun perusteella luoda muiden puiden materiaalit Speedtree 8 ja exportata tehdyt puumallit Unityyn.

Jatkoin karttojen laittoa eilen aloittamani puun lehdille ja varmistin, että materiaalit toimivat oikein. Puun runko pysyy käytännössä samana läpi vuoden Virtuaalitodellisuus mallissa. Keväällä lehdet ovat hieman pienempiä ja vaaleampia, kesällä lehtiä on enemmän ja ne ovat kookkaampia kuin keväällä ja niiden väri on tummempi, Syksyllä lehtiä on taas vähemmän ja ruskan värjäys tulee mukaan ja talvella lehtiä ei näy.

Speedtree 8 toimii siten, että puun runkoon liitetään lehti. Lehti sisältää materiaalin ja materiaali koostuu tekstuuri kartoista. Materiaaleista voi tehdä materiaali settejä, kuten kesä, kevät, syksy ja jokaiselle setille voidaan määrittää graafinen käyrä, milloin mikäkin vuodenaika on päällä. Esimerkiksi jos kesä setillä on tumman vihreät lehdet ja vuodenaika määritetään x ja y koordinaatistossa piste järjestelmän kautta. X on tässä koordinaatistossa vuoden aika ja Y on lehtien määrä, joka piirretään puuhun. Esimerkiksi kesän lehdet merkataan X koordinaatistossa ajalle kesä - elokuu ja Y on kasvava ja käänteisesti laskeva käyrä sen suhteen kuinka lähellä kesän alkua ja loppua ollaan. Suurimmillaan Y arvo on luonnollisesti keskikesän kohdalla. Syksy taas omaa arvot syksyn huippu pisteessä ja alkaa hieman ennen kesän loppua suurentamaan omaa Y arvoaan. Päällekkäisyyksien avulla saadaan kerrottua Speedtree 8 ohjelmalle vuodenaikaa säätäessä esimerkiksi milloin ohjelman tulee laittaa syksy vuodenaika päälle ja vihreiden lehtien määrä alkaa vähentyä ja punaisten lehtien määrä kasvaa.

Kun materiaali setit on säädetty kohdalleen, voidaan vielä lisäksi kasvin parametreistä käydä laittamassa lehden kokoa ja muita asioita mieleiseksi jokaiselle vuodenajalle erikseen. Kun olin saanut ensimmäisen puun materiaali setit kohdalleen ja mieleisekseni oma tekemilläni Substance Designerillä tehdyillä tekstuuri kartoillani, exporttasin puun Unity pelimoottoriin.

Export toiminnossa tärkeää on exportata LOD mallit sekä billboard malli ja tekstuurit wrap around modessa, jotta saamme parhaan lopputuloksen Unity:n sisällä. Kun puu on onnistuneesti exportattu kaikissa kasvu vaiheissaan ja jokainen vuosimalli erikseen tuotu mooottoriin meillä on tästä yhdestä puusta käytettävissä 3 eri kasvuvaihetta sekä jokaisesta kasvuvaiheesta eri lehtiversiot vuodenaikojen mukaan. Näistä versioista kootaan Unityn sisällä prefab tiedosto, jota pelimoottoriin kirjoitettu koodi käsittelee vuodenaikojen vaihtuessa Virtuaalitodellisuus mallissa. Kun vuodenaika vaihtuu Virtuaalimallissa, koodi valitsee mikä lehtiversio näytetään puusta. Esimerkkinä jos kyseessä on syys aika poistaa koodi kaikki muut 3 vuodenaikaa näkyvistä ja lisäksi moottori katsoo mikä pituus versio kuuluisi olla näkyvillä.

Tehtyäni yhden puun oli toisten puiden teko vain saman toistoa ja iltaan mennessä minulla oli kolme neljästä mallista eri vuodenaikoineen tehtynä. Huomasin ikäväkseni, että muutamissa puissa oli edelleen aivan liikaa polygoneja, koska taas kasvien lehtiä oli tehty yksitellen eikä clustereina. Tämä tulee korjata ensi viikolla.

Lisäksi kävin videomateriaalia ja referenssi kuvia läpi todellisen maailman paikasta, jotta sain paremman kuvan missä mikäkin puu oikeasti sijaitsee. Tutkin satelliittikuvia saaresta ja aloin laittamaan pisteitä niihin paikkoihin mihin arvioihin puiden kuuluvan sekä väri koodasin puut eri väreillä karttaan, jotta osaan sijoittaa mallit Virtuaalimallissa samoihin paikkoihin. Aikaisemmassa virtuaalitodellisuus mallissa oli summittain sijoitettu kasveja ympäriinsä.

Onnistuimme saamaan lähes kaikki mallit valmiiksi, ensi viikon maanantaihin jäi muutaman puun loppuun saattaminen ja paikalleen laitto Unityssä.

Viikkoanalyysi

Viikko onnistui erittäin hyvin ja pystyin viemään asioita käytäntöön, joita opin viikon aikana Speedtree 8 ohjelmasta sekä Substance ohjelmistoista. Muutamia haasteita aiheutui työryhmän hieman vaillinaisen pelimoottori kokemuksen takia. Pelimoottorien suorituskyvyn optimointi ja moottoreiden rajoitteet eivät olleet kaikille tuttuja ennestään. Opin parempia työtapoja työskennellessäni, jotka vien käytäntöön yrityksen sisäisen dokumentaation myötä.

Dokumentaatioiden kirjoittaminen osoittautuu äärimmäisen tärkeäksi osaksi työskentelyä, kun tiimiin otettiin lisää ihmisiä auttamaan. Nyt ongelmaksi osoittautui selkeästi, ettei rele-

vanttia tietoa ollut löydettävissä yrityksen sisäisistä kanavista helposti esitetyllä tavalla vaan parhaat toimintatavat olivat ikävä kyllä ihmisten takana. Tämä tulee jatkossa korjata, sillä on täyttä ajanhukkaa, jos dokumentaatiota ei ole tehty. Oikein dokumentoidut työtavat ovat helposti ja nopeasti monistettavissa seuraaviin projekteihin tai jos työryhmän jäsenet vaihtuvat. Dokumentaation teko helpottaa myös esimiesten kykyä arvioida projektin etenemistä aikataulujen mukaan. Näkisin, että vähintään viikoittainen dokumentaatio teknisesti sekä jonkinlainen viikon työtehtävien kirjaaminen helpottaisi paljon yrityksen toimintoja. Jokaisella yrityksellä on myös sen tarpeisiin sopivat parhaat käytännöt, jotka tulisi olla kaikkien tiedossa, sillä etenkin pelimoottoreiden kanssa asioita voidaan tehdä monella tapaa. Mahdollisten ongelmien välttämiseksi nimenomaan yhteisesti sovitut tavat ja niiden jakaminen ja selittäminen auki on helppoa onnistuneen dokumentaation kautta, jota pidetään yllä ja päivitetään tekniikoiden kehittyessä. Itselläni on runsaasti varaa kehittyä dokumentaatioiden teossa ja oma ongelmani lienee kiireessä se, että en jaksa enää lähteä tekemään lisäkirjoitus töitä. Tähän ratkaisuna näkisin, että jokaiseen viikkoon on budjetoitu tunti tai kaksi dokumentaation tekoa ja dokumentaatiota vaadittaisiin yrityksen taholta työntekijöiltä.

Tiimityön parhaat työtavat ovat myös nousseet selkeäksi teemaksi viikon aikana, kun ihmisten tietotaidossa pelimoottoreiden osalta on eroa. Samoin myös oma ymmärrykseni kasveista on osoittautunut varsin puutteelliseksi. Jokaisella tiimin jäsenellä on oma erikoisosaamisensa ja näiden osaamisten yhteistyöllä meillä on mahdollisuus valmistaa paras mahdollinen tuote. Tärkeää onkin juuri saada yhteistyö toimiaan saumattomasti yhteen. Oma näkemykseni on, että aiemmin hoidettu oikeaoppinen dokumentaatio ja parhaiden työtapojen kuvaus auttaisi tässä. Ei ole järkevää suinkaan, että kaikki tiimin jäsenet opettelevat pelimoottoreista tai kasveista kaiken vaan pienten työryhmien välinen kommunikaation onnistuminen takaa hyvän lopputuloksen. Teimme mielestäni älykkäästi viikon kuluessa, kun teimme nopean teknisen määrittelyn ja raamit mm. mallien polygoni määrille ja tekniselle suorituskyvylle. Tämä tekninen määrittely vähensi huomattavasti kysymysten määrä ja antoi hortonomillemme paremmat lähtökohdat Unity:n Speedtree 8:n mallien valmistamiseen. Tekninen määrittely tulee dokumentoida ja kirjata ylös kohta kohdalta, jotta muutkin tiimit osaavat rakentaa toisiinsa sopivia 3D malleja jatkossa.

Realistisen Virtuaalimaailman luonti pelimoottorin sisällä on melko haastavaa. Ihminen on äärimmäisen hyvä tunnistamaan todellisuutta jäljittelevässä ympäristössä miksi asiat eivät näytä oikealta. Tästä syystä monissa Virtuaalituotteissa ei tähdätä realismiin vaan tyyliteltyyn esitykseen tosimaailmasta. Graafisesti tyylitelty, ei fotorealistisen näköinen versio ei saa käyttäjää kokemaan, ettei hän olisi osa ympärillä näkyvää Virtuaaliympäristöä. Sen sijaan Virtuaalimalli, joka tähtää täydelliseen fotorealismiin eikä onnistu siinä saa aikaan

tunteen käyttäjässä siitä, että jokin on pahasti vialla. Tätä kutsutaan peleissä ja muissa kulttuuri tuotteissa Uncanny valley ilmiöksi. Ilmiö on usein liitetty ihmishahmoihin mutta toimii lähes kaikissa graafisissa osa-alueissa teoriana. Mitä realistisemmalta asiat näyttävät sitä oudommalta luonnosta poikkeava näyttää. (Toward data science 2019.) Etenkin virtuaalitodellisuudessa, joka simuloi näkö ja kuuloaistin kautta siirtymistä toiseen paikkaan ja aikaan ja huijaa käyttäjää kokemaan epätosia asioita voi käyttäjä kokea selittämätöntä pahan olon tunnetta, jos maailma ei tunnu luonnolliselta. Käyttäjä ei siis välttämättä aluksi tiedosta mistä outo tunne johtuu mutta hän alitajuisesti kuitenkin havainnoi ympäristöään. Jos virtuaalimallin puiden lehdet eivät heilu tuulessa, maassa ei ole ruohoa tai pieniä kiviä, hyönteiset ja eläimet puuttuvat tuntuu muuten realistisen näköinen maisema staattiselta ja luotaan työntävältä. Tärkeää onkin virtuaalimallin tuotteissa siis joko tuoda riittävä määrä aistiärsykkeitä malliin mukaan tai sitten tyylitellä maailman audiovisuaalinen ilme niin, ettei se simuloi todellista maailmaa sellaisenaan vaan on jonkinlainen karrikatyyri siitä.

3.5 Seurantaviikko 05

Viikon tavoitteena on saattaa Virtuaalitodellisuus tuotteet valmiiksi perjantain PlehatCon tapahtumaa varten ja jakaa työtehtävät tapahtumaan. Perjantaina tuotteet esitellään isommalle alalla olevalle yleisölle.

29.4.2019

Tavoite päivälle oli selvittää, kuinka Speedtree 8:ssa saadaan tehtyä vähemmän polygoneja sisältäviä puita muutamista kesken jääneistä puista ja kuinka kasvien oksista saadaan tehtyä clustereita ja kuinka clusterit toimivat Speedtree 8:ssa ja Unityssä.

Viikkoa aloitettiin maanantai palaverilla, jossa käytiin läpi kuluneen viikon asiat ja edistymiset yleisesti. Palaverin aikana käytiin läpi viikon aikatauluja PlehatCon tapahtumaan valmistautumisesta ja käytiin läpi, kuinka lähellä virtuaalitodellisuus tuotteet ovat maalejaan. Minun tulee kuluvan viikon aikana osallistua molempien virtuaalitodellisuus tuotteiden valmistamiseen ja loppuun saattamiseen, jotta tuotteet ovat varmasti esittelykunnossa.

Yleisen viikkopalaverin jälkeen siirryttiin tekniikka palaveriin, jossa katsottiin, kuinka paljon kullakin työryhmän jäsenellä on vielä teknistä tekemistä omien osiensa kanssa. Tärkeää on, että perjantaina ollaan valmiita esittelemään tuotteet.

Palaverin jälkeen aloitin Speedtree 8:n kanssa työskentelemään puiden oksa Clustereiden kanssa. Tällä hetkellä muutamissa puissa oli vielä ongelmia lehtien materiaalien ja polygoni määrien kanssa. Liian iso polygoni määrä aiheuttaa tarpeetonta rasitusta grafiikan piirrossa, joka näkyy mm. laskevana ruudun päivitysnopeutena. Etenkin VR tuotteissa alhainen Frame Rate aiheuttaa käyttäjälle ongelmia ja mahdollisesti pahoinvointia. Alhainen Frame Rate aiheuttaa käyttäjälle Virtuaalitodellisuudessa tunteen siitä, että oma pää ei pysy mukana välikorvan tuottamissa aistimuksissa, kun käyttäjän silmät eivät aisti kuvia riittävän nopeasti. Tästä syystä pyrimme aina pääsemään mahdollisimman korkeaan Frame Rateen minimoidaksemme pahoinvointia käyttäjillä. Istuminen voi auttaa joitain käyttäjiä pahoinvoinnin hallitsemisessa, kun tasapaino elin ei joudu samaan aikaan niin paljoa töihin. Näin etenkin käyttäjillä, jotka eivät ole tottuneet VR laitteisiin.

Puiden clustereita ei oltu aikaisemmin tehty ohjelman uudessa versiossa, joten ensin oli tarpeen selvittää kuinka ne toimivat Speedtree 8:ssa. Osaan puista oli laitettu lehdet siten, että jokainen puun lehti oli laitettu runkoon ja oksiin kiinni omana yksilönään johon tuuli ja valot vaikuttavat. Tämä on graafisesti realistisin ja kaunein tapa tehdä malleja pelimoottoreiden käyttöön. Realistisesti käyttäytyvät lehdet ja kasvit näyttävät erittäin hyvältä katsojan silmään Virtuaalitodellisuudessa mutta miinuspuolena huomattava määrä pelimoottorin laskettavia tuulessa heiluvia lehtiä aiheuttaa tehon laskua. Yhdessä kasvissa voi olla kymmeniä tuhansia lehtiä ja kasveja satoja yhdessä mallissa. Edellä kuvatussa tavassa jokainen puun oksa ja lehti ovat omia täysin 3D renderöityjä objektejaan omine polygoniverkkoineen ja jokainen osa ja sen liike lasketaan erikseen pelimoottorissa.

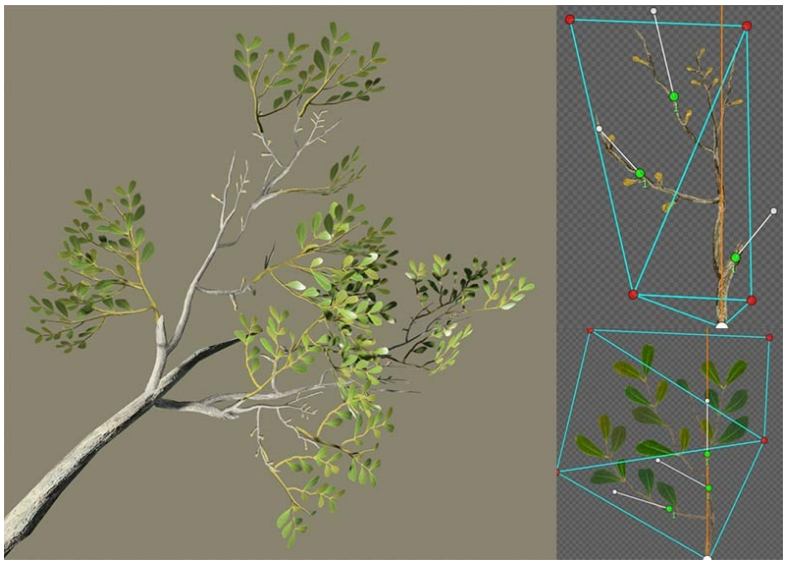
Clusteri järjestelmässä (Kuva Cluster_01) kasvin pienimmät oksat ja lehdet korvataan kuvilla. Isot oksat pysyvät edelleen 3D malleina ja sisältävät isomman määrän polygoneja mutta pienemmät oksat muuttuvat yksinkertaisiksi kuvien heijastuksiksi. Harmaalla pohjalla (Kuva Cluster_01) näemme presentaation Clusterista ja vasemmalla puolella kuvaa näkyy kuinka Clusteri kiinnittyy 3d Mallinnettuun oksaan. Tämän järjestelmän avulla puun graafinen laatu ei juuri kärsi merkittävästi mutta polygon määrä tippuu parhaimmillaan 250 000 polygonista 30 000 polygoniin, kun jokainen lehti ei enää ole oma 3d objektinsa. Yksi clusteri sisältää siis useita lehtiä sekä oksan ja käyttäytyvät yhtenä kokonaisuutena. Tarkkaan katsomalla käyttäjä huomaa milloin kyseessä on clusteri ja milloin kyseessä on 3D mallinnettu kasvin osa, kun Virtuaalimalli simuloi tuulta kasvimallien oksistoissa. Tämä ei kuitenkaan ole merkittävä haitta sillä pääosin lopputulos on hyvätasoinen ja tekemässämme tuotteessa tärkeintä ei ole täysi fotorealismi vaan funktionaalisuus yhdistettynä kauniiseen visuaaliseen ilmeeseen.

Speedtree 8:ssa Clusterit luodaan tekemällä kokonaan uusi kasvi, joka koostuu yhdestä oksasta, johon lehdet laitetaan kiinni. (Kuva Cluster_01, harmaa pohja). Clusterista tehdään eri vuodenaikamallit erivärisine ja kokoisine lehtineen, aivan kuten normaalipuussakin, jotta vuodenaikoja voidaan vaihtaa tarpeen vaatiessa virtuaalitodellisuus mallissa. Kun kasvin yksi oksa on luotu 3D malliksi ja se näyttää hyvältä siihen laitetaan aiemmin puussa olleet lehdet. Tässä kohtaa Clusterimalli ei siis sisällä yhtään vähempää ja voi sisältää itseasiassa enemmän polygoneja kuin alkuperäinen puumallin oksa. Polygoni määrän vähentäminen tapahtuu, kun Clusteri tuodaan varsinaiseen puuhun mukaan.

