# 02285 AI and MAS, F24
# **Programming Project**

Thomas Bolander, Andreas Garnæs, Martin Holm Jensen,
Mikkel Birkegaard Andersen, Mathias Kaas-Olsen

This document describes the main programming project of the course. The goal of the project is to construct an advanced client for the *MAvis* server using the hospital domain. So essentially the problem is the same as in the warmup assignment, except here we expect you to experiment with advanced techniques and architectures to be able to solve the widest possible range of multiagent levels within the hospital domain efficiently. The client of the warmup assignment is not particularly fine-tuned or efficient, so you are strongly encouraged to rebuild your own client from scratch. You can obviously take inspiration in the warmup assignment, but building your own client from scratch and thinking about how to e.g. represent states efficiently is a very valuable exercise, and will also make it easier for you to afterwards extend the client with more features.

**Expectations.** The project is purposely designed to be very open-ended and flexible, allowing many different group sizes, types of solutions and levels of ambition. Successful implementations of the project range all the way from basic solutions using standard techniques to highly research relevant multi-agent systems. However, a basic extension of the warmup assignment will **not** count as a sufficient solution. You are not going to be able to create an efficient client by simply further tweaking the heuristics and/or use a slightly modified variant of graph search. What you deliver in the programming project has to be **significantly** more advanced and efficient than what you did in the warmup assignment.

In the programming project you will only work with multi-agent levels. You might of course use single-agent levels to try out your algorithms for controlling the individual agents of a multi-agent system, but in the final competition all levels will be multi-agent levels.

The programming project constitutes approximately 3.75 of the 7.5 ECTS in the course. This means that the expected total workload of the programming project is approximately 100 hours per student (including participating in the competition, preparing the video, etc).

**Group sizes.** Group sizes can range from 2 to 5 students with 3–4 as the recommended numbers. Groups of 2 will be expected to deliver the same as groups of 3. Groups of 2 students are by default **not encouraged** as it gives you a bit too few resources for a project of this size and level of ambition. Groups of 5 students are also by default not recommended as it can be hard to distribute tasks between you in such a large group. But if you think you can manage and are able to handle your project management efficiently, it will of course give you more ressources to create a nice solution. Groups of 4–5 are expected to deliver a somewhat more substantial and thorough solution to the assignment than a group of 3, however, of course realising that a group of 4 can't necessarily produce 30% more than a group of 3, and similarly when going from 4 to 5.

**Competition.** Towards the end of the course there will be a competition where your different clients will compete against each other. You will be given a number of multi-agent levels, and your implemented client should then try to complete each level as fast as possible and using as few joint actions as possible. In the competition, the environment server will time clients out after 3 minutes, so your AI only has 3 minutes to complete each level. Furthermore, your client can use at most 20.000 joint actions to solve a level.

Your client will get two scores for each level, an *action score* and a *time score*. These are both numbers between 0 and 1 with 1 being the best. If a level is not solved within the time and action limits, both scores will be 0. Otherwise, the scores are calculated as follows:

$$\text{action score of your client} = \frac{\text{fewest number of joint actions among all successful clients}}{\text{number of joint actions used by your client}}$$

$$\text{time score of your client} = 1 - c\log\left(\frac{\text{time spent by your client}}{\text{time spent by the fastest among all successful clients}}\right)$$

where $c$ is a constant such that 0.1 is the lowest score of any solved level (the essential property is that the scoring is logarithmic to avoid that the chosen programming language or hardware used influences the scores too much). Both your action and time scores for the individual levels will be summed up, and two winners will be announced: one for having the best action score, and one of having the best time score. There will be prizes for the winners. We will also sum up the action and time scores, and give prizes to the top-5 teams overall. Finally, there will be a prize for the most interesting and novel level submitted to the competition.

**Competition and assessment.** The goal of the competition is to motivate you to make the most efficient possible solution, and to allow comparison of the strengths and weaknesses of the various submitted solutions. Note, however, that your result in the competition will not directly affect your grade. A group might for instance challenge itself by trying out advanced novel techniques or focus mostly on optimising a certain aspect of the solution, e.g. multi-agent communication and coordination. That group might not be able to produce the most efficient solution, but the novelty and quality of both solution and video could still be very high.

**Submission of competition levels.** Each group has to design and submit one multi-agent level (i.e., containing at least 2 agents). The submitted levels will be included in the set of competition levels. These levels are required to be of size at most 50x50 grid cells. Note that a level does not necessarily have to be of big size in order to be challenging, as for instance a small level with tightly coupled subgoals will by default be much harder than a big level with more or less independent subgoals. You should only submit levels that your own client can solve within the time and action limits. *Any level submitted for the competition that the group's competition client cannot itself solve will automatically be excluded from the competition results!* The final competition levels will be a combination of the levels submitted by the students and potentially additional levels designed by the teachers. Levels will be selected to ensure variation in difficulty and features.

