Møte 3, 08.03.24

Sak 0 - Standardprotokoll

- Valg av ordstyrer Egor
- Valg av referent William
- Oppmøte/fravær Trym, William, Tarjei, Egor, / Kevin (Jobber)
- Godkjenning av innkalling Ok
- Godkjenning av saksliste Ok
- Regler for innlegg/replikk Ok

Prosjektrapport

Hvordan fungerer rollene i teamet? Trenger dere å oppdatere hvem som er teamlead eller kundekontakt?

- For det meste fungerer rollene i teamet veldig bra, men tester er noen ganger tungt. Trenger kanskje ekstra hjelp med det.

Trenger dere andre roller? Skriv ned noen linjer om hva de ulike rollene faktisk innebærer for dere.

- **Trym | Systemarkitekt**
  - Rollen innebærer å tidlig finne refferanse punkt fra andre prosjekter til å designe vår arkitekttur
- **Tarjei | Support**
  - Hjelper der det trengs og kommer med nytt innsyn.
- **William | Testansvarlig**
  - Lager tester der det trengs og hjelper til hvis noen trenger hjelp
- **Kevin | Design og UX**
  - Designer textures og UI (skaper en god brukeropplevelse)
- **Egor | Projektleder**
  - Har kontroll innenfor projektet, sånn at hvis andre spørr om hjelp så kan han hjelpe. Har oversikt over oppgavene og innleveringsdatoene. Passer også på flyt og jobbing i projektet.

Er det noen erfaringer enten team-messig eller mtp prosjektmetodikk som er verdt å nevne? Synes teamet at de valgene dere har tatt er gode? Hvis ikke, hva kan dere gjøre annerledes for å forbedre måten teamet fungerer på?

- Peer programming har vi testet ut og det fungerer bra. Men XP metodikken fungerte dårlig, da vi ikke er gode nok til å skrive tester.
- Issueboard har vi vært dårlig til å bruke, men tenker å bruke det mer i fremtiden.
- Vi har skrevet javadoc for det meste etter vi er ferdig, men vi bør bli bedre på å skrive det underveis.
- Temet fungerer bra, men vi er litt dårlig på å komme i gang, men først kommer i gang klarer vi å holde flyten oppe.

Hvordan er gruppedynamikken? Er det uenigheter som bør løses?

- Gruppedynamikken fungerer bra når vi jobber fysisk i lag, men vi har ikke testet hvordan det fungerer å jobber individuelt.
- Vi kom litt dårlig i gang på grunn av mye sykdom i teamet i startet av projektet

Hvordan fungerer kommunikasjonen for dere?

- Kommunikasjonen fungerer bra både fysisk og på nettet, men beskjeder kommer litt sent.

Gjør et kort retrospektiv hvor dere vurderer hva dere har klart til nå, og hva som kan forbedres. Dette skal handle om prosjektstruktur, ikke kode. Dere kan selvsagt diskutere kode, men dette handler ikke om feilretting, men om hvordan man jobber og kommuniserer.

- Hvis vi bruker issueboard mer aktivt kan man hjelpe seg selv og hverandre på å ha en bedre oversikt over prosjektet.
- Det som har fungert bra er peer programing og sam-programmering.

Under vurdering vil det vektlegges at alle bidrar til kodebasen. Hvis det er stor forskjell i hvem som committer, må dere legge ved en kort forklaring for hvorfor det er sånn. Husk å committe alt. (Også designfiler)

- Kevin jobber med design og sitter derfor mye i photoshop. Han jobber også med sitt eget prosjekt i et testmiljø.
- William bruker mye tid å gjøre seg kjent med prosjektet for å lage tester
- Tarjei hjelper ofte og det kan skje gjennom par-programmering og det vil derfor ikke gjenspeiles i git historikken.

Referat fra møter siden forrige leveranse skal legges ved (mange av punktene over er typisk ting som havner i referat).

Bli enige om maks tre forbedringspunkter fra retrospektivet, som skal følges opp under neste sprint.

- Vi kan forbedre oss på å bruke issueboard oftere.
- Bedre planlegging av møter og arbeidsøkter.
- Bedre kommunikasjon av det som skal gjøres.

Krav og spesifikasjon

Oppdater hvilke krav dere har prioritert, hvor langt dere har kommet og hva dere har gjort siden forrige gang. Er dere kommet forbi MVP? Forklar hvordan dere prioriterer ny funksjonalitet.

