Practical 02: Pacman deals with limited visibility

(Version 1)

1 Overview

In the previous practical, Pacman had access to all the information in the environment, so it was *fully observable*. In this practical, we start to deal with the situation in which Pacman has to deal with a world that is *partially observable*.

2 Get a clean copy of Pacman

Some of your comments on the first practical made me realise that some of the same Pacman agents that I had given you were cheating. That is, they were accessing aspects of the world without going through the API. Since we're insisting that your code **only** accesses the environment throgh the API, that seemed wrong. So I rewrote the agents, and modified the API to make accessible information that was previously only accessible by other means. The changes are not backwards compatible, so:

1. Download:

   pacman.zip

   from KEATS — it is under the Practical for Week 2, and you should make sure it is labelled on KEATS as version 1.3 or higher.

2. Save that file to your account at KCL (or to your own computer).

3. Rename the folder you were using before (which probably has the same name (`pacman`) which the zip archive will create), to prevent any problems with overwriting code you wrote before.

4. Unzip the archive.

   This will create a folder `pacman`

5. Checkout the changes in `sampleAgents`, comparing the version you just downloaded with the version from last week.

   One change to note is the use that `RandomishAgent` makes of the constructor function to create some static storage in the agent, allowing you to maintain some state information from one call of `getAction` to another. `RandomishAgent` needs this to remember what it did on the previous move.

6. Copy the agents you wrote last week into the new `pacman` folder and get them to run.

   To do this you will have to adapt your code to work with version 1.1 of the API — that is the version in the new `pacman.zip`

3 Partial observability

Now it is time to start to grapple with partial observability. To do this you will use a(nother) new version of the API. This newest version, version 2, only returns information on those objects (ghosts, food, capsules) which are 5 steps or less away from Pacman. Thus Pacman has a much more limited view of its environment, often, for example, being unaware of where the ghosts are.

You should:

1. Download a copy of version 2 of `api.py` from KEATS. This can be found under the Practical for Week 3.
2. Add version 2 of `api.py` to your `pacman` folder.

   I would make a copy of `pacman` first so that I kept the materials from Practical 01 and Practical 02 separate.
3. Run `SensingAgent`

   ```
   python pacman.py --pacman SensingAgent
   ```
4. Notice that when it starts up,you get output like:

   ```
   Legal moves:  ['West', 'Stop', 'East']
   Pacman position:  (9, 1)
   Ghost positions:
   Distance to ghosts:
   Capsule locations:
   []
   Food locations:
   [(4, 1), (6, 1), (6, 2), (6, 3), (7, 1), (7, 3), (8, 1), (8, 3), (9, 3),
   (10, 1), (10, 3), (11, 1), (11, 3), (12, 1), (12, 3), (13, 1), (13, 2)]
   ```

   This is the exact same `SensingAgent` as last week, it just isn't getting data on the ghosts and capsules from the new API (because they are too far away). Also, it is showing much less food because it isn't getting data on food that is further away.
5. Also notice that `SensingAgent` is still getting all the information on wall positions.

4 Make Pacman move through the world

To win a game, Pacman needs to eat up all the food while not bumping into a ghost. When Pacman could "see" all the food, it could potentially win by just heading for the nearest bit of food at all times — eventually the nearest bit would be the last bit (provided Pacman stayed alive that long). Since it no longer has access to information about all the food, Pacman can no longer rely on just looking for the next food. Instead it has to do its best to move around the entire environment.

A simple way to do this is to force Pacman to move to the four corners of the environment, and you should develop a `CornerSeekingAgent` which does this. `CornerSeekingAgent` will need to do a few things:

1. Find out where the corners are.

   Version 2 of `api.py` includes the function `corners` which returns a list of the (x, y) coordinates of the extreme points of the world. From this you can extract the coordinates and use them to guide Pacman.

2. Remember where Pacman is going and where it has gone.

   To have a mission, Pacman needs some static storage. You can use the constructor of `CornerSeekingAgent` to create and initialise storage that is attached to the agent, and so persists between calls to `getAction()`. The latest version of `RandomishAgent` shows one way to do this.

Of course just having Pacman go to the corners won't ensure that they eat all the food — you'll need to set some other *waypoints* in addition to the corners to make that happen.

Getting the `CornerSeekingAgent` to a point where, without ghosts (see below), it can sweep up all the food is the minimum you should plan to get done this week. You will likely need more than the one hour of scheduled lab time to do this.

5 Ghostless Pacman

For the coursework, you will need to develop a Pacman that can cope with ghosts. However, while developing your code, it may be helpful to work without ghosts getting in the way. If you want to try your Pacman code in a ghostless world, you can try using:

```
python pacman.py --pacman CornerSeekingAgent --layout mediumClassicNoGhosts
```

This layout is a new addition, so this will only work with the code from version 1.3 or later of `pacman.zip`.

6 More

Other things to think about

1. Try combining the "food seeking" code of `HungryAgent` (form Practical 01) and/or `CornerSeekingAgent` with code to avoid ghosts (as for `SurvivalAgent` from Practical 01).

   You will have to think about how to prioritize seeking food and staying away from ghosts, and you will probably need some additional state information so that Pacman doesn't get distracted.

2. As we saw above, Pacman still has access to information about where all the walls are in the environment. In other words, it has all the information it needs to create a map of the world. Write a `MapBuildingAgent` which creates a datastructure that is a map, and uses this map to track every location that the agent has visited.

3. If you have only run Pacman using the default layout or `mediumClassicNoGhosts` (which has the same layout), make sure your code runs on different layouts.

7 Version list

- Version 1.0, October 8th 2017.