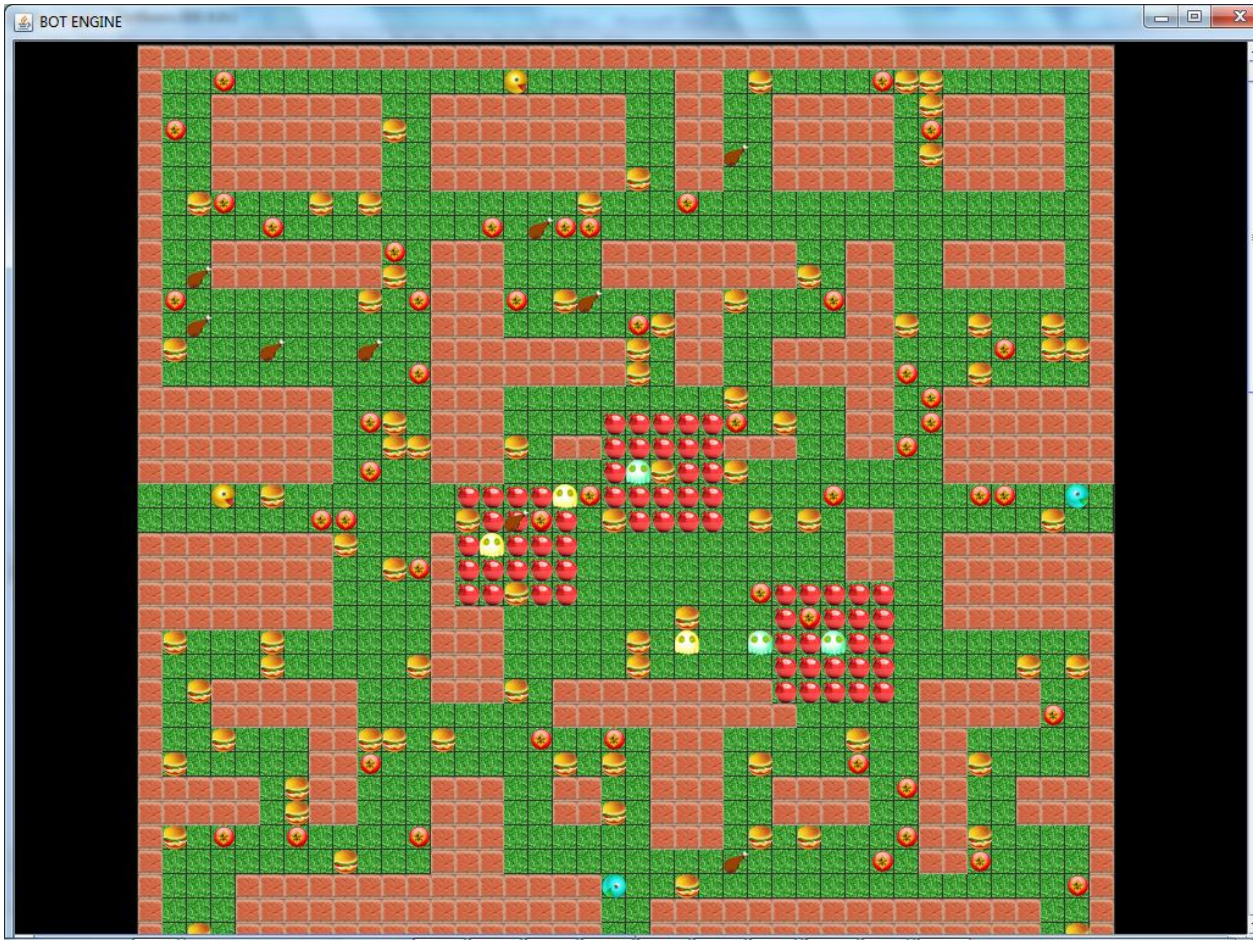
AI Challenge

Pac-Ghost Challenge Game Description

The AI Challenge game, Pac-Ghost Challenge, is played in an environment that looks something like the following figure. A yellow player and a green player compete on an obstacle and food covered 40x40 filed. Each player controls a group of Pac and Ghost, who can pick take the food freeze each other. Players earn points by taking food and placing bomb strategically.



1 The Playing Field:

Pac-Ghost Challenge is played on a field of 40x40 spaces. Spaces are indexed by X and Y**, with X going left to right from 0 to 39 and Y going top to bottom 0 to 39.**

1.1 Obstacle:

A few spaces contain obstacles. If a space contains obstacle it can't contain any food or any Pac or Ghost. Obstacle configuration will vary from game to game. But it will be less than 40% of total grid area. And there will be no obstacle in the initial position of any Pac or Ghost. To keep things fair obstacle will maintain symmetry with respect to yellow and green player. Obstacles are not permanent a bomb field can vanish obstacles.

