jamk.fi

Mobiilipelitestauksen tehostaminen Unity-ympäristössä

Aleksi Toivonen

Opinnäytetyö
Syyskuu 2018
Luonnontieteiden ala
Tietojenkäsittelyn tutkinto-ohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Toivonen, Aleksi	Opinnäytetyö, AMK	Syyskuu 2018
	Sivumäärä 41	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Mobiilipelitestauksen tehostaminen Unity-ympäristössä

Tutkinto-ohjelma
Tietojenkäsittelyn tutkinto-ohjelma

Työn ohjaaja(t)
Niko Kiviaho

Toimeksiantaja(t)
Zaibatsu Interactive Oy

Tiivistelmä

Peliteollisuus on nopeasti kasvava viihteen ala. Pelit kehittyvät jatkuvasti sisällöltään suuremmiksi, viihdyttävämmiksi ja laadukkaammiksi. Merkittävä osa laadun tavoittelua on pelien testaaminen. Testaamalla peleistä löydetään viat ja ne saadaan täten hiottua mahdollisimman toimivaksi ja viihdyttäväksi.

Toimeksiantaja Zaibatsu Interactive on Keski-Suomen suurin pelialan yritys. Zaibatsu tavoittelee peleissään mahdollisimman korkeaa laatua, joten pelien testaaminen perinpohjaisesti on heille tärkeää. Toimeksiantaja halusi tehostaa pelitestausprosessiaan erityisesti Unity-ympäristössä kehitettävissä projekteissa, sillä se on heidän pääasiallinen kehitysympäristönsä.

Toimeksiantajan testausprosessia tehostettiin kehittämistutkimuksen keinoin. Tutkimuksessa hankittiin taustatietoa pelitestauksesta yleisellä tasolla sekä erityisesti Unity-ympäristön tarjoamista työkaluista. Tämän lisäksi haastateltiin toimeksiantajan työntekijöitä heidän kokemuksistaan pelien testaamisesta.

Hankitun teorian ja haastattelujen pohjalta toimeksiantajalle kehitettiin Unity-ympäristössä toimiva ohjelmistopohjainen ratkaisu, joka tehostaa pelitestaamista Unity-peleissä. Ratkaisu on kokoelma työkaluja, jotka helpottavat tai nopeuttavat aikaisemmin käytettyjä testitoimenpiteitä, ja mahdollistavat kokonaan uudenlaisia menetelmiä hyödynnettäväksi toimeksiantajan testausprosessissa.

Kehittämistutkimuksesta saatiin hyödyllistä lisätietoa pelitestauksesta, ja kehityksen tuloksena syntyi toimiva työkalupaketti. Työkalut on helppo ottaa käyttöön uusissa projekteissa ja niiden arkkitehtuuri sallii jatkokehittämisen tulevaisuudessa.

Avainsanat (asiasanat)

Unity, pelikehitys, pelitestaus, ohjelmointi

Muut tiedot (salassa pidettävät liitteet)

jamk.fi

Description

Author(s)	Type of publication	Date
Toivonen, Aleksi	Bachelor's thesis	September 2018
		Language of publication: Finnish
	Number of pages: 41	Permission for web publication: x

Title of publication

Enhancing mobile game testing in Unity environment

Degree programme

Business Information Systems

Supervisor(s)

Kiviaho, Niko

Assigned by

Zaibatsu Interactive Oy

Abstract

The videogame industry is an incredibly quickly growing area of entertainment. Games are constantly evolving in size, entertainment value and quality. A crucial part of pursuing high quality in game development is game testing. Developers need testing to find the flaws in their games and can thus polish their products to function properly and be as entertaining as possible.

The assignor, Zaibatsu Interactive, is the largest game company operating in Central Finland. Zaibatsu develops games with quality as a priority, thus testing is very important to them. The company wanted to enhance their game testing process, especially in projects developed with Unity, because Unity is their primary development environment.

Design research was used as a method to work on the testing process. The research consisted of gathering information about game testing on a general level, and additionally tools available in Unity. Additionally, Zaibatsu's employees were interviewed about their experiences in game testing.

Based on the gathered theory and interviews, a software-based solution for enhancing game testing in Unity environment was developed. The solution is a collection of tools to ease and quicken previously familiar game testing methods, and additionally, allow new kinds of methods to support the game testing process.

The research results present a great deal of useful knowledge and understanding of the industry was learned, and the development portion resulted in a set of tools designed to be easy to use and an architecture allowing further development in the future.

Keywords/tags (subjects)

Unity, game development, game testing, programming

Miscellaneous (Confidential information)

Sisältö

Kuviot

Käsitteet

Bugi: Ongelma, jonka seurauksena laite tai tietokoneohjelma toimii odottamattomalla tavalla ja päätyy tuloksiin, joihin sen ei pitäisi päätyä. Johtuu virheellisestä logiikasta tai logiikan puutteesta. (What is a bug n.d.) Peleissä bugi voi olla esim. näkymätön seinä, vääränäköinen hahmo tai reikä maassa. (Levy & Novak 2010, 5.)

Build: Buildaaminen (*rakentaminen*) tarkoittaa lähdekoodin prosessointia ja kokoamista sellaiseen muotoon, että kohdelaite voi ajaa ohjelman itse. Buildaamisen tulos on buildi, joka on esimerkiksi tietokoneohjelma tai peli. (What is Build? n.d)

Debug: Debug tarkoittaa bugien etsimistä ja poistamista ohjelmasta. (Debug Meaning in the Cambridge English Dictionary n.d.)

QA: *Quality Assurance*, *Laadunvarmistus*. Prosessi, jossa varmistetaan, että kohde täyttää määritykset ja vaatimukset. (What is Quality Assurance n.d.)

Yksikkötesti: Koodi, joka ajaa toisen koodin, ja arvioi tämän toisen koodin palauttamien arvojen kelvollisuutta. Yksikkötesteillä arvioidaan, toimiiko koodi oikein. (JavaScript Unit Testing 2013)

1 Johdanto

Peliteollisuus on nopeasti kasvava ala, ja kilpailu kiristyy jatkuvasti. Videopelit ovat nousseet television ja elokuvien ohi suosituimmaksi ja tuottavimmaksi viihdemuodoksi (D'Argenio 2018). Pelikehittäjien on tehtävä peleistä jatkuvasti aikaisempaa mielenkiintoisempia, viihdyttävämpiä ja laadukkaampia, jotta he pystyvät kilpailemaan kansainvälisillä markkinoilla. Laadukkaita pelejä ei voi tehdä ilman testaamista, sillä testaamalla peleistä löydetään viat ja ongelmat, jotka haittaavat pelikokemusta ja sen myötä myyntiä ja yrityksen menestystä. Peliteollisuuden kasvun sekä pelien ja teknologioiden kehittymisen myötä myös testaamisen tarve kasvaa jatkuvasti.

Toimeksiantaja Zaibatsu Interactive haluaa tuottaa pelaajille mahdollisimman laadukkaita pelejä, joten pelien testaaminen on heille tärkeää. Toimeksiantaja toivoo, että voisi saada kehitettyä pelitestaamisprosessiaan entistäkin tehokkaammaksi. Toimeksiantajan pääasiallinen kehitysympäristö peliprojekteissa on Unity, joten opinnäytetyön tavoitteessa priorisoidaan Unity-ympäristössä kehitetyissä peleissä tapahtuvaa testaamista.

Testausprosessin tehostaminen pohjustetaan ensin tutkimalla pelitestauksen teoriaa, ja sen jälkeen tutustumalla toimeksiantajan pelitestauksen nykytilanteeseen. Teorian tutkimisen ja alkukartoituksen jälkeen kehitetään ohjelmistopohjainen ratkaisu, joka auttaa toimeksiantajaa testaamaan Unity-pelejään tehokkaammin.

2 Tutkimusasetelma

Työn toimeksiantaja Zaibatsu Interactive on 2014 perustettu startup-yritys. Zaibatsu on Keski-Suomen suurin pelialan yritys.

Zaibatsu pyrkii pelikehityksessään korkeaan laatuun. Testaus on erittäin merkittävä osa laadun tavoittelua, joten testauksen kehittäminen on Zaibatsulle tärkeää.

2.1 Tavoite

Tutkimuksen tavoite on hankkia tietoa pelitestauksesta ja tehostaa toimeksiantajan pelitestausprosessia Unity-ympäristössä. Pelitestauksesta hankittavaan taustatietoon

kuuluu pelitestauksen kehitys aikojen saatossa, pelitestauksen teoria, ja Unity-ympäristöön liittyvän pelitestaamisen tutkiminen.

Tutkimuksen tuloksena syntyy ratkaisu, joka tehostaa toimeksiantajan pelitestausta Unity-ympäristössä luoduissa peleissä, sillä suurin osa heidän peleistään toteutetaan Unityn työkaluilla. Ratkaisun tulee nopeuttaa ja helpottaa testausprosessia ja auttaa toimeksiantajaa ylläpitämään tuotteidensa laatutasoa vähemmällä vaivalla.

2.2 Tutkimusmenetelmät

Opinnäytetyö toteutetaan kehittämistutkimuksena. Kehittämistutkimukseen liittyy halu saada ymmärrystä ilmiöstä ja saada aikaan muutos parempaan suuntaan. (Kananen 2012, 26.) Kehittämistutkimus sopii toimeksiantoon siksi, että toimeksiantajalla on ongelma, johon halutaan kehittää ratkaisu, ja aiheesta, eli pelitestaamisesta, halutaan lisää ymmärrystä. Tutkimuksessa käytetään kvalitatiivisia eli laadullisia menetelmiä. Ongelman ratkaisun kehittämiseen tarvitaan ideoita ja mielipiteitä toimeksiantajan työntekijöiltä, ja tähän tarkoitukseen käyttökelpoisia vastauksia saadaan paremmin laadullisilla kuin määrällisillä menetelmillä. Laadullisen tutkimuksen tärkeimmät tiedonkeruumenetelmät ovat havainnointi, haastattelut ja erilaiset dokumentit. (Kananen 2012, 93.) Näistä käytetään ainakin havainnointia, kun osallistutaan toimeksiantajan projekteihin, sekä haastattelua, kun tarvitaan kuulla työntekijöitä.

2.3 Tutkimuskysymykset

Tutkimuskysymyksiksi valittiin seuraavat kysymykset:

- Mitä on pelitestaus?
- Millainen on toimeksiantajan nykyinen testausprosessi?
- Miten toimeksiantajan testausprosessia voisi tehostaa?

Ensimmäiseen kysymykseen vastataan hankkimalla tietoa pelitestauksen taustasta ja menetelmistä. Toiseen vastataan osallistumalla toimeksiantajan projekteihin ja kartoittamalla heidän pelitestausprosessiaan. Kolmanteen vastataan kehittämistutkimuksen tuloksilla. Pelitestausprosessin selvitys ja kehitys rajataan Unity-ympäristössä työskentelyyn, koska Unity on toimeksiantajan pääasiallinen kehitysympäristö.

3 Pelitestaus

Tässä luvussa tutustutaan pelien testaamiseen: miten pelejä on testattu aiemmin ja miten niitä testataan nykyään. Lisäksi selvitetään mikä on pelitestaajan tehtävä, mihin tavoitteisiin testaamisella pyritään ja mitä menetelmiä testauksesa käytetään.