Kun olin tyytyväinen Clusterin ulkonäköön ja muotoon tein siitä eri vuodenaikoja varten mallit eri värisin lehdin ja erimäärillä lehtiä. Kun kaikki 4 versiota olivat valmiita, tämä Cluster oksa laitetaan 3D mallina 2D projisointiin suoraan ylhäältä päin Speedtree 8:n sisällä. Speedtree 8:n sisällä 3D mallinnetusta oksasta otetaan suoraan ylhäältä kuva, tähän kuvaan luodaan polygon verkko, joka on näytetty kuvassa 1 sinisinä viivoina. Clusterin 3D malli sisältää yleensä noin 30 000 polygonia itsessään ja luomalla yksinkertaisen 2D projisoinnin oksasta ja yksinkertaisen polygon verkon, tippuu oksan polygon määrä noin 3 - 5 polygoniin.

Kun Clusterin 2D projisointi ja uusi matalapolygoninen oksa malli on valmis, tuodaan Clusteri alkuperäiseen puuhun ja korvataan pienet oksat Clusteri polygoneillä. Polygoneja voidaan muuttaa vielä 3D tilassa hieman, jotta niihin saadaan vaihtelevuutta x,y ja z suunnissa tarpeen mukaan. Näin Clusterit eivät tunnu toistavan itseään.

Toin kaikki 4 eri vuodenaika Clusteria puihin ja korvasin vanhat korkea polygoniset pienet oksastot niillä. Tein Speedree 8:ssa vuodenaika mallit clusterit sisältäville puille, jotta voimme exportata mallit Unity pelimoottoriin toimivina malleina eri kasvuvaihe malleina.



Kuva1

30.4.2019

Tavoite oli saada saada puut täysin valmiiksi puuttuvista lajeista kaikissa kasvuvaiheissa ja clustereilla eri vuodenaikoina Unityyn, jotta loppuviikosta voidaan keskittyä muihin tärkeisiin asioihin.

Jatkoin eri korkuisten 3d mallien parissa työskentelyä ja tein puista eri vuosimallit kaikissa vuodenajoissa Virtuaalitodellisuus mallia varten. Valitettavasti huomasimme, että puiden eilen tehdyt oksat olivat pilalla mallin tekstuurissa olevien väriongelmien takia. Jouduin siis luomaan materiaalit kokonaan uusiksi Substance ohjelman avulla puihin, jotta ne olisivat yhteneviä toistensa kanssa.

Lisäksi Opacity mapit aiheuttivat lehtien reunoihin artifakteja, koska normal mapit eivät olleet oikean kokoisia ja olivat skaalautuneet väärin. Korjasin arfitaktit kutistamalla Opacity mappia hieman Photoshopin avulla ja sain lehdet näyttämään hyvältä jälleen. Lehtien tekstuurien ja materiaalien korjaaminen oli suurimmaksi osaksi aikaa vievää toistoa, joka kuitenkin jouduttiin tekemään käsin jokaiselle kasvumallille ja vuodenajalle erikseen export toimintoa varten. Korvasin kaikkien puiden pienet oksat Clustereilla ja sain aikaan toimivia ja matala polygonisia puita.

Exporttasin puut unityyn ja generoin kaikista puista 4 eri LOD tasoa, joista LOD 4 on billboard malli puusta. Tuotuani puut Unityn sisään loin niistä uudet Prefabit ja korvasin virtuaalitodellisuus mallin käytetyt prefabit uusilla. Loin puut suoraan vanhan prefabin päälle ja tallensin prefabin uudella nimellä näin saaden aikaan edelleen toimivan järjestelmän mut-

ta vaihdetuilla malleilla. Samalla huomasin, etteivät LOD tasot mene päälle, koska käyttäjä ei koskaan mennyt riittävän kauaksi kasveista jotta Unity alkaisi vaihtamaan puita matala resoluutio versioihin. Vaihtamalla Unity pelimoottorin asetuksia on mahdollista säätää milloin LOD tasot kytkeytyvät päälle ja pois mutta jätin tämän seuraaviin päiviin tehtäväksi.

Keskipäivällä oli toimistokokous, jonka aiheena oli perjantaina järjestettävä PlehatCon. Kokous kesti noin 1.5h ja kävimme läpi mitä päivän aikana tapahtuu, mitä vastuita eri ihmisille jaetaan ja miten asiat etenevät päivän aikana.

Kokouksen jälkeen jatkoin Unreal Enginen Virtuaalitodellisuus mallin parissa. Unreal versiossa oli aiemmin tehty ainoastaan Talvi ja kesä vuodenajat toimimaan. Koodi oli melko puutteellisessa tilassa, vaikka toimikin tässä tapauksessa halutulla tavalla. Nyt kuitenkin tarve oli lisätä kaksi uutta vuodenaikaa kevät ja syksy mukaan malliin kasvien ja ympäristön muutoksineen.

Aloitin avaamalla tekemäni visuaalisen scriptauskielen koodin ja tutustuin noin puolituntia siihen, kuinka olin aikaisemmin koodin tehnyt. Koodin ongelmana oli se, ettei se tulisi hyäksymään lisää vuodenaikoja, koska tällä hetkellä se katsoi vain, onko Virtuaalitodellisuudessa menossa talvi tai kesä ja asetti sen mukaan asiat päälle tai pois peliobjektien tagien mukaan.

Ongelmana on siis, että jos jokin objekti olisi samanlainen keväällä ja kesällä ei järjestelmä pystyisi laittamaan niitä halutusti pois ja päälle tai sitten jokaisesta objektista pitäisi olla jokainen mahdollinen permutaatio olemassa eri tageillä. Tämä ei kuitenkaan ole järkevä tapa tehdä asiaa, joten oli fiksumpaa muuttaa koodi toimimaan oikeaoppisesti vuodenaikojen kanssa.

Ensimmäisenä toin kaikki vuodenajat Unreal Enginen karttaan päällekkäin, jolloin kaikki kasvit ja ympäristöt ovat oikeissa kohdissa riippumatta vuodenajasta. Sitten tein koodin, joka katsoo kuukausi numeroiden mukaan mikä vuodenaika on kyseessä ja piilottaa tarpeettomat peliobjektit näkyvistä. Käytin olemassa olevaa Enum- luokkaa, jonka olin tehnyt ja kutsuin sen arvon koodissa tarkastaakseni mikä kohta on menossa.

Aikaisemmin asetin talven ja kesän pois käyttämällä monimutkaisia koodi kokonaisuuksia, joissa jokainen kasvi katsottiin yksitellen saadaksemme selville mitä vuodenaikaa kukin edusti. Loin uudet tagit moottoriin vanhojen rinnalle, jolloin edellisen Summer ja Winter tagien rinnalle tuli Spring, Fall ja NotWinter. NotWinter tagi sisältää objektit, jotka ovat aktiivisina, kun kyseessä ei ole talvikausi virtuaalitodellisuus mallissa.

Kirjoitin 5 uutta funktiota, jotka asettivat kasvit joko näkyviin tai pois näkyvistä tagien mukaan. Aiemmin tein jokaisen muutoksen suoraan pääkoodissa, mutta selkeyden vuoksi päätin tehdä jokaisen kokonaisuuden omaksi muuttujakseen.

Tarkastettuani Enum- muuttujan arvon pääkoodissa asetin neljä eli vaihtoehtoista haaraa koodiin, joissa asetin peliobjektien tagien mukaan eri asioita pois ja päältä. Koodista tuli selkeän näköistä ja toimivaa uusien funktioiden avulla. Tarkistin ja bugi testasin koodin ja nyt kaikki vuodenajat menivät oikein päälle poistaen näkyvistä väärät asiat. Esimerkiksi lumikasojen ei virtuaalitodellisuus mallissa tule näkyä keväällä, kesällä eikä syksyllä, joten jos mikään noista vuoden ajoista on aktiivisena lumikasat ovat poissa päältä käyttäen NotWinter tagia ja talvella taas vihreä nurmi menee pois päältä koska Winter tag on päällä. Tämän muutoksen kanssa yhtäaikaa tarkastetaan samassa pääkoodin kohdassa mitä muuta näytetään. Esim syksyllä lumikasat ovat poissa koska NotWinter tagia ei näytetä mutta samalla kuitenkin puiden lehtien tulee tippua puista ja muuttaa värinsä ruskan värisiksi.

2.5.2019

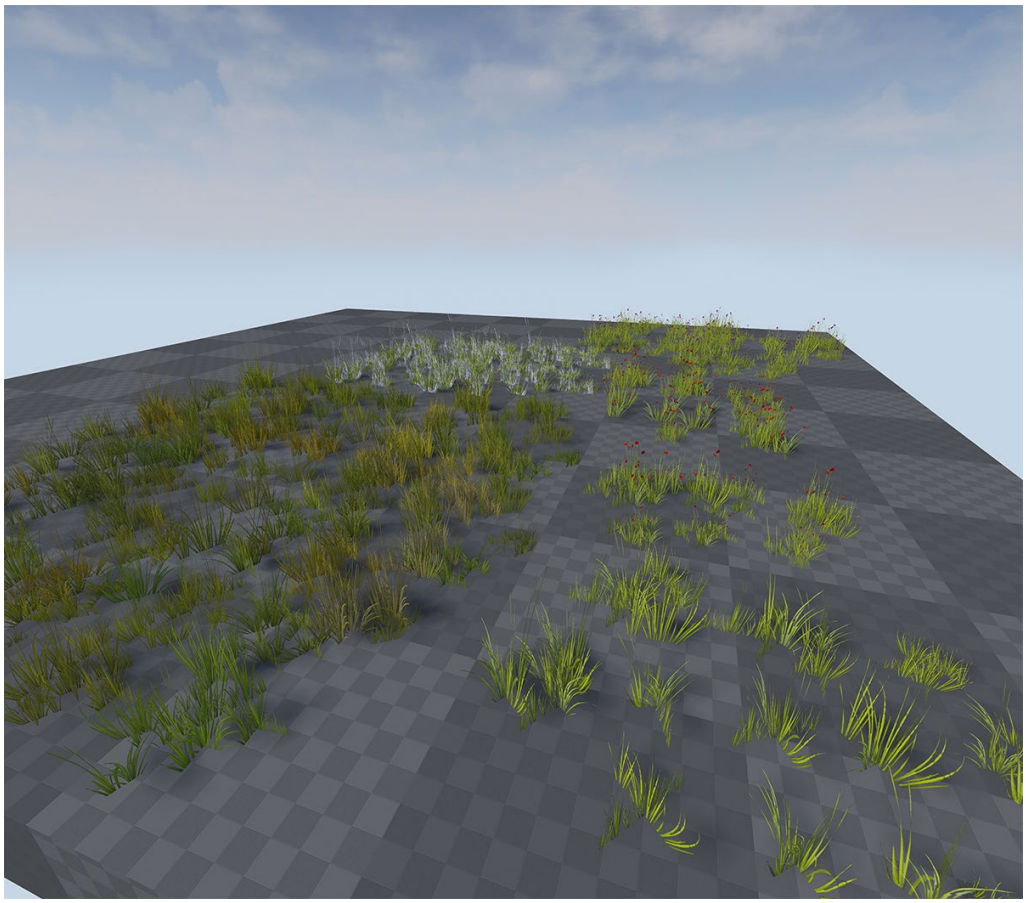
Tavoite oli saada loppuun Unreal malli perjantaita varten koodin ja toimivuuden osalta ja tehdä viimeiset optimoinnit Unity version graafiseen puoleen.

Unreal Engine mallin käsittely jatkui heti aamusta hortonimme avustuksella, joka oli listannut mitä ongelmia mallissa vielä oli ja mitä koodia ja toiminnallisuuksia piti korjata.

Edellisenä päivänä kirjoittamani koodi hoiti 3D mallien käsittelyn virtuaalitodellisuus mallissa mutta ei tehnyt mitään Unrealissa instansoiduille 3D malleille eli foliage järjestelmän generoimille malleille. Foliage järjestelmän 3D mallit eroavat siis merkittävästi yksittäisistä kasveista pelimoottorissa. Jokainen yksittäinen 3D malli voi olla erilainen toisistaan ja sisältää dataa ja omat sääntönsä kutenTagit siitä, milloin se on päällä ja milloin piilossa tai minkä korkuinen kasvi se on eri vuosina. Huonona puolena 3D malleissa on raskas piirtokäskyjen määrä. Tätä varten on luotu kevyt instansoitujen 3D mallien järjestelmä Foliage System.

Foliage 3D mallit piirretään yhdellä piirtokäskyllä ja ovat toistensa kopioita toisin kuin täysiveriset 3D mallit, jotka voivat sisältää omaa dataansa. Hyvä puoli foliagessa on siis, ettei se vie läheskään yhtä paljon suorituskykyä kuin normaalit 3D mallit mutta huonona puolena on, ettei mallin parametreihin päästä käsiksi, koska ne ovat instansoituja. Foliagea

käytetään pääosin pienten kasvien ja puskien tekoon täyttämään pelimaailman isoja tarkkoja 3D mallien välisiä alueita, jotta saamme luotua uskottavan 3d ympäristön kopion todellisesta maailmasta. (Kuva2)



Kuva2, foliage käytössä Unreal Enginessä, jokainen ruohonkorsi on osa foliage järjestelmällä luotua kasvillisuutta

Tein uuden Blueprint koodin muutoksen Foliage Actor järjestelmän sisään, johon hain vuoden aika Enum- muuttujan ja asetin sen vaihtamaan kaikki foliage actorit tiettyyn toiseen malliin. Hortonomimme oli tehnyt aiemmin kaikki vuodenaika ja kasvumallit kaikista pienistä kasveista. Tehtäväkseni jäi kirjoittaa foliage koodi, joka vaihtaa tiettyinä vuodenaikoina tietyt mallit käyttöön. Koska tageja ei voida hyödyntää Instansoiduissa 3D malleissa piti järjestelmä kirjoittaa eri tavoin. Tein jokaiselle 6:sta eri Foliage mallista oman blueprintin, joissa käskin pelimoottoria vaihtamaan esim. kesämallin pienen foliage puskat syys mallin punalehtisiin versioihin mallista ja samalla tarkastamaan mikä kasvuvaihe oli kyseessä. Normaalisti mallin materiaalit voitaisiin vaihtaa, mutta koska instansoiduissa mesheissä koko meshi piti vaihtaa, tuli järjestelmä tehdä eritavoin kuin 3D malleille.

3D mallien kanssa löytyi Unreal Enginestä bugi, joka aiheutti kasvien menemisen harmaaksi ja kadottamana materiaalinsa. Tämä johtui siitä, että jos kasvin vaihdon tekee

ennen kuin Unrealin on antanut käsitellä kasvin Foliage actorina ei se ole voinut luoda vielä tarpeellisia meta tiedostoja kasvin toiminnasta. Koska asetin puut suoraan koodista vaihtumaan toisiin 3d malleihin ei Unreal koskaan kerennyt käsittelemään malleja ja pilasi tehdyt 3D mallit ja ei suostunut enää laskemaan kuinka kasvin tekstuurishaderit tulee laskea.

Tämä korjattiin siten, että otin vanhemman version projektista auki koneelleni ja kopioin sieltä kasvit uudestaan moottoriin ja nimesin nämä kasvit eri nimillä ennen kuin Unreal pääsi käsiksi niihin ja toin kasvit eri nimisinä uusina objekteina moottoriin. Tämän jälkeen manuaalisesti lisäsin kasvin paikalleen Foliage actoriksi, jolloin Unreal laski mallin tarvitsemat shaderit uudestaan. Kun olin varmistanut kaikkien 6 kasvin ja niiden kaikki 4 vuodenaika mallia toimiviksi sekä käyttänyt ne manuaalisesti Unrealin Foliage Systeemin läpi pystyin aktivoimaan koodin uudestaan ja kaikki toimi taas kuten kuuluukin.

Unity virtuaalitodellisuus järjestelmässä huomattiin todella iso ongelma myöhään illalla, kun esimieheni kerkesi katsomaan mallia. Leppä puun malli oli otettu suoraan Speedtree 8:sta ja se ei ollut millään osa alueella oikean näköinen Tervaleppä. Ongelma on siis se, että Speedtree 8 tekijät eivät ole Hortonomeja eivätkä maisema-arkkitehtejä enkä itsekkään ymmärrä kasveista riittävästi ymmärtääkseni Leppä puiden eroja. Itse en siis ollut ymmärtänyt, mitä eroa on Tervalepän ja Speedtree 8:n verkkokaupasta lataamani Leppäpuun välillä, itselleni ne näyttivät lähes samalta. Esimieheni selitti mikä vika puussa oli ja miksi se ei näyttänyt oikealta ja luonnolliselta. Loin kokonaan uuden puun ohjeiden mukaisesti kaikissa kasvumalleissa käyttäen kaikki oppimiani tapoja ja sain muutamassa tunnissa ne exportoitua Unityn Virtuaalitodellisuus malliin ja asetin ne toimimaan oikein koodin kanssa. Samalla poistin käytöstä kaikkein pienimmät leppäpuun versiot tarpeettomina tässä mallissa ja korvasin ne isomman kasvuvaiheen malleilla Prefabien sisään.

Päivän aikana autoin muuta tiimiä luomaan grafiikoita ja parantelin virtuaalimallin muuta vistuaalista ilmettä. Lisäksi korjasin Unity version Lod järjestelmän niin, että LOD tasot menevät päälle oikeaan aikaan lyhyemmällä etäisyydellä ja saimme pudotettua mallin näkyvät polygonit 12 miljoonasta 1.2 miljoonaan näkyvään polygoniin ilman että visuaalinen ilme muuttui ollenkaan. Lod järjestelmä on äärimmäisen tehokas työkalu, kun optimointia suoritetaan.