1.2 Free spaces:

Spaces without obstacle may contain different types of food, bomb field, power field, Ghost, Pac

1.2.1 Food:

Food can be of three kinds Positive-Food, Negative-Food, Power-Food. These foods will be randomly distributed on free spaces. Food can only be taken by a Pac. Positive-Food will increase a player's point by **5**. Negative-Food will decrease player's point by **2**. And Power-Food will increase point by **10** and add an additional power field surrounding the Pac. In above figure

Positive Food is 

Negative Food is 

Power Food is 

1.2.2 Bomb Field:

Bomb Field can only be created by a Ghost. A bomb field has duration of **8** time slots. A ghost can set a bomb after **20** time slot of previous bomb set by it. Bomb field is created as a **5x5** field surrounding the Ghost that set the bomb. A Pac of any Player will freeze up to bomb duration if it steps in a bomb field. A bomb will smash obstacles falling within its field.

1.2.3 Power Field:

A power field is a **5x5** field that will shield a Pac if it takes a Power-Food. And this power field has duration of **8** time slots. It will be automatically created soon after taking the Power-Food and it will move along with the creator Pac piece. A ghost of any player will freeze if it falls in a Power field.

1.2.4 Pac

A player will have to control **2** Pac pieces. Their aim is to gain food cleverly to maximize the total point.

1.2.5 Ghost

A player will have to control **3** Ghost pieces. Their aim is to freeze Pac pieces by setting bomb cleverly to hinder mobility and minimizing opponent's scope to gain points.

Following table gives the symbol used to encode each when the game state is communicated to the player.

Space Contents	Encoding	Data Structure	Description
Obstacle	0,1	Obstacle Array	1 if there is an obstacle
Food	0,1,2,3	Food Array	0 if no food 1 if Positive-Food 2 if Negative-Food 3 if Power-Food
Bomb Field	0-8	Bomb Field Array	0 if no bomb Else the remaining duration of the bomb field of latest set bomb over that cell (maximum of all set bomb duration on this cell)
Power Field	0-8	Power Field Array	0 if no bomb field Else the remaining duration of the Power field of latest set Power field over that cell (maximum of all set Power field duration on this cell)

2. Playing the Game:

The game proceeds through a series of **500** turns. Each player controls a team of **2** Pac pieces and **3** Ghost pieces. At the start of a turn, the player reads a description of state of the field from standard input and then prints a desired action for each piece to standard output.

2.1 Pieces Attributes:

A Pac or Ghost may be directed to move Left, Right, Up, Down. And a ghost may be directed to set a bomb field on the next move if set bomb time duration expires (after **20** time slot of previous bomb set by very this ghost).**Wrap movement is allowed only for the Pac pieces.** Wrapping move for a ghost is an invalid one.

2.2 Pieces Domain and Visibility:

Any Piece a Pac or a Ghost has a visibility square of **7x7** dimensions having itself in the center. It can only get information of food, obstacle, bomb, power, Pac pieces, and Ghost pieces that belong to this **7x7** square space. So a Player gets information of 5 visibility squares of **7x7** sizes one for each of his/her pieces.

2.3 Earning Points:

Only Pac pieces can earn point. If a Pac goes to a cell having Food .Point corresponding to that food is added (or subtracted in case of Negative-Food) to score. Point distribution is cited above.

2.4 Winning the Game

At the end of the game, the winner is the player with the highest score. To break Tie rematch will be used.

3 Directing the Team:

3.1 Pac Direction: Player needs to decide the move for each of his/her Pac pieces. There position will be given after each move along with the visibility info to go to next position Player decision could be

Left: write 'L' in standard output

Right: write 'R' in standard output

Up: write 'U' in standard output

Down: write 'D' in standard output

Idle: write 'F' in standard output

*full I/O specification is given in latter section

3.2 Ghost Direction:

Same as Pac direction .But a Ghost may set bomb if it wishes and if it's now valid to set a bomb field. To set a bomb on the next move player need to write 'Y' in the standard output for the corresponding ghost.

4 Simulation Operations:

After receiving a player's actions for each Piece, the game checks the actions to make sure they are legal. If an action can't be performed, if it can't be completed or if it conflicts with an action from another Piece, the engine just give a random valid decision for that action.

