Java for Interfaces and Networks — Project

HT2016

1 Overall Objective

The overall goal of this project is to implement a distributed framework for agent-based simulation. An agent-based simulation consists of a set of Agents that "live" in an environment which is often a 2D continuous map (i.e., positions are points with double values). In our distributed versions a client manages the reasoning of an individual agent receiving the agents' perception and returning an action (with parameters) that the agent selected/computed. The overall environment is managed by a server who also displays the positions of the agents and animates the dynamics of the simulation.

2 Simulation Framework

The aim of this project is to create a client-server agent-based simulation with different types of agents controlled by clients "living" on a map that is handled, updated and visualized by a server.

You will need to implement the following elements (and put them together):

- A Simulation server that handles virtual time, administers the current state of the simulation and manages the agent clients
- Representation of the environment (map) as a data structure that represents the current state of the simulation. This data structure is updated and visualized by the Simulation server.
- A set of classes representing the actual agent-based system structure (its "ontology") including the data structures that are to be specified for perceptions (what the server sends to the client in each update cycle) and for actions (what the client returns to the server after processing its perceptions). An ideal system would have an abstract representation layer of abstract classes and interfaces. From this abstract specification-type of layer, model-specific implementation of entities are to be done. You may do a foodweb model with classes for Predator and Prey agents (and non-agent food items) or a flocking simulation following the BOIDS model (`https://en.wikipedia.org/wiki/Boids`) classes.
- Animation and data collection on the server side, including a GUI to manage the running simulation with Start, Pause, etc. interaction between human observer and simulation.

In the following, we give more details on these elements. The specification cannot be detailed in a way that there are no ambiguities, but leave space for your own design ideas.

3 Simulation server

The simulation server is responsible for handling and managing the running simulation. It shall take a simulation configuration consisting of, for example, the size of the map, attribute values

to be recorded, etc. The server waits for clients to connect with information about their agent type and initial values. Clients should be allowed to enter the simulation at any time. The server locates the incoming agent-clients on the map either given their specification or randomly.

The server starts the simulation which shall be a so-called time-stepped simulation: The server calculates what the agents could see on the map, creates a perception object and sends it to the corresponding agent-client and waits for a particular time for an action returned from the client (with timeout). After executing all actions, the simulation time counter is increased (and possibly data is recorded for logging). The server executes the given action on behalf of the agent-client and changes the environment representation by, for example, moving the agent, etc.

An interesting question here concerns how to achieve concurrency of the agent's decision making and actions. There are in principle two variants:

1. The server calculates all perceptions from the environment and at the same time sends the individual perception to all agents. When agents return their actions, the server either executes them in a random order (or in the incoming order) and tests whether actions may conflict (e.g., resulting in a not admissible overlap of agent shapes). If actions conflict, either a randomly selected action or no action will actually be executed.

2. A simpler, yet less correct way of handling simulated parallelism is to handle each agent one after the other (in a random sequence): The server calculates the perception on an agent, sends and waits for the action and immediately executes the action, then the next agent is handled.

4 Representation of the overall Environment

The agents move on a 2D continuous map (all coordinates are of type double). Each agent possesses a shape (Rectangle, Circle) with some extension. They also have an orientation, that may (or may not) determine what they can perceive on the map. The server manages the map, based on distance calculations, it can determine which other entities an agent can actually perceive. Overlapping shapes may be admitted or not depending on what you want to implement. The map may – especially in the foodweb case – contain non-agent entities or `Resources` that agents may also perceive and act on (eat, avoid...). These non-agent entities may be updated by the world process before or after all agents have been updated.

5 The Agents

An agent is represented by two parts: a `body` that is located on the map and that may have physiological properties like an energy level. If you use those body-level properties, the perception object must also contain information on the body state of the agent. The second part of the agent is its "mind" – located in a remote client. You may assume that the remote client is started and connects to the server which then creates the body on the map and maintains the connection to the client. During the simulation, the agent-client receives a perception object that represents what the agent may see on the map. The perception object must contain information on in which direction another object is visible and what type this object has. Transforming the global coordinate system of the map to a local coordinate system of the agent might be challenging. The agent client parses the perception and applies some calculations to determine its next action. These decision making rules may be rules in form of if-then-else cascades ("If food is ahead, then makeStep" in the predator/prey foodweb simulation) or calculations of movement direction and stepsize (in case of the BOIDS example).