Huomiselle jäi Virtuaalitodellisuus mallien buildaaminen Unityssä ja Unrealissa sekä viimeisten tarkistusten ja bugi korjausten teko esittelykäyttöä varten

3.5.2019

PlehatCon tapahtuma Otakaari 5A:ssa. Tavoite oli esitellä päivän aikana mahdollisimman monelle virtuaalitodellisuus tuotteita ja keskustella asiakas yritysten tarpeista ja koittaa myydä tuotetta kiinnostuneille ja mahdollistaa myyntiä tulevaisuudessa hyvän esittelyn kautta.

Aamu alkoi ikävästi sillä, että neljä tuntia ennen vieraiden saapumista työkaverini oli löytänyt Virtuaalitodellisuus tuotteesta Unitystä erittäin ison bugin. Kaikki kasvillisuus muuttui virtuaalitodellisuus mallissa yöaikaan kuun valossa kirkkaan valkoiseksi kaukana näkyvissä puumalleissa.

Tähän lähdettiin etsimään syytä ensimmäisenä Tenkoku luonnonkierron simulaatiosta tuloksetta. Lopulta ongelmaksi selvisi Speedtree 8:n Shaderi ongelmat. Speedtree 8:n mukana tuleva Shaderit tekevät jotain puiden materiaaleille Unityssä ja nostavat niiden kiillon maksimi arvoille näin tehden niistä kiiltäviä tietyssä valon kulmassa. Tämä tietysti korostuu erittäin paljon yöaikaan ja näyttää karmealta ja epärealistiselta. Lähdimme korjaamaan ongelmaa ennen vieraiden tuloa tekemällä mahdollisimman monta testiä mistä ongelma johtuu tarkalleen, mutta emme kyenneet löytämään ongelmaan suoraa ratkaisua tai syyllistä. Kokeilimme kaikkia mahdollisia permutaatioita asetuksista, joita keksimme kokeilla tuloksetta. Lopulta huomasimme, että ongelma on nimenomaan LOD tasoissa ja siksi ongelmaa ei oltu huomattu aiemmin. Kun muutin edellisenä päivänä LOD Bias asetuksia Unityssä, jotta LOD:it kytkeytyvät päälle ongelma tuli ensimmäistä kertaa esiin. Tähän asti ongelmaa ei oltu huomattu, koska näkyvissä oli aina ollut LOD 0 taso.

Koska emme kerenneet korjata ongelmaa kokonaan ennen esittelyiden alkua päätimme laittaa Lodit kokonaan pois päältä kaikista puumalleista, jolloin ongelma katoaa. Laitoimme LOD Bias asetuksen takaisin 10 tasolle korjaamastani tasosta 1. Tämä poisti hohtavien puiden ongelman, mutta laski ruudun päivityksen tasoa, joten optimoimme hieman valaistuksen laskemista saadaksemme paremman frameraten aikaan. Tämä onnistui ja buildasimme uuden version koneelle ja siirsimme koneet esittelypaikalle.

Vedin nopeasti sähköt ja laitoin Virtuaalitodellisuus majakat kiinni seiniin ja konfiguroin järjestelmän käyttöön esittelytilaan niin, että kaikki koneet olivat valmiina esittelyä varten. Tein room setupit VR laitteistoille koneilla ja tein nopean bugi testauksen laitteille. Kun olin varmentanut, että kaikki toimii autoin muissa järjestelyissä ja netin asentamisessa esittelytilaan.

Loppu päivän esittelin Unity tuotetta potentiaalisille asiakkaille ja vieraille sekä kävin keskusteluja alaan liittyen. Pyrin jakamaan käyntikortteja ja tekemään jatkomyynnistä mahdollisimman tehokasta ja helppoa esittelemällä tuotteen ominaisuuksia parhaani mukaan.

Viikkoanalyysi

Virtuaalitodellisuus poikkeaa merkittävästi muista peleistä ja järjestelmistä koska se tuo aivan erilaisen kokemuksen todellisuudesta. Muut kulttuurituotteet tai teolliset järjestelmät eivät saa aikaan tuntemusta oman kehollisuuden menettämisestä ja siirtymisestä kuvitteelliseen tilaan. Tämä silmien ja kuuloaistin huijaaminen aiheuttaa oikein tehtynä parhaimmillaan käyttäjälleen tuntemuksen siirtymisestä toiseen todellisuuteen. Mutta koska käyttäjän keho pysyy edelleen sidoksissa tosimaailmaan ei täyttä immersiota pääse tapahtumaan. Jos käyttäjän keho saataisiin liikkumaan samalla tavalla kuin virtuaalitodellisuudessa pahoinvointia ei pitäisi syntyä koska koko keho luulisi näkemänsä ja kuulemansa olevan totta.

Kun keho on paikallaan mutta käyttäjän silmät ja korvat aistivat liikettä aiheutuu monille käyttäjille pahaa oloa Virtuaalitodellisuuden käytössä etenkin aluksi. Testiemme pohjalta olemme huomanneet, että kiihtyvä ja äkillisesti pysähtyvä liike ovat usein pahan olon lähteitä. Paha olo tuntuukin usein käyttäjästä samalta kuin matkapahoinvointi tai merisairaus. Sisäkorva siis viestii käyttäjälle hänen olevan paikallaan mutta ristiriitaiset muut aistimukset luovat käyttäjälle ongelman. Virtuaalitodellisuuden käytöstä johtuvia pahoinvoinnin oireita ovat mm. pahan olo, oksetus ja pyörryttäminen (Inside Science 14.8.2019.). Jotkin käyttäjät eivät koskaan totu virtuaalitodellisuuteen ja kokevat lyhyenkin käytön jälkeen oireita mutta suurin osa ihmisistä tekemiemme testien perusteella tottuu pikkuhiljaa järjestelmään. Oma kokemukseni on, että itselleni ei tule pitkästäkään käytöstä juuri minkäänlaista pahaa oloa nykyään.

Parhaat käytännöt pahan olon välttämiseen virtuaalitodellisuudessa on pitää Frame Rate mahdollisimman korkeana. Mitä alhaisempi ruudunpäivitys taajuus, sitä suurempi mahdollisuus pahaan oloon. Laitevalmistajat pitävät yleensä noin 60 – 90 kuvan näyttämistä sekunnissa riittävänä määränä pääosalle käyttäjistä. Lisäksi tulee huolehtia siitä, ettei käyttäjää yhtäkkiä kiihdytetä tai hänen liikkumistaan hidasteta. Nämä nopeat muutokset liikkeeseen aiheuttavat käyttäjälle oudon tuntemuksen liikkeen äkillisestä muutoksesta ja koska keho ei oikeasti ole liikkunut missään vaiheessa kokemus voimistuu. Varsinaisesti tasainen jatkuva arvattava liike ei yleensä aiheuta ihmisille pahaa oloa koska pää pystyy käsittämään tällaisen. Tätä voidaan myös kiertää sillä, että käyttäjä käyttää esimerkiksi omaa kehoaan tai käsiään ohjaimina ja ikään kuin tarttuu maailmasta kiinni ja liikuttaa sitä

ympärillään. Tällöin käyttäjä kokee olevansa paikallaan ja maailma liikkuu hänen ympärillään. Pahaa oloa voidaan myös vähentää sillä, että käyttäjän ympärille tehdään kehikko, joka sitoo hänet yhteen paikkaan. Tämä yhteen paikkaan sidottu käyttäjä ikään kuin matkustaa kehikon sisällä, jolloin kokemus on lähempänä matkustamista esim. auton kyydissä. Tekemässämme virtuaalitodellisuus malleissa liikkuminen on toteutettu teleporttaamalla paikasta toiseen käyttäen ohjainta.

Tällainen yhtäkkinen siirtyminen paikasta toiseen ei aiheuta pahaa oloa koska liikettä ei paikkojen välillä tapahdu. Aivot pystyvät käsittämään ja orientoitumaan uusiin paikkoihin nopeasti ja pelaaja ikään kuin pysyy paikallaan ja maailma siirtyy. Tässä kannattaa kuitenkin käydä mustan kautta siirtymien välillä siten, että ruutu pimennetään muutaman framen ajaksi siirryttäessä. Tämä ikään kuin resetoi pään uuteen tilanteeseen. Omat testini muissa projekteissa ovat osoittaneet, että liike on mahdollista pysäyttää kerralla kovastakin vauhdista ilman, että pahaa oloa muodostuu. Jos käyttäjä voi käyttää omaa kehoaan ohjaimena ja hallita hidastumista jollain tapaa se vähentää pahan olon syntymistä ja kun vauhti on juuri pysähtymässä, ruutu asetetaan muutamaksi frameksi täysin mustaksi. Tämän jälkeen käyttäjä teleportataan paikalleen siihen kohtaan mihin pysähdys olisi tapahtunut. Näin käyttäjä siis ei koskaan pysähdy vaan ainoastaan löytää itsensä toisesta paikkaa ja voi uudelleen orientoitua tilanteeseensa.

Virtuaalitodellisuuden optimointi on haastavaa laajoissa kokonaisuuksissa. Pelimoottorin erilaisia osia, jotka vaikuttavat toisiinsa pelimoottorin sisällä voi olla tuhansia yhtä aikaa käynnissä. Tähän lisätään pelimoottorin fysiikkamallinnus ja virtuaalitodellisuus lasien asettamat kahden silmän laskemisen aiheuttamat lisärasitukset, asettaa se nykypäiväiset näytönohjaimet ja tiedonsiirtokapasiteetit äärirajoilleen. Tärkeää onkin toiminnan varmistamiseksi optimoida kaikissa kohdissa, joissa voidaan.

LOD järjestelmät ovat omiaan vähentämään piirtokäskyjä pelimoottoreissa ja näin parantaa tuotteen toimivuutta. Piirtämällä kaukana olevista objekteista vain pienempi polygoniset mallit saamme säästettyä piirto budjetista leijonan osan, eikä käyttäjä huomaa muutosta koska objekti on kaukana. Muita optimoinnin polkuja löytyy valaistuksen laskemisessa, materiaaleissa ja erilaisten Terrain Mesh järjestelmien kanssa toimiessa.

3.6 Seurantaviikko 06

Viikon tavoitteet ovat PlehatCon:in läpikäynnissä ja Plantabib järjestelmän uudelleen aloittamisessa.

6.5.2019

Päivän tavoitteena oli saada kaikki tietokoneet ja laitteet paikalleen PlehatConin jälkeen ja asentaa VR laitteistot takaisin seiniin ja tehdä kaikille koneille ”Room Setup” uudelleen. Ensimmäiset purkupalaverit sovittiin aloitettavaksi tämän päivän aikana. Plantabib projektin uudelleen aloittaminen ja siihen orientoituminen tauon jälkeen olivat myös päivän asialistalla. Plantabib:in osalta tulevien projektin osien suunnittelu sekä ohjelmiston uudelleen määrittely syyskuuhun asetettua takarajaa varten teknisesti ja myynnillisesti tulee hoitaa samalla.

Aloitin päivän auttamalla toimiston uudelleen järjestelyllä. Käytimme toimiston koneita ja kalusteita PlehatCon:in järjestämiseen. Autoin kantamalla kalusteita paikalleen ja laittamalla tietokoneet ja näytöt paikalleen sekä asentamalla reitittimet sekä nettipiuhat paikalleen kaikille koneille. Asensin VR majakat ja laitteet paikalleen seiniin ja tein Room Setupit laitteille sekä asensin tarpeelliset ohjelmistot ja niiden päivitykset koneisiin. Verkkojen kytkeminen ja piuhojen veto vie aina hieman ylimääräistä aikaa. Samalla uudelleen järjestelimme toimiston tilat siten, että ohjelmisto tiimi pysyi yläkerrassa ja alakerrasta yksi työntekijä siirtyi meidän ryhmäämme. Samalla keskikerroksen neuvottelutila siirrettiin alakertaan.

Aamupalaverissa käytiin läpi perjantain PlehatConia. Tärkeää oli tarkastella, kuinka tapahtuma sujui yleisesti. Lisäksi kävimme läpi paikalla olleet vieraat ja keiden kanssa keskusteluita käytiin ja mitä keskusteluissa sovittiin. Tästä datasta koostettiin taulukko ja jaettiin jatko tehtäviä kontaktointiin tapahtuman jälkeen. Tapahtuma onnistui tiimin mielestä hyvin ja muutamia hyviä huomioita tehtiin tapahtuman aikana seuraavaa vuotta ajatellen. Palaverissa käytiin läpi myös muita yleisiä asioita sekä tulevan viikon työtehtäviä.

Teknisessä viikkopalaverissa käytiin läpi Unity virtuaalitodellisuusmallin tila ja havaitut ongelmat. Virtuaalitodellisuus mallin ongelmat kierrettiin PlehatConia varten asettamalla malleista vain tietyt versiot näkyville ja keskustelimme, kuinka tämä ongelma ratkaistaan tulevaan versioon. Palaverissa tarkasteltiin myös, kuinka asiakkaat olivat ottaneet virtuaalitodellisuus tuotteen vastaan. Keskusteltiin tuotteen yleisestä suunnasta myyntiä varten ja mitä vielä pitää tehdä, jotta tuotetta voidaan alkaa markkinoimaan. Kävimme läpi miten uusia asioita tullaan kehittämään virtuaalitodellisuusmallissa, jotta siihen saadaan upotettua uusia esiin nousseita tarpeellisia osia. Käytiin läpi tulevan viikon teknisiä tehtäviä kaikkien osalta ja myös esimieheni osallistui tekniseen kokoukseen, kun puhe siirtyi Plantabib ohjelmistoa koskevaan osioon. Teimme teknisen määrittelyn ohjelmiston minimi tarpeista ja suunnasta sekä teimme selvityksen ohjelmiston aikataulusta. Selvityksessä käytiin läpi mitä ominaisuuksia julkaistava ohjelmisto sisältää ja mitä päivityksiä ohjelmisto saa julkai-

sun jälkeen sekä päivitysten aikataulu. Tavoite on saada julkaistavaan ohjelmistoon noin 30 kasvia valmiiksi. Valmiit kasvit sisältävät BIM- ja FBX mallit sekä kasvien tieteelliset ja muut lisätiedot.

Palaverien jälkeen aloitin työni Plantabib ohjelmiston parissa. Olen päävastuussa tuotteen suunnittelusta ja valmistamisesta. Lisäksi työtehtäviini kuuluu serverirajapinnan valmistaminen yhteistyössä Lahden serveri asiantuntijan kanssa sekä Plantabib käyttöliittymän suunnittelu. Lisäksi ohjelmistoa varten tulee luoda kasvikorttien käsittelyn mahdollistava lisäosa. Kasvikortit ovat Excel muodossa olevaa dataa, joka pitää muuntaa muotoon, jota Unity pystyy lukemaan ja luomaan tästä käsiteltäviä kasvi objekteja. Muut projektissa olevat tekevät C# kielellä lisäosaa Revit sekä Archicad ohjelmiin.

Aloitin suunnitteluni lukemalla aikaisemmin tekemän työni dokumentaation ja muut tekemäni huomiot ohjelmasta. Järjestin lyhyen testaus kierroksen toimistolla laittamalla teknisen tiimin käyttämään tekemääni alpha versiota ohjelmasta muutamien minuuttien ajan ja ohjeistin heidän antamaan vapaamuotoista palautetta ohjelman toimivuudesta ja ulkoasusta.

Kävin palautteet läpi ja koostin niistä itselleni uuden listan tulevista työtehtävistä HackNPlan järjestelmään. Yksi isoimmista havainnoista oli, että ohjelmisto ei noudattanut muista ohjelmista tuttuja rakenteita. Tein Photoshopissa itselleni Mockup kuvat ohjelman eri osista ja toiminnoista. Käytin kuvien teossa mallina Adobe ohjelmistojen väri teemoja sekä rakennetta ja lisäksi otin mallia Blenderin valikko ja toiminta rakenteesta. Tein Mockup kuvat alkuvalikosta, projektin luonti valikosta sekä itse Plantabib valikosta ja lisäsin ylös perinteisen File valikon yläpalkkeineen. File valikko luo eroa ohjelman ja nettisivu pohjaisen järjestelmän välille ja luo ohjelma maista rakennetta lisää Plantabib:iin.

Olen tyytyväinen päivän saldoon ja Mockup kuviin. Oli pitkästä aikaa mukavaa päästä takaisin jatkamaan Plantabib ohjelman tekoa. Muutamia isompia asioita tulee selvittää tällä viikolla serverin API:n ja serverirajapinnan teosta sekä itse ohjelman valikkojärjestelmän valmistamisesta.

7.5.2019

Tavoite on jatkaa Plantabib ohjelman tekoa valikoiden valmistamisella ja luoda seuraavat kaksi ohjelman pää tilaa. Lisäksi tarkoitus on aloittaa sijaintitieto järjestelmän sekä kartta näkymän tekoa. Kaikki datan hallinta on tarkoitus tehdä JSON:in kautta Lisäksi tavoite on aloittaa Jetpack Jumper mobiilipelin parantaminen illalla.

Aloitin Plantabib ohjelman parissa avaamalla nykyisen Scene rakenteen Unity pelimoottorissa ja aloitin selvittämään mitkä osat ohjelmasta ovat toimivassa kunnossa ja mitä osioita vielä puuttuu. Kartoituksen jälkeen jatkoin HackNPlaniin asialistaa tulevista tehtävistä. Aloitin luomalla Unityn sisällä kaksi uutta Sceneä nimiltään Project ja Welcome screen. Project Scene sisältää perus osat Plantabib Scenestä ja kirjoitin koodin, jolla Scenejen välillä vaihtaminen onnistuu helposti ruudulla näkyviä nappeja painamalla. Tämän on tarkoitus olla vain testi Scenejen vaihdosta ja siksi en käyttänyt tähän juuri aikaa. Welcome screen sisältää tärkeimmät serveriä koskevat osat eli varsinaisen kirjautumisruudun ja käyttäjätietojen syöttö valikot.