Fra vår mvp har vi fått gjort følgende:

- Vise spillbrett
- Plassert forsvarer ved hard koding
- Plassert angriper ved hard koding
- Spiller kan dø
- Game start screen(trenger redesign)
- Game Over screen(trenger redesign)
- Quick Reset funker
- Beskytter gir skade på angriper

Vi er ikke helt forbi MVP ennå, da noe av funksjonaliteten vi hadde som mål ikke er laget helt ennå.

Vi prioriterer nå ny funksjonalitet gjennom brukerhistorier som vi har fått skrevet skikkelig.

For hvert krav dere jobber med, må dere lage 1) ordentlige brukerhistorier, 2) akseptansekriterier og 3) arbeidsoppgaver. Husk at akseptansekriterier ofte skrives mer eller mindre som tester.

Dersom dere har oppgaver som dere skal til å starte med, hvor dere har oversikt over både brukerhistorie, akseptansekriterier og arbeidsoppgaver, kan dere ta med disse i innleveringen også.

Forklar kort hvordan dere har prioritert oppgavene fremover

- Vi ønsker å fjerne hardkoding av fiender og tårn. Dette krever en form for UI, så dette vil også være en prioritet. Vi ønsker å vise level stats, slik at spiller har en oversikt over hvordan hen ligger ann.

Har dere gjort justeringer på kravene som er med i MVP? Forklar i så fall hvorfor. Hvis det er gjort endringer i rekkefølge utfra hva som er gitt fra kunde, hvorfor er dette gjort?

- Vi har valgt dytte punkt 4 og 9 bak i prioritet, da disse ikke implementerer mye ny logikk, men heller bare viser frem tall som vi i stor grad allerede har tilgjengelig i vår kode.

Oppdater hvilke krav dere har prioritert, hvor langt dere har kommet og hva dere har gjort siden forrige gang.

Vi har til nå prioritert følgende krav:

- Defenders
- Enemies
- Stats
- Options
- HowToPlay
- UI.

Hittil har vi i stor grad fullført følgende:

- Backend for: Defenders, Enemies, Spillbrett

Vi vil derfor fremover prioritere å lage en UI, HowToPlay, options i meny og fremvisning av stats.

Husk å skrive hvilke bugs som finnes i de kravene dere har utført (dersom det finnes bugs).

Bugs vi har utbedret:

- Forsvarer sin angreps radius ble ikke overholdt.
- Angripere beveget seg vertikalt i en tile sin bredde, og horisontal i en tile høyde.

Kravlista er lang, men det er ikke nødvendig å levere på alle kravene hvis det ikke er realistisk. Det er viktigere at de oppgavene som er utført holder høy kvalitet. Utførte oppgaver skal være ferdige.

Produkt og kode

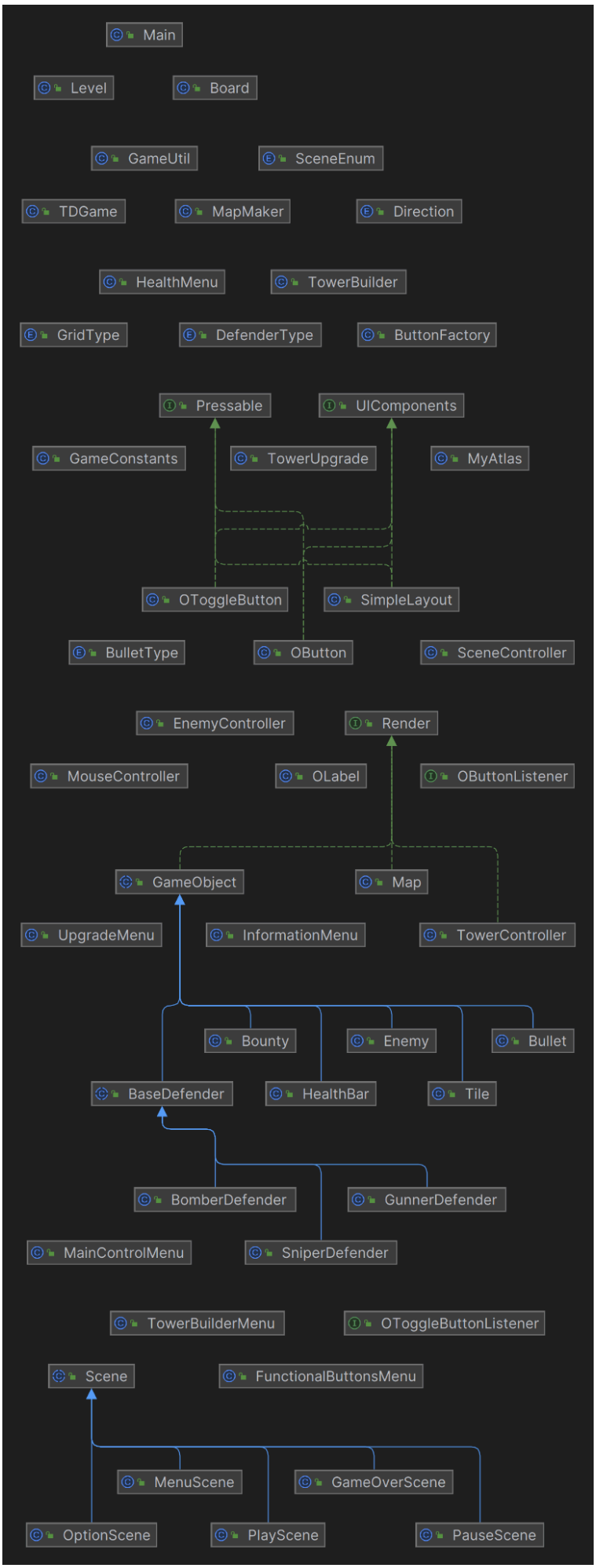
Utbedring av feil: **hvis dere har rettet / forbedret noe som er påpekt tidligere, lag en liste med «Dette har vi fikset siden sist», så det er lett for gruppelederne å få oversikt.**

