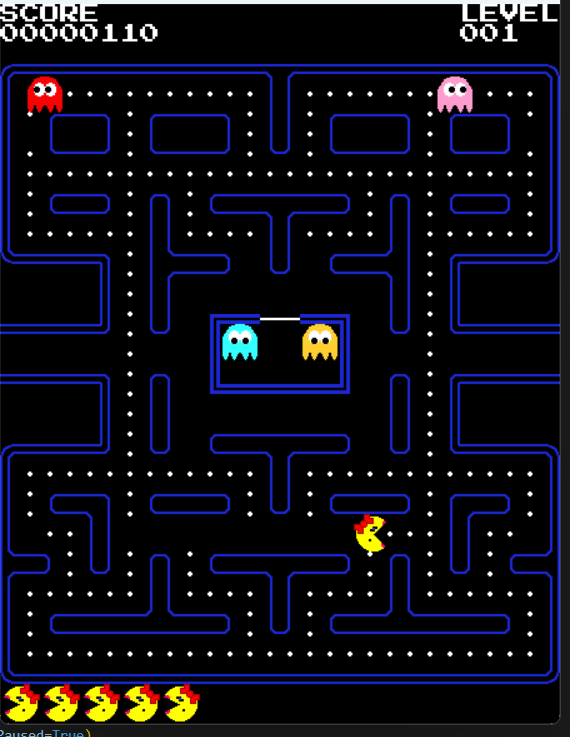
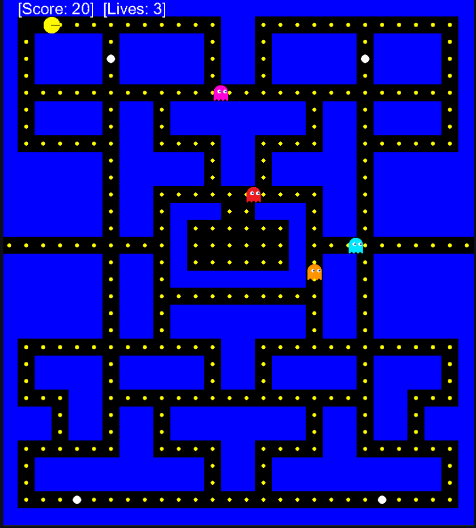
# MMI513 Term Project Report – Enhancing Pac-Man

## Introduction

We would like to start by introducing ourselves as a team. We are a team of two, two friends who have met in our master’s in MMI; Nuri Baran Ayana, an Electronic & Electrical Engineer; and Ali Ozan, a Civil Engineer and a Game Developer.

We have investigated several existing repositories that might be suitable for our term project to use as a base project to build upon. We have experimented with nearly 10 projects, considering their recency, scope, complexity, similarity to the original game of Pac-Man, ease of modifications, etc.

One of the main projects that we have tried our luck with was the project from <https://pacmancode.com/> which is a very good implementation of the original game with good visuals, animations, style, and functionality. We wanted to use that as a base project first, but the way the walls are rendered with their curved shapes raised a major challenge. The project uses them as static maps, but creating these curved corners with separate rotation matrices and layouts was too big of a challenge for random maze creation. So, we shifted our focus to projects that have rigid shapes and corners for the walls.

 Eventually, we settled on the project from the following repository: <https://github.com/janjilecek/pacman_python_pygame> . The project has a basic implementation of the original game with its core functionality present, but lacking in the visual art style. Since art is not very important in our context, we have happily adopted the project. The original code is a bit crowded and stacked on a single Python file but it met our expectations. It has a replica of the original Pac-Man map, but defined with ASCII characters; zeros for walls, ones for open paths, “G” for ghosts, “P” for player or Pac-Man, and finally “O” for powerups that frighten the ghosts and lets the player eat them. It also has the teleport gates in the middle, and the portal function is handled correctly. The ghosts have two modes which are “Chase” and “Scatter”. In Chase mode, the ghosts get a path to the current position of the player and try to get to them. In the Scatter mode which gets activated when the player picks up the power-up, the ghosts pick a random reachable location as their target and try to get to it. The problematic part of the original project is that, when a ghost picks a target, such as the player, it does not actively follow the player. It takes a snapshot of its position and the player’s position at that time and follows the resulting path from that snapshot. Only when it gets to its destination, it takes another snapshot to determine its next target location. We have tried to handle this problem in our project as best as we could, the details will be discussed later.

