SavannaGame

UseCase FURPS

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **// Functionality**   |  |  | | --- | --- | | **Use Case Name** | Start Simulation | | **Scope** | Funktions stadie, ingen grafisk brugerflade. | | **Level** | *User goal or sub function* | | **Primary Actor** | Observant | | **Stakeholders and interests** | None | | **Preconditions** | *What must be true on start and worth telling the reader* | | **Success guarantee** | En simulering af mindst 50 runder | | **Main success scenario** | 1. Observant starter simuleringen 2. Simulering kører I minimum 50 runder | | **Variations list** | *Varying I/O methods and data formats* | | **Frequency of occurrence** | *Influences investigation, testing and timing of implementation* | | **Miscellaneous** | *Such as open issues* |   **// Useability**   |  | | --- | | Se Scope… |   **// Reliability**   |  | | --- | |  |   **// Performance**   |  | | --- | | Et minimum af nedbrud ;) |   **// Supportability**   |  | | --- | |  |   **+**   |  | | --- | | Dette er en Consol applikation | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **// Fully dressed**   |  |  | | --- | --- | | **Use Case Name** | Start Simulering | | **Scope** | Funktions stadie, ingen grafisk brugerflade. | | **Level** | *User goal.* | | **Primary Actor** | Observant | | **Stakeholders and Interests** | Klassens elever: ønsker at se et funktionelt program der kører uden bugs  Klassens Lærer: ønsker at se elevens (mig) lærte viden inden for fagene programering og systemudvikling  Udvikleren (mig): ønsker at vise et funktionelt program der lever op til de krav stillet | | **Preconditions** | * ingen | | **Postconditions** | * Simulering kørt i minimum 200 runder * Kaniner og løver har flyttet sig på spilpladen * Kaniner og løver har formeret sig i simuleringen * Kaniner og løver har spist i simuleringen * Kaniner og løver er døde i løbet af simuleringen | | **Success Guarantee** | * Ingen fejl meldinger | | **Main Success Scenerio** | 1. Observant starter simuleringen. 2. Simuleringen kører 3. Dyr flytter rundt på spilpladen 4. Kanin møder en kanin af modsatte køn og formere sig, der tilføres 4 nye kaniner til simuleringen 5. Løve møder en løve af modsatte køn og formere sig, der tilføres 2 nye løver til simuleringen 6. Løve møder en kanin, kaninen bliver spist, løven tager på i vægt og forlænger derved sin levetid, kaninen slettes fra simuleringen 7. Kanin møder græs, græsset bliver spist, kaninen tager på i vægt og forlænger derved sin levetid, græsset slettes fra simuleringen | | **Basic Flow** | Simuleringen kører automatisk og der er derfor ingen interaktion med observanten | | **Variations list** | Simuleringen køres som en consol applikation | | **Frequency of Occurrence** | *Influences investigation, testing and timing of implementation* | | **Miscellaneous** | * Simuleringen stopper pga fejlmeddelser * Simuleringen bliver meget langsom ved mange dyr | |

Domain Model



SystemSekvensDiagram



GRASP

Arkitekturen er lag-opdelt med en decideret Controller klasse, så der på sigt let ville kunne indføres en grafisk grænseflade.

Programmet er modul opdelt for at øge muligheden for at udskifte enkelte komponenter nemt.

For at opnå samhørighed er metoder lagt i de klasse hvor den største del information til metoden allerede ligger.

De fleste metoder ligger i Board klassen da den indeholder lister med dyr og grund tiles.

Controlleren og board klassen er begge singletin klasser for at lette tilgangen til dem ellers er variabler private og metoderne public bortset fra enkelte metoder som er private.

SekvensDiagram: Add\_Lion

SekvensDiagram: PrintBoard



DesignClassDiagram



Eks på kode:

public override void Get\_Eaten(List<Animal> al,List<Ground> gl)

{

List<Ground> EatList = new List<Ground>();

foreach (Animal a in al)

{

if (a is Rabbit)

{

Point[] arr = new Point[8];

if (a.p.X > 0 && a.p.Y > 0)

{

if (a.p.X < 19 && a.p.Y < 19)

{

arr[0] = Board.Get\_Instance().GameBoard[a.p.X - 1, a.p.Y - 1];

arr[1] = Board.Get\_Instance().GameBoard[a.p.X - 1, a.p.Y];

arr[2] = Board.Get\_Instance().GameBoard[a.p.X - 1, a.p.Y + 1];

arr[3] = Board.Get\_Instance().GameBoard[a.p.X, a.p.Y - 1];

arr[4] = Board.Get\_Instance().GameBoard[a.p.X, a.p.Y + 1];

arr[5] = Board.Get\_Instance().GameBoard[a.p.X + 1, a.p.Y - 1];

arr[6] = Board.Get\_Instance().GameBoard[a.p.X + 1, a.p.Y];

arr[7] = Board.Get\_Instance().GameBoard[a.p.X + 1, a.p.Y + 1];

}

}

foreach (Ground g in gl)

{

for (int i = 0; i < arr.Length; i++)

{

if (g.p == arr[i])

{

if (g is Grass)

{

EatList.Add(g);

}

}

}

}

}

}

foreach (Ground g in EatList)

gl.Remove(g);

}