**Presentation made by:**

**Aravind Sagar 1100104**

**V Priyan 1100136**

PAC-MINION

Minor Project on Artificial Intelligence, 2014

An interactive 3D maze game.

Table of Contents

[Problem Specification 2](#_Toc385841692)

[Problem Description 3](#_Toc385841693)

[For Ghosts 3](#_Toc385841694)

[For PAC 3](#_Toc385841695)

[Algorithms Used 4](#_Toc385841696)

[For Ghost 4](#_Toc385841697)

[For PAC 4](#_Toc385841698)

[IDA\* based PAC Simulation 4](#_Toc385841699)

[MiniMax based PAC Simulation 5](#_Toc385841700)

[MiniMax based PAC Simulation with Tree Pruning 5](#_Toc385841701)

[Test Cases 6](#_Toc385841702)

[Testing Ghost Behaviour 6](#_Toc385841703)

[Testing PAC Behaviour 6](#_Toc385841704)

[IDA\* based PAC Simulation 6](#_Toc385841705)

[MiniMax based PAC Simulation 6](#_Toc385841706)

[MiniMax based PAC Simulation with Tree Pruning 7](#_Toc385841707)

[Comparison between different PAC AI Algorithms 7](#_Toc385841708)

# Problem Specification

**“Intelligence is what you use when you don’t know what to do. “**

**-Jean Piaget**

To create a 3D PAC-Minion game, where ‘PAC’, the main character of the game will be chased by ‘Ghosts’, the smart villains of the game.

The problem consists of two parts:

1. Giving ‘Ghosts’ behavior and intelligence to efficiently catch ‘PAC’.
2. Giving ‘PAC’ intelligence to smartly escape the ghosts during the simulation mode. (PAC can be controlled manually as well)

# Problem Description

## For Ghosts

* **Goal of the Ghosts:** Corner the PAC to eat it in NORMAL mode and wander aimlessly during PANIC mode.
* **Maze:** Game takes place in a maze, which is divided into 28x31 grids. Each ghost knows the present tile of the PAC.
* **Ghost’s Targets:**
  + **Blinky:** ‘To run down or pursue’. The target of Blinky is the PAC’s current tile. Most aggressive ghost of all four.
  + **Inky:** ‘Fickle minded, moody’. Uses most complex targeting scheme. A vector from Blinky’s position to two tiles in front of PAC is taken and it is doubled to get the target.
  + **Pinky:** ‘To perform an Ambush’. The target of Pinky is four tiles away from PAC, in the current direction of the PAC.
  + **Clyde:** ‘Pretending Ignorance’. The target of Clyde is PAC’s current tile, if it is eight units or more away from PAC; otherwise its target is one of the corners of the maze.
* **Ghost Restrictions:** 
  + It cannot reverse its direction.
  + Cannot take turns as fast as the PAC.

## For PAC

* **Goal of the PAC:** To eat all the food without getting caught by the ghosts. When ghosts are in panic mode, PAC can eat them to score more points.

# Algorithms Used

## For Ghost

* To find the path to the Target tile, the ghosts use **Iterative Deepening A\* algorithm** (IDA\*).
* The next direction is selected according to path returned by the IDA\*.
* The previous tile of the ghost is blocked before running IDA\* so that the ghost cannot reverse its direction.
* For some ghost, the present PAC tile is blocked so that the ghost will take a different path to reach in front of the PAC to smartly corner it.
* **Heuristic Used for IDA\*:** Manhattan Distance.
  + Manhattan distance is both consistent and admissible.
  + The maze is fairly simple, so Manhattan distance gives satisfactory results.
* The cost of moving from one tile to another is 1.
* The target tile might be out of the maze or in an invalid position, then the algorithm finds the nearest reachable tile from the given target tile by constructing a reachability matrix.

## For PAC

We have three different AI modes for PAC simulation, depending upon the algorithm used.

