Unity Projekti Kurssi: Ketterä kehittäminen, Peliprojektin yleiset käytännöt, Pelin myynti/pitch, Versionhallinta kertaus

8.9.2020 & 9.9.2020

Discord ryhmän linkki! GitHub Repo

Olli-Pekka Juhola

Päivän aiheet

- Ketterä Kehittäminen peliprojekteissa
  - + Hyödyllisiä työkaluja siihen
- Peliprojektien yleisiä käytäntöjä
  - Tips & Tricks
  - Fail Fast Mentaliteetti
- Versionhallinta (Kertausta edelliseltä kurssilta)
  - Miksi käyttää versionhallintaa peliprojekteissa
  - Miten toimia oikein versionhallinnan suhteen

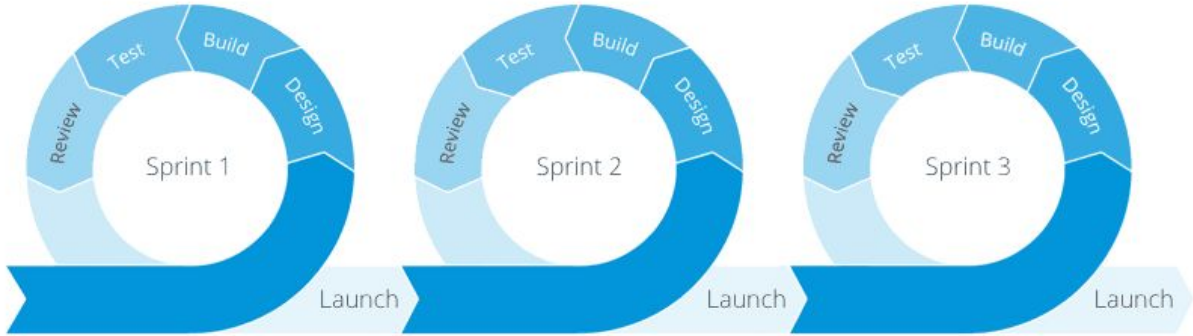
Päivän aiheet

- Peliprojektin aloitus ja suunnitelmien läpikäynti
  - Ryhmien pelisuunnitelman läpikäynti ja itse pelikehityksen aloittaminen
- Kuinka pitchata peliä julkaisijoille ja investoreille
  - + Pitch Deck
- Post Mortem

Ketterä Kehittäminen (Scrum)

Scrum Tarjoaa pelikehitykseen (ja sovelluskehitykseen) mallin, jonka mukaan projektia ohjataan oikeaan suuntaan. Tavoitteena oleva tuote (peli) kehittyy pikkuhiljaa täyteläisemmäksi useiden kehitys jaksojen aikana.

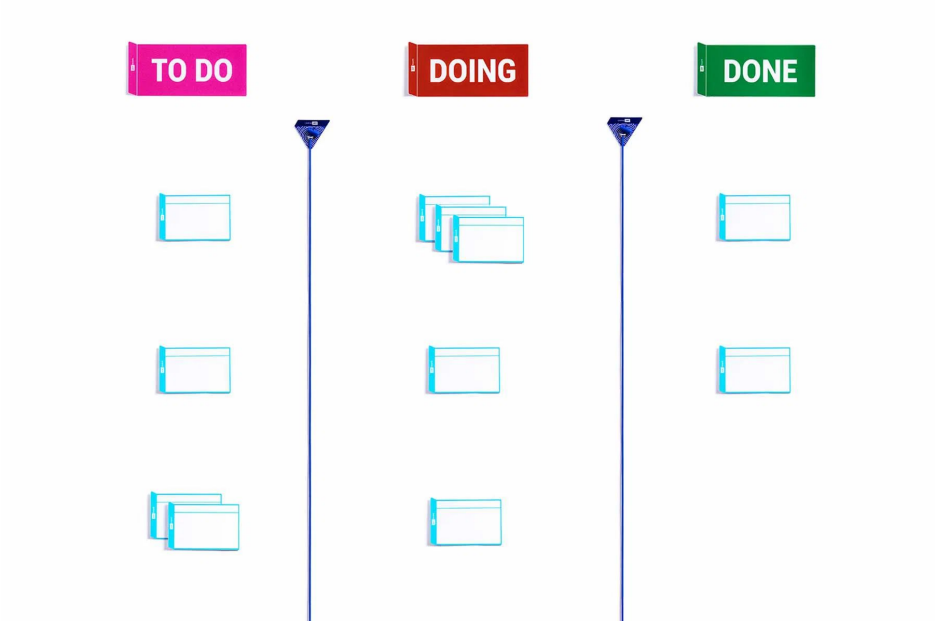
Kehitysjaksoa kutsutaan sprintiksi, joka voi olla 1-4 viikon mittainen aikaraja, jonka sisällä tuotetaan mahdollisimman vakaa versio tuotteesta.