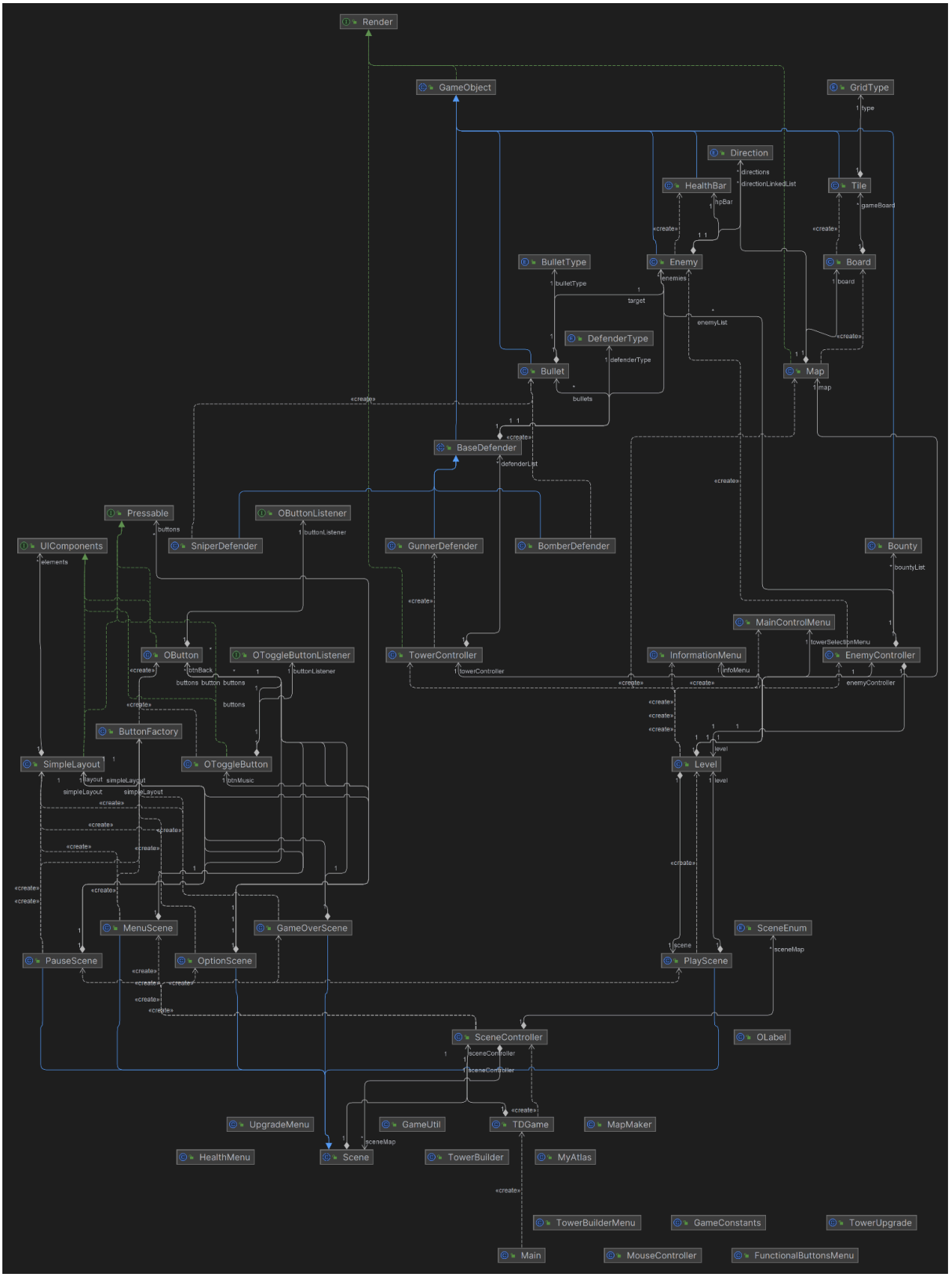
- Sist innlevering var vi fortsatt i startfasen i prosjektet, vi har derfor fokusert på å produsere mer kode. Vi hadde derfor ikke noen konkrete feil vi ville utbedre fra sist innlevering.

