G54DIA:
Designing Intelligent Agents

Lecture 9: Coursework 1: Getting Started With Your Project

Natasha Alechina

School of Computer Science

nza@cs.nott.ac.uk

Outline of this lecture

- description of the coursework
- some hints to help get you started
  - classifying the task environment
  - designing the agent architecture
  - approaches to implementation
- structure and content of the report

Coursework

- the *coursework* involves the design and implementation of a single agent
- assessed by submissions describing your agent and the associated code – due *Friday 27th of February*

Specification

- the *specification* states what you are going to do:
- the *assumptions* you are going to make for the purposes of your project, e.g.:
  - properties of the task and environment
  - percepts and actions available to an agent
- the specification shouldn't say *how* you agent does things, only *what* it does

Standard task environment in detail 1

- the environment is discrete and consists of a grid of cells
- the environment contains randomly distributed stations and wells
- stations periodically generate *tasks* – a request for a specified amount of water (max 10,000 litres)
- tasks persist until they are achieved (a station has at most one task at any time)
- wells contain an infinite amount of water
- there is a single fuel station in the centre of the environment that contains an infinite amount of fuel

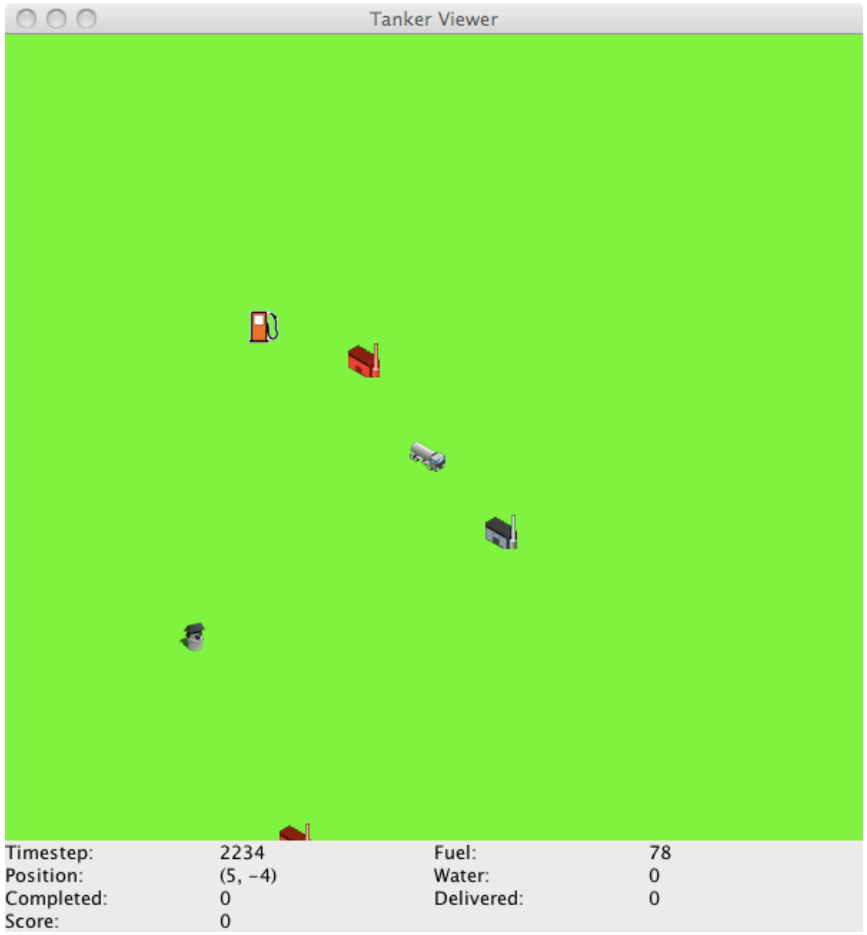
Standard task environment in detail 2

- the agent can see any stations and wells within 12 cells of its current position
- if a station is visible, the agent can see if it has a task, and if so, how much water is required
- the agent can carry a maximum of 100 litres of fuel and 10000 litres of water
- the agent moves at 1 cell / timestep and consumes 1 litre of fuel / cell
- filling the fuel and water tanks and delivering water to a station takes one timestep
- if the agent runs out of fuel, it can do nothing for the rest of the run

Standard task environment in detail 3

- the agent starts out in the centre of the environment (at the fuel station) with 100 litres of fuel and no water
- a run lasts 100,000 timesteps
- the success of an agent in the task environment is determined by its score at the end of the run
- the agent's score is given by the amount of water delivered × number of (completed) deliveries

