Wildfire Season Report #4
(Agent Requests and Refined Strategy)

Multi-Agent Systems Group 6: Daniel Brüggemann, Carsten Orth, Karsten Seelert

Agent Requests

This week, we extended our communication model by integrating requests; An agent can send a request to other agents (within a certain range) either to receive information of a certain type in a certain position (“is there a fire at position x/y”, “where are the other foresters positioned”) or to get other agents to help by executing a certain action at a certain position (“Extinguish the fire at x/y”). This is the extended communication-related class diagram:

https://eleum.unimaas.nl/courses/1/2015-200-KEN4111/groups/_31128_1//_513677_1/Agentclass.png

We also introduced the `Action` class to represent time-consuming actions like extinguishing, patroling (going through the forest searching for fires) etc. Actions also define a reward, a bounty, for contributing to the action’s goal and/or for fulfilling this goal (`incrementalBounty`, `finalBounty`). However, the bounty is not yet integrated into the decision making process. While actions like extinguishing fire are valuable even if just done partially, actions like preemtively clearing parts of the forest to prevent fire from spreading are only useful if done completely; otherwise, this could even count as destructive behavior.
Actions are contained within an `Intention` instance storing the position where the action should be executed (if necessary). Each forester agent decides on its current intention by evaluating the environment, requests he agreed upon and requests from other agents.

The following flow diagram describes the refined steps of the agent’s communication and action strategy:

https://eleum.unimaas.nl/courses/1/2015-200-KEN4111/groups/_31128_1//_513672_1/generalBehavior.png

Each of the main steps (`checkNeighborhood`, `doRequests` etc.) are priotized scheduled methods, meaning they all happen in a single time step.
First, information gained from the surrounding environment is gathered. Then, requests are created and sent out, either to obtain certain information or to ask for help with a certain task.
Then, answers to these requests are sent to the request creator. `InformationRequests` are just answered by the corresponding information class (given that the agent has such information), while `ActionRequests` are answered in the form of `RequestOffer` objects. In the following step, these responses are checked by the request creator and evaluated (for now simply by their distance; the closest agent(s) are chosen).
The chosen agents are notified via `RequestConfirm`. This is a binding confirmation, the notified agents are expected to help. This is why an agent will only send out one offer to help per time step, and this offer expires in the next time step. If an agent does not receive a confirmation, he knows he was not chosen and can make offers to other requests.
Finally, in the `doAction` step, the agent decides on his intention (got a confirmation? Set intention. No intention yet? Start patroling) and either moves toward the target position

where the corresponding action should be executed or, if these preconditions are given, executes the action.
An agent also needs an option to cancel his participation to a request or the request in its whole under certain circumstances, e.g. if the task is done or impossible (fire got extinguished, fire burned down the forest tile, agent found a request with higher priority and closer nearby). For this the `RequestDismiss` class is used.

Exploration

It is most efficient to extinguish fires before they spread, which means they have to be found early. Before, our agents quickly clustered together and moved together to known fires, which leads to fast extinguishing, but fires in areas far away are not noticed.
We used a simple rule set to try to spread the agents across the forest grid:

- Check for other agents in a certain range, then move away from the closest one.
- Only move to tiles where there is still forest (if possible)

There is no random movement anymore.
This is a small video showing the agent's movement (yellow) through the grid:

https://eleum.unimaas.nl/webapps/blackboard/execute/groupFileExchange?course_id=_39741_1&action=LIST&group_id=_31128_1

Conclusion and Perspective

The strategy got a lot more complex. Still not all data gained from the environment is used in the decision process, but the request model leads to a more organized distribution of (agent) resources.
We created a `Statistic` class which is put in the context as an intermediate data storage and enables to show certain values in percent. This way, we can now show different values in one graph and compare tendencies better. Also, we fixed the dropping of the "Fire extinguished" value when an agent dies by creating a `GraveyardStatistic` where they can rest in piece (while we can still keep their contribution).
Messages can now be send to a specific agent, which is essential for the information exchange when handling a request.

TODOs for next week:

- Bounties and costs have to be integrated into the decision making process.
- Creating the model took a lot of time, we could not test simulations and evaluate the graphs that much. This has to be done next week.
- The preemptive actions Wetline and Woodcutting still have to be implemented and integrated.