INF5820/INF9820: Obligatory Assignment 2

The deadline for the submission of this assignment is **October 21st, 23:59**. The assignment must be submitted via the Devilry portal:

`http://devilry.ifi.uio.no`

Please don't hesitate to contact me if you have any questions or problems.

1 Description of the assignment

In this assignment, you will be asked to develop a small dialogue system for a simple, simulated human-robot interaction scenario. The scenario works as follows: you have a small humanoid robot (e.g. a Nao) located on a small table where various visual objects can be perceived,as shown in Figure 1.

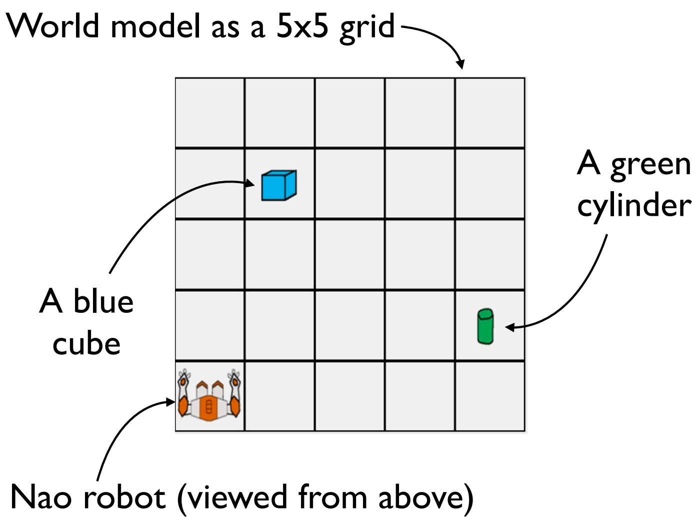

Figure 1: The simulated world model

The robot can move around using 4 basic operations: move forward, move backward, turn 90 degrees left, and turn 90 degrees right. It can also pick up the object in front of him, and put down an object it currently holds. That makes 6 physical operations in total.

In addition, the robot can also perceive spoken utterances and synthesise spoken responses. The user can then interact with the robot to ask him to perform specific actions (e.g. “move forward”, “pick up the blue object”, etc.), and the robot should then react appropriately.

Your task is to implement a simply *dialogue policy* telling the robot what to do upon receiving a new user utterance. Ideally, the policy will include a few simple routines for e.g. natural language understanding (using e.g. shallow matching techniques), dialogue management and generation. The dialogue policy should be able to handle the following:

1. Greetings (“hi”) and closings (“goodbye”)
2. Commands to move the robot (“go forward”, “turn left”, “turn right”)
3. Talking about visual objects (“do you see an object?”, “what do you see?”)
4. Picking up and releasing objects (“pick up the cylinder”, “put down the blue cube”)
5. The robot must also be able to produce clarification requests (“sorry could you repeat?”, “could you confirm that...”) when things are unclear.

You are free to decide what kinds of utterances you want to exactly include in your domain, as long as it captures the behaviours listed above. You will have to provide a speech recognition grammar for these utterances. (cf. below).

2 Connection to AT&T API

The speech recognition and synthesis will be performed “in the cloud”, by relying on the API provided by AT&T. You must first register to their services on the following webpage:

`https://service.research.att.com/smm`

After registering, you should read Chapter 4 of the Speech Mashup Application Developer’s Guide available on the website[1] to understand how to write simple rule-based grammars. You can easily test your grammar by using the online “ASR test” page on the portal. You can also test the TTS engine in the same way.

[1]You can skip the other chapters, which are not relevant for you.

3 Software integration

Once you have created and tested your first recognition grammar, you're ready to integrate these elements in a simple dialogue system. Download the package for the assignment, using SVN[2]:

```
svn co https://opendial.googlecode.com/svn/branches/oblig2
```

The code contains a `README.txt` file to get you started. The easiest way to compile and start the code is via Ant – see the readme file for details. You can develop your code either in Java or in Python (provided you use the Jython implementation, which makes it possible to use Java libraries in a Python script). In any case, you will have to change the `uuid`, `appname` and `grammar` parameters in `Main.java` or `Main.py` to the right values for your AT&T profile!

The initial code only contains a dummy DialoguePolicy which simply repeats what the user just said. Your task is to implement a more sophisticated policy which is able to capture the dialogue domain described in the previous section. More specifically, you have to provide an implementation for the method:

```
public Action processInput
     (NBest u_u,
     DialogueState dstate,
     WorldState wstate) { ... }
```

The method takes a new user utterance as input, as well as a dialogue state (including the dialogue history) and a world state (describing the robot position as well as the objects in the scene). The method outputs an `Action`, which might be a `DialogueAction` (an utterance to synthesise), a `PhysicalAction` (a movement to execute), a `VoidAction` (nothing to do), or a `CompoundAction` (a combination of several basic actions).

Of course, you won't be able to implement this complex procedure in one single method – you will have to factor it into several independent functions (e.g. shallow parsing, action selection ,etc.). You are free to decide how to structure your code, and which approach you want to adopt to address each subtask.

4 Delivery

You should submit on Devilry the full code (please include the full package, it will make my life easier to test your code), as well as a *short* (1-2 pages) report on your system and the design choices you made.

Good luck!

[2]I decided to use SVN in order to ease the debugging and refactoring process. Simply install it from `http://subversion.apache.org/` if you don't have it yet.