**Introduction**

Pac-Man is a maze arcade game developed and released by Namco in 1980. The player controls [the titular character](https://en.wikipedia.org/wiki/Pac-Man_(character)) through an enclosed maze; the objective of the game is to eat all of the dots placed in the maze while avoiding four colored ghosts — Blinky (red), Pinky (pink), Inky (cyan), and Clyde (orange) that pursue him. When all of the dots are eaten, the player advances to the next level. If Pac-Man makes contact with a ghost, he will die and the game ends. Each of the four ghosts have their own unique, distinct [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence) (A.I.), or "personalities"; Blinky gives direct chase to Pac-Man, Pinky and Inky try to position themselves in front of Pac-Man, usually by cornering him, and Clyde will switch between chasing Pac-Man and fleeing from him. Pac-Man was a widespread critical and commercial success, and has an enduring commercial and cultural legacy. The game is important and influential, and it is commonly listed as one of the [greatest video games of all time](https://en.wikipedia.org/wiki/List_of_video_games_considered_the_best). The success of the game led to several [sequels](https://en.wikipedia.org/wiki/List_of_Pac-Man_video_games), merchandise, and two television series, as well as a hit single by [Buckner and Garcia](https://en.wikipedia.org/wiki/Buckner_%26_Garcia). The Pac-Man video game franchise remains one of the highest-grossing and best-selling game series of all time, generating more than $14 billion in revenue (as of 2016) and $43 million in sales combined.

**Objectives**

* To design a game that is similar to the original Pac-Man
* To create AI or personalities of the four ghosts in the game

**Agent Description**

In Pac-Man, the four ghosts namely Blinky, Pinky, Inky and Clyde are the agents. Each of these agents works with the same algorithm that is finding the shortest path to target but the target for the agents are different.

|  |  |
| --- | --- |
| **Performance** | Selected path length (Best performance is selecting the shortest path) |
| **Environment** | Walls, Nearest nodes, Pac-Man |
| **Actuator** | Moving toward the nearest node |
| **Sensor** | Position of Pac-man, Agent Position, Nearest nodes position, Blinky position (For Inky) |

Each agent has three modes in which they act. The modes are: Chase mode, Scatter mode and Frighten mode. In each of the modes, the percept, goal and environment of the agents change.

**For Chase mode**

In this mode, the agents try to ambush Pac-Man in their own way.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Agent** | **Percept** | **Action** | **Goal** | **Environment** |
| Blinky | Pac-man Position, Nearest nodes | Select most nearest node | Shortest path to Pac-Man | Walls, Nearest nodes, Pac-Man |
| Pinky | Pac-man Position, Nearest nodes | Select most nearest node | Shortest path to 5 tiles ahead of Pac-Man’s moving direction | Walls, Nearest nodes, Pac-Man |
| Inky | Pac-man Position, Blinky Position, Nearest nodes | Select most nearest node | Shortest path to mirrored position Blinky with reference to Pac-man | Walls, Nearest nodes, Pac-Man, Blinky |
| Clyde | Pac-man Position, Nearest nodes | Select most nearest node | Shortest path to Pac-Man when Pac-Man in more than 8 tiles away, Shortest path to Bottom-Left corner otherwise | Walls, Nearest nodes, Pac-Man |

**For Scatter Mode**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Agent** | **Percept** | **Action** | **Goal** | **Environment** |
| Blinky | Nearest Nodes | Select most nearest node | Shortest path to Top-Right corner | Walls, Nearest nodes |
| Pinky | Nearest Nodes | Select most nearest node | Shortest path to Top-Left corner | Walls, Nearest nodes |
| Inky | Nearest Nodes | Select most nearest node | Shortest path to Bottom-Right corner | Walls, Nearest nodes |
| Clyde | Nearest Nodes | Select most nearest node | Shortest path to Bottom-Left corner | Walls, Nearest nodes |

**For Frighten Mode**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Agent** | **Percept** | **Action** | **Goal** | **Environment** |
| Blinky, Pinky, Inky, Clyde | Nearest Nodes | Select a random node from nearest nodes | Move randomly | Walls, Nearest nodes |

**Agent Environment**

The agent environment consists of walls which the agents should avoid, Pac-man whom the agents target, nearest nodes from which the agents decide where to move. The environment properties are:

|  |  |  |
| --- | --- | --- |
| **Observable** | Fully | All the possible states are visible to the agent |
| **Deterministic** | Deterministic | Next state is determinable from the current state |
| **Episodic** | Sequential | The output to be determined depends on previous output |

|  |  |  |
| --- | --- | --- |
| **Static** | Dynamic | The state of the agent depends on position of Pac-man which is constantly changing |
| **Discrete** | Discrete | The agents can move through only a set of defined nodes |
| **Agents** | Multi-Agents | For each agent, the position of Pac-man must also be considered |

**Problem Specification**

The goal of all the agents is to get to the position of Pac-man. The state of each agent is defined by the tile position they are in which is represented by two values i.e. (x, y). The x value ranges from 0 to 29 and y value ranges from 0 to 26. The agents start from the location (11, 13). The agents can move from current tile to adjacent tiles which are not walls. The valid operators are (-1, 0) to go left, (1, 0) to go right, (0, -1) to go up and (0, 1) to go down.

