Lab class: minimax with α-β pruning

Steven Schockaert

Week 4

At the end of the exercise class, please post a message on the discussion board on learning central, indicating which parts of the assignment you have been able to solve and what problems you faced. Feedback in the subsequent lecture will address the issues that were highlighted in this way.

PacMan game

This lab is based on T. E. G. Grenager's CS121 Introduction to Artificial Intelligence course at Stanford University, Summer 2006, using a modified version of his Pac-Man game environment. See `http://www-nlp.stanford.edu/~grenager/cs121/materials.html`. Note that the version of Pac-Man used for this exercise and the coursework is notably different from the arcade version.

The Pac-Man game is played in a 2D maze build on a regular grid that contains dots and the player controls Pac-Man, a yellow disc with a mouth. The maze also contains four ghosts in different colours which move around according to their own strategy. When Pac-Man runs into a dot he eats it. When he runs into a ghost, he dies. When Pac-Man dies, he loses one of his three lives. Once he has eaten all dots in a maze, he moves on to the next level with a new maze. The aim of Pac-Man is to eat as many dots as possible without wasting time, while trying to avoid dying.

The objective is to design and implement the strategy taken by Pac-Man. For the lab class, a player based on a standard implementation of minimax is provided. Your goal is to improve it by implementing α-β pruning. This will enable Pac-Man to anticipate a larger number of moves, making it harder for the ghosts to catch him.

Getting Started

Download `lab3-source.zip` from learningcentral. In the archive, you will find the source files of the Pac-Man environment and a minimax based implementation for the ghosts and for Pac-Man itself. To compile the code command-line on linux or mac, you can use:

```
javac */*.java
```

To compile the code on windows, you can use:

```
javac ghosts\*.java pacman\*.java player\*.java util\*.java
```

To run the game, there are various command-line arguments which can be set to control the behaviour of the game. If you want to play the game yourself, you can use the following (for windows replace / by \):

```
java pacman/Game -display gui
```

If you want to execute the game with Pac-Man controlled by a computer player, use

```
java pacman/Game -display gui -pacman player.MinimaxPlayer
```

The class `player.MinimaxPlayer` is a minimax based implementation of Pac-Man, and can be replaced with your own implementation. Similarly, you can set the number and type of ghosts as follows:

```
java pacman/Game -display gui -ghosts ghosts.MinimaxGhost,ghosts.MinimaxGhost
```

In this example, two minimax based ghosts are used, but you can choose more ghosts or ghosts of a different type. A number of different ghost implementations are included in the framework, and can be found in the `ghosts` directory.

Assignment

Your assignment for this lab class is as follows:

1. Add α-β pruning to the minimax implementation of Pac-Man (`player/MinimaxPlayer`).
2. Add iterative deepening, and ensure that Pac-Man makes his move within 100ms.