3.1 Historia

Peliteollisuuden alkuaikoina pelikonsolit vastasivat tehoiltaan enemmän laskimia kuin tietokoneita. Niissä oli erittäin vähän laskutehoa ja muistia, ja yhdellä konsolilla pystyi pelaamaan yhtä peliä, tai parhaimmillaan variaatioita kyseisestä pelistä. Pelien yksinkertaisuuden takia pelitestaamiselle ei ollut tarvetta. Henkilö joka koodasi pelin, myös testasi sitä. (Levy & Novak 2010, 6.)

Vuonna 1976 julkaistu Fairchild Channel F-konsoli muutti peliteollisuutta merkittävästi. Ensimmäistä kertaa yhdellä konsolilla pystyi pelaamaan loputtomasti erilaisia pelejä, kun konsoliin kiinnitti pelin sisältävän ROM-moduulin. Muistirajoitteet helpottuivat ja peleistä saattoi näin tulla monipuolisempia, ne pystyivät nyt sisältämään esim. hahmoja, taustakuvia ja vihollisia. Pelit olivat siltikin helppoja testata, yksi testaaja pystyi muutamassa tunnissa varmistamaan, että peli toimii odotetulla tavalla. (Mts. 7.)

1980-luvun alussa videopelien menestys oli nousussa ja pelit toivat julkaisijoille suuria määriä rahaa. Tästä seurauksena markkinoille alkoi ilmestyä huonolaatuisia pelejä. Ahneet julkaisijat päästivät rahan toivossa myyntiin nopeasti tehtyjä ja huonolaatuisia pelejä, mistä seurasi videopelimarkkinoiden romahtaminen Yhdysvalloissa. Japanilainen Nintendo kuitenkin selvisi pelimarkkinoiden romahtamisesta vahingoitta. Nintendo julkaisi Famicon (myöhemmin länsimaissa NES) laitteensa, joka oli monella osa-alueella aikaansa edellä. Famicon/NES herätti itsestään Yhdysvaltojen pelimarkkinat uudelleen henkiin. Laite kykeni ajamaan huomattavasti laajempia ja laadukkaampia pelejä kuin mitä aiemmin oli nähty. Mittavampien pelien myötä pelikoodaajat eivät enää voineet itse todeta, ettei pelissä ole virheitä. Lisäksi pelijulkaisijat alkoivat vaatimaan peleiltään enemmän, ja asettivat tuotteilleen laatustandardeja

(esim. Nintendo's Seal of Quality). Ensimmäistä kertaa videopelien historian aikana tarvittiin ammattimaisia videopelitestaajia. (Mts. 8–10.)

Nykyään pelejä pelataan yhä vaativammilla ja monimutkaisemmilla laitteilla. Pelikonsolit sisältävät omat käyttöjärjestelmänsä ja moniydinsuorittimet. Pelit tukevat monia eri resoluutioita ja internetin yli pelaaminen on erittäin yleistä. PC-pelien täytyy tukea satoja eri laitteisto- ja ohjelmistokokoonpanoja. Nämä kaikki muuttujat ovat moninkertaistaneet pelitestaamisen tarpeen ja vaativuuden. (Mts. 20–21.)

3.2 Hyvin testatun pelin ominaisuudet

Davisin (2016) mukaan ohjelmistojen ja pelien kehityksessä on hyvä tavoitella seuraavaksi selitettäviä ominaisuuksia. Myös tuotteen testaamisen onnistumista voidaan arvioida sillä, miten hyvin tuote täyttää seuraavat ominaisuudet.

Kokonaisuus
Tekeekö koodi kaiken sen, mitä sen on tarkoitus tehdä?

Virheettömyys
Toimiiko koodi ilman virheitä?

Suorituskyky ja resurssienkäyttö
Onko koodin suorituskyky ja resurssienkäyttö hyväksyttävällä tasolla?

Luotettavuus/vakaus
Voiko koodiin luottaa? Osaako koodi käsitellä odottamattomia tapahtumia ja huonoa dataa?

Hauskuus (pelikehityksessä)
Tekeekö peli kaiken tämän, ja on kaiken lisäksi hauska pelata?

Kaikissa näissä ominaisuuksissa ei voi yhtäaikaisesti tavoitella täydellisyyttä. Todella suuri osa ohjelmistokehityksestä on valintojen tekoa ja osa-alueiden tasapainottamista, jotta työ voidaan saada joskus päätökseen.

Pelikehityksessä listan viimeisin ominaisuus eli hauskuus on usein ensimmäisenä mielessä peliä testatessa. Täytyy kuitenkin muistaa, että jos listan muita ominaisuuksia ei

ota huomioon, pelin hauskuus kärsii. Peli ei voi olla hauska, jos se ei toimi. (Davis 2016)

3.3 Pelitestaaja

Pelitestaaminen ei tarkoita sitä, että istuu päivät kotisohvalla pelaten uusia pelejä uusimmilla konsoleilla. (Levy & Novak 2010, 28) Schultzin ja Bryantin (2017, 21) mukaan pelitestaajalla on 2 tarkoitusta. Ensimmäinen on löytää vikoja pelin koodista tai designista. Toinen tarkoitus on osoittaa, mitkä pelin ominaisuuksista toimii tarkoitetulla tavalla, ja mitkä eivät. Levyn ja Novakin (2010, 49.) mukaan testaajan tarkoitus on ensisijaisesti löytää bugeja, etsiä askeleet bugin toistamiseen, ja ilmoittaa bugeista. Testaajien toissijainen tarkoitus on varmistaa, että bugit on korjattu, ja että peliä on hauska pelata.

Pelitestaajan työ vaatii muutakin kuin halua ja taitoa pelata pelejä. Pelien testaaminen vaatii kurinalaisuutta, kypsyyttä ja sinnikkyyttä. Tavalliset pelaajat pelaavat pelejä pitääkseen hauskaa, mutta pelitestaajan täytyy kyetä pelaamaan peliä niin, että tarkoituksena on olla pitämättä hauskaa. (Levy & Novak 2010, 37.) Pelitestaajat suurimman osan ajastaan testaten pelejä, jotka ovat niin keskeneräisiä, ettei niitä vielä ole ollenkaan hauska pelata. Senkin jälkeen, kun peli on jo hauska pelata, testaustehtävät eivät yleensä ole ollenkaan hauskoja. Testaaja voi myös joutua pelaamaan pelejä, jotka eivät häntä kiinnosta ollenkaan. (Raga, 2017.)

Usein pelit ovat niin isoja, että niitä testaa suuri joukko testaajia. Tästä seuraa se, että yksittäinen testaaja usein testaa vain yhtä pelin osa-aluetta, eikä saa aina pelata peliä kokonaisuutena. Pelitestaajan työ isoissa projekteissa on usein sitä, että testaaja pelaa kymmeniä tunteja pelin samaa kohtaa, yrittäen saada se toimimaan väärällä tavalla. (Most Common Game Testing Questions, N.d.)

Pelitestaajan täytyy osata kommunikoida tehokkaasti. Testaajan täytyy osata ilmoittaa löytämästään bugista niin tarkasti, että pelikehittäjä osaa korjata bugin kysymättä lisätietoa. Jos pelitestaajan ja koodaajan täytyy käydä keskustelua siitä, miten bugi toistuu, on testaajan raportointi ollut vajaata, ja aikaa on mennyt hukkaan. (Most Common Game Testing Questions, N.d.)

3.4 Pelitestauksen menetelmät

Jotta pelien ongelmista voitaisiin tunnistaa mahdollisimman suuri osa, niitä testataan monilla eri tavoilla. Tässä luvussa tutustutaan yleisimpiin pelitestauksen menetelmiin sekä niiden käyttötarkoituksiin.

Black box & white box

Kaiken erilaisen pelitestaamisen voi jaotella näihin kahteen kategoriaan. Suurin osa pelitestaamisesta on ”black box”-testaamista. Tämä tarkoittaa sitä, että testaajalla ei ole pääsyä pelin lähdekoodiin. Testaajalla on käytössä samat sisään- ja ulostulolaitteet kuin tavallisella pelaajalla, esim. tietokone, hiiri, näppäimistö ja näyttö, tai pelikonsoli, ohjain ja televisio. Testaaja antaa pelille syötteitä, esim. napin painallus tai liiketunnistimen heilautus, ja tarkkailee pelin tulosteita, kuten kuvaa, ääntä, ohjaimen värinää tai tallennettuja tiedostoja. Testaaja reagoi tulosteisiin ja muuttaa sen perusteella seuraavaa syötettään. Testaaja simuloi näin tavallista pelaajaa, joka on ostanut pelin ja pelaa sitä omalla laitteellaan. (Schultzin & Bryant 2017, 120–121.)

White box-testaamisessa testaajalla on mahdollisuus tutkia ja käyttää pelin lähdekoodia, toisin kuin tavallisilla pelaajilla. White box-testit ovat esimerkiksi yksikkötestejä, tai muita vastaavia ohjelmoituja testejä, jotka testaavat koodin toimivuutta erilaisissa tilanteissa. White box-testaus on hyödyllistä, kun halutaan testata joitain pelin yksittäisiä moduuleja tai jotain tiettyä osaa koodista. Testien syöte ja tuloste riippuvat täysin siitä, mitä dataa testattava koodi vaatii toimiakseen, mitä vaikutuksia koodilla on sen ympärillä oleviin muuttujiin, ja mitä arvoja koodi palauttaa. Pelin testaaminen ainoastaan white box-menetelmillä ei ole järkevää, pelkästään koodia lukemalla on yleensä mahdotonta arvioida kaikki mahdolliset tilanteet, joita pelissä voi ilmentyä. (Schultz & Bryant 2017, 122–123.)

Black box-testaaminen sallii tuotteen testaamisen oikean käyttäjän tavoin, eikä testaajan tarvitse osata mitään ohjelmointikieliä. Testauksen voi myös toteuttaa muut kuin tuotteen kehittäjät, ja testit voidaan suunnitella ennen kuin koodia on kirjoitettu. Black box-testien syötteiden määrä kuitenkin rajoittuu siihen, mitä testaaja kykenee syöttämään, joten kaikkia mahdollisia lopputuloksia ei yleensä voida testata. (Black Box Testing – Software Testing Fundamentals. N.d) White box-testit vaativat osaavia tekijöitä; testaajan täytyy ymmärtää lähdekoodia ja osata ohjelmoida testejä.

Koska white box-testit liittyvät suoraan lähdekoodiin, ne ovat erittäin yksilöllisiä, joten niiden ylläpito voi myös viedä aikaa. White box-testejä voidaan kuitenkin toteuttaa ilman käyttöliittymää, joten toisin kuin black box-testausta, white box-testejä voidaan hyödyntää ennen kuin tuote on täysin toimintakykyinen. White box-testeillä pystytään myös käymään läpi huomattavan paljon enemmän mahdollisia lopputuloksia kuin black box-testeillä. (White Box Testing – Software Testing Fundamentals. N.d)

Combinatorial

Combinatorial eli kombinatorinen- tai yhdistelmätestaaminen tarkoittaa pelin testaamista mahdollisimman monilla eri muuttujien yhdistelmillä, esimerkiksi pelin eri kenttien testaamista eri laitteilla, eri grafiikka-asetuksilla, eri hahmoilla ja eri vaikeustasoilla. (Schultz & Bryant 2017, 153–154.)

Clean room

Clean room testaaminen tarkoittaa pelin pelaamista niin kuin erilaiset pelaajat pelaavat sitä. Peliä täytyy testata eri kokemuksen ja eri pelityylin omaavien pelaajien tyyleillä. Jos testaaja pelaa peliä vain omalla tavallaan, häneltä voi jäädä huomaamatta jokin vika pelissä. Osa pelaajista pelaa rauhallisemmin, osa taas painelee eri näppäimiä todella nopeasti ja osaa käyttää erilaisia näppäinyhdistelmiä. Vähemmän kokeneet pelaajat tekevät vain asioita joita heille opetetaan pelin aikana, kokeneemmat keksivät omia lähestymistapojaan pelin haasteisiin, ja jotkut taas tekevät kaikkensa huijatakseen pelissä. (Schultz & Bryant 2017, 223–226.)

