

To Play Ms. Pac-Man Automatically

Xiaocong Gan
sliant13@sina.com

Wenhui Liu
qdn163@163.com

Yun Bao
bybnu_1010@sina.com

2008-5-23

1 Introduction

We developed the program here to play the game “Ms. Pac-Man” automatically, which is a competition for WCCI 2008. We hope this article useful for those who are interesting in the competition.

The competition requires that the program captures the screen, processes the captured image, makes a decision of which way to go (up, down, left, right), then pressed the corresponding key. We’ll have some analyses about the game in section 2. Our program consists of 3 parts: image processing, searching algorithm, and decision making, discussed in section 3, 4, 5 respectively. Section 6 talks about some useful program-optimization techniques that we have employed.

2 Some analyses about the game

2.1 Random behaviors

We have developed a simple program that presses predetermined key based on the Ms. Pac-Man’s current position in the game. We run the program many times, and the ghosts behave differently. It’s quite possible that the game contains a quasi-random number generator which may use the current time and inputted keys to generate, and the ghosts’ movements are based on those random numbers.

2.2 Movement

The speed of the four ghosts’ and Ms. Pac-Man’s are almost the same in normal states.

During the game, a ghost is always moving and never stays still. We find that:

- If a ghost is on a dead end, it will go back.
- If it’s in the middle of a road, it will go straight forward and never go back.
- If it’s in a corner with 2 possible next directions to go, it will go ahead and never go back.
- If it’s on an intersection, it will choose a road that is closer to the Ms. Pac-Man’s current position. If such roads are more than one, choose one randomly. If such road doesn’t exist, choose a road randomly if it doesn’t go back.
- Every 20 seconds or so, all ghosts are forced go backward despite any above rules.

2.3 How to score high

The program which scores the highest will be considered Champion in the competition. Here’s some techniques to score high:

- Once you have eaten a power pill, all the 4 ghosts will become eatable and move slower for a short while. During this period, eat the first eatable will give you 200 points, and eat the second will give you 400 points, the third 800, the fourth 1600. Once this period has gone, or a new power pill has been eaten, all the counts will restart.
- The fruits give you many points. A fruit in the first stage give you only 100 points, but several stages later, a fruit may give you thousands of points. So a possible strategy to score high is to stay in a stage, wait for fruits and do not pass to next stage.

2.4 Cycles at the boundary

Ghosts and Ms. Pac-Man can pass through the right border into the left border. When Ms. Pac-Man passes through the border, it will disappear for a second. When a ghost passes through the border, or even when it’s near the border, it will move slower, and disappear for about 2 seconds.

3 Image Processing

3.1 Consider the paths in the maze as a network

Obviously this is a good way to represent the structure of the maze. The roads that pass through the border are connected vertices in the network.

3.2 Using the blocks as the vertices

The game window can be divided into 28×31 blocks. A block is a square of 8×8 pixels. Each block can be either a wall or passable. Each block is considered to be a vertex in the network if it's passable, and is connected to all its neighbor-blocks that are also passable.

3.3 Using the pixels as the vertices

The game window is of 224×248 pixels. One may think that a road in the maze can be better represented by a line of pixels than a line of blocks. We have developed a version in this way, and sadly find that, as fine as it is, it doesn't work nice. The reason is that, the game objects in the game window are recognized as connected pixels sets of the same color, and it's hard to say which pixels is the best position for a whole set of pixels.

However, this may be a good way to improve the program in the future.

3.4 Game objects recognition

Once the game window has been captured, we look for the connected pixels sets of the same color, and using the size and the color of a connected set to determine which objects this connected set represents.

Of course this is a vague way to recognize, especially when some game objects cover each other, or when they are passing through the border so they are invisible. A better way to recognize needs to analysis the outline of the image and remembers the object's last positions, and decide where the objects is with intelligence.