Standard task environment



Software design

- the *software design* states how you are going to achieve the project specification
- it is an *abstract* description of how you are going to solve the problem:
  - high level: what sort of architecture your agent has
  - low level: how the agent decides which tasks to perform in which order, and which wells to use for each task
- it should *not* be a list of classes and methods

Documenting your design

- you need to describe your design and the *reasons for each design decision* clearly in your report
- a good approach is first to say which general type of architecture your agent has (and why)
- then explain the main components or steps in its operation in outline
- then describe each component or step in detail

Software implementation

- high level description of the implementation:
  - which data structures were used
  - how the algorithms were implemented etc.
  - why the approach adopted was chosen
- try to focus on the ‘interesting’ bits of the implementation
- a full code listing is *not* required

Evaluation & discussion

- how well does your agent (or agents) work, e.g.:
  - what score does it achieve (on average)
  - how does this compare to other agents (e.g., simpler versions of your agent)
- why is your solution appropriate for the task environment
- which features of the task environment are critical – how would you expect it to perform in other task environments?

Possible report outline

- your name, email address, student id and “G54DIA coursework 1 report”
- introduction
- relevant background material
- specification
- design
- implementation
- evaluation (average score over at least 10 runs)
- discussion/conclusions
- references

How it will be marked

- aim of the module is to understand the relationship between an agent's task environment and its architecture
- to do well, you need to develop an agent (or agents) that work well, and demonstrate that you *understand why they work well*
- marking is therefore based on:
  - the capabilities of the implemented agent(s), including the quality of the specification, design and implementation
  - the degree to which the specification, design and implementation are clearly documented in the report
  - clarity of presentation in general (including grammar, spelling and punctuation)

Assessment guidelines

- very broadly, a basic implementation of the minimal requirements (and corresponding report) would gain a pass mark
- extra credit will be given for submissions that demonstrate a clear understanding of the relationship between the specified task environment and the architecture of the implemented system
- this does not necessarily involve implementing one of the extensions– it is possible to get a first class mark by doing an excellent implementation of the minimal requirements and an excellent report
- full assessment guidelines on the module web page

Some hints …

Classifying the task environment

- a good way to start is by classifying the (standard) task environment given as part of the problem
- what properties do the agent's task, percepts and actions have?
- hint: see lecture 2 …

Designing the architecture

- once you understand the features of the problem, think about their implications for the agent's architecture
- use the features of the task environment to help you make *and justify* high-level decisions about the design of your agent
- e.g., is the environment observable? – if so, what does this mean for the architecture of your agent?

Low-level design

- once you have made the high-level decisions, think about how each aspect of the agent could/should be implemented
- e.g., how will the agent search the environment, or decide what to do next
- will your agent always use the same action selection function, or will the action selection function vary with time, etc.
- you can use algorithms from agent case studies in the lectures, from previous AI courses or AI textbooks, or just invent your own solution

How to start implementation

- once you have a design you will want to start thinking about how to implement it
- you may want to try a simple case first which is an agent that can collect water from the nearest well
- basic capability that will be required by almost any design
- aim is to ensure that you are familiar with the toolkit, and get some practice using it

Collecting water from the nearest well

- starting from the fuel station, your agent must be able to:
  - find the nearest well
  - collect water from the well
  - return to the fuel station
  - without running out of fuel

Finding the nearest well

- finding the nearest well requires:
  - finding a well
  - somehow ensuring that the selected well is the closest to the fuel station
- several approaches are possible:
  - design the search for wells so that the first one found must be the closest; or
  - having found a well, check to ensure that there is no other well(s) closer to the fuel station

Collecting water

- collecting water requires navigating to the selected well (easy)
- returning to the fuel station (also easy)
- not running out of fuel in the process (see the demo agent)

Tutorials

- the project work is supported by group and individual tutorials
- **group tutorials** cover the use of the Java agent package
  - group tutorials start this week, during lecture slots. The first one is at 11 today.
- **individual tutorials** cover the design and implementation of your project
  - individual tutorials are scheduled during G54DIA lecture and tutorial slots for two weeks starting 17 February
  - email Lavindra de Silva (Lavindra.Desilva@nottingham.ac.uk) or me (nza@cs.nott.ac.uk) to make an appointment. I am only available in the week of the 23rd of February (not next week).

The next *lecture* (3 March)

Multi-Agent Systems

Suggested reading:

- Ferber (1999), chapter 1

The rest of this week: group tutorials during lecture slots

- Coming up: Java coursework package