I `README.md`: Dere må dokumentere hvordan prosjektet bygger, testes og kjøres, slik at det er lett for gruppelederne å bygge, teste og kjøre koden deres. Under vurdering kommer koden også til å brukertestes.

Prosjektet skal kunne bygge, testes og kjøres på Linux, Windows og OS X – dere kan f.eks. spørre de andre teamene på gruppen om dere ikke har tilgang til alle platformene. *OBS!* Den vanligste grunnen til inkompatibilitet med Linux er at filnavn er *case sensitive*, mens store/små bokstaver ikke spiller noen rolle på Windows og OS X. Det er viktig å sjekke at stiene til grafikk og lyd og slikt matcher eksakt. Det samme vil antakelig også gjelde når man kjører fra JAR-fil.

Lag og lever et klassediagram. (Hvis det er veldig mange klasser, lager dere for de viktigste.) Det er ikke nødvendig å ta med alle metoder og feltvariabler med mindre dere anser dem som viktige for helheten. (Eclipse har forskjellige verktøy for dette.)


Main
Level
Board
GameUtil
SceneEnum
TDGame
MapMaker
Direction
HealthMenu
TowerBuilder
GridType
DefenderType
ButtonFactory
Pressable
UIComponents
GameConstants
TowerUpgrade
MyAtlas
OToggleButton
SimpleLayout
BulletType
OButton
SceneController
EnemyController
Render
MouseController
OLabel
OButtonListener
GameObject
Map
UpgradeMenu
InformationMenu
TowerController
Bounty
Enemy
Bullet
BaseDefender
HealthBar
Tile
BomberDefender
GunnerDefender
MainControlMenu
SniperDefender
TowerBuilderMenu
OToggleButtonListener
Scene
FunctionalButtonsMenu
MenuScene
GameOverScene
OptionScene
PlayScene
PauseScene



Kodekvalitet og testdekning vektlegges. Dersom dere ikke har automatiske tester for GUI-et, lager dere manuelle tester som gruppelederne kan kjøre basert på akseptansekriteriene.

Hittil har vi fokusert på å få ned basisfunksjonene. Vi er i et mellomstadium der vi holder på å rydde opp grunnstrukturen for å forbedre kodekvaliteten. Vi har lagt litt fokus på bruk av datastrukturer og java dokumentasjon.
Når det gjelder tester, har vi begynt med tester for basis funksjoner som finnes i akseptansekriteriene.

Statiske analyseverktøy som SpotBugs eller SonarQube kan hjelpe med å finne feil dere ikke tenker på. Hvis dere prøver det, skriv en kort oppsummering av hva dere fant / om det var nyttig.

- Vi har ikke fått tid til å teste disse verktøyene ennå, men vi satser på å få gjort dette til neste innlevering.

Automatiske tester skal dekke forretningslogikken i systemet (unit-tester). *Coverage* kan hjepe med å se hvor mye av koden som dekkes av testene – i Eclipse kan dette gjøres ved å installere *EclEmma* gjennom Eclipse Marketplace.

Utførte oppgaver skal være ferdige. Slett filer/kode som ikke virker eller ikke er relevant (ennå) for prosjektet. (Så lenge dere har en egen git branch for innlevering, så er det ikke noe stress å fjerne ting fra / rydde den, selv om dere fortsetter utviklingen på en annen gren.

- I vårt prosjekt har vi nå implementert en midlertidig UI, som benytter seg av et knappesystem som vi ikke tenker å bruke på sikt. Kevin holder nå på å gjøre seg kjent med libGDX sitt eget knappesystem, slik at vi kan implementere dette i vår UI. Vi har foreløpig det gamle knappesystemet lokalisert i UI mappen.