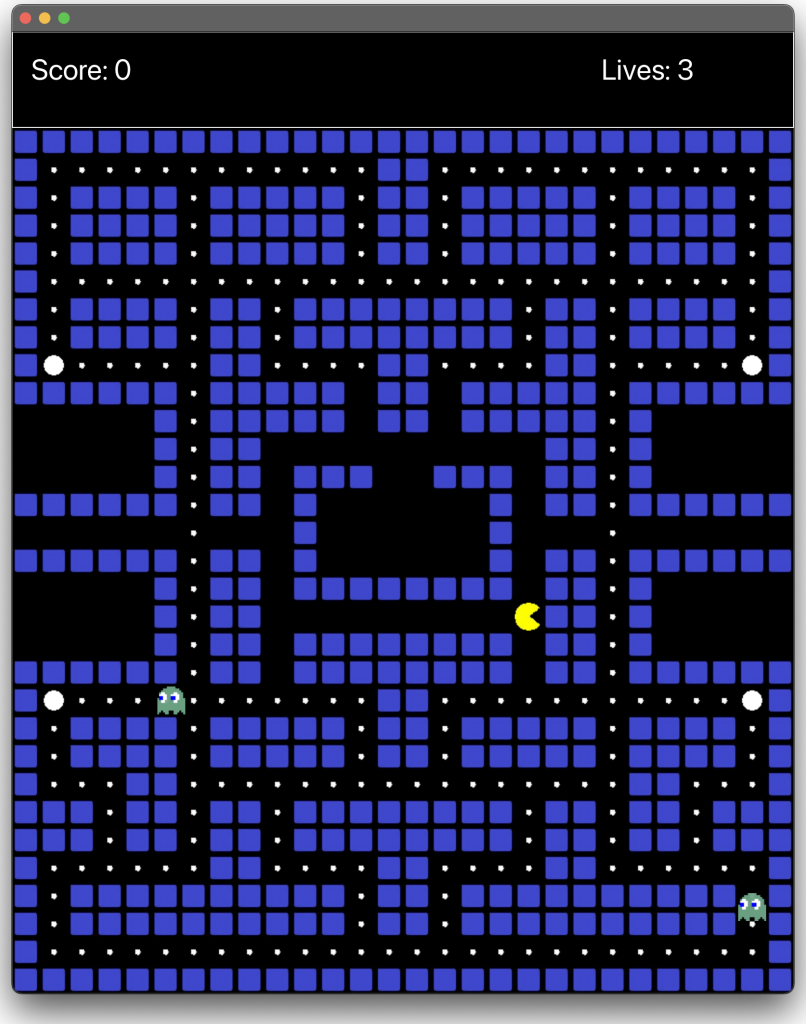
Objektorienteret Programmering Projekt PacMan

Andreas K. L. Aske W. F. Magnus R. K.

26. maj 2025



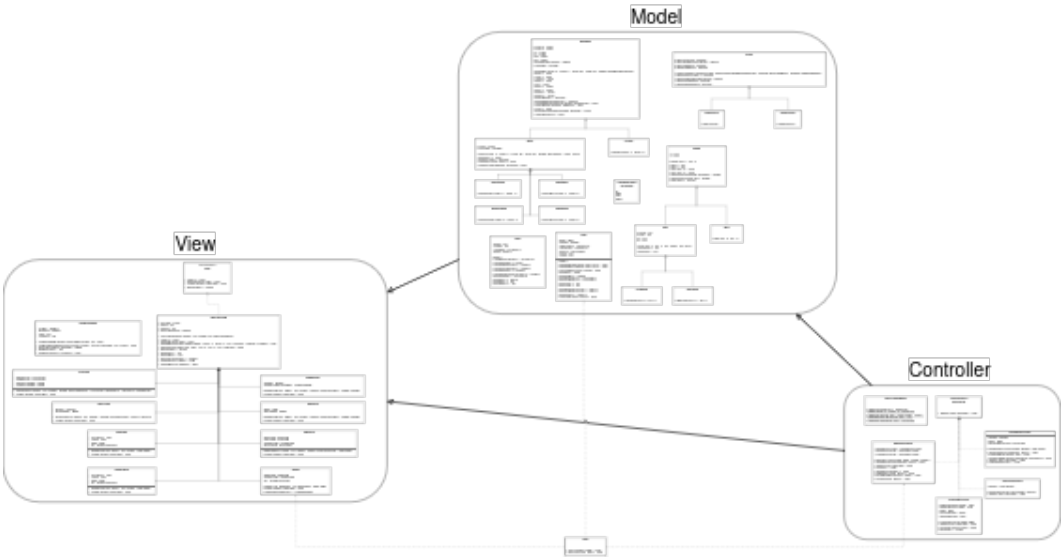
Indhold

1 Projektbeskrivelse

Til fordel for at sikre en så tro kopi til orignalen som overhovedet muligt, er dette projekts hovedformål, at udvilke spilfunktionalitet mht. kravsspecifikationen. Målet herefter, er at udvide både spillets funktionalitet samt dets brugervenlighed.