Loin ohjelmaan yläpalkki järjestelmän, joka sisältää File valikon ja paikat muille mahdollisesti myöhemmin lisättäville valikoille. Koska Unity ei sisällä suoraan Windowsin kaikkia toiminnallisuuksia joudun kirjoittamaan valikko järjestelmän alusta asti itse. Kirjoitin koodin siten, että valikko ei aukea kuin klikkauksesta. Kun valikko on avattu klikkaamalla mitä tahansa valikon osaa tein koodin, joka katsoo minkä elementin osan päällä hiiren kursori on. Jos kursorin kuljettaa toisen valikon elementin päälle avautuu tämä ala valikko ja muut valikot sulkeutuvat. Klikkaaminen mihin tahansa valikon osaan aktivoi valikon sen osan ja sulkee kaikki ylävalikon elementit. Koodin on tarkoitus toimia kuten Windows Formin valikkorakenne, jota Unity Pelimoottori ei tue. Unity ei tue automaattisesti myöskään Mac järjestelmää, joten OSX:n upotetut valikot eivät toimi suoraan Unity:llä.

Koska en voi tässä vaiheessa olla varma mitä yläpalkin valikoita lopullisessa tuotteessa tullaan tarvitsemaan päädyin tekemään vain File ja Edit valikkoon sisältöä. Projektin etenemisen kannalta on tärkeää tehdä työtä optimoidusti ja välttää turhan työn tekemistä. Lisäsin myös yläpalkin oikeaan yläkulmaan Windowsista tutut sulku, pienennys ja koko-ruutu napit ja koodasin näille toiminnot.

Valikkojen koodaamisen jälkeen vaihdoin toiseen projektiin tekemään mobiilipeliä Jetpack Jumper. Tavoite on tehdä täysin uusi rakenne pelin kulkuun siten, ettei pelaaja enää avaa hahmoja pelin sisäisellä valuutalla vaan hän ostaa taitoja, joita voi päivittää ja siirtää hahmolta toiselle. Tästä aiheutuu iso määrä uutta työtä, sillä tähän asti hahmot ovat sisältäneet jokainen omat ominaisuutensa. Halusimme tehdä uuden järjestelmän siten, että hahmot lentävät hieman eritavoin toisiinsa nähden, jotta jokainen voi löytää oman suosikkinsa. Kaikilla hahmoilla tulee kuitenkin päästä yhtä pitkälle pelissä riippumatta minkä hahmon pelaaja valitsee. Hahmojen oston sijaan pelaaja voi päivittää kaikkia hahmoja yhtä aikaa ostamalla lisää bensiiniä tai voimaa rakettireppuunsa sekä muita erikoistaitoja. Tavoite uudella järjestelmällä on parantaa ”Day 7 retentionia” sekä pelikokemusta yleises-

ti. Pelin koodin tekee kollegani, joten itse keskityn valikkojen säätämiseen ja hahmojen tasapainotukseen sekä pelikokemuksen yleiseen parantamiseen.

Olin aiemmin tehnyt listan HackNPlan järjestelmään tarvittavista muutoksista peliin, joten aloitin katsomalla listan läpi. Pelissä oli vielä muutamia bugeja, jotka korjasin ensimmäisenä. Kun yksinkertaiset bugi korjaukset oli tehty jatkoin hahmojen testaamisella ja tasapainottamisella, jotta kaikki hahmot toimisivat samalla tavalla toistensa kanssa. Hahmojen testaaminen tapahtuu pelissä lentämällä hahmoja vuorotellen samoilla ominaisuuksilla varustettuna, kaikilla hahmoilla tulisi päästä likipitäen samoihin lukemiin. Lensin aina noin 10 lentoa jokaisella hahmolla ja laskin lentojen keskipituuden. Lisäksi testasin kaikki hahmot korkeilla teho tasoilla ja matalilla ja lensin uudet 10 lentoa ja otin huomioon kuinka hauskaa kullakin hahmolla lentäminen on. Jotkin hahmot pyörivät liikaa ja toiset eivät pyörineet tarpeeksi, joten hahmojen tasapainotus tapahtui muutenkin kuin matkan perusteella.

Muutin liikaa pyörivien hahmojen ”Rigidbody 2D” komponenttien asetuksia siten, että kaikki hahmot vastaavat paremmin toisiaan ja poistin kitkan aiheuttamaa vaikutusta fysiikka moottorissa hahmoilta, jotka eivät pyörineet tarpeeksi. Osalla pelihahmoista on osia, jotka roikkuvat hahmon takana, jotka aiheuttavat hahmon lukkiutumisen taaksepäin suuntaavaan asentoon ilmanvastuksen vaikutuksesta, jolloin lentäminen ei ollut hauskaa.

Testasin hahmoja kolmen tunnin ajan ja sain lähes kaikki hahmot tasaisiksi, vaihdoin hahmojen järjestystä siten, että kaikkein parhaat hahmot ovat loppupäässä avaus järjestystä, jotta parhaat hahmot maksavat eniten. Suunnittelin kollegani kanssa hahmojen avaus mekaniikan hienosäätöä ja pohdimme, mikä on paras tapa ohjata pelaajaa ostamaan taitoja oikeassa järjestyksessä. Päädyimme ratkaisuun, jossa pelaajan käyttäytymistä ohjataan selkeillä huomioväreillä ja kehotuksilla pelin valikoissa.

Pelin progressio parantui huomattavasti toimien seurauksena ja aloitin pelin sisäisen rahajärjestelmän kehityksen parissa työskentelyn. Aiemmin pelaaja avasi hahmoja, jotka olivat aina edellistä hahmoa voimakkaampia ja kalliimpia. Uudessa järjestelmässä pelaaja avaa hahmojen lisäksi erilaisia taitoja ja voimia hahmoille ja jokaisesta avatusta hahmosta tai taidoista pelaajan ”Level” nousee yhdellä. Pelaajan saavuttaessa tietyn level:in hän voi aukaista erikoistaitoja käyttöönsä. Jokainen taito maksaa aina hieman enemmän kuin edellinen ja tämä tasapainotus tehdään myöhemmin testaamalla progressio tarkemmin. Testasin tunnin ajan raha progressiota eri taidoille ja löysin sopivan tasapainon mukavan etenemisen ja riittävän haasteen välille.

Kun hahmojen perustaidot olivat kunnossa tein tasapainotuksen vielä muutamiin hahmojen lentotyyleihin sekä koodasin systeemin, jossa uudelle pelaajalle opetetaan hahmojen ja taitojen avaaminen ja näiden merkitys pelissä etenemiseen.

Lopulta sain valmiiksi tasapainotuksen myös erikoistaidoille ja viimeistelin hahmovalikkojen graafista tyyliä ja värimaailmaa sekä lopuksi yhtenäistin kaikkien hahmojen tasapainopisteet.

8.5.2019

Tavoite päivälle oli jatkaa Plantabib valikon tekoa ja aloittaa serveri API:n kanssa toimiminen. Tavoite päivälle oli ymmärtää, kuinka rajapinnan tulisi toimia ja kuinka serverin API toimii käytännössä ja kuinka siihen otetaan yhteys Unity:llä.

Jatkoin yläpalkin viimeistelyä ja löysin muutamia bugeja, jotka korjasin lisäämällä ylimääräisen kerroksen järjestelmään, jonka avulla avoimet valikot saadaan suljettua, mikäli käyttäjä klikkaa ohi valikosta, jonka on avannut. Tämän tehtyäni valikko oli riittävän valmis ja kopioin sen muihin Scene:ihin.

Loin sisään kirjautumista varten Scenen ja siihen tekstikentät sekä kirjautumis- napit. Sisään kirjautumisen hoitava koodi tallentaa käyttäjän kirjoittamat tekstit Unityn Input Field:eistä ja kirjoittaa ne talteen muistiin jotta tiedot voidaan lähettää serverille. Huomasin samalla, että tekstit näyttivät epätarkoilta ruudulla ja syyksi paljastui, ettei Unity käytä automaattisesti TextMesh Pro järjestelmää vaan Unity:n omaa teksti järjestelmää UI grafiikoissa. Vaihdoin teksti komponentit uudempaan TextMesh Pro versioon ja lisäsin valikkoon Toggle napin kirjautumistietojen tallentamista varten. Kun Toggle on valittu koodi tallentaa käyttäjän kirjautumistiedot koneelle ja seuraavalla käynnistyskerralla luen tiedot sieltä ja syötän ne automaattisesti kirjautumiskenttiin valmiiksi.

Pidin video puhelun Lahteen, jossa serveri API:n valmistajan toimisto sijaitsee. Videopuhelussa kävimme läpi API:n keskeisimmät toiminnot ja hän esitteli minulle, kuinka serverin API toimii sekä kuinka serveriin otetaan yhteys. Asensin Postman ohjelman serverin testaamista varten, jotta voin helposti tehdä serverille testikyselyitä.

Serverillä löytyy tällä hetkellä muutaman kasvin tiedot valmiina ja tavoite on saada Plantabib:in ja serveri API:n kommunikaatio toimimaan mahdollisimman nopeasti. Plantabib ottaa serveriin yhteyden aluksi domainin tai IP- osoitteen avulla. Kun yhteys on saatu, lähetän serverille käyttäjän kirjoittaman käyttäjänimen sekä salasanan JSON tiedostona

lähetyksen Body osiossa ja Header osiossa kerron, että kyseessä on autentikointi tiedosto JSON muodossa. Kun tämä tieto on serverillä lähettää serveri takaisin minulle JWT Token tiedoston, joka on satoja merkkejä pitkä avain. JWT Token:it vanhentuvat tietyn ajan sisällä, jotta voimme varmistaa, että käyttäjällä on lisenssi edelleen voimassa. Tämä avain tulee tallentaa aina uudelleen käyttäjän tuleviin viesteihin Header osioon ja lähettää kaikkien serverille lähetettävien kyselyiden mukaan.

Opettelin mitä API:ssa post, get ja put toiminnot tekevät ja miten ne eroavat toisistaan ja selvitin kuinka Unityllä tehdään API kyselyitä. Opiskelin mitä Header ja Body tekevät ja miten JSON tiedostot toimivat sekä miten serveriin voidaan ottaa yhteys Postman ohjelmalla.

Päivä sujui melko onnistuneesti, mutta serveriin yhdistäminen ei onnistunut. Opin todella paljon siitä, kuinka serverit toimivat ja miten yhteys suojataan. Huomenna tulee selvittää miksi serveriin yhdistäminen ei onnistunut sekä serverin toiminnan jatko opettelua, jotta saan putken auki serverin suuntaan Postman ohjelmalla ja sen jälkeen Unity:llä.

9.5.2019

Tavoite oli saada serveri yhteys auki Postmanilla ja myös Unity:llä sekä lähettää dataa serverille ja tallentaa palautettava data koneelle.

Koska serveri ei toiminut postmanilla edellisenä päivänä, kysyin serverin API:n tehneeltä henkilöltä mitä teen väärin ja hän lähetti minulle postmaniin valmiin configure tiedoston. En ollut ymmärtänyt ohjeista, että käyttäjän tiedot lähetetään serverille Headerissä Raw muodossa. Käytin Form muotoa, jota Postman ja Unity automaattisesti tarjoavat. Muutettuani asetukset tähän muotoon lähettämäni Post komento palautti minulle JWT Tokenin. JWT Token on käyttäjä kohtainen varmistus avain, jonka avulla pidämme huolen, ettei kuka tahansa voi käyttää ohjelmistoa ilman ostettua lisenssiä. Kopioin Tokenin sisältämän tekstin ja liitin sen Postman ohjelman Security osion kohtaan ”Token”, jotta saan Get komennot menemään läpi API:lle

Kun putki lokaalilta koneelta API:lle oli saatu toimimaan tein tunnin ajan erilaisia testejä, kuinka serveri toimii ja luin JTW tokenin käytöstä Unity:ssä ja kuinka JWT Token tallennetaan JSON muotoon.

Aloitin ensimmäisen testauksen Unity:n API rajapinnan tekemisestä lukemalla foorumeilta muiden tekemiä rajapintoja ja minkälaisia ongelmia heillä on ollut. Ensimmäisenä tehtävä

listallani on opetella kommunikoimaan serverin API:n kanssa Unity:n sisältä. Jotta serveriin saadaan yhteys, tarvitaan IP osoite serveriin ja verkkopolut. API:n osoitteet olivat muotoa /API/Check-IN. Tähän polkuun lähetetään käyttäjän salasana sekä käyttäjänimi ja vastauksena serveri lähettää kyseisen käyttäjän sen hetkisen JWT Token arvon.

Polulla /API/Plants voidaan serveriltä pyytää kasvien tietoja, joiden avulla lokaalisti luodaan kasvi objekteja käyttäjälle. Plants komennon jälkeen lisätään tekstiin kasvin ID numero, jota käytetään Get komennolla ja tällä yksilöidään mitä kasvia tällä kertaa kutsutaan. Get ja Post komennot eroavat toisistaan siten, että Get palauttaa vain pyydettyä dataa mutta Post myös lähettää dataa serverille.

Get komennot on tehty toimimaan Tokenilla ja Post salasanalla ja käyttäjänimellä. Token:in avulla varmistamme, ettei serveriltä lähde tietoa ulospäin henkilöille, joilla ei ole käyttöoikeutta ohjelmaan. Koska serverin IP osoitteen kirjoittaminen on työläs tapa, selvitin palveluntarjoajaltamme Nebulalta, kuinka voimme itse hallinnoida Domaineja, Halusimme laittaa IP osoitteet osoittamaan plantabib.com osoitteeseen, jotta testaaminen on helpompaa ja tulevaisuutta ajatellen tästä ei ole mitään muuta kuin hyötyä. Selvitin Nebulalta kuinka hallintaportaali toimii ja pyysin serverin API:n valmistajaa linkittämään serveri API:n IP osoitteen domainiimme.

PlehatCon tapahtuman täysi purkupalaveri pidettiin loppupäivästä. Kävimme läpi kaikki kontaktit ja miten tapahtuma oli mennyt ja keskustelimme mitä voidaan jatkossa tehdä paremmin sekä missä onnistuttiin hyvin. Samalla käytiin läpi potentiaaliset ostajat.

Illalla jatkoin Jetpack Jumper mobiilipelin hahmojen lentomekaniikkaan tasapainotusta, koska emme olleet vielä tyytyväisiä kaikkien hahmojen lentotyyliin. Samalla vaihdoin ensimmäisten taitojen ja hahmojen avaamisjärjestystä, jotta pelaaja saa vielä paremman käsityksen pelissä etenemisestä. Tasapainotin taitojen hintoja sekä taitojen hinnan nousun kertoimet.

Harmillisesti serverin kanssa toimiminen on haastavampaa kuin oletin ja vaatii enemmän perehtymistä API toimintaan. Lisäksi ongelmia aiheutti Unity:n www rajapinnan koodaus.

10.5.2019

Tavoite oli saada kirjautuminen serverille onnistumaan Unity:n kautta ensimmäistä kertaa ja saada selvitys siitä, miten JSON tiedostoja käytetään Unity:n sisällä. Tavoite on selvittää kuinka JSON tiedostoja voidaan serialisoida ja deserialisoida.

Aloitin etsimällä esimerkkejä Unity:n www rajapinnan käyttämisestä ja löysin eräältä foorumilta koodin, jossa joku oli tehnyt oman API:n kirjautumista varten. Kopioin tämän koodin ja loin Scriptin, jonka yhdistin UI nappiin. Tein tähän koodiin kohdan, joka lukee login scenen tekstin syöttölaatikoihin syötetyn tekstin ja tekee siitä oikean laisen teksti muodon, joka voidaan lähettää serverille. Kun käyttäjä on syöttänyt tekstin syöttö kenttiin omat salasanansa ja käyttäjätunnuksensa lisään nuo tekstit luomaani staattiseen serveri tekstiin. Staattinen tekstimuuttuja on muotoa {“username” : “käyttäjänimi”, “password” : “salsana”} ja tämä teksti tallennetaan koneen muistiin JSON tiedosto muodossa.

Tätä varten minun tuli osata käsitellä ja luoda JSON tiedostoja. JSON on tiedostomuoto, jota käytetään perus datan käsittelyyn, lähettämiseen, vastaanottamiseen tai tallentamiseen. Yksinkertainen tietomuoto on erittäin käytännöllinen juuri tämänkaltaiseen järjestelmään. Toinen syy JSON:in valitsemiseen tulee siitä, että serveri ja Unity molemmat ymmärtävät JSON:ia. JSON mahdollistaa eri järjestelmien välillä liikkuvan tieton tallentamisen ja lukemisen tiedostosta ilman ongelmia, kunhan käännös tehdään molemmissa päissä samalla tavoin.

Opeteltuani JSON tiedostojen serialisoinnin ja kuinka pelimoottorin teksti muutetaan JSON tiedostoksi käyttämällä Unityn JSON Utility:ä sain tiedoston tallentumaan koneelle. Tämän jälkeen loin koodin, joka ottaa yhteyden serveriin käyttämällä Unityn sisällä olevaa www järjestelmää. Tähän käytin aiemmin netistä löytämääni koodia. Loin koodissa tekstimuuttujan nimeltä Header ja lisäsin siihen serverin tarvitseman tiedot string muodossa “authentication” /json”, jotta serveri ymmärtää tulevan tiedon olevan varmistus dataa JSON muodossa. Tämän lisäksi tämä piti tehdä RAW muotoisena datana, joten tämäkin piti kertoa Unity:lle. Unity lähettää normaalisti kaiken datan automaattisesti Form datana eikä tarvitsemassani RAW muodossa.

Sain serverin ottamaan vastaan datan Headerissä ja muutamien kymmenten kokeilujen jälkeen vastauksen serveriltä JWT Tokenin muodossa. Token tuli seuraavaksi saada syötettyä talteen Unityyn. Loin uuden muuttujan koodiin, joka deserialisoi serveriltä tulevan JSON datan tekstimuotoon ja syöttää sen muuttujaksi teksti komponenttiin jolloin JWT Token on tallennettuna Plantabib:ssä. Tämän jälkeen on mahdollista ottaa yhteys vihdoin serveriin ja ladata tiedostoja käyttämällä Get komentoa ja ottamalla yhteys oikeisiin osoitteisiin oikeaa polkua käyttämällä.

Onnistuin tänään siis saamaan putken vihdoin auki serverin suuntaan ja ymmärtämään hieman paremmin JSON tiedostomuotoja.