Ketterä Kehittäminen (Kanban)

Kanban on suosittu tapa, jota käytetään ketterän ohjelmistokehityksen toteuttamisessa. Projektin työtehtävät näkyvät visuaalisesti kanban -taululla, jolloin tiimin jäsenet voivat nähdä jokaisen työn tilan milloin tahansa.

Kanban menetelmää ei suosita käyttöö projektin alkuvaiheissa ja kriittisissä oloissa, mutta jokainen tuote ja projekti on erilainen.
Kanban antaa hieman enemmän vapautta tiimille ja sen takia se on hyödyllinen ottaa käyttöön, kun tuote on julkaistu markkinoille ja stabiilissa tilassa.



Kanban vai Scrum?

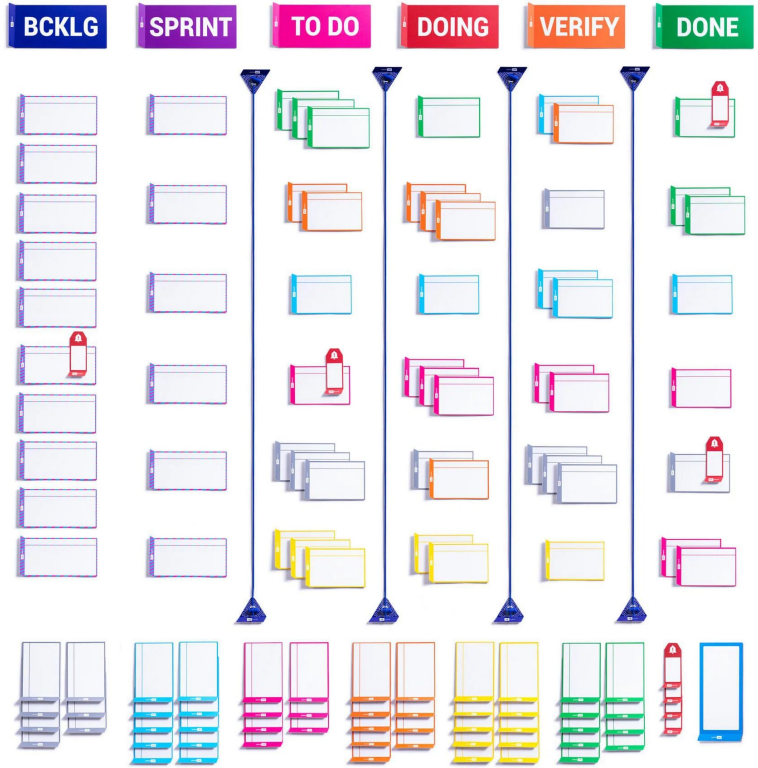
	Scrum	Kanban
Cadence (rytmi)	Jatkuvat 1-4 viikon sprintteinä	Jatkuvaa (kehitystä)
Release methodology (Julkaisu metodologia)	Sprinttien lopussa	Jatkuva toimitus
Roles (roolit)	Product Owner, Scrum Master ja Development tiimi	Ei erillisiä rooleja
Key Metrics (tärkeimmät menetelmät)	Nopeus	Toimitusaika, jaksonaika
Change philosophy (Suunnitelmien vaihto)	Muutoksia suunnitelmiin ei tehdä sprintin aikana	Muutoksia voi tapahtua milloin vain

Max Rehkopf, https://www.atlassian.com/agile/kanban/kanban-vs-scrum

Ketterän kehittämisen työkaluja

Kanban taulukko on hyvin yksinkertaistettu versio Scrum taulukosta, vaikka molemmat näyttävät hieman samoilta, on mieluisempaa käyttää peliprojekteissa Scrum menetelmää. Mikäli peliprojekti ei ole kokoaikaista työtä vaativa, voidaan mieluummin käyttää Kanban menetelmää, jossa ei ole aika rajoitteita tai deadlinejä (sprinttejä).