State = (x, y)

x = 0 to 29

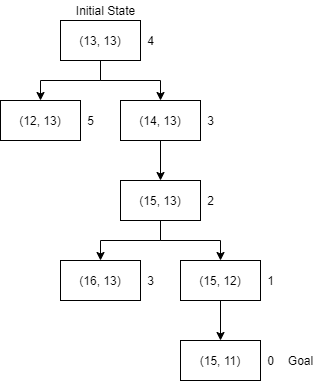
y = 0 to 26

Start state = (11, 13)

Goal state = Pac-man state

Operators = (-1, 0), (1, 0), (0, -1), (0, 1)

**Space State Representation**



**Fig: State Space with Position of Pac-man (15, 11)**

**Algorithm Used**

**i) Dijkstra's algorithm**

Dijkstra's algorithm (or Dijkstra's Shortest Path First algorithm, SPF algorithm) is an [algorithm](https://en.wikipedia.org/wiki/Algorithm) for finding the [shortest paths](https://en.wikipedia.org/wiki/Shortest_path_problem) between [nodes](https://en.wikipedia.org/wiki/Vertex_(graph_theory)) in a [graph](https://en.wikipedia.org/wiki/Graph_(abstract_data_type)), which may represent, for example, [road networks](https://en.wikipedia.org/wiki/Road_network). It was conceived by [computer scientist](https://en.wikipedia.org/wiki/Computer_scientist) [Edsger W. Dijkstra](https://en.wikipedia.org/wiki/Edsger_W._Dijkstra) in 1956 and published three years later.

**Pseudocode**

function dijkstra(G, S)

for each vertex V in G

distance[V] <- infinite

previous[V] <- NULL

If V != S, add V to Priority Queue Q

distance[S] <- 0

while Q IS NOT EMPTY

U <- Extract MIN from Q

for each unvisited neighbour V of U

tempDistance <- distance[U] + edge\_weight(U, V)

if tempDistance < distance[V]

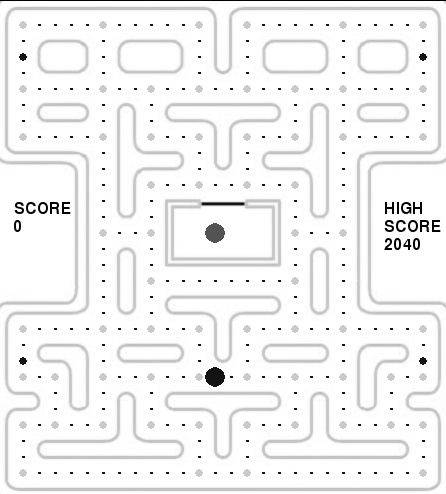
distance[V] <- tempDistance

previous[V] <- U

return distance[], previous[]

In the given algorithm, the code u ← vertex in *Q* with min dist[u], searches for the vertex u in the vertex set Q that has the least dist[u] value. length(u, v) returns the length of the edge joining (i.e. the distance between) the two neighbor-nodes u and v. The variable alt on line 18 is the length of the path from the root node to the neighbor node v if it were to go through u. If this path is shorter than the current shortest path recorded for v, that current path is replaced with this alt path. The prev array is populated with a pointer to the "next-hop" node on the source graph to get the shortest route to the source.

In the game, each tile position where the agents can change their direction are represented as Nodes. The Nodes are shown as red dots.



**Fig: The Node Positions**

For the agents to find the shortest path to the position of Pac-man, they need to know the true cost from their nearest nodes to nearest nodes to Pac-man. Dijkstra’s algorithm is use to find the true cost from a node to every other node.

**ii) A\* algorithm**

A\* (pronounced "A-star") is a [graph traversal](https://en.wikipedia.org/wiki/Graph_traversal) and [path search](https://en.wikipedia.org/wiki/Pathfinding) [algorithm](https://en.wikipedia.org/wiki/Algorithm), which is often used in computer science due to its completeness, optimality, and optimal efficiency. A\* achieves better performance by using [heuristics](https://en.wikipedia.org/wiki/Heuristic_(computer_science)) to guide its search.