The action repertoire that the agent client may select/configure an action to be send to the server, needs to be carefully designed as it also determines the behavioral complexity that the client may express. You may send a simple atomic string or Enum such as "MoveOneStep" that translates the agent one unit of movement in the direction it looks into. Alternatively,

you may foresee categories of actions that are specified with additional parameter such as $< MOVE, speed, direction >$ expressing a translation with given speed into the given direction. Think carefully about what actions you want the agent to be able to do. This is highly depending on the actual application. The predator/prey model seems to be more suitable for a more token-like approach with MoveStep, Turn, Eat, Reproduce,... while the BOIDS model must use more elaborate move commands.

You may select what agent-based simulation model you actually want to implement. We suggest two alternatives:

- Predator/Prey simulation. There are two classes of agents both inheriting from the Entity class: Prey agents randomly walk around on the map and eat all passive food items that they encounter. Each eat action – if successful – increases some energy value. If the energy is high enough (threshold), the Prey agents reproduce, that means create new offspring Prey agents (with appropriate client). Predator agents also wander on the map in search for Prey agents to feed-on. For both agent types, every movement costs energy. If the energy level falls below a threshold, the agent dies. Foodweb models are very popular, yet finding a good parameter setting might be challenging.

- A simpler model is the BOIDS model first published by C. Reynolds. It reproduces how birds, fishes and other social animals move in groups. The movement speed and direction of an agent is calculated based on the movement speed and direction of its neighbors. For the exact rules for calculation see the link above. This model appears to be simpler, yet the action representation may be more complex. One may enrich this simulation with obstacles scattered over the map.

If you encounter another model that you would prefer to implement in this way, this would be also possible, if the model is not too complex. Contact Franziska in this case.

In principle, agent-based models are hard to calibrate - that means the parameters influencing agent behaviors are many and have a huge and often non-linear effect on the overall outcome. So, don't despair if your populations die early or the swarm clumps, try a different set of parameters.

6 Animation and Data Generation

The server also manages a visualization of the map. It may after each executed action or after finishing an update cycle update the visualization of the map showing icons, circles or rectangles for each of the agent. You may encode energy levels in colors, etc. Yet, don't invest too much time into how to visualize your agents. You may also add a listener to each entity – displaying agent properties when a human observer/user clicks on an agent icon/representation.

As an optional extension you may record what happens in the simulation or display it during the simulation in a different form: For example in the Predator/Prey model you may log/display the current numbers of agents of each type; in the BOIDS model, the overall average speed or distance between agents, etc.

7 Project Report and Hand-in

This assignment leaves a lot of possibilities for code-reuse and nice code design. You are strongly advised to spend some time thinking about the contents of the project before you start programming so that you can maximize code-reuse through inheritance and clarity through using interfaces, etc. Support your design by using UML diagrams for classes and interaction. Make sure that you have properly documented your code, including Javadoc documentation. Projects are individual, i.e., team working is not allowed in this assignment. The source code will be analyzed with tools designed to detect code plagiarism and all cheating will be reported. Each student must submit a short report written in English containing:

- Introduction to the assignment: clearly write a description of what you are asked to do
- Design of the simulation platform: describe your idea of how to implement basic server and client and support your idea with the diagrammatic techniques proposed above (UML).
- Implementation of your particular predator/prey or flocking model
- Discuss your design and implementation: mention any problems or pitfalls that you met during the implementation of your proposing design, and discuss what you could have done better if you knew from the beginning.
- Give some information on how you tested your platform and describe its performance

7.1 Handing in your project

The project is the only mandatory element of the practical part of the JAVA course. Projects are individual! The deadline for this assignment is Tuesday, January 3rd, 2017. You must send your material (project source code and report) to in a zip-file both, Uwe (`uwe.kockemann@oru.se`) and Franziska (`franziska.klugl@oru.se`) by that date. We will prepare sufficient slots for demonstration of your simulation framework during the rest of week 1 for those who handed in in time and passed a pre-check of the submitted material. State in the email and the report, in which class your main method resides. We will create a doodle so that students who handed in can choose a slot of demonstration. Slots for demonstrations for slightly delayed hand-ins of projects will be available in week 4 and during the week after the re-exam in March (week 12). Without handing in a report, it is not possible to demonstrate. During the demonstration you will not be examined in a harsh way, we just want to make sure that you understood what you did. We might request updates for the report during demonstration.

Do not hesitate to ask if you have questions, if something is unclear or you are not sure whether your approach may lead to some result. We want you to pass this course!