Loistava työkalu peliprojektien suunnitteluun ja ketterään kehittämiseen on HacknPlan.
Enemmän Kanban ystävällinen työkalu on Trello mutta myös perinteiset GitHub issue boardit käy tähän myös.



HacknPlan

- HacknPlan on ilmainen työkalu, joka yhdistää projektityöskentelyn pelisuunnittelun kanssa. Se on pääsääntöisesti tarkoitettu pelkästään peliprojektien ketterään kehittämiseen.

- HacknPlan sisältää perinteisen Kanban taulukon, johon voidaan yhdistää Scrumin sprintit.

- Sisältää sisäänrakennetun game design dokumentointi alustan joka on rakennettu ketterän kehittämisen menetelmiä ajatellen.

- HacknPlan antaa myös tarkat metriikat ja raportit projektista, joka auttaa projektin tehtävien hallinnassa ja suunnittelussa.

Pelikehityksen Tips & Tricks

1. Think big, Start small. Pelikehitys on haastavaa, ja vielä haastavampaa jos haluaa lähteä kehittämään todella laajaa peliä. Pelisuunnittelun aikana, kannattaa ottaa huomioon jokaisen tiimiläisen taidot ja lähteä suunnittelemaan peliä niiden avulla.

2. Pelaa samankaltaisia pelejä mitä olet kehittämässä, se voi antaa ideoita ja ajatuksia siitä millaisen pelin haluat todella kehittää. Jos haluat kehittää oman korttipelin, on hyvä tutkia markkinoita ja katsoa millaisia pelejä on jo olemassa, millaiset pelit ovat suosittuja ja millaiset eivät.

3. Testing, testing and more testing! Uusien ominaisuuksien testaus on tärkein asia pelikehityksessä. Jos sinun työsi on kehittää peliin jokin todella pieni asia, esimerkiksi uusi ase, kannattaa se suoraan testauttaa editorissa ja buildatussa pelissä, sillä mikäli se ominaisuus on rikki ja sitä ei ole testattu, voi se olla aikaa vievä asia tulevaisuudessa.

Pelikehityksen Tips & Tricks

4. Innovointi. Mikäli sinulle iskee päähän jokin todella hullu idea, esimerkiksi pelimekaniikka, kannattaa se suoraan kirjoittaa ylös ja lähteä testaamaan. Joskus todella hullu ominaisuus, esimerkiksi hauska ase tai vihollinen, voivat olla sellaisia asioita jotka jäävät pelaajien päähän.

5. Suunnittele ennen lisäystä, Design → Implement. Ilman kunnollista suunnittelua, pelin ominaisuus voi karata käsistä, jäädä epäselväksi muulle tiimille tai suoraan rikkoa pelin. Tämän takia jokaisen ominaisuuden suunnittelu, esimerkiksi taskin lisäys kanban taulukkoon, auttaa sen ominaisuuden kehittämisessä ja lisäyksessä.

6. Kehityksen flow. On helppoa aloittaa pelikehitys mutta kehityksen rytmin pitäminen voi olla haastavaa. Kannattaa suunnitella tarkasti milloin ja kuinka kauan työskentelet pelin parissa, mikäli aikaa on vähän. Todella hyvä puhe tähän aiheeseen!

Fail Fast mentaliteetti

Viimeinen hyödyllinen asia mikä kannattaa pistää muistiin on Fail Fast mentaliteetti.

Fail fast (epäonnistuminen) filosofia arvostaa laajaa testausta ja kehitystä, jonka tarkoituksena on selvittää onko idealla mitään arvoa. Fail Fast mentaliteetin tärkein tavoite on vähentää pelin ja työskentelyn tappioita, kun pelin / ominaisuuden testaus paljastaa, että jokin asia ei toimi halutulla tavalla ja siitä suoraan siirtyä uuteen asiaan.

Tätä voidaan käyttää pienten tai isojen ominaisuuksien kehittämisessä, esimerkiksi jos tietty pelin ase ei toimi halutulla tavalla, lähdetään suoraan testaamaan uutta, mikäli sekään ei onnistu jätetään se asia pois pelistä. Fail Fast mentaliteettiä käytetään myös koko peliprojektin testauksessa ja selvittää, onko järkeä jatkaa pelin kehitystä vai ei.