All actions take only one turn, but they don't all occur at the same time. In a turn, all move actions are performed followed by setting bomb actions. Move action sequence is Pac1, Pac-2, Ghost-1, Ghost-2, and Ghost-3. That is a cell may be free before this turn but can be occupied by Pac-2 so that cell can't be obtained by Ghost-1.If two pieces try to go to same cell first one in the sequence will be given the chance and random move will be given to the next one. After the move operation bomb set operation are performed. And first player will be given chance before the second player. During tournament we will decide randomly who will be the first one.

5 Player/Game Interface

Player implementations are external to the game itself. A player is a separate executable that communicates with the game via standard input and standard output. The player is executed when the game starts up and continues running until the game is finished. At the start of each turn, the game engine sends the player a description of the game state. The player reads this description from standard input chooses a move (an action for each piece) and sends it back by writing it to standard output.

5.1 Player Input Format

The game state is sent to the player in a plain-text format. The first line contains a turn number, t. Under normal operation, the value of t will start with zero and increment by one with each game state snapshot received by the player. The next line contains a report of the current game score, first the score for the Player himself/herself then the score for the opponent player.

The score report is followed by position of each piece of the player. Five pair of integers denotes the position of respective piece in sequence pac1, pac2, ghost1, ghost2, ghost3. Each position is space separated two integers and each position constitutes a single line. That is in the following format:

x0 y0

x1 y1

.. ..

x5 y5

Than another Five pair of integers are given denoting the position of the pieces of opponent in the same format and same sequence.

Then for Pac1 visibility range,

A 7x7 matrix containing food configuration surrounding the Pac1.This matrix is given in row column order. Each element indicates a food type if there is any.0 if no food, 1 for Positive Food, 2 for Negative food, and 3 for Power food. In case a cell is out of our board it will be wrapped.

1 0 1 1 1 0 0

1 0 1 1 2 3 0

1 0 1 0 3 0 1

. . . . . .

. . . . . .

2 0 3 1 2 1 2

1 0 0 1 3 2 1

A 7x7 matrix containing bomb configuration surrounding the Pac1.This matrix is given in row column order. Each element indicates bomb field duration if there is any.0 if no bomb, 1-5 for bomb duration of that cell. In case a cell is out of our board it will be wrapped.

0 0 0 0 0 0 0

0 0 2 2 2 2 2

0 0 2 2 2 2 2

. . . . . .

. . . . . .

0 0 0 0 0 0 0

0 0 3 3 3 3 4

A 7x7 matrix containing Power field configuration surrounding the Pac1.This matrix is given in row column order. Each element indicates Power field duration if there is any.0 if the cell is not part of any power field, 1-5 for Power field duration of that cell. In case a cell is out of our board it will be wrapped.

0 0 0 0 0 0 0

0 0 2 2 2 2 2

0 0 2 2 2 2 2

. . . . . .

. . . . . .

0 0 0 0 0 0 0

0 0 3 3 3 3 4

A 7x7matrix containing obstacle configuration surrounding the Pac1.This matrix is given in row column order. Each element indicates obstacle if there is any.1 if the cell has obstacle, else 0. In case a cell is out of our board it will be wrapped.

0 0 0 0 0 1 1

0 0 1 1 1 1 0

0 0 0 1 1 1 0

. . . . . .

. . . . . .

0 0 0 0 0 0 0

0 0 1 0 1 0 1

That is 4 visibility information is given for the Pac1.

In the same format and same sequence visibility information for all other pieces will be given. That is 4 matrixes for every pieces will be given.

(** A sample code corresponding to input will be given)

5.2 Player Output Format

At each turn, the player is to print a desired action for every piece to standard output along with the bombing decision for the ghost pieces. Player output will be just one line in format