Eventuelle afvigelser fra kravsspecifikationen ses dokumenteret/diskuteret i det følgende.

2 design



Figur 1:

Projekts design følger *MVC* (Model-View-Controller) modellen. Det vil sige at vores *Model* repræsenterer hvordan PacMan spillet er bygget op med spøgelser, væge, piller osv. Så har vi vores *Controller* som står for alt logikken med hvordan ting skal kolliderer og bevæge sig, og hvornår de forskellige stadier af spillet sker. Til sidst har vi vores *View* som står for at vise spillet, med alle billederne og animationerne, samt score tekst og liv osv.

Designet kan ses i vores *UML*-diagram, hvor man kan se vi har opdelt koden i de tre dele fra *MVC* modellen, samt en *main* fil til at starte spillet, og initialiserer de andre klasser.

2.1 Model

I *Model* (se Figur 2) har vi lavet et abstrakt klasse der hedder `Moveable`, som er en abstrakt enhed der kan bevæge sig. Dette tillader os at *nedarve* fra den når vi skal lave ting der skal bevæge sig som `Ghost` og `PacMan`. Ud over dette er `Ghost` også en abstrakt klasse, så vi kan udvide de enkelte spøgelser fra den. Udover `Moveable` klasse, har vi også en abstrakt klass, `Pos2D`, til at beskrive positioner som ikke skal kunne bevæge sige. Fra denne klasse kan vi så nedarve klasser som `Pill` og `Wall`. På denne måde benytter vi *klasseafhængighedsprincippet* til at simplificerer koden, og gøre det nemmere at udvide med nye features.

Figur 2: UML-diagram til *Model* delen af projektet.

2.2 View

I *View* (se Figur 3) har vi benyttet en *grænseflade* ved navn `View` som specificere hvilke metoder et *View* skal have. Så har vi lavet en abstrakt klasse `AbstractView`, som implementerer dette interface. Vi nedarver på denne måde fra det abstrakte *View* hver gang vi laver et nyt *View* som står for at vise noget andet. På denne måde benytter vi princippet om et enkelt ansvar.

Figur 3: UML-diagram til *View* delen af projektet.

- Giv en kort beskrivelse af jeres diagram:
  - Hvad er de forskellige dele?
  - Har I anvendt designmønstre i jeres design? I så fald, hvor i diagrammet findes disse? Det er ikke et krav at anvende designmønstre, men kan være en god idé.
- Dokumentér designbeslutninger hvor I har anvendt SOLID, DRY, eller andre OO-principper.
- Hvis I i løbet af projektet har forfinet jeres design, giv da en kort beskrivelse af hvilke ændringer I har foretaget og hvorfor.

3 Implementation

- Formålet med denne rapportsektion er at give den interesserede læser et overblik over de interessante implementationsdetaljer, som er værd at kigge nærmere på i jeres kodebase, samt nødvendige detaljer for at køre jeres kode.
- Angiv hvilken version af Java I har brugt til at teste og kompilere jeres kode, og inkludérkorte instruktioner til hvordan man kompilerer og kører koden.
- Giv en beskrivelse på højniveau af interessante implementationsaspekter. F.eks., aspekter, I har brugt særligt meget tid eller energi på.
- Det kunne f.eks. være mere avancerede aspekter såsom hvordan I håndterer AI, hvordan I håndterer spilhandlinger, animation, eller andet.
- Hold beskrivelsen overordnet. Vi kan læse jeres kode for detaljerne.

4 Kvalitetssikring

- Beskriv hvordan I har testet, at jeres kode lever op til kravsspecifikationen. Har I, f.eks., benyttet unit tests? Manuelle tests?
- Ville I have taget en anden tilgang til kvalitetssikring hvis I skulle designe og implementere projektet forfra?

5 Proces

- Arbejdede I i faser i løbet af projektet?
- Hvordan gik samarbejdet, og hvordan sikrede I lige deltagelse?
- Brugte I tekniske værktøjer til at få samarbejdet til at glide nemmere på tværs af maskiner?
- Har I brugt AI som støtte under udviklingen af jeres projekt? I så fald, hvordan?

6 Diskussion

- Ville I gøre noget anderledes hvis I skulle implementere projektet forfra?
- Var der dele af projektbeskrivelsen I ikke nåede? I så fald, hvordan er disse dele kompatible med jeres design? Ville I foretage ændringer for at imødekomme ændringer?