Fail Fastin ideana on estää “sunk cost effect”, joka tarkoittaa ihmisten taipumusta investoida aikaa johonkin, mikä ei selvästikään toimi.

Pelin myynti

Pelin pitchaus / myyminen toimii samalla tavalla kuin minkä tahansa tuotteen pitchaus. Esität ja yrität myydä tuotteesi asiakkaille tai yrittää saada investorin investoimaan tuotteeseen. Pelien suhteen, usein yritetään saada myytyä idea myös julkaisijoille, jotka julkaisevat pelin heidän nimellään.

Pelin myyntiä, markkinointia ja näkyvyyttä on vaikea saada ilman ulkopuolista apua. On mahdollista julkaista peli ilman mitään apua, mutta yleisesti julkaisija ja investori on tähän pakollinen, etenkin jos kyseessä on kallis ja laaja peli.

Peli esitellään asiakkaille, investoreille ja julkaisijoille vertical slice vaiheessa, mutta se voidaan myös esitellä prototyyppi vaiheessa, mikäli kyseessä on todella laaja idea.

Pelin myynti

Paras mahdollinen paikka etsiä pelille investoreita ja julkaisijoita on pelikonferenssit. Suomessa pidetään useita eri tapahtumia ympäri vuoden esimerkiksi Pocket Gamer Connects Helsinki, joka on yleisesti Syksyllä pidettävä mobiili & PC/Konsoli tapahtuma. Siellä pidetään esimerkiksi Big Indie Pitch kilpailu, jossa pitchataan peliä tuomareille, jotka voivat olla investoreita, julkaisijoita tai journalisteja.

Muut tapahtumat ovat erilaiset IGDA Finland (International Game Developer Association) järjestämät tapahtumat, joita pidetään joka kuukausi eri puolilla suomea.

Lisätietoa IGDA:n tapahtumista

Pelin Pitch

Pelin pitch on samankaltainen kuin hissipuhe. Pelin voi pitchata jo suunnittelu vaiheessa, mutta kannattaa kuitenkin olla lyhyt prototyyppi valmiina. Oman pelin pitchaus voi olla todella stressaavaa, sillä usein kyseessä voi olla yli 10 000€ investointi pelin kehittämistä varten.

Pelin pitchausta varten kannattaa valmistautua kuukausia etukäteen, miettiä kaikki kysymykset läpi jota voidaan kysyä ja laatia kunnollinen pitch deck esittelyä varten.

Pitch Deck

Pitch Deck on lyhyt esitys jonka avulla yleisö, investori tai julkaisija saa nopean yleiskuvan tuotteen suunnitelmasta. Se on yhtä kattava kuin High Concept Document, mutta tehdään usein PowerPointilla tai sen tyylisellä työkalulla.

Pitch Deck kannattaa pitää tiiviinä, mutta sen pitäisi näyttää projektin koko suunnitelma, mitä ollaan tehty, miksi ja myös mihin tähdätään.

Pitch Deck Sisältö

- Title
- Pelin yleiskatsastus
  - Genre, mekaniikat, yhteenveto
- Markkinat
  - Kohderyhmä, alustat, markkina analyysi
- Bisnes malli
  - Monetisaatio, suunnitelma (roadmap)
- Kilpailijat
  - Samankaltaiset pelit? Ainutlaatuiset ominaisuudet, miten olette parempia kuin muut
- Tiimi
  - Millainen tiimi teillä on, kokemukset, osaamiset
- Talous
  - Rahoitus tähän mennessä, resurssien tarve, rahoituksen tarve, mihin rahoitus käytetään

Pitching 101

1. **Tunne yleisösi**. Ennen kuin otat yhteyttä mahdolliseen julkaisijaan, tutki minkä kaltaisia pelejä julkaisija yleisesti julkaisee. Jos kehität mobiilipeliä, todennäköisesti PC/Konsoli julkaisijat eivät ole kiinnostuneita ideastasi. Monet julkaisijat julkaisevat tiettyjen genrejen pelejä, esimerkiksi pelkästään fantasy pelejä.

2. **Laadi jotain erityistä ja ainutlaatuista.** Muista, että myyntipuheen tarkoituksena on saada rahoitus projektillesi, se voi olla kymmeniä tuhansia euroja, joten tee esityksestä erityinen.

