TCSS 435 - Artificial Intelligence
Assignment 1 - Pac Man

Guidelines

This assignment consists of programming work. Solutions should be a complete working JavaScript program including your original work or **cited contributions** from other sources. These files should be compressed in a .zip file for submission through the Canvas link.

This assignment is to be completed on your own or in a group of two or three. If you choose to work in a group this must be clear in your submission. Please see the course syllabus or the course instructor for clarification on what is acceptable and unacceptable academic behavior regarding collaboration outside of a group of two or three.

Assignment

The game of Pac Man is a classic video game created by Namco in 1980 (we will use a modified web clone created by Lucio Panepinto and modified by Chris Marriott). You will program an artificial intelligence to play as Pac Man to live as long as possible, collect as many points as possible and to reach the furthest level possible.

You are provided with a Pac Man simulator. The provided agent selects its move randomly when it reaches an intersection. You must replace its selectMove method with your own strategy. I recommend considering simple strategies first. A good first attempt would be to mimic the ghost behaviors (available in ghosts.js). Do not modify any other code except to create new variables as needed by your strategy. You should do this all in the agent.js file.

Your agent will be evaluated by allowing it to play Pac Man ten times. Grades will be assigned based on the performance of your agent by considering the highest score achieved by your agent and the highest level reached by your agent.

Specifications

The Pac Man implementation uses the following map from directions to numbers:

1. Right
2. Down
3. Left
4. Up

In your selectMove method you will call movePacman(direction) **once** where direction is a var containing the number corresponding to the direction you wish Pac Man to move.

To help with your move selection I have created a GAMEBOARD data structure that represents the Pac Man world. The GAMEBOARD is a 2D array of cell objects. (Note: In the Pac Man world not every space can be occupied and in the GAMEBOARD array these locations are represented by null. You are free to change this if you like.) Each cell in the GAMEBOARD has 8 Boolean properties:

- bubble - does this cell have a bubble
- superBubble - does this cell have a super bubble
- eaten - is the bubble or super bubble eaten
- pacman - does the cell contain pac man
- inky - does the cell contain inky
- pinky - does the cell contain pinky
- blinky - does the cell contain blinky
- clyde - does the cell contain clyde

You are welcome to use this GAMEBOARD to select your moves and you may modify it however you like to store more information. Below are variables from the pac man game engine that can be inspected but not changed. I inspect many of these variables to build the GAMEBOARD.

You can check Pac Man by inspecting one of these variables:

- PACMAN_DIRECTION - the direction that Pac Man is facing/moving by number
- PACMAN_POSITION_X - Pac Man's x-coord on the Canvas
- PACMAN_POSITION_Y - Pac Man's y-coord on the Canvas
- PACMAN_MOVING - boolean is true if Pac Man is moving
- PACMAN_DEAD - boolean is true if Pac Man is dead
- PACMAN_SUPER - boolean is true if Pac Man is super

Note that the ghosts in Pac Man have different names:

- Blinky - Red
- Pinky - Pink
- Inky - Blue
- Clyde - Orange

You can check the ghosts by inspecting one of these variables (using the ghost's name):

- GHOST_BLINKY_POSITION_X - Blinky's x-coord on the Canvas
- GHOST_BLINKY_POSITION_Y - Blinky's y-coord on the Canvas
- GHOST_BLINKY_DIRECTION - the direction that Blinky is facing/moving by number
- GHOST_BLINKY_MOVING - boolean is true if Blinky is moving
- GHOST_BLINKY_BODY_STATE - the state of Blinky's body
- GHOST_BLINKY_STATE - Blinky's state (numerical)
- GHOST_BLINKY_AFFRAID_STATE - boolean is true if Blinky is afraid
- GHOST_BLINKY_TUNNEL - boolean is true if Blinky is in the tunnel

You can check the bubbles by inspecting one of the elements of this array:

- BUBBLES_ARRAY[i] - element i is the i^{th} bubble
    - Every bubble is a string in the form “x,y;row;col;type;eaten” for example “26,26;1;1;b;0”
    - x - x-coord of the bubble on the canvas
    - y - y-coord of the bubble on the canvas
    - row - the row of the bubble in the maze
    - col - the col of the bubble in the maze
    - type - will be “b” for normal bubbles and “s” for super bubbles
    - eaten - will be 0 if the bubble is not eaten and 1 if it is

Other helpful functions:

- canMovePacman(direction) - returns true or false based on whether Pac Man can move that direction
- oneDirection() - returns a random direction (number between 1 and 4)

Submission

The following files are provided for you:

- Pac Man code base - html, css, and js files for the Pac Man game
- agent.js - includes a simple example agent

You will submit only:

- agent.js - your agent code