Ad hoc

Ad hoc on latinaa ja tarkoittaa karkeasti ”tiettyä tarkoitusta varten”. Ad hoc testaamisessa pyritään vastaamaan tiettyyn kysymykseen. Testit voidaan jakaa kahteen alalajiin, *free* ja *directed*.

Vapaissa (*free*) testeissä testaaja saa pelata peliä omalla tyylillään, ja samalla improvisoida testejä. Testaaja pohtii, ”mitä tapahtuu, jos teen asian X?” ja testaa pohdintojaan. (Schultz & Bryant 2017, 273–274.)

Ohjatuissa (*directed*) testeissä testaaja pyrkii pelaamalla löytämään vastauksen ennen pelaamista määritettyyn kysymykseen, esim. ”voinko ostaa ainutlaatuisen esineen useammin kuin yhden kerran?” (Mts. 276–277.)

Gameplay/Playtesting

Pelikehityksessä ei riitä, että peli toimii suunnitellulla tavalla. Hyötyohjelmistoissa toimivuus on riittävää, mutta pelin täytyy olla toimivan lisäksi viihdyttävä. Pelin hauskuudelle (”Fun factor”) ei ole sääntöjä, joten pelin hauskuutta arvioidaan kokeilemalla peliä. Ei kuitenkaan riitä, että peli on hauska ensimmäisellä kerralla. Pelin täytyy pystyä säilyttämään viihdearvonsa, ja tämä taataan testaamalla peliä uudelleen ja uudelleen, arvioiden samalla sen mekaniikkoja. (Levy & Novak 2010, 66.)

Aiemmin mainitut testausmenetelmät vastaavat testikysymykseen faktalla, esimerkiksi ”bugi X on korjattu, bugi Y tapahtuu edelleen”. Gameplay-testaamisen tulos arvioi subjektiivisia arvoja, kuten pelin hauskuutta, tasapainoa ja vaikeutta, esimerkiksi vastauksella ”pelin ensimmäinen puolisko on viihdyttävä, mutta puolivälin jälkeen vastustajat ovat aivan liian vaikeita ja peli on turhauttava”. (Schultz & Bryant 2017, 284.)

3.5 Testaus pelikehityksen eri vaiheissa

Davisin mukaan (2016) pelin testauksen ei pitäisi olla erillinen vaihe pelin kehityksessä tai eri osaston vastuulla, vaan testaus on jatkuvasti osa pelin kehitystä. Testaus on ketterä prosessi, joka kehittyy pelin kehityksen myötä.

Pelien kehitysprosessi voidaan jakaa useaan eri vaiheeseen. Pelitestaus ilmenee näissä vaiheissa eri tavoilla: joissain vaiheissa testaus keskittyy koodin toimivuuteen, toisissa taas pelin viihdyttävyyteen. Seuraavaksi vertaillaan pelitestauksen ilmenemistä näissä eri vaiheissa.

Prototyyppi-vaihe

Pelin prototyyppi on vain suuntaa antava kokeilu pelistä. Prototyypin ei tarvitse olla toimiva kokonaisuus, joten prototyyppi vaiheen testaaminen on turhaa. Prototyyppiä tarvitsee testata vain silloin, jos siitä aletaan jatkokehittämään varsinaista tuotetta. (Davis 2016.)

Esituotanto

Pelien testaaminen alkaa heti tuotannon alussa. Alkupään testaaminen on lähinnä white box testaamista: koodista ja sisällöstä täytyy alusta asti tehdä toimivaa. Projek-

tin alussa on myös hyvä suunnitella testaamista. Silloin asetetaan testaamiselle säännöt ja ohjeistukset, määrätään testauksen johtaja ja määritellään peliltä vaadittavia ominaisuuksia. (Schultz & Bryant 2017, 94–97.)

Alpha

Alphavaiheeseen siirrytään, kun peli täyttää alussa määritellyt ominaisuusvaatimukset. Pelin ydin on nyt pääosin sellainen, kuin siitä aluksi suunniteltiin. Alphavaiheessa pelin ominaisuuksia testataan ja hiotaan tai hylätään. Alphassa peliin kasataan runsaasti uusia ominaisuuksia, joten myös uutta testattavaa syntyy paljon. (Schultz & Bryant 2017, 108)

Alphavaiheen tarkoitus ei ole tehdä bugivapaata peliä, vaan tehdä ominaisuuksiltaan ”valmis” peli, jonka voi pelata alusta loppuun. Pelissä kuitenkin on vielä paljon bugeja ja jonkin verran väliaikaista sisältöä, pääasia onkin saada ominaisuudet paikoilleen. (Levy & Novak 2010, 52.)

Beta

Kun pelin ominaisuudet ja sisältö on lyöty lukkoon, siirrytään Betaan. Betassa alkaa niin sanottu ”kiillotus”. (Levy & Novak 2010, 53.) Uusia ominaisuuksia ei enää lisätä, joten testaaminen on bugien metsästämistä, laitteistojen yhteensopivuuden testaamista ja paljon muuta laadunvarmistamiseen liittyvää työtä. Tässä vaiheessa testaajat varmistavat sen, että pelistä voi nauttia, se toimii ja siinä ei ole liikaa bugeja. (Schultz & Bryant 2017, 110–112)

Betavaihe voi olla joko suljettu tai avoin. Suljettu beta on juurikin aiemmin mainittua pelin kiillottamista ja pelituntuman viihdyttävyyden varmistamista. Avointa betaa käytetään jatkuvasti yleistyvissä online-peleissä, joita pelataan internetin kautta. Pelikehittäjät eivät usein kykene itse testaamaan online-pelejä, varsinkaan MMO, eli massively multiplayer online pelejä. MMO-peleissä voi olla pelaamassa yhtä aikaa jopa tuhansia pelaajia, joten pelipalvelun stressitestaamiseen tarvitaan ulkoista apua. Ulkoiset testaajat ovat yleensä tavallisia pelaajia ilman mitään työsopimusta; heidän palkkio on se, että he saavat pelata peliä ilmaiseksi ennakkoon. (Levy & Novak 2010, 54–55.)

Gold

Kun julkaisu lähestyy, siirrytään Gold vaiheeseen. Tässä vaiheessa varmistetaan, että kaikki merkittävät tai pelin ”rikkovat” bugit eli pelisession keskeyttävät, tai pelikokemuksen pilaavat bugit on korjattu. Peleihin jää väkisinkin joitain pieniä bugeja. Näistä suurin osa olisi hyvä olla tiedossa, ja pelaajille tulisi ilmoittaa tiedetyistä ongelmista, joita ei julkaistavasta versiosta ehditä korjata. (Schultz & Bryant 2017, 112–114)

Julkaisun jälkeen

Julkaisun jälkeen testaamisen tarve vähenee. Jos pelistä tulee ilmi merkittäviä ongelmia, on peliin syytä julkaista päivitys, jossa näitä korjataan. Päivityksien testaaminen suunnitellaan päivityskohtaisesti. Jos julkaistaan useampia päivityksiä, täytyy lisäksi testata, että päivitykset asentuvat ongelmitta pelin aikaisempien versioiden päälle. (Schultz & Bryant 2017, 116.)

4 Työkalut

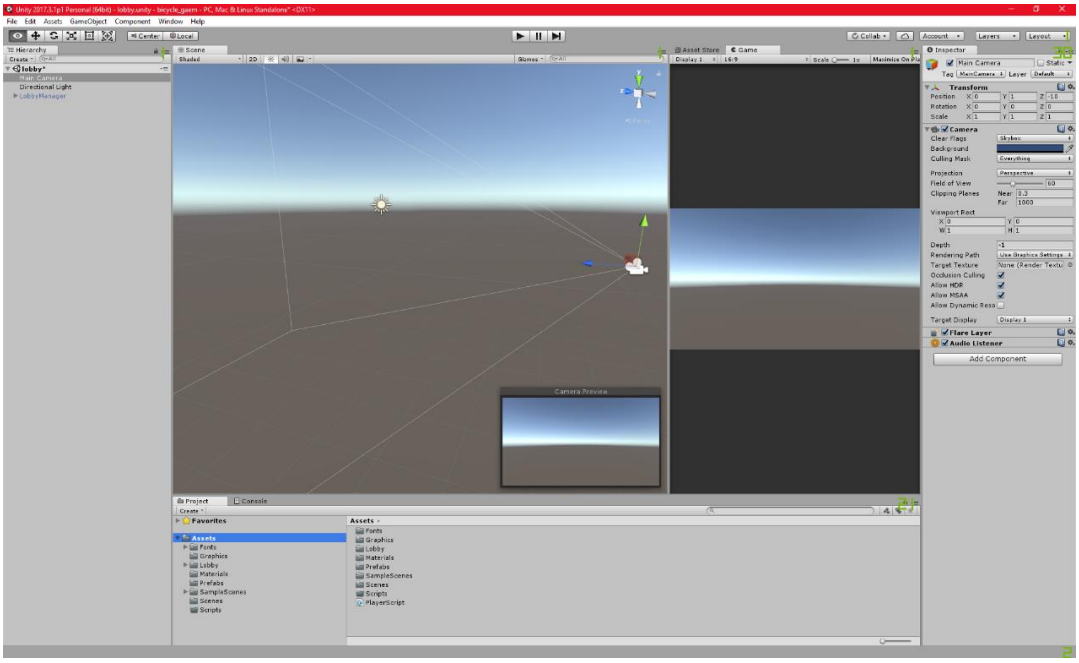
Seuraavaksi tutustutaan niihin toimeksiantajan käyttämiin työkaluihin, joilla on yhteys pelitestaukseen.

4.1 Unity

Unity on Unity Technologiesin kehittämä alusta, jolla voi luoda 2D, 3D, VR (virtuaalitodellisuus) ja AR (lisätty todellisuus) pelejä ja sovelluksia. Unity tunnetaan parhaiten heidän pelimoottorista ja siihen sisältyvästä editorista (Kuvio 1.), mutta Unity tarjoaa näiden lisäksi myös työkaluja ja muita resursseja, esimerkiksi Unity Asset Store verkkokaupan, Unity Analytics analytiikkapalvelun ja Unity Ads mainospalvelun. Unity on suosituin pelimoottori mobiilipelien kehityksessä, 34 % suosituimmasta 1000 ilmaisesta mobiilipelistä vuonna 2016 oli tehty Unitylla (Unity – Fast Facts n.d.)

Toimeksiantajan omat peliprojektit, sekä monet asiakasprojekteista tehdään Unity-moottorilla ja niissä myös hyödynnetään Unityn muita palveluja.

Unity tarjoaa käyttäjilleen muutamia työkaluja pelien testaukseen. Osa näistä löytyy editorista ja osa on palveluita tai sovelluksia, jotka kommunikoivat pelin kanssa joko editorissa tai valmiissa buildissa.