### IDA\* based PAC Simulation

* **Target:** IDA\* requires a target tile. The target tile is set as follows.
  + If no ghost is nearby, nearest Food tile is the target.
  + When at least one ghost is nearby,
    - If the ghosts are in PANIC mode, set the target to the nearest ghost.
    - Else set a random food tile as the target.
* **Heuristic:** Manhattan Distance.
* The cost of moving from one tile to adjacent tile.
  + If tile empty, 14.
  + If tile is a ghost and ghost are not in PANIC mode, cost is large constant divided by distance from the PAC.
  + If the tile is a ghost, and the ghosts are in PANIC mode, cost is 0.
  + If normal food tile, cost is 10.
  + If power pellet tile, cost is sum of ghost distances by 10.
  + These values where chosen after experimentation and testing.

### MiniMax based PAC Simulation

* Explores every possible move of the PAC and the reaction of the ghost to its motion, to decide the best possible move.
* This algorithm generates a state space tree considering both PAC and ghost, up to a given maximum depth, and evaluates the score for each state to find the best possible move (Less the score, better for the PAC).
* The score **(Heuristics)** for each state depends on the following variables:
  + Distance from the PAC to the other ghosts.
  + Number of food particles.
  + Distance of the nearest food particle to the PAC.
  + Distance of the nearest power pellet.
  + Ghost mode.
* The weightage of the each variable given above was chosen after extensive experimentation.

### MiniMax based PAC Simulation with Tree Pruning

* It is based on the MiniMax algorithm mentioned above.
* The above mentioned MiniMax is computationally expensive as it has exponential boom while generating the state space tree.
* The algorithm prunes the state space tree when it knows that the subsequent moves doesn’t improve the score for the PAC (i.e. score will not reduce). This reduces the CPU and memory requirements considerably.
* The restrictions to the ghosts’ movement are also considered to further prune the tree.
* This algorithm also gives the PAC, some knowledge about the behavior of the ghost to help it make smarter moves to escape from delicate situations.
* The PAC will use the Manhattan distance to find the next direction of the ghost, where the target tile of the ghost is determined by the knowledge provided to the PAC about the ghost.

# Test Cases

## Testing Ghost Behaviour

* For testing the ghost behavior, we initially made the PAC static and tested the movement of the each ghost separately depending on their respective behavior.
* The maze passed to the each ghost was modified in such a way that they behave differently according to their intended behavior (like avoiding the path directly leading to the PAC, so that it attacks the PAC from the front).
* Then the PAC was moved manually by us, to check how well the ghosts cover the PAC, and their targets were changed accordingly.
* The game was played in manual mode many times and the score was noted to make sure the AI makes the game neither too easy nor too difficult.

## Testing PAC Behaviour

PAC performance can be measured using the ‘score’ and the number of times it got killed by the ghosts (deaths). Score is incremented by 10 for every food pellet eaten and by 50 for ever ghost eaten in PANIC mode. It is decremented by 50 whenever it’s eaten by a ghost.

### IDA\* based PAC Simulation (Press ‘1’ in the game to run)

* Problems:
  + The path finding is done using a static maze, assuming the ghosts in the maze remain constant. Thus the PAC could get cornered easily by the ghosts.
  + Neglecting the Food during PANIC mode. Could otherwise finish the dangerous parts of the maze.
* The IDA\* uses, non-trivial heuristic to decide the target tile for the PAC which has been locally optimized for game as described in algorithm section.
* Many trial runs were made with different combinations of heuristic to decide the target tile for the PAC.
* The cost of a tile in the maze changes with the content at that position like ghost, food pellet, power pellet or empty tile. The costs for these tiles were chosen after many trials until we reached acceptable performance.

### 

### MiniMax based PAC Simulation (Press ‘2’ in the game to run)

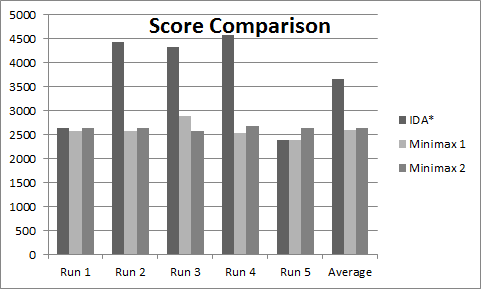
* Problems:
  + Due to very high computational requirements, the depth was limited because of which the simulation became highly sensitive to the heuristic chosen.
  + The algorithm got stuck in local minima when the number of food particles is less.
* The heuristic to score each game state was chosen after multiple trials to give an optimal performance.
* The weightage given to each variable was varied continuously and checked multiple times consecutively, as it the change in the simulation was highly sensitive on the weightage.