3. **Puhu vain tärkeistä ominaisuuksista**. Keskity pelin tärkeimpiin ominaisuuksiin. Kerro lyhyesti millainen peli on kyseessä parilla lauseella siten, että siitä ymmärtää koko pelin idean. Älä kerro jokaisesta levelistä, aseesta tai aarteesta mitä pelissä on. Esimerkiksi mikä tekee pelistä ainutlaatuisen, mikä motivoi pelaajia yms.

Pitching 101

4. **Tuo mukanasi kaikki tarvittavat materiaalit.** Myyntipuheessa kannattaa olla valmiina kaikki tarvittavat dokumentit ja myös prototyyppi/vertical slice pelistä. Valmistelu auttaa esityksen flown kanssa.

5. **Harjoittele!** Harjoittele pitchiä muutamaan otteeseen, projektiryhmälle ja tutuille. Omat ryhmänjäsenet huomaavat helposti mitä puheesta puuttuu ja jos omat tutut eivät ymmärrä pelin ideaa pitchistäsi, niin ei ne julkaisijat/investorit myöskään ymmärrä.

6. **Harjoittele lisää!** Pitchit voivat olla kolmesta minuutista puoleen tuntiin, joten on hyvä harjoitella pitchiä kunnolla ja yrittää ajoittaa se oikein!

   Oma kokemus PGC Helsinki Big Indie Pitch kilpailusta: 3 minuuttia, 6 pöytää. Joka pöydälle sama pitch jonka kesto oli 3 minuuttia. Mutta erilliset tapaamiset voivat viedä jopa 30 minuuttia ja yleisesti käydään enemmän asioita läpi!

Pitching 101

7. **Itsevarmuus.** Ole varma pelistä ja taidoistasi! Näytä heille miksi juuri teidän peli tulee menestymään ja miksi juuri teidän tiimi on voi toteuttaa sen!

8. **Kehonkieli.** Älä pidä käsiä ristissä, pidä kontakti kenen kanssa puhut ja jos mahdollista näytä kädet osoittamaan avoimuutta ja luottamusta. Kuuntele ja näytä mielenkiintoa henkilöön jonka kanssa puhut.

9. **Lisämateriaali ja uuden ajan sopiminen.** Puheen jälkeen, kannattaa ojentaa heille kopiot materiaaleista, jotta he voivat näyttää niitä kollegoilleen. Keskustele ja sovi mahdollisesta follow up tapaamisesta, ja osoita heille, että voitte antaa muitakin materiaaleja mikäli he ovat jotain kyselleet.

30 Things I Hate About Your Game Pitch - GDC Puhe

Post Mortem

Post mortem on dokumentti joka sisältää pelin kehityksen ja suunnittelun tiedot, pelin kehityksen historia. Post Mortem dokumenttia ei näytetä muille, se ei ole markkinointi dokumentti. Post Mortem sisältää kaikki mikä kehityksessä meni hyvin ja mikä huonosti.

Post Mortem laaditaan pelin julkaisun jälkeen, jolloin voidaan nähdä mitä kehityksessä meni vikaan, mikä meni oikein ja mitä voisi parantaa seuraavaa projektia ajatellen. Esimerkiksi jos pelikehityksen aikana kommunikointi oli heikkoa ja sen takia projektityöskentely hidastui, se on tärkeä lisäys Post Mortem dokumenttiin.

Keskustelkaa tiimiläisten kanssa mikä meni vikaan ja mikä meni oikein, joten seuraavassa projektissa voidaan oppia edellisen projektin ongelmista.

Esimerkki post mortem Diablo II kehityksestä

Versiohallinta

Versiohallinta on oleellista lähes jokaisessa ohjelmointiprojektissa, tämän takia se on myös todella tärkeä peliprojekteissa.

Versiohallinan avulla voidaan luoda helposti eri versioita projektista, työskennellä eri ominaisuuksien kanssa erillään ja varmistaa, että vakaa versio projektista on aina saatavilla tietyllä branchillä.

Hyvä versionhallintaohjelmisto tukee kehittäjän suosittua työnkulkua asettamatta yhtä tiettyä työskentelytapaa

Versiohallinta

Versiohallintatyökaluista suosituin on Git, mutta muitakin vaihtoehtoja on saatavilla.

Git on hajautettu versionhallintajärjestelmä.
Git versiohallinnalle on monia eri isännöitsijä palveluita, näistä tunnetuin on GitHub.
Muita isännöitsijä palveluita ovat Bitbucket, GitLab & AWS Code Commit.