Kuvio 1. Kuvakaappaus Unity editorista

Test Runner

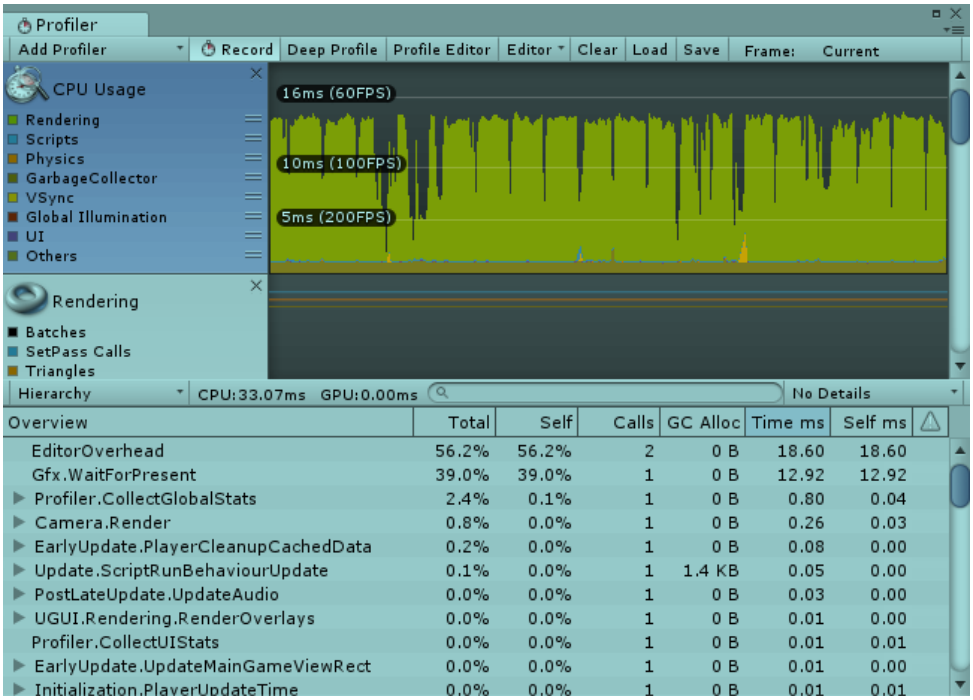
Editorista löytyy Unity Test Runner niminen työkalu, jolla voi yksikkötestata pelin koodia. Testejä voi ajaa sekä Edit-tilassa (peliä ei ajeta) että Play-tilassa (peliä pelataan editori-ikkunassa). (Unity – Manual: Unity Test Runner n.d.)

Unity Analytics

Unityn analytiikkapalvelu Unity Analytics soveltuu erinomaisesti ulkoiseen alpha ja beta pelitestaukseen. Pelin kehittäjät voivat kerätä Analyticsiin tilastoja haluamistaan pelin ominaisuuksista ja toiminnoista. Analytiikan avulla voidaan mitata pelaajien etenemistä pelissä ja arvioida pelin eri ominaisuuksien toimivuutta (Analytics | Unity 2018)

Profiler

Suorituskykytestausta varten Unity-editorista löytyy Profiler-ikkuna. Profiler havainnollistaa käyrillä ja prosenttilukemilla (Kuvio 2.), kuinka paljon resursseja pelin eri toimintojen (esimerkiksi animaatiot, ruudulle piirtäminen, logiikka) suorittamiseen käytetään. Sen avulla voidaan paikantaa pelissä olevat suorituskykyä haittaavat suunnitteluvirheet ja bugit. (Unity – Manual: Profiler n.d.)



Kuvio 2. Kuvakaappaus Unity editorin Profilerista

Unity Remote

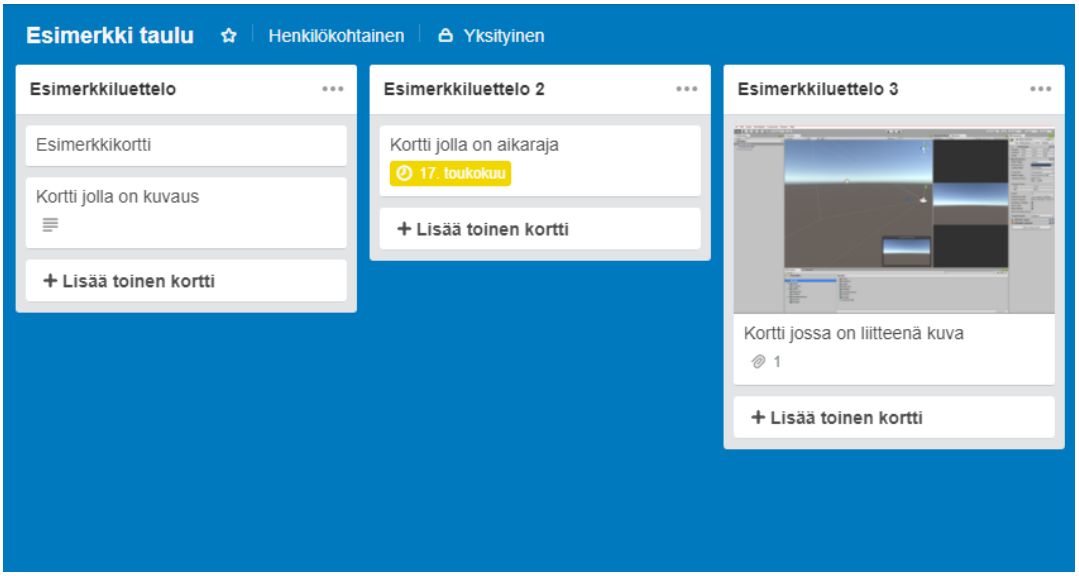
Mobiilitestaamista varten Unity on kehittänyt Unity Remote sovelluksen, joka on saatavilla Android, iOS ja tvOS alustoille. Remote sovellusta käytetään yhdessä Unity editorin kanssa. Kun editorissa laitetaan Play-tila päälle, editori lähettää USB-kaapelilla kytkettyyn mobiililaitteeseen pelikuvaa, ja vastaanottaa laitteesta syötteitä, kuten kosketukset, kiihtyvyys- ja kallistusanturien lukemat ja GPS koordinaatit.

Peliä ei tässä tilanteessa oikeasti ajeta mobiililaitteella, siinä vain näytetään pelikuvaa tietokoneelta, joten Remotea ei voida käyttää mobiilipelin suorituskykytestaukseen. Remoten avulla mobiililaitteella saadaan kuitenkin testattua aiemmin lueteltuja mo-

biililaitteille tyypillisiä syötteitä poikkeuksellisen nopeasti, sillä Remote poistaa tarpeen buildata sovellus tai peli mobiililaitteille joka kerta, kun halutaan testata jotain muutoksia. (Unity – Manual: Unity Remote n.d.)

4.2 Trello

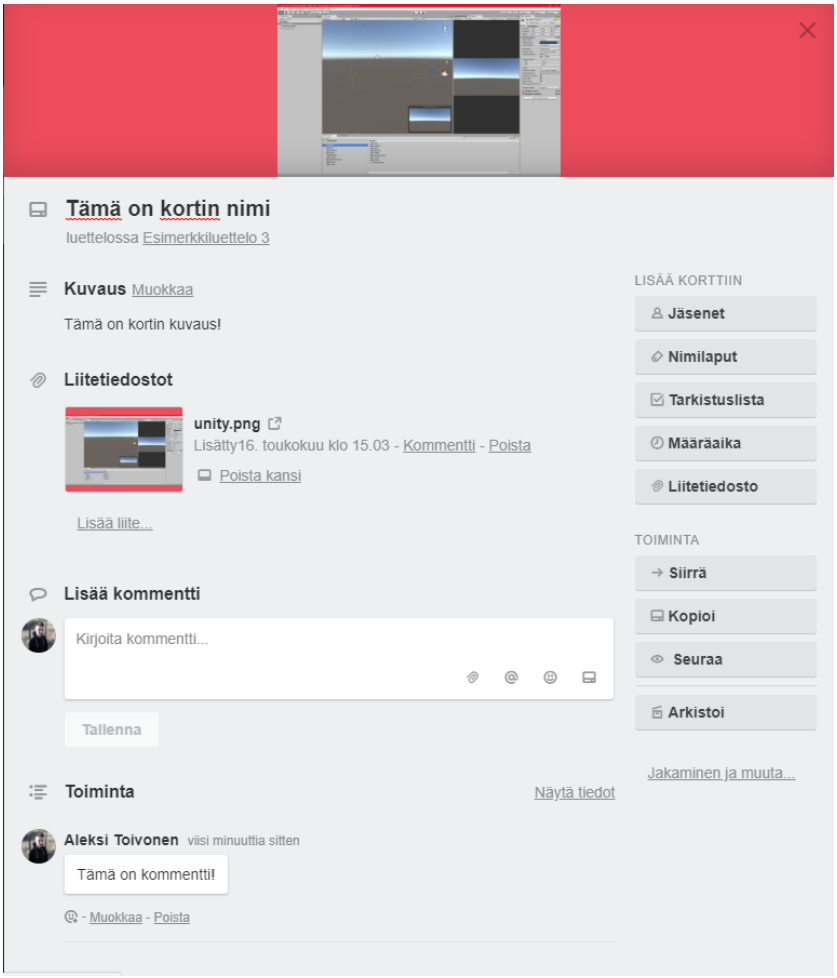
Trello on projektinhallintasovellus, jossa luodaan tauluille luetteloja ja kortteja projektien seuraamisen helpottamiseksi. Luetteloiden nimet voivat olla vaikka projektin tehtävien eri vaiheet, esim. ideat, tehtävät, kesken, ja valmiit. Kortit ovat yksittäisiä tehtäviä, kuten ”soita henkilölle X” tai ”osta asia Y”. Kortteihin voi lisätä liitteitä, kuten tiedostoja, tarkistuslistoja, tai aikarajoja. Kokeneemmat käyttäjät voivat myös halutessaan käyttää lisäominaisuuksia (”Power-Up”), joihin kuuluu esim. korttien ikääntyminen ja äänestäminen. Trelloa voi käyttää verkkoselaimella, sekä työpöytä- ja mobiilisovelluksella. (Trello n.d.)



Kuvio 3. Kuvakaappaus Trello-taulusta

Kuviossa 3 näkyy Trello-taulu nimeltä ”Esimerkki taulu”, jolla on 3 luetteloa (Esimerkkiluettelot 1-3). Jokaisessa luettelossa on kortteja, joissa on erilaista sisältöä.

Kortin voi avata painamalla sitä, jolloin kortin sisällön näkee tarkemmin (Kuvio 4.). Tässä näkymässä korttiin voi lisätä esimerkiksi kuvauksen, kommentteja ja liitteitä.



Kuvio 4. Trello-kortin tiedot

5 Alkukartoitus

Ennen ratkaisun kehitystä täytyi selvittää, millaista pelitestausta toimeksiantajalla tehdään. Kartoittamalla toimeksiantajan lähtötilanne voidaan selvittää mitkä pelitestauksen osa-alueet vaativat kehittämistä.

Alkukartoitus rajattiin Unity-ympäristössä tehtäviin projekteihin. Se toteutettiin osallistumalla toimeksiantajan projekteihin, ja keskustelemalla toimeksiantajan työntekijöiden kanssa heidän kokemuksistaan Unityn työkalujen käytöstä. Osallistumalla projekteihin saatiin kattava käsitys toimeksiantajan testausprosesseista eri kokoisissa, sekä toimeksiantajan omissa, että asiakasprojekteissa.

5.1 Laitteisto

Pelitestauksessa laitteistolla on suuri merkitys, kaikki bugit eivät välttämättä toistu kaikilla eri laitteilla. Pelien testaaminen eri laitteilla on osa kombinatorista testaamista. (Schultz & Bryant 2017, 154.)

