Assignment 2 rapport - INF‑1400

Robin Kristiansen

12. april 2021

# Introduksjon

Oppgaven ba oss replikere en flokkalgoritme kalt «Boids» i Python med hjelp av pygame biblioteket. Vi skulle objekt orienterte programmeringsprinsipper slik som klasser, metoder og arv i prosessen.

# Teknisk bakgrunn

*Boids* originalt skapt i 1986 av Craig Reynolds bygger en flokksimulasjon basert på tre enkle regler: unngåelse (avoidance/separation) for å holde avstand til andre boids; innretting (alignment) for å rette seg etter den generelle retningen til flokken; og til slutt samhold (cohesion), rette seg mot midten av flokken. Med disse tre enkle reglene skal man kunne skape en nærmest livaktig simulasjon til hvordan f.eks. en gruppe fugler flyr i flokk.

Boids har en motpart, et rovdyr kan man si, hoiks. Dette er en mindre gruppe eller enkelt boid-liknende flyvende sak som har som mål å drepe boids. Den flyr rundt med nesten de samme reglene som gjelder for boids, med unntak av unngåelse. Den skal rette seg mot en flokk med boids og prøve å spise/drepe en av dem.

I oppgaven bruker jeg aktivt OOP-prinsipper slik som klasser, metoder og arv.

Boid, hoik og main-klassene

Alle klassene har mange metoder

Jeg arver fra pygame.sprite klassen

Should be short and cover the necessary topics without mentioning your specific implementation and design. A rule of thumb would be to write a mini article about techniques used, ex. Polymorphism or inheritance.

# Design

*Give an overarching view of the structure of your solution.*

Applikasjonen starter i fila main.py som inneholder klassen Game, dette er den overordnede klassen som håndterer, instansierer og kaller på metodene definer i boid.py og hoik.py. Det er definert noen variabler i \_\_init\_\_()-metoden som blir brukt aktivt utover i simulasjonen, blant annet weights, screen og delta\_time. Game-klassen setter også opp vinduet som simulasjonen tar sted i, den laster inn «bilder» for boids og hoiks, blitter litt tekst og håndterer input.

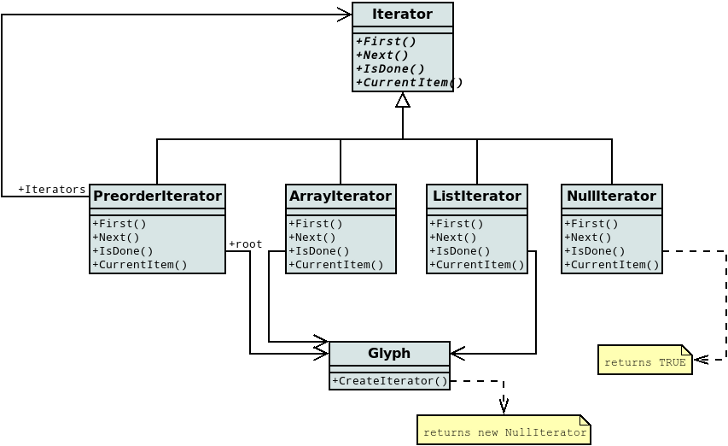
Boids klassen inneholder alle metoder som er spesifikk for boids og som trengs for å oppfylle de tre reglene som kjent med boids. Det er hovedsakelig reglene jeg blir å beskrive her i design delen: alignment() tar inn en pygame.sprite.Group (heretter kalt «sprite group» eller «sprite gruppe») og returnerer en pygame.Vector2 (heretter «Vector2»). Først sjekkes det om det er 0 naboer, hvis sann returner nåværende retningsvektor (self.vel). Om det fins mer enn 0 naboer kjøres selve «rettingen», lager en ny vektor som skal inneholde retningen som til slutt blir returnert, kjører en for-løkke

Sjekker om det er noen naboer, da det ville vært unyttig å kjøre koden med 0 naboer og vi havner i en matte-felle hvor vi ikke kan dele på 0

Itererer gjennom lista og summerer opp alle naboers velocity, etter løkken deler jeg på antall naboer, og til slutt returneres vektoren.

Tar enkelt gjennomsnitt av velocity.

Describe how your objects fit together, a figure like figure 1 must be included, and you should refer to it in the design section. Remember to refer to figures, such as Figure 1 below, in your text.

**Figure 1**: Example UML diagram

# Implementasjon

*Describe implementation details, particularly those that are not obvious choices.*

For the implementation of the paddle, the visual representation on the screen is different from the internal representation used for collision detection, by representing the paddle in this way we achieve...

# Evaluering

*Examine if your submission fulfils the requirements and what shortcomings exist.*

In this solution, all requirements are fulfilled, but collision detection between the ball and paddle is inaccurate, due to differences between the visual representation and the implementation...

# Diskusjon

*Discuss what could be done better, problems you had, experiences etc. (we also appreciate feedback on the assignment or group sessions).*

The implementation of the paddle-ball collision could be done some other way, but due to some reason, the current implemetation is better. After spending two days trying to write the report in LATEX, I gave up, and wrote it in Word instead.

# Konklusjon

*Sum up the previous sections.*

I have implemented a solution that fulfills the requirements, the implementation is moderately buggy, but does not crash too much..

Jeg har implementert en løsning som nesten fyller alle kravene satt i oppgaven. Implementasjonen er rimelig fungerende, bare ikke skap mer enn et par hundre boid-instanser, da blir programmet ganske tregt, som følge av implementasjonen min.

# Referanser

|  |  |
| --- | --- |
| [1] | The Dia Developers. Dia website, 2014.  URL http://dia-installer.de/shapes/UML/index.html.en. |
| [2] | Python Software Foundation. Python language reference, version 3.4, 2014.  URL http://www.python.org. |