Møte 13, 03.05.24

Sak 0 - Standardprotokoll

- Valg av ordstyrer Egor
- Valg av referent William
- Oppmøte/fravær Trym, William, Tarjei, Egor, Kevin
- 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?

- Underveis når vår designer satt i gang med å utvikle UI, pådro han seg fort mye mer arbeid enn tiltenkt. Det oppstod da en fordel og en ulempe: Han fikk en rolle som liknet mer på systemarkitekt, noe vi burde ha unngått. Fordelen var at vi hadde et render-system på plass tidlig som gjorde at resten av utviklerne kunne lage grunnleggende strukturer under ledelse av den utvalgte systemarkitekten.

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

- **Trym | Systemarkitekt**
  - Underveis har denne rollen innebært å ta avgjørelser på hvor metoder skal lages, og hvordan klassene skal samhandle med hverandre.
- **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 prosjektet, sånn at hvis andre spørr om hjelp så kan han hjelpe. Har oversikt over oppgavene og innleveringsdatoene. Passer også på flyt og jobbing i prosjektet.

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?

- Vi har fortsatt med peer programming.
- Issueboard har vi blitt mye bedre på å bruke. Når noe som trengs å gjøres blir oppdaget, legges dette til raskt. Enten blir riktig person satt til oppgaven, eller så assigner man oppgaven til seg selv når man har fullført den.
- Kommunikasjon har vi blitt mye bedre på, slik at det er lettere å jobbe med metoder og klasser andre har laget.

- Teamet fungerer bra. Vi har jobbet ekstra mye sammen de siste ukene for å fullføre spillet.

Hvordan er gruppedynamikken? Er det uenigheter som bør løses?

- Gruppedynamikken fungerer bra. Vi jobber for det meste sammen, men også alene når vi jobber hjemme. Dette fungerer bedre ettersom vi har begynt å bruke issue board mye mer. Derfor er det mye lettere å se hva som trengs å gjøres.

Hvordan fungerer kommunikasjonen for dere?

- Kommunikasjonen ehar som sagt blitt mye bedre. Dette har ført til at vi jobber mye raskere og bruker mindre tid på å finne ut hva som trengs å gjøres, siden dette allerede er dokumentert i issues.

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.

- Vi har klart å få til et godt system på hvordan vi får til fremgang på en mest mulig effektiv måte ved bedre kommunikasjon, i tillegg til at vi gjør oss kjent med hva hverandre gjør så det blir enklere å jobbe med det.
- Peer programming i tillegg til programmering sammen rundt et bord er det som har fungert best for oss.

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)

- Vi føler at commits har blitt jevnere over tid naturlig.
- Kevin er som sagt ui designer som gjør at han jobber mye med photoshop, noe som ikke vises så mye i committs.
- Vi har også sagt tidligere at William bruker mye tid med å gjøre seg kjent med prosjektet, noe som heller ikke vises i commits historikken.
-

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

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.

1. Vise et spillbrett
2. Mulig å plassere en "beskytter" (i koden)
3. Mulig å generere "angripere" (i koden)
4. Liv, poeng og bølger vises
5. Spiller kan dø (mister liv)

6. Game Start Screen
7. Game Over Screen
8. Quick Reset av spillet
9. Pause Screen
10. "Beskytter" skyter/ skader "angriperen"

- Vi er nå ferdig med vår MVP. Det vi har prioritert mest er punktene på MVPén vi ikke hadde fullført. Dette satt vi som Prioritized to-do på issue board for å ha det i fokus.

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 ikke gjort noen særlige endringer annet enn at hovedfokuset siden sist har vært å gjøre spillet spillbart.
- Utenom det har vi bare prøvd å få gjort alle punktene ferdig.

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

Punkter vi har prioritert ekstra siden sist:

- Gjøre spillet spillbart
- Fikse ulike bugs
- Skrive tester
- Nye scenes som how to play, map selection og options scene.

Vi har fullført alt vi har proritert

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

Bugs vi har utbedret:

- Hitboxen følger nå enemy
- Vi har ikke tilebased towers, så de trenger ikke plasseres midt på tile
- Vi har endret til en lesbar font i hele prosjektet
- Spawn delay baseres nå på wave

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.**

- Vi har nå gjort at det er en viss vanskelighetsgrad i spillet ved å justere pris på towers, ulike waves og lagt til oppgraderinger på towers som legger til et strategisk element til spillet.

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.)

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

Vi har nå nådd testdekning på over 75% i tillegg til at de testene som trenger det, kjøres hodeløst. Scene-klassene testes manuelt ved å kjøre spillet og teste at de ulike knappene gjør oppgaven sin.

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.

- Alle ubrukte klasser og metoder skal nå være slettet.