Versiohallinta

Git:in käytölle on monia eri käyttöliittymiä. Suosituin kaikesta on alkuperäinen komentorivi ikkuna. Monille tämä on ainoa tapa työskennellä Git:in kanssa, mutta muita vaihtoehtoja ovat myös olemassa, jotka lisäävät paremman käyttöliittymän käyttäjille.

Suosituimpia Git käyttöliittymiä ovat:
SmartGit, GitKraken, Git Extensions, SourceTree & GitHub Desktop (Oma suosikki on GitKraken)

Git komentorivi. Käyttöliittymän valitseminen riippuu omasta työskentely tyylistä´.

Versiohallinta Best Practices

1. Versiohallinnassa työskennellään erilaisissa Brancheissä. *origin/master* - branch on ensimmäinen branch joka on automaattisesti luotu. *Master* - branchin tulisi lähes aina heijastaa tuotantovalmiutta ja yleisesti siellä on saatavilla viimeisin *Major* - versio, esim versio 0.1 / 0.2 / .. / 1.0 ja eteenpäin.

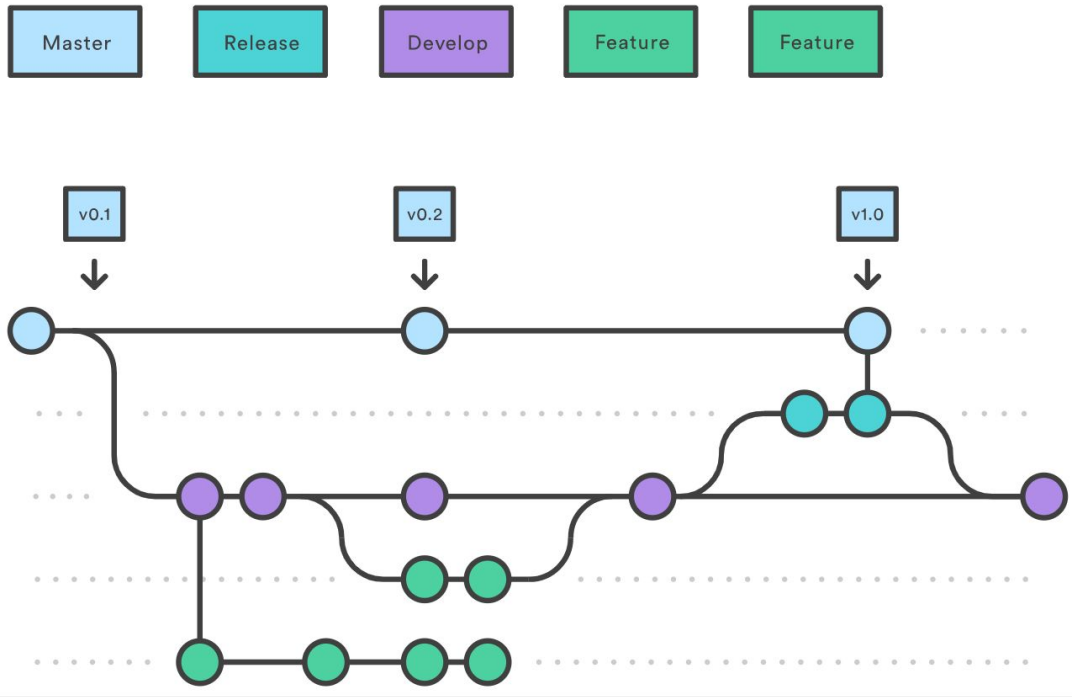
2. *Master* - branchin jälkeen on yleistä luoda *develop* - branch, jonka tarkoituksena on integroida viimeisimmät / uusimmat ominaisuudet saman branchin alle. Monet kutsuvat tätä branchiä “Integration branch”.

3. *Develop* - branchin jälkeen yleensä luodaan *feature/”featurename”* - branchejä, jotka ovat kaikki *feature/* “kansion” alla. Jokainen *feature* - branch on oma ominaisuus / “feature” projektille, jotka yhdistetään *develop* - branchiin.