Viikkoanalyysi

Viikon työtehtävät sujuivat varsin mallikkaasti mielestäni. Kiinnostavaa oli nimenomaan se, että tein lähes pelkästään asioita, joita en ollut koskaan tehnyt aiemmin eikä minulla ollut kokemusta asioista. Uuden oppiminen ja tiedonhaku kykyni on parantunut merkittävästi viime vuosina.

Bugien korjaaminen ja etsiminen on tullut myös rutiiniksi. Pystyn analyyttisesti etsimään ja paikallistamaan koodissa olevia bugeja sekä korjaamaan niitä nopeasti. Toisinaan bugin syy ei löydy helposti, jolloin tärkeää on lähteä miettimään pala kerrallaan mistä bugi voisi ja ei voisi johtua. Käänteinen tapa on osoittautunut toisinaan oikeaksi ja pääsen sitä kautta kiinni nopeammin ongelmaan. Suurin osa bugi testauksesta on koittaa rikkoa ohjelmaa ja tehdä asioita mitä kukaan ei ainakaan pitäisi tehdä. Bugien etsiminen alkaa testauksesta ja testaaminen tuntuu olevan todella tärkeä taito työssäni. Koitan kehittää omaa osaamistani jatkuvasti ja yksi hyvä tapa on katsoa mitä muut alan ihmiset ovat tehneet. Netistä löytyy todella paljon hyviä tutoriaaleja ja parhaiden käytäntöjen ohjeita. Yksi lista mistä on ollut itselleni paljon hyötyä ja jonka oppeja olen koittanut seurata, on Testbytesin listaus, kuinka bugeja etsitään ja testataan (Testbytes 2019.). Valitettavasti bugi testaus tiimin puuttuessa kaikkia kohtia en pysty täyttämään.

Testaus suunnitelma on hyvä tehdä mielestäni etukäteen, sillä ilman jonkin tietyn osan tai järjestelmän testaamista ongelmien löytäminen on hankalaa. Huomasin tämän, kun pyysin kollegoitani kokeilemaan alfa versiota ohjelmasta. Pyysin palautetta ohjelman ominaisuuksista ja miten he parantaisivat sitä. Ehdotukset vaihtelivat aivan laidasta laittaan UI järjestelmän ja sorttaus algoritmin toiminnasta aina grafiikoiden tyyliin. Tällainen yleinen palaute voi olla tärkeää mutta huonon ohjeistukseni takia palautteesta ei saatu kaikkea irti. Ensi kertaa varten pyydän hyvin tarkkaan etsimän tiettyihin asioihin vastaukset ohjelmasta ja pyydän näiden jälkeen havaintoja liittyen ongelma kysymyksiin.

Ohjelmien suunnittelu on mielestäni alkanut sujumaan melko hyvin mutta usein huomaan kuinka vaikeaa ongelmiin on varautua. Olen viime aikoina huomannut katsovani todella paljon, miten osaavat ihmiset ja todistetusti toimivat ohjelmat tekevät asioita ja koitan miettiä miten sen tällaisia järjestelmiä voisi vielä parantaa. Tällä menetelmällä on mahdollista löytää omat parhaat ja kiinnostavat ratkaisut omiin ohjelmiini. Hyvänä esimerkkinä on pelin valikko rakenteen teko tai Plantabib järjestelmän valmistaminen. Pyörää ei välttämättä kannata lähteä keksimään uudestaan vaan olemassa olevia väri ja layout tyylejä kannattaa käyttää pohjana omalle. Kun omaan ohjelmaan tulee uusia ominaisuuksia ja tarpei-

ta muuttuu itse ohjelma ja sen rakenne, mutta layout voi olla järkevää pitää silti samana koska se on todettu hyvänä jo toisessa ohjelmassa. Omista suunnitelmista pitää kuitenkin pystyä luopumaan, jos ne eivät selkeästi täytä laatuvaatimuksia. Tulen omalle tekemiselleni kiireessä ja nopealla tyylilläni hieman sokeaksi, joten ulkopuolinen palaute on todella tärkeää minulle. Etukäteen tekninen laatumäärittely sekä ohjelmiston toimintojen määrittely on tärkeää. Ohjelmistojen suunnittelussa SCRUM pohjainen HackNPlan on todella käytännöllinen.

Toisinaan hyväksi kaavailtu ominaisuus ohjelmassa ei toimikkaan käytännössä. Tästä hyvä esimerkki oli mobiilipelissämme Jetpack Jumper alkuaikoina. Mietimme, mitä pelissä voisi tehdä extraa, jotta pelaajalla olisi enemmän tekemistä. Päätimme lisätä kerättäviä kolikoita ja bensa kanistereita peliin, jotta pelaaja saa lisä boonuksia keräämällä niitä. Tämä osoittautui täysin epäkelvolliseksi järjestelmäksi, koska pelaaja liikkuu pelissä liian kovaa. Kun pelaaja näki ohi vilahtavat bonukset, jätti se lähinnä huonon mielen koko pelistä. Ja kun pelaaja silloin tällöin sattumalta osui bonuksen tielle ja keräsi sen se ei tuntunut ollenkaan palkitsevalta. Päinvastoin. Näin ollen hyvästä ideasta oli tulla todella huono ominaisuus peliin, joka olisi todennäköisesti turhauttanut pelaajia ja vähentänyt peli-intoa. Tästä syystä testaaminen on hyvä tehdä ohjelmiston kehityksen jokaisessa vaiheessa mielestäni. Myös ketterät menetelmät ovat oiva apu omassa työssäni.

Viikon aikana opin uusia asioita paljon kuten JSON järjestelmän luomisen ja mitä serialisointi ja de-serialisointi tarkoittaa käytännössä. Samalla opiskelin mikä on serveri API ja miten rajapinta rakennetaan ja testataan. GET, PUT ja PULL komennot serveri rajapinnan kanssa tulivat jokseenkin tutuksi. Unityn JSON Utilityn käyttöönotto onnistui hyvin ja opin tekemään kirjautumisruudun sekä kommunkoimaan Unity:n ja API:n välillä www kirjastoa käyttämällä. Unity:n sivuilla tosin lukee, että www kirjasto on deprekoitumassa joten voi olla, että joudun kirjoittamaan serverin rajapinta koodin uudestaan tulevaisuudessa. Viikko oli mielestäni siis todella onnistunut, koska alkuviikosta en vielä tiennyt miten serverin API toimii käytännössä enkä ollut käyttänyt JSON järjestelmiä, vaikka niistä hieman tiesinkin. Jälleen kerran menetelmäni etsiä toimivia koodi pätkiä ja selvittää kuinka ne toimivat on osoittautunut nopeaksi tavaksi oppia asioita. Uusien asioiden opettelun osalta mielestäni sen pilkkominen pieniin taskeihin HackNPlan:ssa auttaa todella paljon. On hankala koittaa hallita isoa kokonaisuutta, mutta pienten taskien kanssa on mahdollista toimia. Jaoin opetteluvaiheessa JSON järjestelmän osat omiin taskeihin ja kirjoitin niistä pienet kommentit itselleni aina kun sain yhden osan opeteltua. Näin kokonaisuus alkoi kasvaa ja pystyin hahmottamaan mitä seuraavaksi tulee tehdä. Aion jatkossa myös käyttää HackNPlan alustaa ja sen Kanban boardeja hyödykseni, sillä se pitää myös työn mielekkäänä, kun

taskit vähenevät niiden valmistuessa. Työhön tulee näin mielekäs tekemisen tuntu ja sen suunnitelmallisuus on parempaa.

3.7 Seurantaviikko 07

13.5.2019

Tavoite tälle päivää oli valmistaa serverille pääsy koodi ja saada luotua Coroutine koodit sekä login toimimaan ja tallentamaan salasanat.

Aloitimme aamu palaverilla viikon ja kävimme läpi edellisen viikon tapahtumat ja tulevan viikon työtehtävät ja tavoitteet. Yleisen toimisto palaverin jälkeen pidettiin tekniikka palaveri, jossa käytiin läpi Plantabibin edistyminen ja selitin seuraavat vaiheet auki muulle tiimille. Samalla tehtiin katsaus kasvikortti järjestelmän tarpeista. Kasvikortit ovat osa Plantabib ohjelmistoa. Unity:n Virtuaalitodellisuus malli on edistynyt todella hyvin edellisestä viikosta ja useita tärkeitä ominaisuuksia on lisätty ja myös isoimpia bugeja on korjattu.

Aloitin työskentelyn palavereiden jälkeen Speedtree 8:n kanssa, loin uudet koivupuut eri kokoisina Unity:n virtuaalitodellisuus mallia varten eri vuoden ajoissa. Aikaisempien mallien rinnalle tarvittiin isompia puita. Exporttasin puut Unity pelimoottoriin ja tein uudet prefabit puista. Tämän jälkeen pidimme palaverin Plantabib:in BIM mallien valmistamisesta ja kuinka BIM mallit tulee valmistaa. Aloitin kokouksen jälkeen testit voisiko Speedtree 8 puista luoda yksinkertaistettuja BIM malleja helposti Blenderin avulla. Decimate työkalulla voidaan yksinkertaistaa korkea polygonisia puu malleja. Kun mallit oli decimoitu kokeilin Wraparound työkalulla luoda puun ympärille kuoren, joka muodostaisi yksinkertaisen 3D mallin ja säilyttäisi puun muodon. Tämä ei kuitenkaan toiminut odotetulla tavalla ja on kaiken lisäksi työlästä ja hidasta. Lopulta testieni päätteeksi päädyin siihen, että BIM mallit kannattaa tehdä generoidusti 2D kuvan pohjalta ja tätä varten tulee kirjoittaa oma ohjelmansa.

Plantabib serveri rajapinnan koodin kirjoittaminen edistyi melko hyvin viimeviikolla, kun kirjautuminen onnistui ensimmäistä kertaa Unity:n kautta. Aloitin tekemällä ensimmäiset-Get komennon serverille osoitteeseen /API/Plants/1, josta kasvi 1:en JSON tiedosto ladataan. Loin tätä varten Coroutinen joka kolmesti yrittää ottaa serveriin yhteyden. Get komennon mukana lähetetään JWT Token, joka kertoo serverille käyttäjällä olevan oikeus päästä tiedostoihin käsiksi. Kasvikortti sisältää muutamia datapisteitä kasvista, kuten pituuden, lajin nimi tiedot sekä tiedot siitä mihin serverillä kasvin 3D ja 2D tieostot ovat säi-

löttynä. Serverille on laitettu FBX tiedosto, BIM mallitiedosto, JSON tiedosto sekä kuva kasvista ja näiden osoitteet ovat tallennettuna tekstimuotoon JSON tiedostoon.

Koska kaikki serverin nettisivun osoitteet alkavat aina samalla verkko osoitteella loin muuttujan koodiin nimeltä baseHttp, joka sisältää String muodossa verkkosivun alku osoitteen “http://www.plantabib.com/”. Tähän osoitteeseen voin koodissa lisätä minkä tahansa tarpeellisen osoitteen String muodossa, jolloin pystyn luomaan kokonaisia polkuja eri tarpeisiin. JWT Token lisättiin Headeriin aiemmin koodissa, mutta tokenin mukana tulee lähettää myös “Bearer” ilmoitus, jotta serveri tietää kyseessä olevan oikeanlaisen Tokenin. Tässä oli haasteita koska Unityn sisäinen deserialisointi ei ollut helppo ymmärtää ja käyttää mutta sain lopulta useiden kymmenien yritysten jälkeen koodin toimimaan ja Tokenin muutettua oikeaan muotoon ja lähetettyä sen Get komennon Headerissä serverille. Sain vastauksena oikean JSON tiedoston, joten koodi toimii oikein. JSON tiedoston purin käyttämällä Unityn JsonUtilityä fromJson komennolla ja muutin tiedoston tekstimuotoon Unity:n sisällä.

Nyt serverin putki on auki myös Get komennoilla käyttäen Unity:n www komentoja, joten seuraavaksi vuorossa on koodin muuttaminen dynaamisesti generoiduksi JSON tiedostosta. Tällä hetkellä kaikki osoitteet on kirjoitettu vielä manuaalisesti suoraan koodiin, että saan testattua yhteyksiä.

14.5.2019

Tavoite päivälle oli muuttaa kasvikorttien käsittely dynaamiseksi ja muuttaa osoitteiden loppuosat generoitumaan JSON tiedostoista löytyvien tietojen avulla. Lisäksi tarkoitus oli pystyä luomaan Kansiorakenne ja lataamaan tiedostoja pysyvästi haluamaani kansioon Unity:n kautta. Aiemmin tiedostot ovat kadonneet aina kun käynnistän editorin uudestaan koska ne ovat olleet välimuistissa.

Aloitin koodin kirjoittamisen jatkamalla edellisen päivän manuaalisen koodin päälle. Ensimmäisenä piti pystyä lataamaan tiedosto koneelle, luomaan koneelle uusi kansio, johon tallentaa tiedosto sekä tallentamaan tiedosto. Tein Get komento koodiin lisäyksen, jossa luen kasvikortista kasvin tieteellisen nimen ja tallennan tekstin itselleni String muuttujaan. Kun ohjelma käynnistyy, katson ensimmäisenä, onko kansio PlantData olemassa ohjelman sisällä, jos sitä ei vielä ole luon sellaisen käyttämällä Unity:n System.IO kirjaston komentoja. Luotuani tämän kansion heti ensimmäisenä käynnistyksen jälkeen tallennan tämän kansion polun toiseen muuttujaan. Otan kasvin tieteellisen nimen ja luon toisen kansion edellä luodun PlantData kansion sisälle käyttämällä lukemaani nimeä. Näin jokaiselle

kasville voidaan dynaamisesti luoda omat kansiot, johon kasvien datat voidaan tallentaa. Serveriltä tulee siis aluksi ladata tiedosto, joka sisältää kaikkien kasvien tiedot ja sen jälkeen ladata jokaiselle kasville oma JSON data tiedosto. Tämän jälkeen luodaan kansiot jokaiselle kasville omilla nimillään ja tallennetaan kasvin oma JSON tiedosto generoitujen kansioiden sisään.

Kun kansiorakenne on luotu ja koodi testattu toimivaksi tein seuraavaksi muutamia UI nappeja Unityn Login Sceneen ja yhdistin funktiot nappeihin. Napit ajavat koodin sisällä olevat co- routinet suoraan Unity:n Canvakselta silloin kun tarvitsen niitä ja pystyn testaamaan ohjelman toimivuutta oikean kaltaisessa ympäristössä.

Loin 3 muuta coroutinea pääkoodiin, jotka lataavat kasvin BIM-, FBX- ja kuvatiedostot koneelle ja loin niille napit Unity:n Canvakselle. Näiden tiedostojen sijainnit löytyvät JSON tiedostosta, joten minun tuli kirjoittaa myös uusi JSON deserialisointi koodi, joka poimii kaikki verkkosivupolut tiedoston sisältä ja tallentaa ne muuttujiksi. Muuttujien pohjalta generoin uudet verkkosivuosoitteet API rajapinnan käyttöön. Samalla kirjoitin koodin, joka käyttää kasvin luettua tieteellistä nimeä ja luo sen pohjalta uuden tiedostonimipolun ja lisää tiedostomuoto päätteeksi FBX tai BIM tai PNG riippuen minkälaisen tiedoston haluan ladata serveriltä. Unity:n lataaminen toimii serveriltä siten, että Unity ei varsinaisesti lataa tiedostoja sellaisenaan, vaan siirtää serveriltä tulevat bitit sellaisenaan koneen muistiin ja niistä kirjoitetaan uusi tiedosto kovalevylle. Tiedostolle annan nimeksi kasvin tieteellisen nimen ja loppuun tiedostopäätteen ladatun tiedoston mukaan. Tästä hyötynä on myös se, että jos jokin tiedosto on nimetty väärin aiemmin, saan korjattua nimet vastaamaan nyt oikeita tieteellisiä nimiä kaikille kasvin tiedostoille.

Tiedostojen kanssa toimiminen Unity:n sisällä on melko helppoa mutta isoja haasteita aiheutuu serveriltä datan siirrossa. JSON tiedostojen lukeminen ei meinannut onnistua Unity:llä. Loin alun perin oman Class:in, joka sisältää kasvikortin tiedot ja kirjoittaa tiedot JSON tiedostosta Class:iin tekstimuodossa. Tämä ei kuitenkaan toiminut, koska JSON tiedosto, jonka serveri lähettää on sellaisessa muodossa, jota Unity:n JSON Utility ei osaa lukea oikein. Ongelma on siinä, että tiedosto sisältää saman nimisiä muuttujia useita ja JSON data on ryhmitelty Array muotoon. Koska Unity ei pysty käsittelemään kuin listamuodossa olevaa Serialisoitua dataa, en saanut luettua kuin ensimmäisen kasvin Arrays:ta.

Etsiessäni vastausta JSON ongelmaan huomasin samalla, että Unity on poistamassa käyttämääni www rajapintaa käytöstä ja sen tilalle on tullut uusi Webrequest kirjasto, joka on monipuolisempi. Sen ongelmana on, ettei sen käytöstä löydy kunnollisia ohjeita mis-

tään, vaan ainoastaan hyvin pelkistetyt komentojen kuvaukset. Unity:n ongelmana voidaan pitää alati muuttuvaa ympäristöä tällä hetkellä. Koska en ole koskaan tehnyt aiemmin mitään serverin API:iin liittyvää oli minulla melko suuria vaikeuksia ymmärtää miten uusi koodi tulisi toimimaan. Tulin kuitenkin siihen tulokseen, että koodi tulee kirjoittaa uusiksi, jotta se toimi vielä vuosien päästä oikein. Tämä jäi seuraavalle päivälle.
Olin melko tyytyväinen päivän saldoon koska tiedostojen ja kansioiden luonti toimi nyt ja pystyn lataamaan tiedostoja serveriltä.

15.5.2019

Tavoite oli kirjoittaa uudestaan kaikki www koodi Webrequest muotoon ja opetella käyttämään Unity:n uusia Header ja Body toimintoja koodissa.

