DOLLYWOOD

"SUPER ADVANCED SHEEP SIMULATOR"

Implemented in Java

Operating systems and multicore programming (1DT089)

Project proposal for group 5

Viktor Andersson (900409-4471)

Jimmy Helmersson (920425-1970)

Marcus Münger (931007-5198)

Elin Parsjö (910915-5128)

Samuel Svensäter (831130-2957)

2014-04-16

Version alpha 0.1

Table Of Contents

1. Introduction

The plan for the project is to do a small simulation of an ecosystem in a sheep pasture. There will be several actors that are dependent of each other; sheep that eat grass, wolves that eat sheep, grass that grows and more. There are many features that can be implemented but are not necessary, this makes the project easy to adapt to the situation and therefore open ended.

There were other ideas for the project than the sheep pasture. One of those ideas was to make some kind of game. But rather quickly, that specific idea was closed down. In a simulation the time can be spent on making lots of objects that work with each other concurrent instead of spending time implementing support for a player to interact with the world. Another idea was to create some sort of ticketing system. This would have been a good project for concurrency, but the sheep pasture simulation gives more practice on using graphics alongside concurrency.

The main reason the sheep pasture simulation was chosen as the project is because it makes a clear use of concurrency and that it is easily expandable. The expandability makes the project fun since there are many possibilities for improvements and the ideas of what you can do are endless. Even though it is easily expandable it is also very easy to limit which is important since there is a time limit on the project. Compared to the idea with the ticketing system a sheep pasture felt more interesting due to the possibility to learn more about advanced graphics, which was desired by the group.

During the project there is hope to learn how to program in Java using threads, actors, processes and the communication between these. An interesting part to learn is how to do a programming project from scratch without any given specifications, this will hopefully be very valuable. Another important thing the project hopefully will teach is how to collaborate and work as a group, partially by doing pair programming.

The project is interesting since it will simulate a close to real life problem, an ecosystem. It is also very expandable making it easy to add features and creating endless possibilities. Who knows, maybe the project will span larger than just a sheep pasture.

The main challenges will be how to communicate between actors and the dependencies between them. What does the "environment" look like around the actors? Since actors will interact with each other there might be some sort of pathfinder algorithm implemented, to find shortest path to other actors. How to use graphics in Java is another interesting challenge.