Versiohallinta Best Practices

Kuvassa näkyy yleisin git workflow. *Master:iin* menee Major päivitykset, *Develop:iin* yhdistetään *feature* - branchit, jonka jälkeen kun on aika julkaista uusi major versio, liitetään *develop* branch *master* branchille (tai erilliseen *release* branchille, mikäli sellaista käytetään)

Hotfix - branchin voi myös lisätä, missä korjataan julkaisu version bugit ja ongelmat

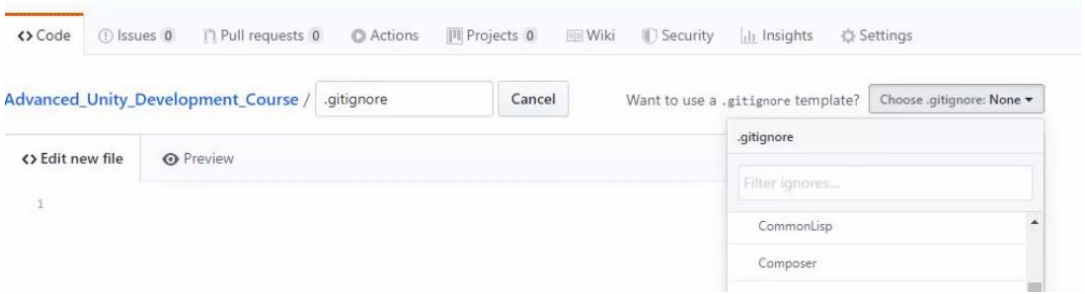


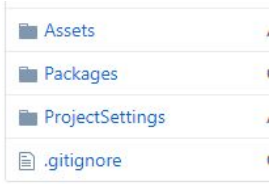
Lähde: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Versiohallinta Best Practices

Ennen kuin aloitat projektin, lisää repositorioon *.gitignore*. *Gitignore* tiedostoja saa netistä valmiiksi, esimerkiksi GitHubissa kun luodaan uutta repositoriota, voit lisätä *.gitignoren* todella helposti repostorioon. *Gitignore* - tiedosto hylkää halutut tiedostot ja niitä ei työnnetä repositorioon, Unity projekteissa nämä ovat TODELLA tärkeitä.

GitHubissa *.gitignore* luodaan kätevästi luomalla uusi tiedosto, ja kirjoittamalla nimeksi ".gitignore". Tämän jälkeen oikealle puolelle ilmestyy "Want to use a .gitignore template?" ja drop down lista useasta eri .gitignoresta.





Unity .gitignore kannattaa asettaa Unity projekti kansion sisälle

Versiohallinta Best Practices

1. Tee yksinkertaisia ja puhtaita committeja, etenkin bugien ja feature lisäysten kanssa.
   - Kun lähdet lisäämään uutta ominaisuutta, luo sille ominaisuudelle oma *feature* - branchi. Kehitä tiettyjä ominaisuuksia vain omalla branchillä, älä koske muihin ominaisuuksin vaikka niistä löytyisi ongelmia (sillä joku tiimiläinen on jo voinut ne korjata toisella branchillä)

   - Jos tarvitsee mennä versiossa taaksepäin, se tekee siitä paljon helpompaa
2. Commit:taa muutokset usein. Jos työskentelet *feature* branchillä, jonka saattaminen voi viedä aikaa, se auttaa pitämään koodisi ja ominaisuudet päivitettynä uusimmalla *develop* branchillä, jotta vältetään tulevaisuudessa konfliktit.

3. Kirjoita merkityksellisiä commit - viestejä. Kun committaat, on helpompi lukea “Fixed issue with movement” kuin “Fix”.

Versiohallinta Best Practices

- Kun työskentelet versiohallinan kanssa, suhtaudu prosessiin rauhallisesti. Jos on hätäinen committien, push:in ja mergen kanssa, voi tulla vakavia ongelmia.

- Versiohallinan kanssa toimii hyvin suunnittelu työkalut, kuten Hack n' Plan, jolla voidaan nähdä, mitä tiimiläiset tekee.
  - Esim. Tiimiläinen on asettanut itselleen issuen, jossa hän lisää peliin Main Menun, tästä voidaan olettaa heti, että tiimiläinen työskentelee branchillä *feature/main_menu.* Kun tiimiläinen on siirtänyt issuen "Testing" - tauluun, voidaan olettaa että *feature/main_menu*, branchiä voidaan testata, mikäli main menu toimii halutulla tavalla ilman ongelmia, voidaan *feature/main_menu* yhdistää *develop* branchiin.