

# How Pacool works

Abbas Mehrabian                      Arian Khosravi  
abbas.mehrabian@gmail.com        arian.khosravi@gmail.com

September 14, 2007

## 1 Introduction

Pacool 1.0 is a project that has been developed to play “Microsoft Revenge of Arcade Ms. Pac-Man Version 1.0” as an agent. The program has been written using JDK 1.6 in eclipse environment, and was submitted to compete in “IEEE Congress on Evolutionary Computation 2007” Competitions section. In this paper we study its algorithm.

The game situation is modeled as a  $31 \times 31$  table, known as the *gametable*. This table is stored in class `pacman.GameState`. Every entry in this table can contain a (part of a) wall, or a *gameobject*, i.e. MsPacman, ghost, edible, dot, energizer, or snack. The program essentially runs as an infinite loop, and the following procedures are carried out in each iteration:

1. Capture the game window from screen and detect the graphical items.
2. Try to recognize graphical items as gameobjects, and set the gametable entries.
3. Evaluate raw parameters in the updated gametable (e.g. nearest ghost to MsPacman, available dots).
4. Analyze the general situation using the raw parameters.
5. Find the best neighbor of MsPacman’s current position, according to a linear combination of the parameters, and press the corresponding key.

Items 1 and 2 have been discussed in section 2. Items 3, 4 and 5 have been discussed in section 3. Ideas for future work on the project have been discussed in section 4.

## 2 Image processing phase

Capturing a window from screen is a time-consuming task, but there was no way to completely avoid it. The program then iterates on the pixels of the taken image, and extracts connected pieces of same color from the image. Extraction is done using functions in `util.ImageProcessor`. Some of these pieces correspond to gameobjects. Walls are not extracted, since the structure of the mazes are known beforehand, we use this to reduce the image-processing cost.

A position of the gametable is assigned to each of the extracted gameobjects. Together with the position of each gameobject, we store its direction, which is used, for example, to predict movement of the ghosts.

This extraction, however, is not perfect. When two gameobjects intersect, or an object is completely covered by others, the resulting gametable may not be completely correct. Moreover, special situations should be handled in which there is a gameobject in a warp tunnel.

The visible part of the gametable is  $28 \times 31$  in fact. We have added 3 columns to model the warp tunnels, and the gametable is treated as a cylinder, i.e. the left and right edges are connected to eachother.

## 3 Decision making phase

The functions of class `pacman.Evaluator` calculate some raw parameters of the game each time, for instance:

**minimum time of ghost** The minimum time that a ghost can reach MsPacman's current position. When computing this time, we consider many factors such as the differences between speed of ghosts and MsPacman in warp tunnels.

When these parameters are computed, the functions of class `pacman.Agent` decide about the general situation, as follows:

**Danger** if ghosts are near MsPacman or they are trying to surround MsPacman, and MsPacman should focus on escaping or eating an energizer.

**Hunting** if there are edibles that MsPacman should go after and hunt.

**Safe** if MsPacman is in a safe position, and should eat more and more dots with no concern.

**Normal** if none of the above cases occur, and MsPacman should perform a combination of those actions.

When the situation is known, the coefficients of the parameters are set. For example, when in Danger situation, the coefficient of “minimum time of ghost” should be increased.

Once the situation is known, for every possible position on the gametable a score can be computed. The score is a function of the coefficients (that are related just to the current position of MsPacman) and the parameters of the specific position. Practically, the score is computed for the available adjacent positions of MsPacman, one with the highest score is selected and the corresponding key is pressed.

## 4 Future work

There are still some situations to cover, which rarely occur in the game. But due to intricacy of the game it is almost impossible to guarantee a flawless performance. Since the coefficients mentioned in the paper are tuned manually, a big step will be to use Learning Techniques to do the adjustment and improve the outcome.

**Acknowledgements.** The authors are indebted to Mr. Halvati for his support and ideas.