Zaibatsulla on käytössä sekä Mac-, että Windows-tietokoneita. Applen Mac laitteita on pakko olla, jos haluaa kehittää ja julkaista sovelluksia Applen sovelluskauppaan. (Unity - System Requirements n.d.) Monia eri-ikäisiä ja -tehoisia tietokoneita Zaibatsulla ei ole. He eivät kuitenkaan kehitä tietokonepelejä, eikä tarvetta kombinatoriselle tietokonepelien testaukselle ole, joten tämä ei ole ongelma.

Toimeksiantaja kehittää pääasiassa mobiilisovelluksia ja -pelejä, joten sillä alueella laitteiston täytyisi olla monipuolista. Applen iOS laitteita ja eri valmistajien Android laitteita löytyy yhteensä 8 kappaletta.

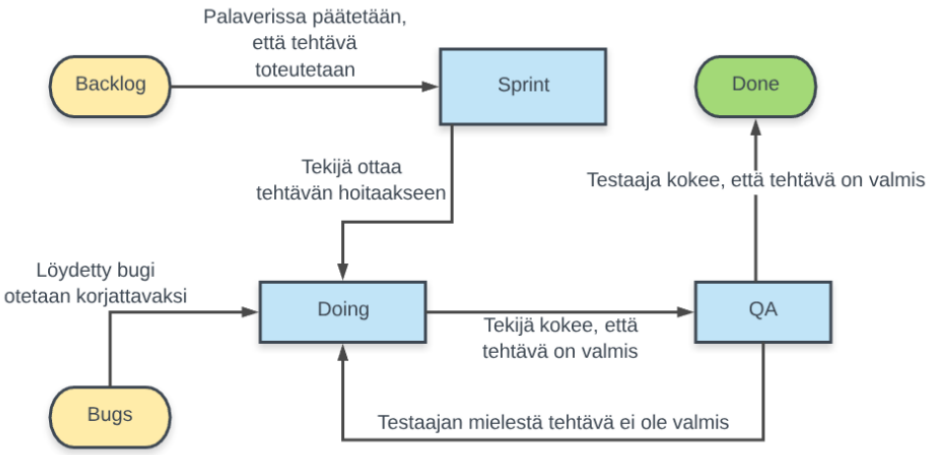
Saatavilla olevat mobiililaitteet ovat kaikki 2–6 vuotta vanhoja keskihintaisia laitteita. Uusia ja tehokkaita, tai matalatehoisia ja vanhoja mobiililaitteita ei ole. Unity-pohjaisten mobiilipelien minimi Android-versio on 4.1 ja minimi iOS-versio on 7.0. (Unity - System Requirements n.d.) Zaibatsun laitteiden vanhin Android-versio on 5.1 ja iOS-versio on 9.3.5, joten juuri ja juuri minimivaatimukset täyttäviä testilaitteita ei ole. Myöskään uusimpia käyttöjärjestelmiäversioita omaavia laitteita ei ole hankittu.

Laitteistossa siis esiintyy puutteita sekä ala- että yläpäässä kun katsotaan tehoja, käyttöjärjestelmäversioita ja laitteiden ikää. Tätä onnistutaan jonkin verran paikkaamaan sillä, että työntekijät testaavat tuotteita joskus omilla laitteillaan, jolloin kattavuutta saadaan lisää, mutta äärimmäisen hyvät ja huonot laitteet puuttuvat silti.

5.2 Trellon käyttö

Merkittävä osa toimeksiantajan sisäisestä pelitestauksesta tapahtuu yksittäisten Trello-korteissa määriteltyjen tehtävien laadunvarmistuksena. Zaibatsun jokaisella projektilla on oma Trello-taulu, joissa pyritään noudattamaan samoja periaatteita. Jokainen projekti etenee hieman eri tavalla ja tarvitsee erilaisia luetteloita, mutta tauluilta löytyy lähtökohtaisesti luettelot Backlog, Sprint, Doing, QA, Done, Bugs, Documentation.

Backlog-luettelossa sijaitsee tehtävät, joita on jossain projektin vaiheessa tarkoitus toteuttaa. Sprint-luettelossa sijaitsee ajankohtaiset tehtävät, Doing-luettelossa on sillä hetkellä työstettävät tehtävät ja QA-luettelossa on testausta tarvitsevat tehtävät. Bugs-luetteloon luodaan projektin bugeja ja ongelmia kuvaavat kortit. Documentation-luettelo sisältää hyödyllistä tietoa projektista, esim. käytettävien työkalujen versiot tai hyödyllisiä muistiinpanoja ja ohjeita. Korttien liikkuminen luetteloiden välillä havainnollistetaan kuviossa 5.



Kuvio 5. Trello-kortin kulku luettelojen välillä

Pelitestaus ilmenee siis Trellossa seuraavasti.

- Tehtävä on tekijän mielestä valmis, ja hän siirtää kortin QA luetteloon.
- Testatessa löytyy bugi tai ongelma, jolloin testaaja luo kortin Bugs-luetteloon. Kortille määrätään korjaaja riippuen ongelman tyypistä. Korjaaja ratkaisee ongelman ja siirtää kortin QA luetteloon.
- Projektin QA vastaava, tai muu sillä hetkellä testausta toteuttava henkilö käy läpi QA luettelon kortteja, ja testaa ja arvioi, täyttyykö kortin tehtävän kuvaus. Jos täyttyy, kortti siirretään Doneen.

5.3 Unityn työkalujen hyödyntäminen

Toimeksiantaja hyödyntää Unityn testityökaluja vaihtelevasti. Jotkin työkalut ovat olleet merkittävissä asemissa projekteissa, kun taas toisia ei ole käytetty lainkaan.

Profiler

Unity editorin profilerin käytön hallitsee vain osa Zaibatsun koodaajista. Profilerin käytölle ei ole koettu olevan paljoa tarvetta, joten sen käyttö on jäänyt vähäiseksi. Kaikki Zaibatsun koodaajat eivät ole koskaan käyttäneet profileria.

Test Runner

Unityn Test Runneria ei ole Zaibatsulla käytetty lainkaan. Uuden koodin testaamiseen käytettävät white box menetelmät rajoittuvat koodin muuttujien arvojen tarkkailuun editorissa pelaamisen aikana.

Unity Remote

Lähes kaikki Zaibatsun Unity-ympäristössä työskentelevät henkilöt ovat kokeilleet Remote sovellusta aikaisemmin. Sovellusta ei kuitenkaan koettu tarpeeksi helpoksi tai hyödylliseksi, joten sen käyttö Zaibatsulla on lopetettu.

Unity Analytics

Analytiikka on Zaibatsulle tärkeä osa ulkoista alpha- ja beta-vaiheiden testaamista. Zaibatsu käyttää Analyticsin avulla kerättyjä tilastoja apuna pelien suunnittelussa ja kehityksessä. Zaibatsun Unity Analyticsia hyödyntävät pelit lähettävät ennalta määrätyissä tilanteissa Analyticsiin merkin, että pelaaja on edennyt tiettyyn pisteeseen tai suorittanut tietyn toiminnon. Pelin pelaajista saadaan näin tilastoja, joiden pohjalta voidaan päätellä mikä pelissä on hyvää ja mikä huonoa, tarkastelemalla esimerkiksi, että missä vaiheessa peliä suurin osa pelaajista lopettaa. Zaibatsulla analytiikkaa on käytetty erityisesti pelikehityksen alphavaiheessa auttamaan kehittäjiä päättämään, mitä pelin osa-alueita kannattaa erityisesti kehittää eteenpäin. Zaibatsun työntekijöistä kovinkaan moni ei ole Analyticsia itsenäisesti käyttänyt, vaan analytiikkatoteutukset ovat olleet samojen henkilöiden vastuulla.

Pohdinta Unityn työkalujen käytöstä

Profilerin osaaminen on Zaibatsun työntekijöiden keskuudessa erittäin vähäistä. Profiler on Unityn omista työkaluista ainoa, joka keskittyy puhtaasti suorituskyvyn testaamiseen, mutta vain pari työntekijää kertoo hyödyntäneensä sitä. Kaikkien sitä ei tarvitse osata, esimerkiksi graafikot, jotka eivät työstä koodia, eivät hyötyisi profiler osaamisesta paljoa, mutta koodaajien olisi hyvä osata käyttää sitä.

Test runneria ei ole käytetty lainkaan. Uudet lisätyt ominaisuudet testataan usein black box- ja whitebox-menetelmien sekoituksilla, esimerkiksi tarkkailemalla muuttujien arvoja pelin lokiin lähetetyistä debug-viesteistä samalla kun testaaja pelaa peliä. Test runnerin käyttöä kannattaisi kuitenkin harkita tulevaisuudessa, jos jokin projekti sisältää niin monimutkaista logiikkaa, ettei sitä black box-menetelmillä pysty luotettavasti testaamaan.

Unity Remote sovellus ei aikaisemmin soveltunut tarpeisiin. Sovellus koettiin riittämättömäksi ehkä myös siksi, ettei tiedetty mihin kaikkeen se pystyy. Tulevaisuudessa sitä kannattaisi kokeilla uudestaan silloin, kun sen tuomat edut säästäisivät aikaa. Remotella voidaan esimerkiksi vähentää kertoja, jolloin GPS-signaalia hyödyntävä peli tarvitsee buildata.

Analyticsin käyttö on ollut toimivaa ja hyödyllistä, sen käytön laajentamiselle ei tällä hetkellä näyttäisi olevan tarvetta. Voisi kuitenkin olla hyvä, että useampi koodaaja osaisi tehdä peliin Analytics toteutuksen, koska tämän hetken osaaminen ei välttämättä riitä, jos käynnissä sattuisi olemaan useampi analytiikkaa tarvitseva projekti.

6 Toteutus

Toimeksiantaja toivoi, että heille toteutettaisiin Unity-ympäristössä toimivia työkaluja, jotka helpottavat pelien testausta.

6.1 Suunnittelu

Kehitystyötä aloittaessa tarvittiin toimeksiantajan työntekijöiden toiveita ja ideoita Unity-ympäristön pelitestauksen tehostamiseen. Työntekijöitä haastateltiin keskustelunomaisesti ryhmissä. Ryhmäkeskustelujen ansiosta jokaisen henkilön esittämistä ideoista pystyttiin saman tien käymään ryhmässä arvioivaa keskustelua, ja yksittäisten henkilöiden ideat saattoivat herättää uusia ideoita toisissa henkilöissä. Ryhmäkeskusteluiden avulla ideoista pystyttiin rajaamaan kehitykseen eniten mielenkiintoa ja kannatusta herättäneet ideat. Haastatteluissa myös kartoitettiin työntekijöiden kokemus Unityn tarjoamien testityökalujen käytöstä.

Keskustelua käytiin pelitestauksen ongelmista ja hidasteista, sekä toiveominaisuuksista, joita testityökaluilta halutaan. Työntekijät halusivat pääasiassa tehdä testaamisesta yleisesti helpompaa ja nopeampaa. Toiveista yleisin oli aikaisempien projektien testiominaisuuksien toteuttaminen uudelleenkäytettävään muotoon niin, ettei kaikissa uusissa projekteissa tarvitsisi käyttää aikaa samojen ominaisuuksien luomiseen.

Koska Zaibatsu tekee lähes poikkeuksetta mobiilipelejä ja -sovelluksia, päätettiin työkalut tehdä ensisijaisesti mobiilialustalla pelin aikana käytettäväksi. Työkalut toimivat kuitenkin myös ainakin tietokonepelien kehityksessä.

