G54DIA: Designing Intelligent Agents

Lecture 6: Getting Started With Your Project

Brian Logan

School of Computer Science

bsl@cs.nott.ac.uk

Outline of this lecture

- requirements for the interim submission
- some hints to help get you started
  - classifying the task environment
  - designing the agent architecture
  - approaches to implementation
- requirements for the final submission in a later tutorial

Coursework

- the *coursework* involves the design and implementation of a simple agent (or agents)

- assessed by interim and final submissions:

  – **interim submission** containing the specification and outline design of your agent(s) and an implementation of an agent that can collect water from the nearest well – due *Friday 28th of February*

  – **final submission** describing your agent(s) and the associated code – due *Friday 11th of April*

Interim submission

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

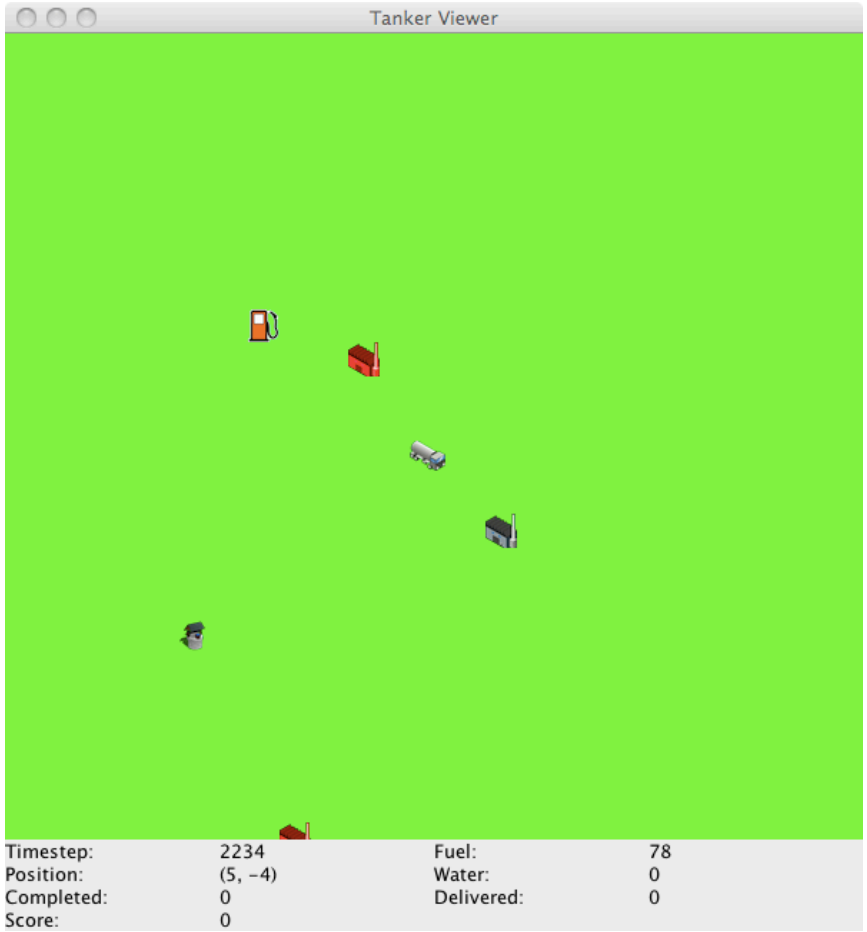
Standard task environment in detail 2

- 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
  - if you want to use multiple agents, which activities will be done by which agents, how the agents will interact etc.
- it should *not* be a list of classes and methods

Contents of the interim report

- if you plan to use the standard task environment all you need is a statement to this effect – you do not need to repeat material from the slides
  - if you plan to modify the task environment please discuss any changes with me *before submitting your report*
- you *must* give a design for a *single agent*
  - the design should be *justified by an analysis of the task environment* (see lecture 2)
  - if you believe the design to be optimal, say why
- you *may also* give a design for a *multi-agent* version
  - please put all material about any multi-agent extension in a separate section

Suggested structure

- your name, email address, student id and “G54DIA interim report”
- Part I: problem specification
- Part II: software design for a single agent
- Part IIa: (optional) software design for any extensions
- Part III: how you implemented an agent that can collect water from the nearest well
- all sections together should not exceed 1,000 words

Interim code submission

- you should also submit an implementation of an agent that can collect water from the nearest well
- submit enough code to demonstrate that your agent can collect water from the nearest well
  - at a minimum, a revised `senseAndAct` method from the `DemoTanker`
  - if you have written/modified other code in the `demo` package, submit the whole `demo` package

How it will be assessed

- the interim report constitutes 10% of the assessment
- assessment will be based on the content and clarity of the report
  - clear statement of the problem specification
  - any changes to the problem specification should be clearly explained and motivated
  - reasonably detailed design for a single agent – design choices should be clearly motivated
  - any multi-agent extension should clearly explain how the agents will interact
- and how your implementation collects water from the nearest well

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 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

Documenting your design

- once you have decided on the design of your agent, you need to describe the design and the reasons for each design decision clearly in your interim 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
- if you find yourself writing more than 1000 words you are going into too much detail for this phase of the project

Initial implementation

- once you have a design you will want to start thinking about how to implement it
- the only requirement for the interim submission 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)

Documenting your implementation

- you should include a brief description of how your agent collects water from the nearest well in your interim report, e.g.,
  - how does it find a well
  - how does it (or you) know that it's the closest
- submit enough code to demonstrate that your agent works
  - e.g., a revised `senseAndAct` method from the `DemoTanker`

Tutorials

- the project work is supported by group and individual tutorials
- **group tutorials** cover the use of the Java agent package
  - the first group tutorial is on Friday the 14th of February at 12:00 in C60
- **individual tutorials** cover the design and implementation of your project
  - individual tutorials are scheduled for 15:00-16:00 on Tuesdays, 16:00-17:00 on Thursdays and 12:00-13:00 on Fridays (starting 14th of March) in C34
  - email me to make an appointment

The next lecture

Deliberative Architectures I

Suggested reading:

- Russell & Norvig (2003), chapter 11
- Wooldridge (2002), chapter 4