Olin aiemmin tehnyt kaikki Coroutiinit www muodolla, joista jokainen rutiini hoitaa yhden osan serveri toiminnoista. Sisäänkirjautuminen on oma rutiininsa ja jokaisella tiedoston latauksella on oma rutiininsa. Kävin kaikki koodit läpi ja yhtenäistin niiden tiedostopolku rakenteet kaikissa samanlaisiksi ja siistin koodia huomattavasti. Yhdistin rutiinit yhteen paikkaan ja pystyn nyt muuttamaan kaikkia rutiineja yhdestä paikasta tarpeen vaatiessa.

Unity on poistamassa www ominaisuutta seuraavista versioista kokonaan käytöstä, koska vanha Unet www järjestelmä ei enää vastaa nykypäiväisten Appien tarpeita monilta osin. Servereiden ja API:en käytännöt ovat muuttuneet paljon vuosien myötä, joten Unity on muuttamassa koko multiplayer ja serveri rajapintaansa kohti uudempaa ja toimivampaa järjestelmää, jossa komentojen käyttö on sujuvampaa ja loogisempaa.

Valitettavasti kaikki viralliset Unity:n dokumentaatiot ovat vielä vanhalla järjestelmällä tehtyjä, joten tiedon hankinta on hidasta ja vaikeaa. Onnistuin kuitenkin löytämään muutamia lähteitä, joissa otetaan yhteys API järjestelmään ja lähetetään autentikointi tiedot Headerissä JSON muodossa ja noudetaan Tokeni käyttämällä Webrequest:ia.

Isoin ero uuden ja vanhan järjestelmän välillä on siinä, että aiemmin riitti kun tieto lähetettiin tekstinä ja muoto oli määrätty www komennon yhtenä arvona. Uudessa järjestelmässä Headeriä ei liitetä suoraan Webrequestiin mukaan, vaan se tulee asettaa heti Webrequest komennon jälkeen. Webrequestissa määritellään siis mihin osoitteeseen otetaan yhteys. Kirjoitin uuden koodin, johon kirjoitin kiinteän nettiosoitteen, joka on serverin pääkasvikortin lataus osoite muotoa /API/Plants. Tämä osoite yhdistetään luomaani teksti muuttujaan, jonka kirjoitin aiemmin nimellä “Base HTTP”. Yhdistän nämä kaksi tekstiä yhteen ja samalla luon myös kansion PlantData tietokoneelle, johon ohjaan tulevat lataukset serveriltä.

Unity:n uusi Webrequest määritetään siis ottamaan yhteys nettiosoitteeseen, jonka jälkeen mukaan liitetään Header, joka sisältää käyttäjätunnuksen, salasanan sekä tiedon missä muodossa salasanat tulevat serverille. Unity haluaa käyttää aina normaalisti Form muotoa, mutta serverimme rajapinta haluaa RAW muodossa tiedot. Tästä syystä Header pitää luoda hieman eritavalla kuin normaalisti käyttäen PUT komentoa Post komennon sijaan. Put komennolla voidaan antaa lähtevälle tiedolle lisä arvo, joka tässä tapauksessa tarkoittaa sitä, ettei komento lähde Form muodossa. Kun Webrequest on luotu asetan Put komennon osoittamaan takaisin Post komentoon seuraavalla rivillä ja luon Download Handler osion koodiin.

Seuraavaksi Header tieto liitetään Post komentoon ja Body osaan liitetään salasanat Set-Header komennolla. Downloadhandler hoitaa Post komennon sisään tulevan tiedoston, joka tässä tapauksessa on JWT tokeni, jonka serveri lähettää JSON muodossa ja se deserialisoidaan tekstimuotoon Unityn sisälle ja tallennetaan käyttäjän asetuksiin. Luon siis uuden Downloadhandler järjestelmän heti Unity Webrequest Headerin alle aina kun lataan tiedostoja serveriltä.

Kun kirjautumis koodi oli vihdoin saatu valmiiksi, aloitin työskentelemään muiden coroutinien parissa ja muutin ne www muodosta Unity Webrequest muotoon. Erona on se, että tällä kertaa käytettiin kaikissa GET komentoa POST ja PUT komentojen sijasta ja salasanojen sijaan lähetetään tallennettu ja deserialisoitu JWT tokeni.

Työ oli todella hankalaa esimerkkien puuttuessa eikä juuri minkäänlaista dokumentaatiota ollut tarjolla. Lopulta onnistuin luomaan Webrequest pohjaisen järjestelmän Plantabib ohjelmistoon, joten ohjelman seuraavat vuodet ovat turvassa vaikka Unity poistaa vanhan www pohjaisen järjestelmän käytöstä.

16.5.2019

Tavoite oli viimeistellä Unityn Webrequest koodi ja korvata kaikki loput www järjestelmää käyttävät Coroutine:t ja aloittaa työ kasvikorttien JSON tietojen parissa. Tarkoitus on tutustua JSON tiedoston käyttöön Unity:n sisällä ja opetella kuinka JSON tiedostot toimivat.

Aloitin päivän jatkamalla www koodin muutoksia ja luin tekemääni koodia. Koodi siistiytyi vielä iteroidessani sitä, ja poistin muutamia turhia rivejä, jotka olin kirjoittanut tehdessäni tiedostopolkuja latauksia varten. Käytin turhia String muuttujia, joten optimoin koodiani

poistamalla turhat String muodossa olevat muuttujat ja yhdistin ne yhdeksi kokonaisuudeksi.

Loin uudet GetAllPlants ja GetOnePlant Webrequestit käyttäen aiemmin oppimaani ja tähän meni aikaa noin kaksi tuntia aiemman kahdeksan sijaan. Saadessani nämä osiot valmiiksi siirryin JSON tiedostojen pariin. Aloitin työn tutkimalla, kuinka JSON tiedostoja luetaan oikeaoppisesti Unity:n sisällä. Aiemmin olin jo oppinut serialisoimaan ja deserialisoimaan JSON dataa JWT tokenien ja salasanojen kanssa toimiessani, mutta nyt oli tarkoitus alkaa rakentamaan jonkinlaista tietokantaa Unity:n sisälle.

Aloitin tutkimalla miltä kasvikortti itsessään näyttää JSON datana tekstieditorissa ja luin sen Unity:n sisään käyttäen JsonUtility komentoja. Latasin itselleni lisäosan Notepad++ ohjelmaan, jolla voin muuttaa JSON tiedoston Neat formaattiin. Neat formaatti tarkoittaa, ettei kaikki data ole yhdellä pitkällä rivillä, vaan se erotellaan omille riveilleen muuttujien perusteella. Näin data on helpommin ihmisen luettavissa.

Samalla kävimme keskustelua kasvikorttien sisältämästä tiedosta projektin tästä osiosta vastaavan henkilön kanssa. Kävimme läpi JSON tiedoston hyötyjä sekä selvitimme mitä kenttiä korttiin lopulta tarvitaan ja miten tietoa käsitellään. Samalla puhuimme myös eri kielisistä versioista. Tulimme lopputulokseen, että kasvikorttien teksteistä tehdään oma JSON tiedosto, josta tehdään eri maihin lokalisoidut versiot. Teemme myöhemmin koodin, jolla voimme vaihtaa kielestä toiseen ohjelman sisällä tarpeen mukaan. Kävimme myös läpi datan määrää sekä tiedosto kokoa, jotta osaamme valmistaa ohjelmasta sujuvamman käyttää. Tulimme johtopäätökseen, että kun käyttäjä on kirjautunut sisään ensimmäisen kerran ohjelma ottaa yhteyden serverille ja lataa kaikki kasvit sisältävän tiedoston ensimmäisenä koneelle. Tämän tiedoston pohjalta voidaan luoda sitten kaikki kasvit ohjelman sisään ja myöhemmillä kirjautumiskerroilla tarkistetaan ainoastaan, mitkä kasvit ovat muuttuneet ja ainoastaan muuttuneet kasvit päivitetään. Näin saamme pidettyä latausten määrän minimissään ja ohjelman pysyy tehokkaana käyttää.

Jatkoin päivää tutkimalla JSON muotoa ja loin ensimmäisen version kasvikortin luku koodista, jotta voin alkaa rakentamaan sen päälle dynaamisesti toimivaa järjestelmää. Illasta pidimme tiimipalaverin, jossa käytiin läpi BIM mallien tekoa.

Tutkiessani JSON:ien käyttöä Unity:ssä törmäsin siihen, ettei Unity pysty käsittelemään JSON tiedostoja joissa, on useita samanlaisia objekteja. Ensin tulee kuitenkin rakentaa järjestelmä, jossa kaikki kasvit sisältävästä JSON tiedostosta luodaan tiedostopolut jokai-

selle kasville ja tämän pohjalta haetaan yksittäisten kasvien JSON tiedostot oikeisiin kansioihin.

Olen hieman pettynyt päivän kulkuun, ongelmat JSON muodon ymmärtämisessä ja Unity:n puutteiden kanssa pelatessa aiheuttivat päänvaivaa. Tein päivän aikana tutoriaalien mukaan useita testauksia, mutta mikään niistä ei johtanut toivottuun lopputulokseen.

17.5.2019

Tavoite päivälle oli löytää ratkaisu JSON tiedoston lukemiseen, kun kyseessä on useita objekteja sisältävä JSON tiedosto. Unity ei pysty tulkitsemaan oikein tämän kaltaista tiedostoa, joten selvitin voisiko JSON.Net lisäosan löytyä ratkaisu.

Aloitin päivän tutkimalla JSON.Net kirjastoa ja kuinka se toimisi Unity:n kanssa ja miten se eroaa Unity:n Json Utility:stä. Json Utility on nähtävästi kaikkein nopein kirjasto JSON tiedostojen käsittelyyn Unity:n sisällä, mutta puutteellinen muutamilta osin. JSON.net sekä useat muut kirjastot ovat täydellisempiä kuin Unity:n tarjoama ratkaisu. Lisäosien ongelmana on, että ne ovat raskaampia käyttää sekä ulkopuolisten tekemiä. Tästä aiheutuu ongelmia siinä, ettei päivityksiä tule riittävän usein Unity pelimoottorin päivittyessä lähes viikoittain. Tämän hetken uusin versio JSON.Net kirjastosta on tehty vuonna 2015 joka aiheuttaa merkittäviä ongelmia .Net frameworkin kanssa. Unity vaihtoi tämän vuoden aikana .net frameworking 2.0:sta 4.0:aan. 4.0 framework tarjoaa merkittäviä uusia ominaisuuksia ja optimointeja sekä kirjastoja, joita 2.0:ssa ei voida toteuttaa. Koska JSON.NET kirjasto on 3.5 Net framework pohjainen, ei se toimi ideaalisti uuden Unity version kanssa. Luettuani dokumentaation läpi päädyin siihen, ettemme voi käyttää JSON.Net kirjastoa, vaikka se tarjoaisi suoran ratkaisun JSON tiedostojen käsittelyyn.

Jatkoin tutkimuksia etsien toista JSON järjestelmää, jossa olisi samat ominaisuudet kuin JSON.NET:ssä, mutten onnistunut löytämään mitään järjestelmää, joka olisi luotettava ja edullinen ratkaisu.

Jätin tutkimukset toistaiseksi tuloksettomina kesken ja siirryin tekemään JSON uudelleen kirjoitusta niiltä osin, kun pystyin, eli yhden kasvin JSON lukua ja tallennusta. Testailin erilaisia tapoja saada luettua yhden kasvin tiedot luomieni Class:ien sisälle. Ongelmana oli, että kaikki data ei tallentunut JSON tiedostosta pelimoottorin käyttöön ja JSON tiedosto ei lukeutunut oikein. Testailin erilaisia JSON muotoja ja sain lopulta järjestelmän toimimaan yrityksen ja erehdyksen kautta.

Kirjoitin uuden Class muuttujan, joka sisältää String muodossa olevaa dataa joihin deserialisoin JSON tiedostosta oikeat osat. Loin ensimmäisenä Class kokonaisuudet nimiltä PlantCard ja GameData. Plantcard sisältää yksittäiset muuttujat ja GameData sisältää kaikki PlantCard instanssit Arraynä. GameData Class on tarkoitettu kaikkien kasvien käsittelyyn, kun taas PlantCard on tarkoitettu yhden kasvin tietojen käsittelyyn. Kun ohjelman käynnistyy se luo ensimmäisenä uuden GameData objektin ohjelman muistiin, johon käyttäjä lataa haluamansa PlantCard datat sisään. Tämän jälkeen voimme käsitellä yksittäisiä PlantCardeja Arrayn sisältä ja käskeä ohjelmaa ottamaan serveriin yhteyttä ja vaikkapa lataamaan tietyt tiedostot liittyen yhteen tiettyyn kasviin. PlantCard Class on tarkoitettu yhden kasvin datan käsittelyyn.

Päivä sujui erittäin hitaasti lukuisten ongelmien kanssa taistellessa. Lopulta sain kuitenkin aivan perus asioista ymmärryksen ja luotua Array:n kasveista, joita latasin ohjelman sisälle. JSON tiedostojen käsittely, joka sisältää useita kasveja jäi täysin kesken, enkä saanut asiaa ratkaistua. Ongelmaksi osoittautui itselläni selkeästi tietämättömyys siitä, kuinka data rakenteita käsitellään ja rakennetaan Unity:n sisällä ja kuinka JSON toimii. Tähän auttaa tutustuminen erilaisiin tietomalleihin ja dokumentaatioiden lukeminen.

Viikkoanalyysi

Viikon aikana opin todella paljon JSON tiedostojen kanssa toimimisesta ja etenkin serverien toiminnasta. Unity:n työkalut muuttuvat jatkuvasti tällä hetkellä, koska moottorin sisällä tapahtuu päivityksiä uuteen rajapintaan. Vanhat poistuvat järjestelmät eivät ole saaneet rinnalleen uutta dokumentaatiota ja tämä vaikeuttaa omaa työtäni huomattavasti. Webrequest dokumentaation totaalisen puutoksen takia oli todella hankala lähteä etsimään kuinka serveri rajapinta tulisi koodata. Onneksi Unity:n foorumit ja muut koodaus palstat sisältävät lukuisia ammattilaisia ja harrastajia, jotka kamppailevat samojen asioiden äärellä ja pystyn näiltä palstoilta hakemaan pieniä tiedon osia.

Serverin opettelu oli todella hyödykäs osa ammattitaitoni kehittymistä. Aloitin lukemalla Postman ohjelman dokumentaation (Postman Learning Center 2019.) ja serveri API:en toiminnasta (MuleSoft 07.09.2016.). Itselleni on todella tärkeää ymmärtää syvemmin, kuinka serverit toimivat ja haluan oppia mahdollisimman tarkkaan mitä minun tulee tehdä. Iso osa kehittymistäni tapahtuukin juuri erilaisia dokumentaatiota lukemalla ja näen tämän olevan oma vahvuuteni koodaajana, kun pystyn omaksumaan tietoa melko nopeasti.

Tiedon haku on toinen todella oleellinen osa omaa kehittymistäni, tällä hetkellä tuntuukin, että lähes kaikki aika kuluu erilaisten koodi snippettien etsimisessä ja niiden tutkimisessa.

Koostan näistä koodin osista itselleni ensin ymmärryksen siitä, kuinka ohjelman pitäisi valmiina toimia ja aloitan aina rakentamalla itselleni kaikkein yksinkertaisimman ja helpoiten toimivan vaihtoehdon. Tämän vaihtoehdon päälle olen tottunut alkamaan rakentamaan uusia toiminnallisuuksia ja saan melko nopeasti laajennettua koodin toimimaan. Ongelmana lähestymistavassa on, ettei koodi ole välttämättä laadultaan riittävällä tasolla ja rutiinit ovat huonosti optimoituja. Tähän olen alkanut puuttua lukemalla koodin optimoimisesta sekä parhaita koodaamisen käytäntöjä (Codeproject 18.10.2010.). Nettisivut ovat hyviä paikkoja löytää tietoa siitä, kuinka koodia voidaan optimoida mutta käytännön esimerkit silti tuntuvat antavan itselleni parhaan lopputuloksen. Teen vapaa-ajallani useita online kursseja ja yksi hyvä paikka kurssien suorittamiseen on Codeacademy sivusto (Codeacademy 2019.). Osa sivustoista on täysin ilmaisia ja osa kursseista on maksullisia. Maksulliset kurssit eivät aina ole välttämättä parempia mutta osasta saa itselleen sertifikaatin kurssin suorittamisesta, josta on hyötyä työmarkkinoilla.

Koodin siistiminen ja kommentointi on oleellinen osa omaa prosessiani sekä dokumentointi ylipäätään. Olen tällä hetkellä monessa projektissa mukana ja koen hyvän dokumentaation helpottavan päivittäistä työtäni. On todella turhauttavaa avata muutamia päiviä sitten kirjoitettua koodia, mikäli sitä ei ole kommentoitu hyvin. On todella vaikeaa muistaa ulkoa, miksi ratkaisuita on tehty ilman kommentteja ja dokumentaatiota. Lisäksi tiimin henkilöstön vaihtuessa dokumentaatio on ainut tapa varmistaa tiedon siirtyminen uusien ihmisten käyttöön. Dokumentaatioiden kirjoittamisesta minulla on vielä paljon opittavaa ja aion paneutua tähän tulevaisuudessa.

3.8 Seurantaviikko 08

20.5.2019

Päivän tavoitteet olivat serverikoodissa selvittää kuinka JSON muotoinen ladattu tiedosto, jonka serveri API lähettää pystytään käsittelemään Unity:ssä. Lisäksi minun pitää tutkia puiden BIM mallien tekoa ja saada ajatus siitä, kuinka Plantabib ohjelman Revit lisäosa toteutetaan.

Aamu alkoi maanantai palaverilla, jossa käytiin läpi ajankohtaisia asioita tälle viikkoa. Tekniikka palaverissa käytiin läpi viimeviikon toteutuneet työtehtävät ja katsottiin missä tilassa tuotteiden kehitys on. Virtuaalitodellisuusmalli Unity:ssä on edennyt taas harppauksin eteenpäin ja uusia ominaisuuksia on kehitteillä muutamia, joista tärkeimmät koskevat kasvien manipulointia reaaliajassa mallin sisällä. Kasvikorttien teko on edennyt Plantabib:in osalta hyvin ja tavoite on päästä testaamaan uudentyyppisiä kasvikortteja

pian käytännössä ohjelman sisällä. Tavoite on saada keskiviikkona serveri koodi täysin valmiiksi, jolloin voidaan siirtyä eteenpäin ohjelmiston kehityksessä. Plantabib serverikoodi jäi kesken edellisellä viikolla, kun JSON Utility:n kanssa tuli seinä vastaan ja koitin etsiä korvaavaa kirjastoa. Palaverissa käytiin aikataulua läpi ja otin keskiviikosta eteenpäin työtehtäviksi lähteä kirjoittamaan Revit lisäosan koodia.