Davis toteaa Testing for Game Development artikkelissaan (2016) että peliä testatessa tulisi testata pelin buildia, eikä pelkästään ajaa peliä editorissa. Davisin mukaan on myös erittäin tärkeää tehdä pelin testaamisesta tarpeeksi helppoa ja nopeaa, jotta testaamisesta saadaan kurinalaista ja se ei jää tekemättä. Tämä tukee täydellisesti työntekijöiden toiveita siitä, että työkalujen tulisi helpottaa ja nopeuttaa testaamista, sekä päätöstä tehdä työkaluista oikeassa mobiililaitteelle asennettavassa buildissa toimivia.

6.2 Ominaisuuksien rajaus

Keskustelujen perusteella työkalujen ominaisuuksiksi rajautui seuraavat pääpiirteet.

Uudelleenkäytettävyys

Työkalut täytyy pystyä ottamaan käyttöön uusissa projekteissa nopeasti ja helposti. Työkalujen päätavoite on testaamisen tehostaminen, ja ajan säästäminen on merkittävä osa tätä tavoitetta. Työkalujen käyttöönoton pitäisi viedä huomattavasti vähemmän aikaa kuin se, että luo sen tärkeimmät ominaisuudet uuteen projektiin tyhjästä.

Scenen vaihtaminen

Unity Scenet ovat eräänlaisia näkymiä. Ne voivat olla valikoita, ympäristöjä tai läpäistäviä tasoja. (Unity – Manual: Scenes. N.d)

Eräs suosittu toive työkaluille oli mahdollisuus siirtyä scenestä toiseen helposti. Yleensä peleissä scenen vaihtaminen voi edellyttää tason läpäisyä tai oikean napin painamista tietyssä valikossa, joten tietyn scenen avaaminen manuaalisesti voi nopeuttaa testaamista huomattavasti. Scenen vaihtaminen on ominaisuus, joka on

usein testaamista varten jouduttu uusiin projekteihin tekemään uudelleen. Toteuttamalla se tähän työkalupakettiin säästetään aikaa tulevissa projekteissa.

Timescalen vaihtaminen

Unity-peleissä timescale tarkoittaa ajan nopeuden kerrointa, 1 on normaali nopeus, 0 tarkoittaa että peli on pysähtynyt. (Unity – Scripting API: Time.timeScale. 2018)

Hidastamalla peliä sitä on helpompi pelata, ja nopeuttamalla voidaan ohittaa tapahtumia, joiden odottamiseen ei haluta käyttää aikaa. Ajan nopeuttamisella voidaan säästää aikaa ja hidastamalla tarkemmin testata oikeita asioita. Säätämällä aikaa voi myös testata eri nopeuksien vaikutusta pelin viihdyttävyyteen.

Bugiraporttien lähettäminen

Kuten alkukartoituksessa ilmeni, Trello on merkittävä osa Zaibatsun projektihallintaa. Kun testatessa pelistä löytyy bugi tai jokin vika, siitä kirjataan kortti Trelloon. Testityökalujen tulee helpottaa Trello-korttien tekemistä, ja liittää Trello-kortteihin tietoa, joita niihin käsin olisi todella hankala liittää. Korttien lähettäminen pelin sisältä säästää merkittävästi aikaa.

Overlay

Overlay tarkoittaa englannin kielessä jonkin asian peittämistä toisella. (Overlay Meaning in the Cambridge English Dictionary. N.d.) Overlay näyttää pelin aikana testaajalle pelin toiminnasta hyödyllistä tietoa piirtämällä pelin päälle tekstiä. Tällä voidaan säästää uusissa projekteissa aikaa, sillä lähes jokaisessa projektissa peliin lisätään testitarkoituksessa tekstikenttiä, jotka tulostavat ruudulle pelin logiikasta ja suorituskyvystä olennaisia arvoja, kuten FPS-lukeman eli ruudunpäivitysnopeuden.

Ruudunkaappaukset ja videonauhoitus

Bugien havainnollistamista varten testityökaluilta toivottiin ruudunkaappausominaisuuksia. Testityökaluilla pitäisi pystyä ottamaan kuvia tai videokuvaa pelistä, ja lähettämään materiaali Trelloon. Kuvat ja videot helpottavat kommunikointia huomattavasti, mutta mobiililaitteilla on aiemmin ollut epäkäytännöllistä lähettää ruudunkaappauksia Trelloon. Kuvien ja videoiden jakamisen helpottaminen tehostaa kommunikaatiota merkittävästi, ja tehokas kommunikaatio todettiin Pelitestaaja osiossa tärkeäksi osaksi pelitestausta.

Debug Log

Debug viittaa bugien löytämiseen ja poistamiseen. (Debug Meaning in the Cambridge English Dictionary. N.d.) Kun Unity-peliä testataan Unity-editorissa, pelin lähdekoodin heittämiä debug-viestejä luetaan editorin Console-ikkunasta. (Unity – Manual: Console. N.d.) Vaikka Console onkin erittäin keskeinen testausväline, mobiililaitteilla sitä ei voi lukea. Testityökaluilla tulee olla mahdollista lukea buildatussa pelissä debug-viestejä, joita näkee tavallisesti vain editorin consolesta, ja suodattaa sieltä tietynlaiset viestit. Pelissä näkyvät Debug log viestit mahdollistaa whitebox-testaamisen elementtien tuomisen buildien testaamiseen.

6.3 Valmiit ominaisuudet

Tekninen toteutus

Työkalut ohjelmoitiin C#-ohjelmointikielellä, koska se on käytetyin ja parhaiten tuettu Unity-ympäristön sallima ohjelmointikieli. (UnityScript's long ride off into the sunset – Unity Blog 2017.)

Kehittämistyön tuloksena syntyi työkalut sisältävä Asset package (.unitypackage tiedosto). Asset package on Unityn tapa jakaa koodia ja muuta sisältöä eri projektien välillä. (Unity – Manual: Asset Packages. N.d.) Paketti tehtiin yhteensopivuussyistä Unityn versiolla 2017.2.0f3, koska se oli aloitushetkellä vanhin aktiivisesti käytetty Unityn versio toimeksiantajan projekteissa. Jos työkalut olisi tehty uusimmalla mahdollisella Unityn versiolla, vanhemmissa projekteissa olisi voinut tulla eteen yhteensopivuusongelmia.

Paketti sisältää Unity-objektin, jonka projektin koodaaja lisää haluamaansa Unity-sceneen, työkalujen käyttämän grafiikan, sekä kehittämisen aikana kirjoitetun koodin ja koodin riippuvuudet, jotka muodostavat työkalujen toiminnallisuuden.

Käyttöliittymä

Työkalujen graafinen käyttöliittymä toteutettiin työajan säästämiseksi Unity-editorin mukana tulevilla elementeillä, ja muutamalla täysin vapaaseen käyttöön lisensoidulla kuvalla, jotka löytyivät internetistä. Käyttöliittymän grafiikat on helppo vaihtaa uusiin, jos toimeksiantaja haluaa parantaa ulkoasua.

Työntekijöiden mielestä oli tärkeää, että työkalut saa helposti esille ja piiloon. Työkaluvalikon saa avattua pienestä rattaan muotoisesta napista, jota voi myös siirtää ruudulla raahaamalla. Työkalut ovat näin piilossa, poissa tieltä ja ne saa helposti avattua tarvittaessa. Työkaluvalikon taustan läpinäkyvyyttä voi säätää riippuen siitä, haluaako testaaja nähdä pelin valikon läpi, vai nähdä pelkän valikon.

Scene-työkalut

Valikossa voi listata kaikki projektin näkymät eli scenet, ja vaihtaa niiden välillä. Testaaja voi myös ladata nykyisen scenen uudestaan, tai ladata oletus-scenen, eli pelin ensimmäisen scenen.

Timescale-työkalut

Työkaluilla voi säätää haluamansa timescalen ja ottaa sen halutessaan käyttöön. Pelin timescale menee automaattisesti nollaan aina kun testityökalut avataan, eli peli pysähtyy täysin, kun testaaja selaa testityökalujen valikoita.

Bugiraportteri

Aikaisemmin Trello-kortit tehtiin aina joko tietokoneella tai Trellon mobiilisovelluksella. Testityökalujen bugiraportterilla Trello-kortin voi nyt tehdä suoraan pelin sisältä. Testityökalut liittävät korttiin monenlaista hyödyllistä tietoa. Korttiin liitetään testilaitteen tiedot, mikä on tärkeä osa kombinatorista testausta. Lisäksi korttiin liitetään tilastoja pelin suorituskyvystä, ruudunkaappaus tai videokuvaa pelistä, sekä muuta projektikohtaista lisätietoa. Projektikohtaisella lisätiedolla voidaan esimerkiksi ilmaista, mitä tasoa pelaaja pelasi löytäessään bugin, ja näin ongelman korjaajan on helpompi paikallistaa vika. Näiden kaikkien lisätietojen lisääminen Trello-korttiin manuaalisesti oli aiemmin joko mahdotonta tai vei useita minuutteja, kun nyt Testityökaluilla tämän saa tehtyä muutamassa sekunnissa.

Overlay

Overlaysta tehtiin helposti päälle ja pois kytkettävä, jotta he, jotka haluavat nähdä tilastoja voivat nähdä niitä. Overlay piirtää pelin päälle elementtejä, jotka näyttävät pelin toiminnasta tietoja, esim. ruudunpäivitysnopeuden tai GPS-koordinaatit. Overlay-elementtien koodista tehtiin mahdollisimman helppolukuista, jotta uusiin projekteihin on nopea koodata uusia projektikohtaisia elementtejä.

Ruudunkaappaukset ja videonauhoitus

Testityökaluilla pystyy ottamaan pelistä kuvan, lisäämään siihen havainnollistavia tarroja (esimerkiksi ympyrä tai nuoli) ja tekstiä. Testityökalut hyödyntävät myös erästä Unityn lisäosaa nimeltä Everyplay. Everyplay on pelivideopalvelu, joka aloitti toimintansa lopettamisen juuri testityökalujen kehityksen aikana. (Everyplay Developers. N.d.) Everyplay kuitenkin julkaisi videotyökaluistaan vielä viimeisen version, jossa lisäosaa voi käyttää ilman internetyhteyttä. Testityökaluihin lisätty Everyplay nauhoittaa jatkuvasti videokuvaa peliä testatessa. Nämä muokatut ruudunkaappaukset ja videoleikkeet on mahdollista liittää bugiraportterissa uuteen Trello-korttiin.

Debug Log

Työkalut kykenevät kaappaamaan pelin taustalla liikkuvat varoitukset, virheet ja muut viestit, joita Unity-editorissa luettaisiin consolesta. Pelien buildeissa näitä viestejä ei normaalisti pysty lukemaan, mutta nämä viestit listataan testityökalujen Debug Log sivulle, josta niitä voi lukea, ja suodattaa tyypin mukaisesti (varoitus, virhe, viesti). Kaapatut viestit myös lisätään bugiraportterilla luotuun uuteen Trello-korttiin, joten testityökalut ovat mahdollistaneet koodivirheiden raportoinnin mobiililaitteelta käsin.

7 Pohdinta

Työn tavoite oli tehostaa toimeksiantajan pelitestausta. Toimeksiantaja Zaibatsu Interactive halusi tehostaa pelitestaustaan kehittämällä Unity-pelien testausta helpottavia työkaluja. Tutkimuksen tuloksena saatiin laajempaa ymmärrystä pelitestauksen eri menetelmistä ja tarkoituksista. Tutkimuksen ja alkukartoituksen pohjalta onnistuttiin kehittämään toimeksiantajan työntekijöiden toiveiden mukainen työkalukokonaisuus, joka täyttää työn tavoitteen. Työkalujen ominaisuudet rajattiin työntekijöiden toiveiden mukaan, ja lopullinen toteutus suunniteltiin tekniikan ja työajan rajoitusten mukaan priorisoiden työntekijöiden yleisimpiä toiveita sekä niitä ominaisuuksia, joiden kehittämistä pelitestaamisen teoriasta hankittu tieto tuki eniten.

