TONIJA

GRPRO Projektdagbog – Tema 3

03. 12 2024 – 08. 12 2024

Gruppe 3:

Nikolai Li

Tore Haugan Engberg

Jakob Cadwallader Juhl

1 Resume

1.1 Objekt orienteret analyse

Substantiv– klasser	Verbum– metoder	Adjektiv– felter
- Ådsel - Svamp - Svampeådsel - Kødæder - Sporer	- Spise (et ådsel) - Dør (creature) - Dø(fungi) - Efterlade (fungi) - Nedbrydes (ådsel) - Afhænger (mængden af kød) - Forsvinder (ådsel) - Hjælper (svampe hjælper ådslet med at nedbrydes. - Kan ikke ses (svamp på ådslet) - Opstå (fungi på ådslet) - Placeret (svamp på kort) - Overleve (fungi) - Sprede sig(fungi) - Række længere (fungi)	- Inficeret - Nedbrudt - Størrelse (på kød og dyr) - Kødædende - Dårligere (ådsel) - Større (ådslet)

2 Design

For at skabe et bedre overblik over samspillet mellem Creature, Carcass og Fungi har vi gennemført en designanalyse. Denne analyse er med til at tydeliggøre, hvilke funktioner Carcass og Fungi skal have, og hvordan de generelt interagerer med hinanden.

2.1 Carcass

Vi planlægger at tilføje et element til spillet, der skal repræsentere døde dyr, som vi kalder Carcass. Carcass kan være enten stort eller lille, afhængigt af dyrets oprindelige størrelse, og at det kan være inficeret eller ej. Tanken er, at Carcass skal have en levetid, der langsomt tæller ned. Hvis det er inficeret, skal det nedbrydes hurtigere. Når levetiden er slut, skal en inficeret Carcass omdannes til svampe (Fungi), mens en ikke-inficeret Carcass forsvinder. Vi vil også indarbejde en mekanisme, hvor der er en chance for, at en Carcass kan blive inficeret, mens den ligger i verden. Chancen skal afhænge af, hvor langt den er i sin nedbrydning:

- I starten skal der næsten ikke være nogen risiko.
- Når halvdelen af levetiden er gået, skal risikoen være højere.
- Når levetiden nærmer sig slutningen, skal der være stor risiko for infektion.

Derudover planlægger vi, at Carcass skal kunne miste levetid, hvis noget spiser af det.

2.2 Fungi

Klassen Fungi skal repræsentere en svamp og skal kunne opstå fra døde og inficerede dyr (Carcass). Fungi kan være enten stor eller lille, afhængigt af størrelsen på det oprindelige Carcass. Tanken er, at Fungi skal have en levetid, som langsomt tæller ned, indtil den dør og forsvinder fra verden. Mens Fungi lever, skal den kunne sprede sig i et lille område omkring sig. Hvis der ligger et ikke-inficeret Carcass i nærheden, skal Fungi kunne inficere det, så det også bliver en del af processen. Planen er, at store Fungi skal leve længere end små.

3 Implementation

I dette afsnit beskriver vi, hvordan vi har tænkt at strukturere og implementere klasserne Carcass og Fungi i simulationen på baggrund af vores analyse og design-valg. Fokus er på de valg, vi har truffet omkring nedarvning, interfaces, felter og metoder, samt hvorfor disse valg understøtter deres funktion i spillet. Vi forklarer også, hvordan de to klasser passer ind i den overordnede struktur af simulationen, og hvordan de interagerer med andre elementer i verden.

3.1 Carcass

Carcass skal implementere interfacet *Actor*, så det kan være en aktiv del af simulationen, og *DynamicDisplayInformationProvider*, så dets udseende kan ændre sig dynamisk.

Felter:

- *infected* og *duration* skal bruges til at holde styr på, om Carcass er inficeret, og hvor lang tid det har tilbage.
- *isBig* skal afgøre, om det er stort eller lille.

Metoder:

- Vi planlægger at bruge metoden act til at styre, hvad Carcass gør i hver simuleringstur.
- *decompose()* skal tage sig af nedbrydningen og fjerne Carcass, når det er færdigt.
- *chanceOfInfection()* skal styre, hvordan og hvornår Carcass bliver inficeret.
- Metoder som *getIsInfected()* og *gettingEaten()* skal gøre det muligt for andre klasser at interagere med Carcass, såsom Fungi.

3.2 Fungi

Fungi skal også implementere Actor og DynamicDisplayInformationProvider, så den kan handle i simulationen og vise forskellige udseender baseret på størrelse.

Felter:

- *isBig* og duration skal bruges til at definere størrelse og levetid.
- *alive* skal bruges til at holde styr på, om svampen stadig lever.

Metoder:

- *act()* skal definere, hvad Fungi gør i hver simulation, f.eks. at sprede sig og tjekke, om den skal dø.
- *spread()* skal sørge for, at Fungi kan inficere Carcass i nærheden.
- *dying()* skal reducere dens levetid og fjerne den fra verden, når den er død.

For at gøre koden overskuelig og genbrugelig har vi valgt ikke at bruge en fælles superklasse til Carcass og Fungi, da deres adfærd er forskellig.

4 Tests og debugging

Til test af kodens metoder har vi udført en række unit tests, hvor vi har brugt *assertions* til at påstå om koden virker som forventet.

Vi har bl.a. oprettet JUnits som tester individuelle metoder:

CarcassDecomposes(), FungiDies(), digWolfHole(), Wolf.hunt(), Wolf.attack()

Vi har også JUnits som tester sammenspil mellem klasser og metoder:

CarcassGetsInfected(), CarcassGetsEaten(), FungiSpreadsSpores(), FungiSpreadRadius(), Wolf.hunt(), Wolf.attack()

Til debugging har vi primært brugt konsollen med *System.out.println()* før dele i koden, for at tjekke om programmet når til forskellige dele som det skal. Hvis ikke programmet når et bestemt punkt i koden, arbejdede vi os langsomt bagud i koden indtil man finder det sted, hvor det går galt. Bl.a. har vi brugt felterne som buffere til, hvornår kode skulle køres, og har kunnet kigge på status af felterne for at se om metoder kører når de skal.