Yritin etsiä vielä korvaavaa kirjastoa Unity:n Json Utility:n tilalle.Tutkiessani korvaavaa kirjastoa luin samalla koodiani läpi ja huomasin ison bugin Log In koodissani. Yritin kutsua muuttujia, jotka tallentavat käyttäjän tiedot kirjautumisen yhteydessä joka kerta luomalla uuden instanssin tietokannasta. Tämä ei ole mahdollista koska käyttäjätunnukset tallennetaan Class:in sisälle. Unity ei voi luoda uutta instanssia Classista kellumaan ohjelman sisälle. Unity on toiminnaltaan erilainen muista koodaus ympäristöistä siten, että jokainen objekti tulee sisällyttää peliobjektiin, näin ollen ei ole mahdollista luoda uutta Class:ia joka sisältää käyttäjätiedot jokaisen kirjautumisen yhteydessä, vaan se tulee tehdä ainoastaan ensimmäisen kirjautumisen yhteydessä. Muutin koodin siten, että Unity katsoo, onko tiedosto jo olemassa ja jos on se ohittaa tiedoston luonnin.

Samalla korjailin koodin rakennetta optimoidummaksi ja hain Class:in sisältämän datan suoraan lukemalla JSON tiedoston ohjelman käynnistyessä sen sijaan, että otan toiseen Scriptiin yhteyden. Ohittamalla turhan koodien välisen kommunikoinnin sain koodista siistimpää ja helpommin ymmärrettävämpää ja nyt käyttäjätiedot ja asetukset tallentuvat oikein Unity:n sisällä.

Löysin lopulta tavan, jolla korjata ongelmat JSON tiedostojen lukemisessa käyttäen JSON Utility:ä suoraan Unity:n sisällä. Kun JSON tiedosto luetaan tekstiksi ja deserialisoidaan ei Unity pysty lukemaan JSON tiedostoa sellaisenaan, koska siinä toistuu kaikkien kasvien kohdalla samat tiedot. Joten JSON tiedostosta tulee ensin luoda Array ja tämän pohjalta lista sen sisältämistä objekteista. Array:tä voidaan sitten käsitellä tarpeen mukaan miten halutaan ja sitä voidaan organisoida eri tavoin ja sille voidaan tehdä foreach looppi joka luo kansiot ja lataa oikeat tiedostot jokaisen kohdalla. Lisäksi jatkoin opiskelua asian tiimoilta pitemmälle ja todennäköisesti tämän päälle rakennetaan vielä library järjestelmä, jolla saadaan JSON:in sisältämä tieto vielä paremmin prosessoitua. Libraryn avulla voidaan tehdä yksittäisestä objektista hakuja. Array:n ongelmana on siis se, että jos haluan katsoa kasvin “105” kentän “15” arvon, joka on esimerkiksi “leppä”, joudun käymään kaikkien kasvien kaikki kentät läpi ennen kuin päädyn oikeaan arvoon. Libraryn etu taas on, että voin ensin kutsua kasvin 105 ilman, että käyn kaikkien kasvien kaikkia osia läpi. Löydettyäni oikean kasvin voin katsoa sen sisältä kentän 15 arvon. Näin säästyy laskentatehoa ja koodi on optimoidumpaa ja toimivampaa

Tutkin lopuksi, kuinka Hi-Poly mallisista puista voidaan tehdä Low poly malleja helposti BIM käyttöön. Tulin siihen tulokseen, että nopein tapa on avata 2D kuvatiedostot puusta Blenderissä ja laittaa ne ortograafisiksi kuviksi kahteen eri suuntaan, esim. Front ja Side näkymiin. Tämän jälkeen luon uuden UV-Spheren projektiin ja aloitan sculptaamaan tätä UV- Sphereä Blender:in sisällä, jotta saan luotua kuvien ääriviivojen mukaisen siluetin puulle. Kun olen tyytyväinen sculptauksen lopputulokseen käytän Smooth brushia ja dyntopo työkalua saadakseni tasaisen siistin pinnan puumallille. Lopuksi lisään modifyer listasta remesher työkalun malliin ja luon puusta matalaresoluutoisen version. Tähän meni aikaa noin 5minuuttia, joka on juuri oikea aika tämän tyyppiselle mallille.

Olen tyytyväinen, että JSON ongelmaan löytyi ratkaisu. Toteuttaminen pitää vielä tehdä huomenna ja muutamia mieltä askarruttavia asioita jäi selvittämättä, mm. kuinka saan Serverin API:sta tulevan datan muutettua oikeaan muotoon libraryä varten.Tällä hetkellä JSON tiedosto ei ole oikeassa muodossa. Sculptaus BIM mallille onnistui todella hyvin ja ehdotan sitä huomenna tiimille työtavaksi BIM mallien tekoon.

21.5.2019

Tavoite oli saada JSON ongelma ratkaistua ja luoda Plantabib:iin toimiva JSON järjestelmä, jossa lataan serveriltä kaikki kasvit sisältävän JSON tiedoston ja josta generoin yksittäiset kasvit ja niiden kansiorakenteen. Tavoite siis saada serveripuolen koodi valmiiksi.

Testasin aamusta ensimmäisenä, miten JSON järjestelmä toimii tällä hetkellä ja mitä minun tulee muuttaa koodissa. Tulin tulokseen, että paras tapa viimeistellä yhden kasvin JSON luku koodi on ensin rakentaa järjestelmä, joka tulee library tyylin ratkaisua. Löysin hyvät ohjeet tähän eräästä netti postauksesta JSON aiheeseen liittyen.

Aloitin tekemällä uuden kaikki kasvit sisältävän JSON tiedoston, johon rakensin korjaus scriptin. Tällä hetkellä AllPlants.json tiedosto sisältää datan muodossa {[{“ID” : “1”, “Nimi” : “Kasvi”},{“ID” : “2”, “Nimi” : “Kasvi2”}]}. Unity ei osaa suoraan lukea tätä dataa oikein vaan se lukee kaikki kohdat JSON:sta ja sen jälkeen päälle kirjoittaa tiedot jokaisen uuden kasvin kohdalla. Kun luen tiedoston en saa tulokseksi kahta kasvia vaan ainoastaan kasvin kaksi tiedot. Tästä syystä en pysty generoimaan useita kasveja.

Tein funktion pääkoodin sisälle, joka ensin lukee JSON tiedoston Unityyn käyttäen JsonUtility.FromJson komentoa ja tämän jälkeen lisää yhdet aalto sulkeiset lisää alkuun ja loppuun sekä lisää sanan “Plants”. Uusi teksti näyttää siis muutosten jälkeen sellaiselta

jota Unity ymmärtää paremmin: {{ “Plants” :[{“ID” : “1”, “Nimi” : “Kasvi”},{“ID” : “2”, “Nimi” : “Kasvi2”}]}}. Seuraavana tein koodin, joka ymmärtää molemmat ID:t omiksi kentikseen foreach loopilla ja lisäsin ne array:n sisään. Lisättyäni kasvit arrayn sisälle pystyin tekemään foreach loopin, joka käy kaikki kasvit pää kortista yksitellen läpi ja tekee jokaiselle kansion luonnin käyttäen nimeä ja ID tietoja.

Näin ollen sain aikaiseksi järjestelmän, jossa alkuperäisestä JSON tiedostosta muuttuukin useita “Plants” objekteja sisältävä tiedosto Unity:n silmissä. Loin myös Unityn deserialisointia varten oman Class:in joka sisältää vain ID integerin sekä Name string muuttujat joihin tallensin lukemani kasvit. Tekemättä jäi vielä classiin sisälle tallentuvat tiedostopolut, jotta pystyn tulevaisuudessa kertomaan mihin tiedostot tulee tallentaa.

Seuraavaksi kirjoitin koodin, jossa teen muuttuvan PlantID funktion. Funktion avulla pystyn Unity:ssä tekstikenttään kirjoittamalla etsimään tietyn kasvin ja tekemään tälle kasville muutoksia. Tällä hetkellä kaikki toimii yhden kasvin kanssa moitteetta, mutta haluan muuttaa koodin siten että käyttäjä saa kertoa minkä kasvin tiedostot hän haluaa ladata koneelle ja ohjelma tietää automaattisesti, minne ladata kasvit. Tein Unityn UI canvakselle Text mesh Pro text input fieldin ja pääkoodin sisälle public muuttujan ja yhdistin kaikki osat toisiinsa. Muutin input fieldin asetuksia siten, että se ottaa vastaan ainoastaan numeroita, jotta käyttäjä ei etsi indexistä kirjaimilla koska tämä aiheuttaisi aina hakutuloksen, joka ei johda mihinkään ja rikkoisi koodin. Otin Textmeshpro.text komponentin osan ja tein sille parse komennolla muutoksen tekstistä integeriksi, joka pakottaa tekstin numeroksi. Näin koodi ymmärtää, ettei kyseessä ole teksti vaan numero vaikka mitä tapahtuisi. Lisäksi laitoin oletusarvoksi tälle uudelle PlantId muuttujalle arvon 0, jotta saan aina varmasti jonkin palautteen, ettei koodi voi katketa.

Huomiselle jäi vielä tehtävää, jotta saan yhdistettyä uuden yksittäisen kasvin lukukoodin ja uuden käyttäjä inputtia ottavan plant id koodin.

Illalla otin vielä yhteyttä serverin API:sta vastaavaan henkilöön ja kävimme keskustelua siitä, kuinka suojata data, jota Plantabib käsittelee. Kasvikorttien datan suojaaminen on tärkeää kilpailuedun takaamiseksi. Tällä hetkellä käsittelen JSON tiedostoja suoraan koneella tallentaen ne koneelle selvyyden vuoksi, mutta tulevaisuudessa data käsitellään ainoastaan serverin päässä JSON muodossa ja upload vaiheessa JSON muodossa mutta muuten binäärisenä, jotta sitä ei voida lukea ja käsitellä käyttäjän toimesta. Näin saamme suojattua tärkeimmän immateriaali osamme Plantabib ohjelmistosta, jotta käyttäjät eivät saa kasvikorttien tietoja käsiinsä suoraan.

Olen tyytyväinen päivän tulokseen, sillä nyt Plantabib toimii kaikilta osin vaikkakin yksittäisen kasvin toimimisen kanssa tulee tehdä töitä saadakseni sen toimimaan kaikilta osin. Koodia ei myöskään ole vielä optimoitu.

22.5.2019

Tavoite päivälle oli kirjoittaa koodi uusiksi ja siistiä turhat asiat pois sekä viimeistellä UI selkeämmäksi ja toimivammaksi Login Screenin osalta. Illalla osallistuminen Meet VC tapahtumaan, jossa pääomasijoittajia on mahdollista tavata ja keskustella rahoituksista ja selvittää miten sijoittajat toimivat.

Aloitin päivän käymällä läpi ohjelman pää koodia ja tutkimalla sitä mahdollisten bugien osalta. Tarkastelin myös koodin rakenteita ja varmistin kaiken olevan yhdenmukaista kaikkien coroutiinien ja osien kohdalta sekä poistin tarpeettomia ja vanhentuneita osia koodista sekä turhia muuttujia.

Koska koodi oli kirjoitettu kolmesti uudelleen tähän asti serveri ja JSON osien osalta tarvitsi koko projekti myös uudelleen järjestelyä sekä asetusten muutosta. Tarkistin projektin asetukset, jotta Net Framework versiot olivat 4.0 asetuksilla sekä muut import asetukset kuvien ja tiedostojen osalta. Loin kaikista tarpeellisista osista prefab tiedostot, jotta voin käyttää niitä useissa eri sceneissä Unity:n sisällä ja laitoin tallentuvat tiedot DontDestroyOnLoad komennolla säilymään vaikka Scene vaihtuisikin.

Päivän keskellä tuli kiireellinen muutos mobiilipeliä koskien. Asensin Sourcetree Git clientin koneelle ja kävin Gitlabista kopioimasta Remote Repositoryn osoitteen ja kloonasin peliprojektin lokaalisti koneelleni. Asensin Unity:n tarvittavat lisäosat IOS kehitystä varten ja avasin projektin. Vaihdoin projektin Build settingsit oikeaan muotoon ja nostin versionumerointi Unity Cloud Buildia varten muutamaa pykälää ylemmäs. Tallensin muutokset ja tein commit ja push komennot serverille ja aloitin Cloud Build prosessin. Asetukset eivät kuitenkaan muuttuneet koska teemme samasta paikasta useita eri tyyppisiä buildeja joten kaikki IOS, Android ja Standalone asetusten versionumerot tuli ensin muuttaa ja sitten toistaa edellä olevat Git komennot ja aloittaa Cloud Build uudestaan. Lähetin version testaukseen julkaisijalle.

Jätin yksittäisen kasvin JSON käsittelyn myöhempään osaan projektia, jossa teen kasvi korttien visuaalisen luonnin.

Illalla olin kouluttautumassa Meet VC tapahtumassa StartUp saunalla Otaniemessä. Tapasin pääomasijoittajia, joiden kanssa keskustelimme mitä sijoittajat hakevat ja minkälaisia riskejä he ovat valmiita ottamaan ohjelmistojen osalta sekä miten Business Enkelit ja VC:t eroavat toisistaan. Illan lopuksi oli runsas joukko pöytä kohtaisia ryhmä koulutuksia asian tiimoilta.

Plantabib ohjelman serveri osa tuli lähes valmiiksi tänään. Meet VC tapahtuma oli erittäin hyvä tapahtuma käydä ja opin erittäin paljon sijoittajien toiminnasta. Sain myös hyviä kontakteja rahoituksen haku varten.

23.5.2019

Tavoite oli saattaa Plantabib ohjelman kansiorakenteen luominen ja serveri integraatio loppuun tältä osin ja saada loppu koodi siistittyä ja bugittomaksi. Tavoite oli saada BIM mallit tuotua Revit ohjelmaan ja opetella tekemään lisäosa ohjelmien Plantabib ja Revit välille.

Tämä oli erittäin lyhyt päivä koska edellinen päivä oli ollut 12h puristus. Aloitin päivän tekemällä koodin korjailun ja iteraation valmiiksi. Siistin viimeiset turhat osat koodista pois ja kommentoin kaikki koodin rivit selkeiksi.

Seuraavaksi aloitin uuden osan Plantabib ohjelman integraatio osaksi Revit ohjelmistoa tutkimalla, kuinka Revit toimii. Revit on arkkitehtien käyttämä sovellus rakennusten tekoon, josta saadaan oikeita toimivia pohjapiirustuksia ja tietoa ulos suunnitelmasta. Pyysin yrityksen kokeneinta Revit käyttäjää näyttämään, kuinka ohjelma toimii ja selvitin hänen avullaan, kuinka lisäosa kuuluu toteuttaa. Havainnointieni mukaan tarpeellisinta on, että käyttäjä voi itse tuoda alussa BIM mallit haluamaansa paikkaan ohjelmaan Import komennolla ja tiedostot tulevat olla Revit Family tiedostoja. Family tiedostot sisältävät 3D mallin sekä tiedon mikä objekti on kyseessä. Tietomalli tulee hakea serveriltä Revittiä varten käyttäen Plantabib ohjelmaa. Tarkoitus on rakentaa lisäosa ohjelma Revittiin joka tuo mallit suoraan käyttöön.

Opittuani riittävästi Revit järjestelmästä loin nopeasti yksinkertaisen Revit Family tiedoston malli kasvista ja tallensin tämän koneelle. Tämän jälkeen siirryin rakentamaan kansion valinta järjestelmää Unity:ssä Plantabib ohjelmaan. Halusin tehdä järjestelmän, jossa käyttäjä voi valita kansion, johon kaikki hänen serveriltä lataamansa tiedostot menevät ja oikeaan kansiorakenteeseen. Tämän halusin tehdä siksi, että käyttäjät voivat itse valita minne ohjelma tallentaa tiedostot helppokäyttöisyyden lisäämiseksi. Näin pystyn kiertä-

mään tässä kohtaa monimutkaisen lisäosa järjestelmän rakentamisen hetkeksi ja pääsemme testaamaan parhaat ratkaisut käytännössä ja kehittämään ratkaisua myöhemmin toimivan järjestelmän päälle.

En saanut koodia vielä toimimaan tämän päivän aikana kansion valintaan mutta löysin lupaavan lisäosan Unityyn, jolla asia todennäköisesti saadaan toteutettua.

Lyhyt mutta tuottoisa päivä koska sain Plantabib serverikoodin nyt loppuun ja siistiksi sekä opeteltua kuinka Revit järjestelmä toimii. Samalla ratkaistiin Archicad ohjelmiston osalta sama ongelma lisäosaien osalta.

24.5.2019

Tavoite päivälle oli saada kansio valinta järjestelmä valmiiksi ja toimimaan, jotta lataukset menevät haluttuun kansioon ja oikeiden kansioiden alle järjestettyinä projektin alle. Tämän jälkeen tavoite oli saada UI suunnittelua parannettua ja luoda kaikki scenet projektiin ja muokata vanhoista Sceneistä toimivia.

Aloitin päivän lukemalla edellisenä päivänä löytämäni ilmaisen Unity Assetin tietoja ja tutustumalla sen toimintaan esimerkki Scenen kautta. Järjestelmän avulla voin avata Unityn sisällä Windows tai Mac kansio rakenteen, jotta käyttäjä voi valita koneeltaan sopivan kansion. Kun käyttäjä painaa Plantabib:ssä lataa nappia avautuu ensin kansion valinta ikkuna, kun käyttäjä valitsee sopivan kansion ja painaa OK nappia, koodi jatkaa eteenpäin ja tallentaa kansion tiedosto polun muuttujaan. Syötän tämän muuttujan Unity:n serverikoodin Downloadhandler järjestelmälle tiedosto poluksi. Näin saan lataukset siirtymään valittuun kansioon ja luotua halutun kansiorakenteen tämän kansion sisälle. Sain koodin toimimaan muutamien kokeilujen jälkeen ja järjestelmä oli valmis. Tein ensimmäisen testi Buildin Plantabib järjestelmästä ja poistettuani turhia tiedostoja ja lisäosia sain ohjelman toimimaan itsenäisenä järjestelmänä toisella koneella testatessani.