L R D F L Y N Y

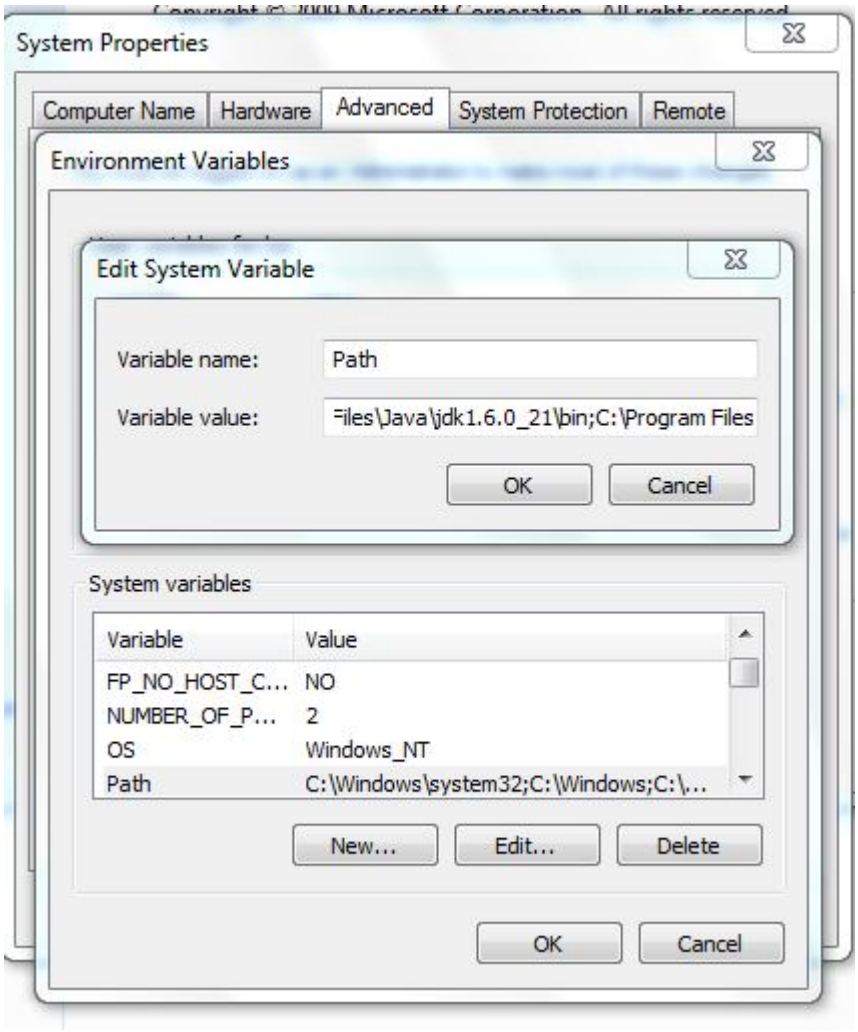
First 5 integers are moving direction for the pieces in sequence pac1, pac2, ghost1, ghost2, ghost3 .Each character corresponds to a direction (see 3.1).Next three characters is bombing decision for the Ghost pieces.

5.3 Real-Time Response Requirement

After a snapshot of the game state is sent to the player, player needs to responds within .5sec.If the player fails to respond or if the response is received too late, the game will assign an idle move to all the player's pieces. The game expects to receive a move for each state that is sent to a player, but the game engine does not maintain a queue of game states on behalf of each player. If a player falls behind in passing game states and responding with a desired move, the engine will discard, rather than queue, subsequent states for the player. A player that is too slow to respond will receive a sampling of the states, and the value of the turn number will indicate that one or more states have been dropped.

6 Usage Instructions

Update your system path variable. Add reference of g++ and jdk.



The game engine is implemented in Java and supports several command-line options to let the user run games with different players and with different output behavior. Since players are implemented as separate executables, the developer is free to use any development environment to code the player behavior. In principle, any programming language may be used to implement a player, but the submission site will only accept players written in C++ and Java.

When you download the game distribution binary, you will receive a zip sample containing a small directory structure for the game. You can unpack this file to create an execution environment for the game. The jar file, BOT.jar, contains the executable code for the game, and other subdirectories contain sample map files and example players implemented in C++ and Java.

****Anyone willing to code at python please mail us we can give you support.**

6.1 Game Visualization

The game distribution binary contains a basic 2D visualization. As described above, this visualization component is intended to help the developer see what's going on in the game, what regions of the field are visible to the player and what actions each child is performing. During the final tournament, we will use a 3D visualization that we hope will provide a more interesting way to watch matches.

6.2 Making Something Happen

If you just want to see a game running, you can unpack the game distribution binary and double click on the BOT.jar file. This will run the game using the java example provided in player0 and player1 folder with total no of move 100;

6.3 Running an Arbitrary Player

Java –jar BOT.jar player0 player1 cpp java 200

This is the command to run arbitrary player code with arbitrary no of moves. First parameter is first player code directory, second is second player code directory. Third and fourth denote language of first and second player code respectively. And the last parameter is optional denoting the no of moves the game will run (100 default).

6.3.1 Debug:

Java –jar BOT.jar player0 player1 cpp java 200 0

Here the last parameter is given 0. This command will run the engine in normal mode if it's set to one that is **Java –jar BOT.jar player0 player1 cpp java 200 1** this command will run the engine in debug mode. In this mode you can simulate each step pressing enter in the black (command) window.

Submission Rules:

Submit Your Code at ai.buet.csefest2013@gmail.com with the subject as AI_[Your Email]

For further query:

https://www.facebook.com/events/142492592584496