A\* is an [informed search algorithm](https://en.wikipedia.org/wiki/Informed_search_algorithm), or a [best-first search](https://en.wikipedia.org/wiki/Best-first_search), meaning that it is formulated in terms of [weighted graphs](https://en.wikipedia.org/wiki/Weighted_graph): starting from a specific starting [node](https://en.wikipedia.org/wiki/Node_(graph_theory)) of a graph, it aims to find a path to the given goal node having the smallest cost (least distance travelled, shortest time, etc.). It does this by maintaining a [tree](https://en.wikipedia.org/wiki/Tree_(data_structure)) of paths originating at the start node and extending those paths one edge at a time until its termination criterion is satisfied.

At each iteration of its main loop, A\* needs to determine which of its paths to extend. It does so based on the cost of the path and an estimate of the cost required to extend the path all the way to the goal. Specifically, A\* selects the path that minimizes

f(n)=g(n)+h(n)}

where *n* is the next node on the path, *g*(*n*) is the cost of the path from the start node to *n*, and *h*(*n*) is a [heuristic](https://en.wikipedia.org/wiki/Heuristic) function that estimates the cost of the cheapest path from *n* to the goal.

**Algorithm**

1. Initialize the open list

2. Initialize the closed list

put the starting node on the open

list (you can leave its **f** at zero)

3. while the open list is not empty

a) find the node with the least **f** on

the open list, call it "q"

b) pop q off the open list

c) generate q's 8 successors and set their

parents to q

d) for each successor

i) if successor is the goal, stop search

successor.**g** = q.**g** + distance between

successor and q

successor.**h** = distance from goal to

successor (This can be done using many

ways, we will discuss three heuristics-

Manhattan, Diagonal and Euclidean

Heuristics)

successor.**f** = successor.**g** + successor.**h**

ii) if a node with the same position as

successor is in the OPEN list which has a

lower **f** than successor, skip this successor

iii) if a node with the same position as

successor is in the CLOSED list which has

a lower **f** than successor, skip this successor

otherwise, add the node to the open list

end (for loop)

e) push q on the closed list

end (while loop)

In the game, A\* algorithm is used to find the shortest path from agent to the Pac-man with the function

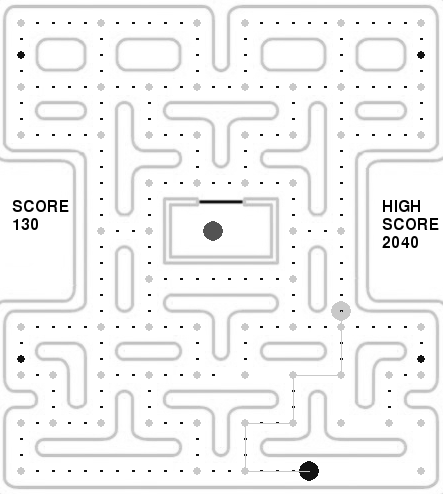
f(n) = g(n) + h(n)

where

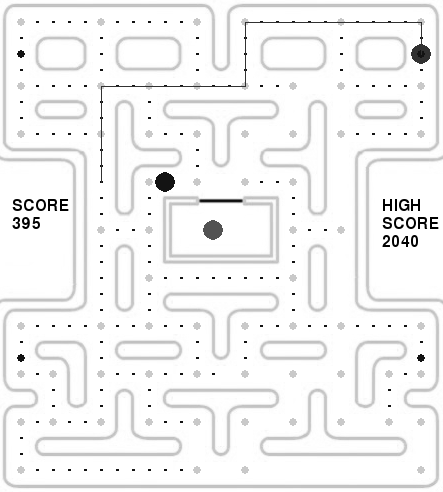
f(n) = Total distance

g(n) = Distance to nearest node

h(n) = Distance from nearest node to Pac-man’s nearest node



**Fig: Shortest path from Blinky to Pac-man**

****

**Fig: Pinky’s shortest path to 4 tiles ahead of Pac-man**

**Outcome Analysis**

The agents have been defined with their own artificial intelligence or “personality”. Each of the agents has its own method of following Pac-man. Blinky moves towards Pac-man in the shortest path. It tries to ambush Pac-man from the back. Pinky moves towards 5 tiles ahead of Pac-man in the shortest path. It tries to ambush Pac-man from the front. Inky mirrors the position of Blinky with respect to Pac-man to create its target. It uses the position of Pac-man as well as Blinky to ambush Pac-man. Clyde moves similar to Blinky but runs away when Pac-man comes near it. All the agents act as expected. The game works in a hard difficulty for the player but with the use of scatter mode and frighten mode, there is a likely possibility the player can with the game.

**Conclusion**

Hence, the Pac-man game was created with each of the agents with their own personality and modes by using Dijkstra’s algorithm and A\* search algorithm. The different personality and modes of the agents create a game that is likely to be exciting for the players.