### MiniMax based PAC Simulation with Tree Pruning (Press ‘3’ in the game to run)

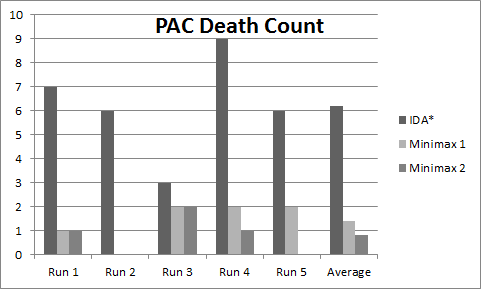
* The tree pruning used doesn’t guarantee optimal solution. Hence the algorithm was tested and the tree pruning strategies were constantly modified until it gave better results than the previous MiniMax.
* The test cases which were used in the previous MiniMax were used here.

## 

## Comparison between different PAC AI Algorithms



|  |  |  |  |
| --- | --- | --- | --- |
|  | Score | | |
|  | IDA\* | MiniMax 1 | MiniMax 2 |
| Run 1 | 2630 | 2580 | 2630 |
| Run 2 | 4430 | 2580 | 2630 |
| Run 3 | 4330 | 2880 | 2580 |
| Run 4 | 4580 | 2530 | 2680 |
| Run 5 | 2380 | 2380 | 2630 |
| Average | 3670 | 2590 | 2630 |



|  |  |  |  |
| --- | --- | --- | --- |
|  | Deaths | | |
|  | IDA\* | MiniMax 1 | MiniMax 2 |
| Run 1 | 7 | 1 | 1 |
| Run 2 | 6 | 0 | 0 |
| Run 3 | 3 | 2 | 2 |
| Run 4 | 9 | 2 | 1 |
| Run 5 | 6 | 2 | 0 |
| Average | 6.2 | 1.4 | 0.8 |

### Observations (For a summary, refer to the table in the next page)

**Score**: From the simulation, we can observe that IDA\* PAC algorithm scores high, but this is due to the fact that in PANIC mode, when a ghost is eaten, it is regenerated instantly at a fixed point, and if the PAC is near the regeneration point, it continuously eats the same ghost again and again in quick succession until the PANIC mode time expires. This happens because the target tile is set as the Ghost position in PANIC mode. This does not happen in the other two modes because the PAC gives weightage to eating up food as well, even in PANIC mode, and will not pursue the Ghosts much.

The scores of MiniMax (1 and 2) simulations were closer to what a human player would usually obtain.

**Escaping Ghosts (number of deaths)**: We can also see that IDA\* algorithm does not escape the ghosts very well in NORMAL mode, and the PAC tends to get eaten up frequently. Thus, if we kept ‘Lives’ (i.e., a limit on the number of times PAC can get eaten by the ghosts), the score of IDA\* algorithm will come down considerably. The number of deaths in both of the MiniMax modes are much lesser, and sometimes they even complete the maze without dying even once.

**Time taken**: Another comparison that can be made is the time taken to complete the maze. IDA\* tends to wander off and circle around the maze, since sometimes, random food tile is set as the target. MiniMax 1 (without pruning) tends to get stuck in local minima once in a while especially when the ghosts are far away, but still, the time taken is usually lesser than that of IDA\*. This happens because the depth is very much limited in this case. MiniMax 2 (with pruning) deals with this problem fairly well, and rarely gets stuck in minima, and takes the least time among the 3 algorithms to finish the maze.

|  |  |  |  |
| --- | --- | --- | --- |
| Comparison Table | | | |
|  | IDA\* | Minimax 1 | Minimax 2 |
| Score | Inconsistent | Average | Average |
| No. of Deaths | High | Low | Low |
| Time taken | High | Medium | Low |

•••