Siirryin suunnittelemaan muita ohjelman User Interface osia. Tein tekemieni mockup kuvien pohjalta toimivan Welcome screenin jossa käyttäjälle tarjotaan kiinnostavaa tietoa Plantabib ohjelman käytöstä ja uutisia sekä hän voi luoda uuden projektin tai avata vanhan projektin. Lisäksi tein järjestelmän, jossa käyttäjä voi valita oman profiilikuvan ohjelmaan. Korjasin vanhasta projektin luonti osuudesta samalla muutamia bugeja ja laitoin serveri koodin toimimaan myös tämän osan kanssa yhteen.

Yrityksen ensimmäisessä Perjantai palaverissa kävimme läpi kuluneiden kuukausien työtehtäviä ja edistymistä ja kävimme läpi mihin suuntaan yritys on menossa ja strategisen katsauksen tulevaan. Palaverin aikana esitimme omat näkemyksemme parhaista toimintatavoista ja perustimme muutamia työryhmiä. Itse otin vetääkseni työryhmän, jonka on tarkoitus luoda yritykseen tietoturvan osalta yhteiset käytännöt sekä salasanojen hallinta järjestelmä käyttöön.

Tämän jälkeen meillä oli vielä iltapalaveri, jossa kävimme läpi yrityksen strategisia muutoksia ja tavoitteita sekä mahdollisia rekrytointeja ja isoja linjauksia.

Sain tänään valmiiksi isoimmat osat Plantabibiä ja meillä on nyt toimiva Beta versio ohjelmasta valmiina. Seuraavaksi lähdetään viemään ohjelmaa ulos betasta ja ensikuun loppuun mennessä ohjelman ensimmäinen versio pitäisi olla lähes valmis. Selvitettävää on vielä tietoturvan osalta, kuinka ohjelma suojataan oikeaoppisesti. Palaverit veivät lähes koko työpäivän mutta sain kuitenkin tarpeelliset koodi työtehtävät tehtyä.

Viikkoanalyysi

Minulla on tällä hetkellä todella monta asiaa kesken yhtä aikaa ja vastuita on todella paljon eri suuntiin oman yrityksen pyörittämisestä sekä työpaikan ohjelmistojen suunnittelusta 3D mallintamiseen.

Olen huomannut, että nykyisin työntekijän tulee osata todella paljon erilaisia taitoja ja startup maailmassa useiden taitojen hallinnasta on todella paljon hyötyä. Teen työssäni tiimin johtamista, koodaamista, video editointia, sopimus neuvotteluita, ohjelmistosuunnittelua ja monta muuta asiaa. Kaiken tämän taidon kerryttäminen voidaan ajatella olevan koko elämän kestävä prosessi ja kaikista oppimistani taidoista on ollut hyötyä jossain kohtaa työuraani. Oman kehittymiseni kannalta on tärkeää entistä suunnitelmallisemmin osata käyttää aikaa hyödyksi.

Kovan työtahdin mukana tulee helposti uupumusta ja väsymystä ja pitkäaikainen stressi voi tehdä todella pahaa toimistotyössä. Lisäksi erilaiset toimistotyöläisen vaivat vaikuttavat fyysisesti jaksamiseen. Luen vapaa-ajallani tällä hetkellä työssä jaksamisesta ja tutkin stressin hallinnan keinoja. Tulevaisuudessa tarkoitus on palkata peliyritykseemme ensimmäiset työntekijät ja tavoite on tarjota heille mahdollisimman kilpailukykyinen työpaikka ja mahdollistaa heille kehittyminen omassa työssään.

Työntekijöiden hyvinvointi on ensiarvoisen tärkeää mielestäni nykyajan työpaikoilla. Teknologia alan osaajista on pulaa ja hyvistä työntekijöistä kannattaa pitää kiinni ja heidän terveydestään kannattaa huolehtia. Itselläni on vielä paljon opittavaa työhyvinvoinnista niin omalta osaltani kuin mahdollisten työntekijöiden palkkaamisen osalta. On hyvä, että tietoa työhyvin voinnista ja stressin hallinnasta on saatavilla runsaasti (Työterveyslaitos 2019.). Työssä jaksaminen on myös yritykselle erittäin kannattavaa sillä terve työntekijä tietenkin pitää vähemmän sairaslomia ja säästää näin yrityksen resursseja. Mielestäni hyvä tapa pitää työntekijöiden kunnosta huolta on tarjota heille myös erilaisia liikuntaetuja.

Koska oma työni on todella kiireistä myös levon ja fyysisen kunnon ylläpito työntekijän ja yrittäjän yksi vastuu alue. Näen, että työkyvyn ylläpitoon kuuluu nykyään myös ruokavalio, fyysinen lihaskunto ja henkinen tasapaino. Yrittäjän tulee pystyä pitämään vapaata kiireen keskellä ja huolehtimaan omasta jaksamisestaan ja oma henkilöjohtaminen on tärkeä osa työtä. Lisäksi uuden opettelu tulee suhteuttaa omaan aikatauluun ja omat rajat tulee pystyä tunnistamaan.

Tällä viikolla asiat tulivat usean projektin osan kanssa päätökseen ja tietty helpotus tuntuu suoraan omassa tekemisessä. On huomattavasti helpompaa siirtyä seuraavaan osaan, kun yksi osa on poissa ajatuksista. Yksi erittäin pitkä päivä keskellä viikkoa rikkoi rytmiä hieman ja siksi oli tärkeää seuraavana päivänä levätä.

4 Pohdinta ja päätelmät

Alkutilanteeni oli melko työtehtävien hoitamiseen. Hyvä perusosaaminen pelimoottoreiden parissa ja virtuaalitodellisuuden kanssa sekä osaaminen 3D mallintamisessa luo hyvän pohjan työtehtävien tekemiseen. Suuri osa aikaisempaa työkokemustani kuitenkin liittyi muuhun kuin koodaamiseen. Opinnäytetyöni aikana iso osa työstäni oli erilaista olio-ohjelmointia C# kielellä. Vielä 3 kuukautta ennen kuin aloitin opinnäytetyöprosessini koodaamis osaamiseni oli melko perus tasolla. Olen opiskellut koodaamista koulussa ja verkko kurssien ja harrastusprojektieni kautta, mutta opinnäytetyön aikana osaamiseni kasvoi todella merkittävästi. Isoin kehityksen osa alue oli koodin ymmärtämisessä ja Unity kirjastojen haltuun ottamisessa. Lisäksi erilaisten serverien ymmärtäminen ja niiden rajapintojen käsittely sekä serialisointi JSON-muotoon ja deserialisointi takaisin Unity:n ymmärtämään muotoon oli itselleni uusi asia.

Oliopohjainen ohjelmointitapa on selkeästi ollut hyvä tapa tehä asioita Unreal ja Unity peli moottoreissa. Ymmärrykseni tämänkaltaisen koodaamisen hyödyistä ennen Plantabib ohjelman aloittamista oli melko hataralla pohjalla. Luin paljon muiden tekemää koodia projektin aikana ja huomasin modulaarisen järjestelmän hyödyt melko nopeasti. Ymmärrykseni koodaamisesta on kasvanut muutenkin projektin aikana, sillä uusia asioita on tullut eteen päivittäin. Näistä opiskelemistani asioista olen koostanut itselleni pienen tietopankin, johon voin palata hakemaan hyvin toimivia koodi snippettejä joita voin kierrättää muissa projekteissani.

Projektin hallinnan työkaluista scrum pohjainen HackNPlan järjestelmä osoittautui arvokkaaksi työkaluksi. Scrum pohjaisen järjestelmän isoin hyöty on tiimin välisen kommunikaation helpottaminen ja aikataulujen suunnittelu sekä vastuualueiden jako. Samalla saadaan luotua projektin työvaiheista automaattisesti dokumentaatio ja hyvä yleissilmäys meneillään olevista tehtävistä. Projekti voidaan jakaa eri boardeille ja milestoneihin jolloin isosta kokonaisuudesta saadaan rakennettua käytännöllinen ja hallittavampi pieni yksikkö. Scrum pohjainen järjestelmä auttaa myös tiimin eri henkilöiden yhteispelin parantamisessa. Koodaajat ja graafikot saavat tarttumapinnan toistensa tavoitteisiin ja näkevät paremmin, kuinka heidän työpanoksensa vaikuttaa muihin projektin osiin. Itse sain huomattavasti oppia projektin läpiviennistä tämän dynaamisen järjestelmän avulla.

Olen kehittynyt todella paljon viimeisen 8 viikon aikana myös yritystoiminnan johtamisen osalta. Erittäin hyödylliseksi koin Meet VC tapahtuman, jossa tapasin pääomasijoittajia ja sain heidän perspektiiviään auki. On huomattavasti helpompaa lähteä keskustelemaan rahoituksesta, kun ymmärtää mitä sijoittaja hakee yrityksestä. Venture Capitalist eli VC

hakee usein nopeaa tuottoa sijoitukselleen exitin kautta, kun taas enkeli sijoittaja voi hakea kiinnostavaa tiimiä, jonka kanssa lähteä tekemään yhteistyötä ja hyötymään pitemmällä aikavälillä sijoituksestaan. Tulin opinnäytetyön aikana siihen tulokseen, että peliyhtiömme hakee tulevaisuudessa nimenomaan enkelisijoittajaa sillä emme hae nopeaa kasvua vaan haluamme kehittyä rauhallisesti ja oppia pelialan mahdollisuudet parhaalla mahdollisella tavalla.

Isoa oppimista edelliseen tapahtui myös dokumentaation järjestämisessä ja lukemisessa. Tieteellisten artikkelien lukeminen todellisen luonnonympäristön dynaamisessa mallintamisessa on äärimmäisen arvokasta. On lähes mahdotonta saada aikaan realistisen oloisia ympäristöjä virtuaalitodellisuudessa, mikäli materiaalien fysikaalisten ominaisuuksien ymmärrys ei ole riittävällä tasolla. Fysikaalisten lain alaisuuksien ymmärtäminen auttaa pelimoottoreiden piirtokyvyn ymmärtämistä ja näin edesauttaa oikeaoppista materiaalien luontia. Toisaalta tieteellinen ymmärtäminen mahdollistaa uusien tekniikoiden keksimisen ja tätä kautta mahdollistaa uutta liiketoimintaa ja innovaatioita. Dokumentaatioiden kirjoittaminen on projektin loppupuolella huomattavasti ammattimaisempaa entiseen verrattuna. Rakenne dokumentaatioissa on järkevämpi ja mietitympi ja toisaalta esimerkiksi koodin kommentointi auttoi suoraan uuden koodin valmistumisessa nopeuttaen toimintaa.

Kehitettävää on vielä runsaasti mm. servereiden ymmärtämisessä ja tietokantojen käsittelyssä. Tähän uskon saavani osaamista lukemalla lisää erilaisista serveri API:stä sekä toteuttamalla erilaisia projekteja ja koulutuksia asian tiimoilta. Itseäni kiinnostaisi toteuttaa rajapinnan koodaamisen lisäksi serveri API ja etenkin tietoturva kiinnostaa itseäni.

En koe olevani vielä ammattikoodaaja, mutta pystyn nyt valmistamaan omatoimisesti isompia kokonaisuuksia ja viemään projektiosaamistani myös koodaamisen puolelle. Oma vahvuuteni projekteissa on ollut aiemmin koordinaattorin tai projekti liidin roolissa. Olen aiemmissa projekteissa ymmärtänyt pääpiirteittäin, kuinka koodi kannattaa toteuttaa, mutta pullonkaulaksi on osoittautunut syvempi tietämys koodin rakenteesta ja parhaista käytännöistä. Pystyn kehittymään valtavasti koodaajana ja mitä syvemmän osaamisen pystyn opettelemaan, sitä parempaa tukea pystyn antamaan tiimilleni.

Päiväkirjamuotoinen opinnäytetyö oli todella hyvä lähtökohta oman tason selvittämiseen ja omien ajatusten jäsentämiseen. Oli todella hyvä pystyä palaamaan viikon tapahtumiin jälkikäteen ja saada selvyyttä itselle missä kohtaa projekti etenee ja mitä seuraavaksi kannattaa tehdä. Mielestäni oli todella vapauttavaa nähdä jälkikäteen missä virheet tulevat ja pystyä myös kirjoittamaan niistä ajatukset auki. Tämän prosessin myötä olen löytänyt omia heikkouksiani ja pystyn kompensoimaan niitä paremmalla valmistautumisella seu-

raavan viikon työtehtäviin. Kirjoittaminen myös poistaa stressiä, kun tieto ei pyöri enää pelkästään oman muistin varassa ja koen päiväkirjamaisen työtavan olevan hyvä oppimisen keino itselleni ja tulen todennäköisesti jatkamaan tämän kaltaista raportointia projekteissani.

Lisäkehitystä koodaamisessa Plantabib:in osalta tulee varmasti niin JSON puolella kuin serveri puolella, kun projekti vuoden loppuun valmistuu. Itseäni kiinnostaisi opetella Unity:n Webrequestin kautta käsittelemään pelien ja ohjelmien sisäänkirjautumista ja tietoturva asioita. Näen täällä potentiaalia liiketoiminnalle, sillä iso osa tulevaisuuden maisema-arkkitehtuuri projekteista pyörii jonkinlaisen monen hengen toiminnallisuuden kautta. Myös peleissä multiplayer on edelleen yhä nousevampi trendi. Seuraavat peliprojektimme pyörivät vahvasti moninpelin ympärillä.

Isoimpia heikkouksiani ovat olleet erilaiset UI järjestelmien suunnittelut. Vaikka osaan käyttää sujuvasti kuvan- ja videonkäsittely työkaluja ja koen olevani vahvasti visuaalinen henkilö ei UI ja UX suunnittelu ollut helppoa. En näe, että tämä osa ohjelmistojen suunnittelua on minulle se kaikkein kiinnostavin osa, joten toivoisin tulevaisuudessa pystyväni keskittymään enemmän toiminnallisuuksien valmistamiseen kuin käyttöliittymän tekoon. Tekemäni mockup kuvat Plantabib ohjelmasta perustuivat isoilta osin olemassa oleviin ohjelmiin. Mielestäni on järkevä työskentelytapa katsoa mallia toimivista ohjelmistoista.

Lähteet

3dEx 7.12.2018. Substance Designer - Layered Rocks Material. Katsottavissa: https://www.youtube.com/watch?v=fG_YlgqBmL4. Katsottu 20.8.2019

Codeacademy 2019. Learn C#. Luettavissa: https://www.codecademy.com/learn/learn-c-sharp. Luettu 20.8.2019.

Codeproject 18.10.2010. Some Best Practices for C# Application Development. Luettavissa: https://www.codeproject.com/Articles/118853/Some-Best-Practices-for-C-Application-Developmen. Luettu 19.5.2019.

Inside Science 14.8.2019. Cybersickness: Why People Experience Motion Sickness During Virtual Reality. Luettavissa: https://www.insidescience.org/news/cybersickness-why-people-experience-motion-sickness-during-virtual-reality. Luettu 20.8.2019.

Jmonkeyengine 2018. Physically based rendering – Part one. Luettavissa: https://wiki.jmonkeyengine.org/jme3/advanced/pbr_part1.html. Luettu: 20.4.2019.

MuleSoft 07.09.2016. What are APIs and how do APIs work? Luettavissa: https://blogs.mulesoft.com/biz/tech-ramblings-biz/what-are-apis-how-do-apis-work. Luettu 15.5.2019.

Postman Learning Center 2019. Luettavissa: https://learning.getpostman.com/?_ga=2.23169145.218526918.1574889508-631719660.1574889508. Luettu 15.5.2019.

Procedural Worlds 2019. Introduction to CTS. Luettavissa: http://www.procedural-worlds.com/cts/?section=documentation. Luettu: 9.4.2019.

Substance 3D 2019 (1). Luettavissa: https://docs.substance3d.com/home. Luettu: 16.8.2019.

Substance 3D 2019 (2). The PBR Guide - Part 1. Luettavissa: https://academy.substance3d.com/courses/the-pbr-guide-part-1. Luettu: 16.4.2019.

Speedtree 2019. Speedtree 8 Modeler Documentation. Luettavissa: http://docs8.speedtree.com/modeler/doku.php. Luettu: 15.4.2019.

Testbytes 2019. How to Find Bugs in Game Testing. Luettavissa: https://www.testbytes.net/blog/bugs-in-game-testing. Luettu 4.5.2019.

Towards data science 2019. The Uncanny Valley in Game Design. Luettavissa: https://towardsdatascience.com/the-uncanny-valley-in-game-design-6a6c38a36486. Luettu 26.4.2019.

Työterveyslaitos 2019. Mitä työhyvinvointi on? Luettavissa: https://www.ttl.fi/perehdytys-tyohyvinvointiin-tyoterveyteen-ja-tyoturvallisuuteen/tyohyvinvointi-yhteinen-asia. Luettu 19.7.2019.

Unity 2019. Luettavissa: https://unity.com. Luettu 16.4.2019.

Unity Book of the Dead 29.6.2019. Book of the Dead: Quixel, wind, scene building, and content optimization tricks. Luettavissa: https://blogs.unity3d.com/2018/06/29/book-of-the-dead-quixel-wind-scene-building-and-content-optimization-tricks/. Luettu 20.8.2019.

Unreal Engine 2019. Luettavissa: https://www.unrealengine.com/en-US/feed. Luettu: 16.4.2019.

Unreal Engine Forum 28.4.2014. How to use Enums in Blueprint. Luettavissa: https://answers.unrealengine.com/questions/36139/using-enums-in-blueprint.html. Luettu 20.5.2019.

Virtual Learning Hub 15.8.2018. Enum (Enumerator) Variable Types - #27 Unreal Engine 4 Blueprints Tutorial Series. Katsottavissa: https://www.youtube.com/watch?v=ZxNrjJhoMVQ. Katsottu 20.5.2019.