Erityisesti työssä onnistui testityökalujen kehitys. Työkaluihin saatiin implementoitua kaikki työntekijöiden keskeisimmät toiveet ja myös ominaisuuksia, joiden

toteutuksen onnistumisesta ei suunnitteluvaiheessa voinut olla aivan varma ajan riittävyyden ja teknisten esteiden takia. Kaikki kehitystyön aikana eteen tulleet tekniset haasteet saatiin ratkaistua, eikä mikään merkittävä toivottu ominaisuus jäänyt puuttumaan kokonaisuudesta osaamisen tai ajan loppumisen takia. Joitain ominaisuuksia täytyi karsia ajan puutteen takia, mutta ne eivät olleet prioriteettejä. Puuttuvista ominaisuuksista kirjoitettiin ylös kuvailevat muistiinpanot siltä varalta, että kokonaisuuteen halutaan tulevaisuudessa kehittää lisäominaisuuksia.

Pelitestauksen teorian osuus testityökalujen kehityksessä ei ollut niin merkittävä kuin se olisi voinut olla. Pelitestauksen historiasta, eri menetelmistä, tyyleistä ja tekniikoista saatiin paljon tietoa, mutta testityökalut toteutettiin enimmäkseen työntekijöiden toiveiden pohjalta. Kun tehdään työkaluja työntekijöille, on tietysti hyvä, että työkalut ovat heidän tarpeiden mukaisia, mutta ehkä työkaluja olisi voinut kehittää myös enemmän teorian näkökulmasta ja alan ammattilaisten oikeiksi toteamien testauskäytänteiden mukaisesti. Pelitestauksen teorian tutkiminen auttoi kuitenkin merkittävästi työkalujen ominaisuuksien suunnittelussa ja priorisoimisessa, sekä ymmärtämään paremmin erilaisten testaustyylien tarpeen.

Lähteet

Analytics | Unity. 2018. Unity Technologies. Viitattu 28.5.2018. https://unity.com/solutions/analytics

Black Box Testing – Software Testing Fundamentals. N.d. Viitattu 6.8.2018. http://softwaretestingfundamentals.com/black-box-testing/

D'Argenio, A. M. 10.7.2018. Statistically, video games are now the most popular and profitable form of entertainment. Gamecrate. Viitattu 10.8.2018. https://www.gamecrate.com/statistically-video-games-are-now-most-popular-and-profitable-form-entertainment/20087

Davis, A. 27.4.2016. Testing for Game Development. Viitattu 3.8.2018. http://www.what-could-possibly-go-wrong.com/testing-for-game-development/?utm_source=ash&utm_medium=code-project-article&utm_campaign=testing-for-game-development

Debug Meaning in the Cambridge English Dictionary. N.d. Cambridge University Press. Viitattu 17.7.2018. https://dictionary.cambridge.org/dictionary/english/debug

Everyplay Developers. N.d. Everyplay Team. Viitattu 17.7.2018. https://developers.everyplay.com/

Fine, R. 11.8.2017. UnityScript's long ride off into the sunset – Unity Blog. Unity Technologies. Viitattu 7.8.2018. https://blogs.unity3d.com/2017/08/11/unityscripts-long-ride-off-into-the-sunset/

Hazem, S. 2013. JavaScript Unit Testing. Viitattu 28.5.2018. https://janet.finna.fi, ProQuest, EBook central.

Kananen, J. 2012. Kehittämistutkimus opinnäytetyönä. Jyväskylän ammattikorkeakoulu.

Levy, L. & Novak, J. 2010. Game Development Essentials. Game QA & Testing. International edition. Delmar Cengage learning.

Overlay Meaning in the Cambridge English Dictionary. N.d. Cambridge University Press. Viitattu 17.7.2018. https://dictionary.cambridge.org/dictionary/english/overlay

Raga, S. 23.6.2017. 12 Secrets of Video Game Testers. Mentalfloss. Viitattu 6.8.2018. http://mentalfloss.com/article/502132/12-secrets-video-game-testers

Schultz, C.P. & Bryant R.B. 2017. Game Testing. All in one. Third edition. Mercury learning.

The Most Common Game Tester Questions. N.d. Gamedesigning.org. Viitattu 6.8.2018. https://www.gamedesigning.org/video-game-tester/

Trello. N.d. Atlassian. Viitattu 15.5.2018. https://trello.com/home

Unity – Fast Facts. N.d. Unity Technologies. Viitattu 8.5.2018. https://unity3d.com/public-relations

Unity – Manual: Asset Packages. N.d. Unity Technologies. Viitattu 7.8.2018. https://docs.unity3d.com/Manual/AssetPackages.html

Unity – Manual: Console. N.d. Unity Technologies. Viitattu 17.7.2018. https://docs.unity3d.com/Manual/Console.html

Unity – Manual: Profiler. N.d. Unity Technologies. Viitattu 25.5.2018. https://docs.unity3d.com/Manual/Profiler.html

Unity – Manual: Unity Remote. Unity Technologies. N.d. Viitattu 28.5.2018. https://docs.unity3d.com/Manual/UnityRemote5.html

Unity – Manual: Unity Test Runner. Unity Technologies. N.d. Viitattu 25.5.2018. https://docs.unity3d.com/Manual/testing-editortestsrunner.html

Unity – Manual: Scenes. N.d. Unity Technologies. Viitattu 6.8.2018. https://docs.unity3d.com/Manual/CreatingScenes.html

Unity - Manual: Writing and executing tests in Unity Test Runner. N.d. Unity Technologies. Viitattu 25.5.2018. https://docs.unity3d.com/Manual/PlaymodeTestFramework.html

Unity – Scripting API: Time.timeScale. N.d. Unity Technologies. Viitattu 6.8.2018. https://docs.unity3d.com/ScriptReference/Time-timeScale.html

Unity – System Requirements. N.d. Unity Technologies. Viitattu 16.5.2018. https://unity3d.com/unity/system-requirements

What is a bug? N.d. Techopedia Dictionary. Viitattu 26.4.2018. https://www.techopedia.com/definition/3758/bug

What is Build? N.d. Techopedia Dictionary. Viitattu 28.5.2018. https://www.techopedia.com/definition/3759/build

What is Quality Assurance? N.d. Techopedia Dictionary. Viitattu 15.5.2018. https://www.techopedia.com/definition/9038/quality-assurance-qa

White Box Testing – Software Testing Fundamentals. N.d. Viitattu 6.8.2018. http://softwaretestingfundamentals.com/white-box-testing/

Liitteet

Liite 1. Työntekijöiden ryhmähaastattelujen kysymykset

Unity profiler

- Oletko käyttänyt? (Miksi et? Miten olet ratkonut suorituskykyongelmia?)

Unity analytics

- Oletko käyttänyt? (Miksi et ole? Miksi et käytä enää?)

Unity Test Runner

- Oletko käyttänyt? (Miksi et ole? Miksi et käytä enää?)

Unity Remote

- Oletko käyttänyt? (Miksi et ole? Miksi et käytä enää?)

Zaibatsun pelitestaaminen

- Millaisia ongelmia pelien testaus ja laadunvarmistus prosessissa mielestäsi on? (Hidasteita, turhaa työtä, epäselvyyksiä?)
- Mitä ominaisuuksia toivoisit pelitestaukseen suunnatulta Unity-assetilta?

Liite 2. Testityökalujen asennus- ja käyttöohjeet

Testityökalujen käyttöohjeet

Contents

Aleksi Toivonen 7/2018

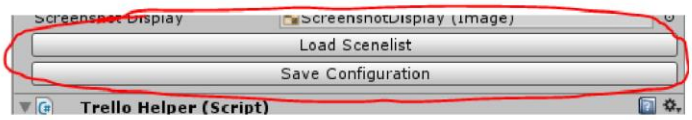
Asennus

1. Importtaa UnityPackage projektiin
    a. Jos importtaa Android Manifestit, buildi saattaa kaatua koska Unity ei kykene mergeemään näitä Manifesteja. Tämä johtuu yleensä seuraavasta rivistä Everyplayn AndroidManifest tiedostossa
    `<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" android:maxSdkVersion="18" />`
    Kyseisestä rivistä voi poistaa tuon maxsdkversion propertyn, jolloin Unity osaa mergeä muiden vastaavien write_externalien kanssa
2. Lisää TestTools/Prefabs kansiosta TestToolsCanvas prefab sceneen, jossa haluat sen initialisoida, mieluusti pelin ensimmäinen scene / splash scene tms.

Scenejen konffaus

Jotta työkaluilla voidaan vaihtaa scenejä, scenelista täytyy ladata editorissa.

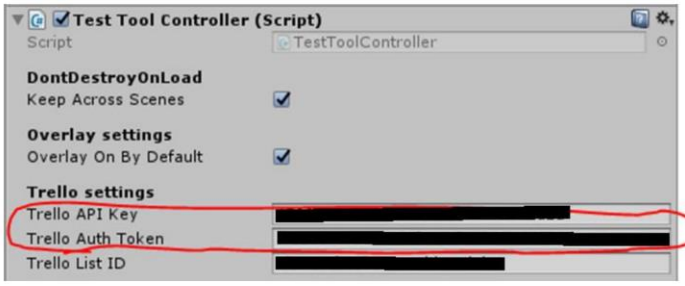
1. Paina TestToolCanvaksen TestToolController komponentin alaosasta ”Load Scenelist” ja sen jälkeen ”Save Configuration”.



Tämä täytyy tehdä uusiksi aina, kun projektin buildsettingsien scenelista muuttuu, tai scenet siirretään projektissa eri kansioon.

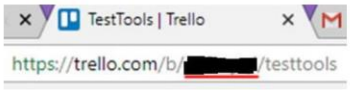
Trellon konffaus

Trellon käyttö vaatii Api keyn ja Auth tokenin. Nämä saa osoitteesta https://trello.com/app-key



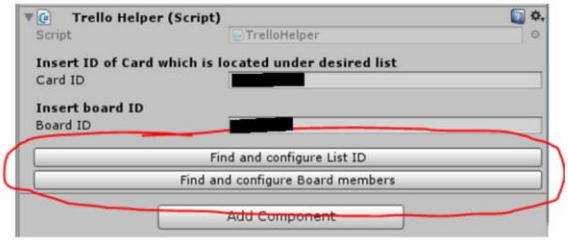
Työkalut tarvitsee Trello-korttien lähettämistä varten Lista-ID:n ja käyttäjien ID:iden hakemista varten Board-ID:n.