4 Search algorithm

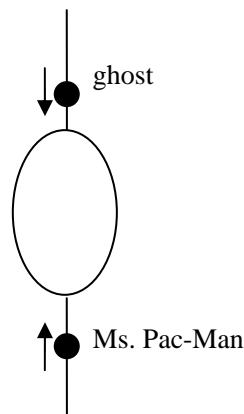
The paths in the maze can be represented by a path network (section 3.1). Giving the network, each state of a game can be represented by the positions and directions of the game objects, including the 4 ghosts, Ms. Pac-Man, eatable objects, pills and power pills.

A "Situation" contains the path network and a game state. Each situation will evolve — all game object moves and the game state changes — until Ms. Pac-Man is on an intersection. Then the situation will branch and generate multiple child-situations.

Thought Ms. Pac-Man can have more than one choice at any time, it would make the searching tree too large. So we assume that Ms. Pac-Man goes only forward and don't go back when the situation is evolving, until it's on an intersection.

In this way Ms. Pac-Man will know if it's been surrounded or not, when it'll be caught up, etc.

In fact, if we branch the searching tree at specific moment, we can get some intelligent behavior. For example, if we allow branch right after a ghost makes its choice, Ms. Pac-Man will know the following situation is safe.



Because we don't know exactly which direction a ghost may go when it's on an intersection (section 2.2), we may want to improve the search algorithm by using game-tree search. However, the version we have developed in this way doesn't perform well due to the too many branches in the search.

5 Decision making

In theory, search algorithm alone is enough to make decisions, and this is the case in some chess games; however, because of the great uncertainty of the game and the image reorganization, search algorithm here is not suitable to make decisions, though it's very useful to provide some basic parameters about the current situation.

Now we use a 3-depth search to decide if the situation is dangerous. If Ms. Pac-Man is very likely to be surrounded, or a ghost is very close, the situation is assumed to be dangerous.

If the situation is dangerous, a new 6-depth search will be performed and Ms. Pac-Man will prefer in order the direction that will eat a power pill, that has a greater probability to live, that can survive longer.

If the situation is safe, Ms. Pac-Man will go to eat the nearest pill, and leave the power pills untouched in case it's dangerous.

In a word, we are using hand-coding here to make a decision, which we are not satisfied. We wish to use some learning techniques to make decisions base on the basic parameters in the future.

6 Program optimization

6.1 When a new round is started

A new round of the game is started when a coin has been inserted and start button has been pushed, or after the last pill has been eaten, or after Ms. Pac-Man has been caught. The yellow text "READY!" will be showing in the game window for about one second. It's a good idea to analysis the maze structure, get the path network (section 3.2) and do many other calculations in this period.

6.2 Capture the game window

The best way is to use a native method to capture the current game window into a pre-allocated array. If the java class "Robot" is used, it'll create a new "BufferedImage" object every time. And get the array directly from "`((DataBufferInt) bufferedImage.getRaster().getDataBuffer()).getData()`" is better than "`BufferedImage.getRGB()`"

6.3 When recognize game objects from a captured image

Remove the known maze walls before reorganization can save a lot of time.

Here you may want to use a quick integer stack of your own instead of an "ArrayList" or something like that.

6.4 Thread priority

Set your main thread's priority to "`Thread.MIN_PRIORITY + 1`" will make your main thread get most of the CPU time while do not delay the game window. Note that "`Thread.MIN_PRIORITY`" is used by GC thread, so it's not recommended.

6.5 Using an array or a "Map" to speed a function up

For those functions that the input variable is in a relatively small domain, you may want to use a "Map" or even an array to speed it up. All the possible input and corresponding output is saved in that "Map" or array beforehand.

6.6 Prevent GC

Programs like this requires a quick responding time. GC may be a big problem for them because a full GC pause the running program for several seconds.

The first step to prevent GC is using "`java -Xmx512m -Xms512m`" or something like that to start you program.

The second step is to call "`System.gc()`" explicitly in your program in an appropriate time to perform a full GC. We did it when a new round is started (section 6.1).

The third step is to check your program, find out where objects are created too many. A buffered pool can be used there.