## Schedule

**13 May at 20.00.** Deadline for submitting competition level. First register your group as one of the groups under "Groups for Programming Project" on DTU Learn. Then pick a group name of at most 9 letters (restricted to the letters `a-z` and `A-Z`), e.g. `DeepGreen`, `MASochist` or `WatsOn`. You can unfortunately not change the group name on DTU Learn, but you're going to use the group name for your level file, and we will refer to this group name in the competition, etc. In the following, we will refer to your chosen group name as `<group_name>`. Then create a multi-agent level for the competition, a plain ASCII text file. The name of the file must be `<group_name>.lvl` and the name of the level after the `#levelname` line inside the file must also be `<group_name>`. So a group named DeepGreen would be submitting a file named `DeepGreen.lvl` in which the level name inside the file is also DeepGreen. Submit the file to DTU Learn using the assignment "Submission of competition level and group registration". Remember that your levels can be at most 50x50 grid cells large and that levels are not allowed to contain tab characters. Levels that are incorrectly named or incorrectly formatted will not be considered for inclusion in the competition levels.

Note that there will be almost no time to improve your implementation after the submission of your competition levels, so your client should be completely finalised when you submit the competition levels, afterwards you have at most a couple of days to polish it further.

**15 May at 08.00(am).** The set of competition levels will be made available in a file `complevels.zip` under `Content/Programming Project` on DTU Learn. You should download and unpack this zip file in a separate directory and then run the server with the following command:

```
java -jar server.jar -c "<client-cmd>" -l "<path-to-complevels-dir>"
                        -t 180 -o "<group-name>.zip"
```

Make sure that the client identifies itself with your group name to the server, so the group name will be correctly registered in the log files! If you're uncertain about whether you're currently doing this correctly, try to produce a single log file of a simple level and check that your group name is correctly stated in the line following `#clientname` in that file. Run

```
java -jar ../server.jar -h
```

for detailed instructions of how to write to log files. When you are done, you should have produced a file `<group-name>.zip` with log files for all the competition levels. You can unzip this file to check how well your client did on the individual competition levels. To replay the log files, use

```
java -jar server.jar -r "<path-to-logfile>" -g
```

The zip file also contains a file `summary.txt` containing a summary of the actions and time spent on each individual level. When you're happy with your results (!), you should upload the zip file produced to DTU Learn, see below. Note that it can take up to 3 minutes to run your client on each level, so it can take several hours to run the server on all competition levels!

**17 May at 20.00.** Deadline for submitting:

1. `<group-name>.zip`: The file produced by running the server on the competition levels, see above. **Don't tamper with the zip file produced by running the server! You can unzip the file when you're still making experiments, but the final zip that you upload should be the completely unmodified zip file produced by running the server!**

2. `code.zip`: A zip-file containing all the source code and executable code of your implemented software. No `.rar`, `.tar`, `.gz` or other formats, please!

Upload both files to the assignment "Submission of competition results and source code" on DTU Learn. Since you hand in your code together with the competition results, you can not change the code afterwards. This is to make sure that you spend the time after handing in your results on preparing the video, not further improving your code. The video is equally important as your implemented system.

**23/5-24 at 14.00.** Presentation of competition results in Arena at DTU Skylab (building 374). We will present the detailed scores of all groups, present head-to-head battles between pairs of clients running on the same level, and announce winners. Each group should be prepared to give a short 1 minute explanation of the behaviour of their client on any of the competition levels. This is the official day of examination in the course, so everybody should be able to participate, and are required to do so (with the exception of students who for some reason have another course in the same schema group and are going to another exam on the same day).

**4 June at 20.00.** Deadline for submission of your video to the assignment "Submission of video" on DTU Learn. The conditions for your video are stated in the separate document `guide_for_video.pdf`. It is very important that your first slide includes a *group declaration.* The group declaration should give a specification of who did what in the project, in terms of ideas, literature search, implementation, preparation of the video, etc (with the same overall conditions as for the Warmup Assignment). The length of the video should be **at most 12 minutes**. Anything after the 12 minute mark will not be watched and assessed. Make sure to include relevant demos in your video, as a minimum a video demo of your client running on your own level, including an analysis of its behaviour (why is it doing what it's doing? How could it potentially be improved?).

Good luck with your project. Be creative! Have fun!