TestToolCanvaksessa on komponentti TrelloHelper. Syötä sille nettiselaimesta Board-ID



ja Card-ID kortilta, joka on siinä listassa, minne uudet kortit halutaan lähettää

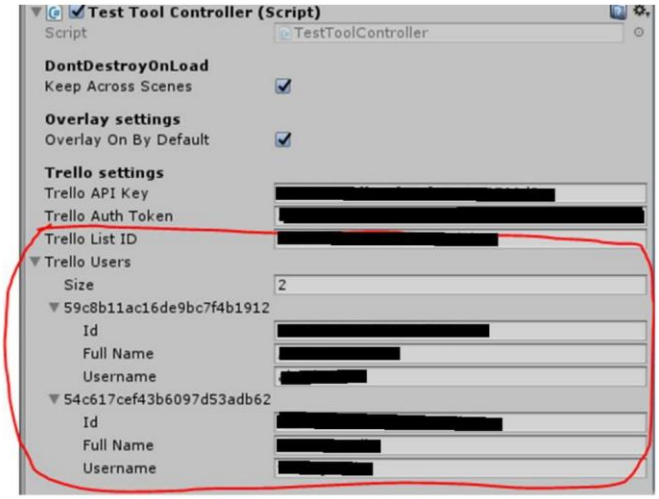
Sen jälkeen paina molemmat Find and configure napit



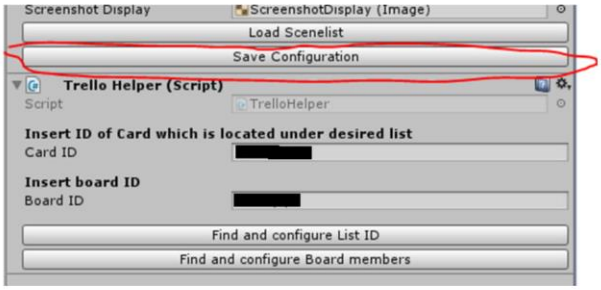
Find and configure List ID hakee annetun Card-ID:n kortin listan ja sijoittaa sen Trello List ID muuttujaan. List ID tarvitaan korttien lähettämistä varten.

Find and configure Board members hakee Board-ID:n boardin ja sijoittaa sen jäsenet Trello Users taulukkoon, jotta jäseniä voidaan tägätä uusiin kortteihin.

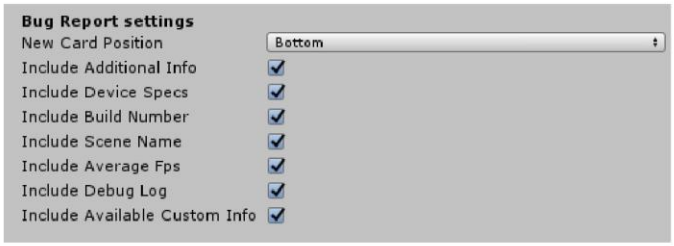
4



Näiden jälkeen paina TestToolControllerista ”Save Configuration”



Uusiin Trello kortteihin lähetettävää tietoa voi rajoittaa näistä asetuksista:



Projektikohtaista Custom infoa pystyy lisäämään kutsumalla
TestTools.TestToolController.Instance.AddReportCustomInfo(string key, string[] value);

Custom infon pystyy tyhjentämään kutsumalla .ClearReportCustomInfo();

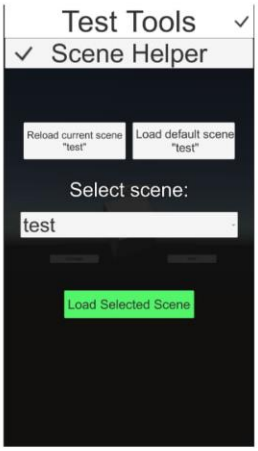
Screenshotit ja video

Työkalujen mukana tulee EveryPlay, joka nauhoittaa taustalla pelikuvaa, jos Record Video on päällä. Videonauhoituksen saa kesken pelin päälle/pois valikosta ”Replay Settings” osiosta. Videon pituutta voi säätää Video Replay Length muuttujalla. Screenshoteditorin tarrat määritetään laittamalla ne ”Stickers” aulukkomuuttujaan.

Käyttäminen

Testityökalut voi avata painamalla iconia

Iconia voi dragata pelissä, joten sen sijainnilla ei skenessä ole niin paljoa väliä.



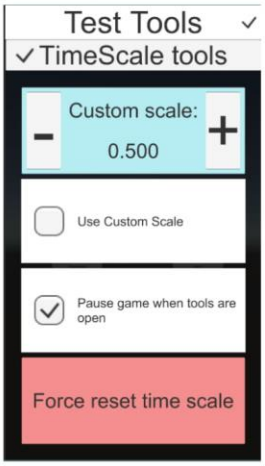
Valikot

SceneHelper

- Reload Current Scene ”scenen nimi”: Lataa nykyinen scene uudelleen
- Load Default Scene ”scenen nimi”: Lataa ensimmäinen scene
- Select scene: (pudotusvalikko) valitse scene jonka haluat ladata
- Load Selected Scene: Lataa pudotusvalikosta valittu scene

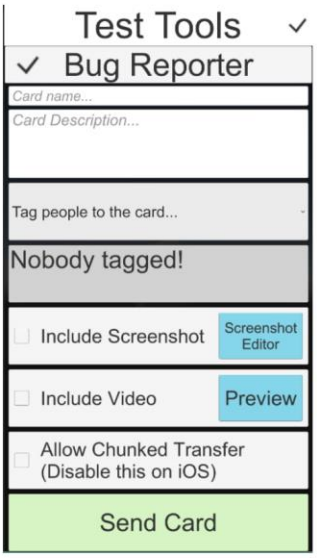
Timescale tools

- Custom scale: näyttää valitsemasi custom scalen
- Plus ja miinus: muuta custom scalea
- Use Custom Scale: käytetäänkö custom scalea. Jos ei, niin peli määrää scalen
- Pause game when tools are open: Jos tämä on päällä, pelin timescale menee aina 0, kun työkaluvalikko avataan
- Force reset time scale: pakota timescale takaisin pelin määräämään timescaleen.



Bug reporter

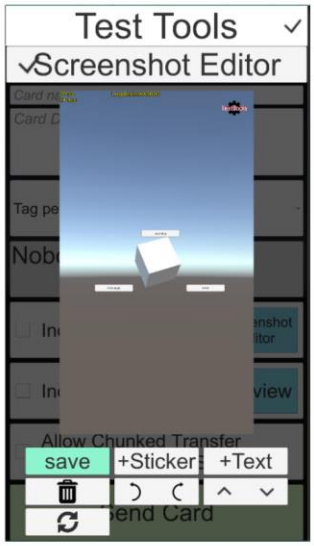
Tällä sivulla lähetetään kortteja määritetylle Trello listalle



- Card name: uuden kortin nimi
- Card Description: uuden kortin kuvaus
- Tag people to the card: pudotusvalikko, josta voit lisätä henkilöitä kortille, tai painamalla uudestaan poistaa henkilöitä kortilta.
- ”Nobody tagged!”: tässä lukee tägätyt henkilöt
- Include Screenshot: lisätäänkö korttiin mukaan screenshot. Screenshot otetaan sillä hetkellä, kun työkaluvalikko avataan. Jos Screenshottia on muokattu Screenshoteditorissa, lisätään sen sijaan mukaan muokattu screenshot.
- Include video: Jos videon nauhoitus on käytössä, tämä liittää editorissa säädetyn pituisen videon korttiin mukaan. Voit esikatsella videon ”preview” napilla.
- Allow Chunked Transfer: http protokollan optimointiin liittyvä ominaisuus, vakauttaa kortin lähettämistä. Ei toimi iOS:lla.
- Send Card: lähetä kortti

Kun painat lähetä kortti, näet huomautuksen ”Uploading”.

Kortin lähetys onnistui, kun näet sinisen tekstin ”Uploaded to Trello” ja lomakkeen kentät palautuvat tyhjäksi. Jos ilmoitusta ei tule, lähetyksessä tapahtui virhe. Debug logiin pitäisi tulla virheviesti. HUOM, Videon lähetys saattaa kestää hieman!



Screenshot editor

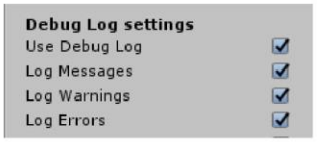
- Kuvassa näkyy screenshot, jota muokkaat
- Save: tallentaa screenshotin Trelloon lähetettäväksi
- + Sticker: lisää tarran, jota tuplaklikkaamalla voi selata erilaisia kuvia
- + Text: lisää tekstikentän.

Tarroja ja tekstikenttiä voi dragata kuvan päällä. Tekstikenttää dragataan tekstin vasemmalla puolelta, johon ilmestyy neliö silloin kun tekstikenttä on tyhjä.

- Roskakori: poistaa viimeksi kosketetun tarran/kuvan
- Kaarevat nuolet: kääntää tarraa/tekstiä
- Ylös/alas nuolet: skaalaa tarraa/tekstiä
- Viimeinen refresh nappi päivittää muokattavan screenshotin uusimpaan. Kuva otetaan aina, kun työkaluvalikko avataan, mutta se ei päivity editorissa itsestään kuin ekalla kerralla

Debug log

Tänne ilmestyy suodattimien mukaan logit, varoitukset ja errorit. Oletusasetukset voi säätää editorissa.

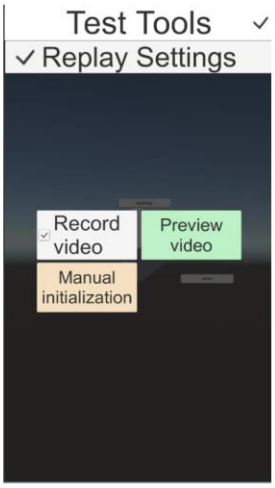


Overlay settings

Täältä löytyy Toggle overlaylle sekä jokaiselle lisätylle overlay-elementille. Togglet luodaan automaattisesti playmodessa, niitä ei tarvitse tehdä itse.

Replay settings

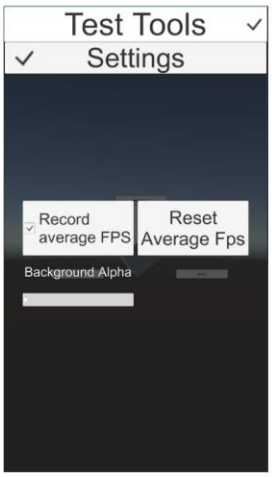
- Record video toggle: nauhoittaako Everyplay taustalla videota
- Preview video: esikatsele video
- Manual init: Jos Everyplay ei lähde toimimaan, tällä voi koittaa manuaalisesti kytkeä sen päälle. Tätä ei pitäisi tarvita painaa.



Test Tools Settings

Testityökalujen asetukset

- Record average fps: lasketaanko fps keskiarvoa
- Reset average fps: nollaa average fps:n kehyslaskuri, eli aloita keskiarvon laskeminen alusta
- Background alpha: säädä valikon taustan läpinäkyvyyttä



Overlay

Overlay tarjoaa tavan näyttää ruudulla jatkuvasti hyödyllistä tietoa kuten fps lukeman tai ilmoituksen debug log viesteistä.

Overlay elementtejä voi lisätä tekemällä uuden gameobjectin Overlayn childiksi, ja lisäämällä siihen sopivan komponentin.

Valmiita elementtejä löytyy painamalla gameobjectista Add component -> TestTools -> Overlay tai Projectin TestTools/Prefabs/Overlay kansiosta.

Komponentteja voi koodata itse lisää. Ne voi tehdä joko itse alusta asti, tai extendaamalla luokkaa OverlayElement. OverlayElementillä on 2 funktiota: UpdateText(string s), joka päivittää tekstin, ja overridettava Initialize(), joka ajetaan Startin lopuksi.

Tässä esimerkkejä.

```
public class FpsCounter : OverlayElement
{
    // Update is called once per frame
    void Update()
    {
        UpdateText(string.Format("{0:0.} fps\n{1:0.} avg", (1.0f / Time.deltaTime), TestToolController.Instance.AvgFpsCumulative));
    }
}
```