2. System architecture

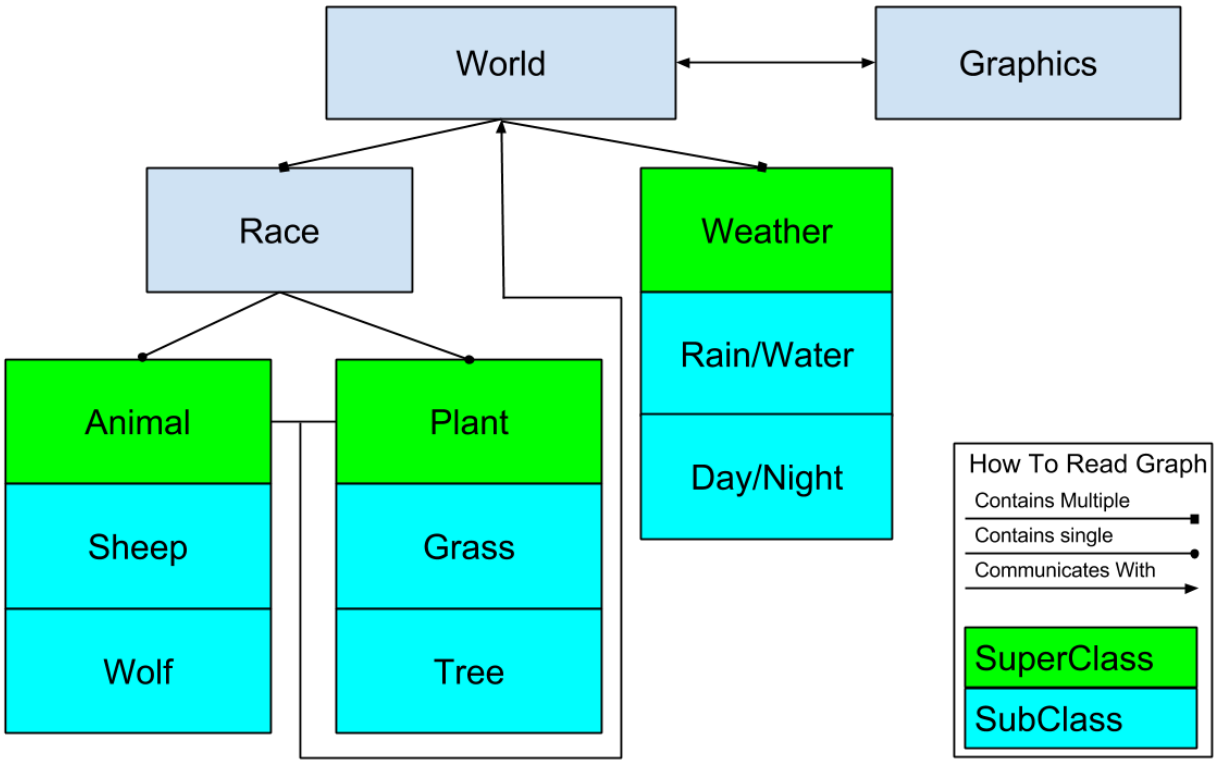

Fig 1

The current system design, see *fig 1*, was made with modularity in mind, where the main class holds a practically infinite amount of races and weather daemons. Each of the races that the main class holds will specialize in a single animal or plant type. The reason for having a race "supervisor" is to make it easier to find a single animal or plant. A sheep can just ask the main class *"is there a wolf next to me?"* and the main class in turn finds the race controlling wolves, that replies *yes* or *no*, there is a/no wolf at that position.

The graphics module, see *fig 1*, will be completely separate from the simulation in the sense that the system does not need to know that the graphics exists. This reinforces the modularity concept making it very easy to switch out the graphics module without having to go through and edit all the simulation objects. The graphical object will also provide the user with a toolbar to edit the current world setup, like re-generating the terrain, spawning new animals, editing global variables and such.

3. Concurrency models

The main method for achieving concurrency will be the use of actors. Since this will be a simulation of an ecosystem where how a sheep will react to a specific situation is dependent

on its surroundings - if a wolf comes along it might be wise to team up with the other sheep - we need the actors representing the sheep to be able to communicate with each other. For all things concerning the simulation we therefore will use actors. In order to do this we will need to use the *Akka toolkit* that is an open-source toolkit that will let us make use of actor-based concurrency in Java.

The implementation of the graphics simulation will, on the other hand, not make use of actor-based concurrency. Instead the graphics will run on a separate thread.

We chose to program this in Java since it is an object-oriented language that will help us with modularity. The inheritance of classes is optimal for this project, where for instance animals have certain things in common such as pathfinding.

4. Development tools

Java will be used as the programming language during this project and Eclipse as the IDE. Eclipse is great because it is a simple tool to work with code, unit testing and documentation. Eclipse also has features to speed up the coding, without modifying, rather than if Emacs or such would be used. For source code management and sharing EGit will be used, which is a plugin for Eclipse that provides the git control system. Java and Eclipse causes a natural choice to use JUnit as unit testing. With the same argument JavaDoc will be used for documentation.

5. Process evaluation

We started to think about what project to make at an early stage so we did the brainstorming a couple of weeks ago. It can be hard to be an entrepreneur and come up with interesting and fun ideas for a project this loose at the reins. One of the group members hatched the idea of this sheep pasture and we all felt quite early that this would be a fun and instructive project for all of us, since it is a wide and extendable project. When the time came to write this proposal we were all more or less set to do the sheep pasture. The brainstorming this week was transformed from thinking about new ideas to thinking of additional features, improving, tweaking and evaluating the original idea.