## Enhancements

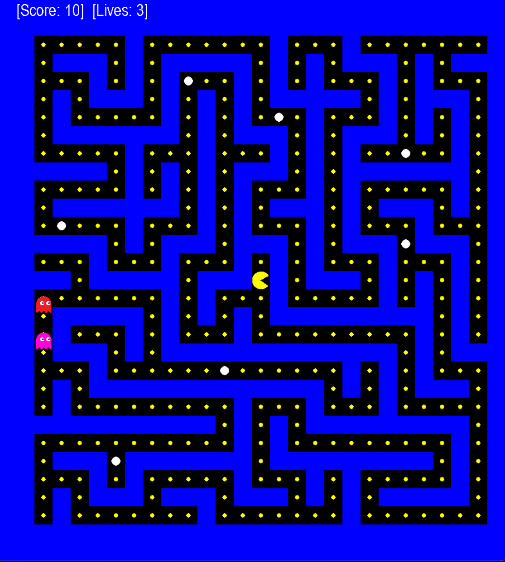
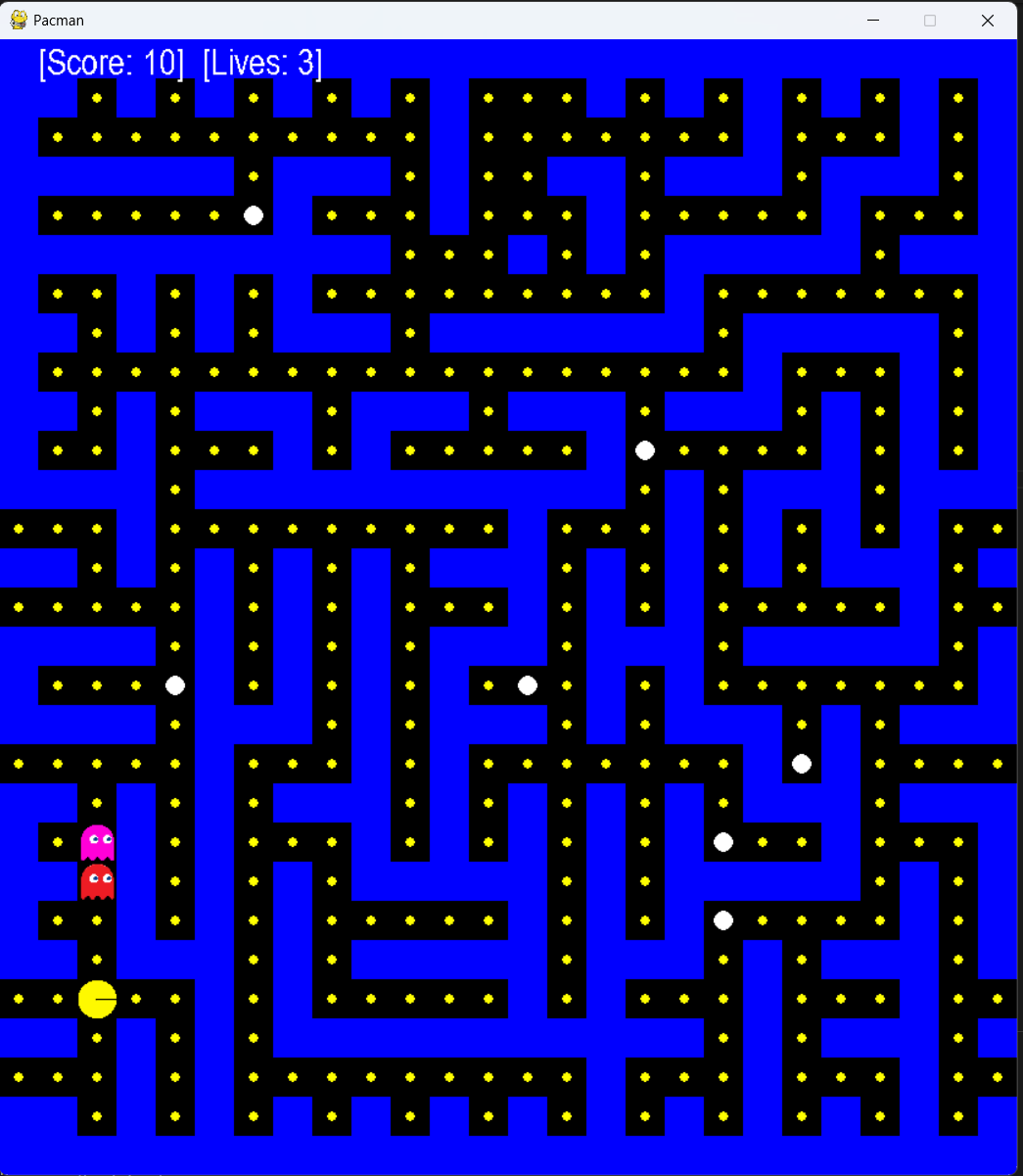
### Maze Generation

As mentioned in the introduction, the original project uses ASCII characters to define a map and then converts it in into necessary rendering input. We have aimed to use the same approach and built our methods and algorithms to result in a 2D grid which defined the coordinates of walls, paths, player, powerups, and finally ghosts (ghosts have been carried out by the Strategy-AI later).

We have implemented the Depth First Search (DFS) algorithm at first and made some observations and experiments with it. Later, we have also implemented the Prim’s Algorithm for maze generation because DFS usually results in longer, winding corridors, often with dead ends, and for a game like Pac-Man this means no way out for the player if there are ghosts at the end. For a more forgiving and flexible gameplay we have implemented the Prim’s Algorithm as our default maze generation method because it results in shorter corridors and more options to choose for the player.

Our workflow of maze generation goes like this;

* Create a 2D grid only consisting of walls (zeros).
* Carve out the corridors using DFS or the Prim’s Algorithm (not including the edges of the map, considering them as a frame).
* Among the suitable locations, pick a random place and place the player.
* Among the remaining path locations, place the powerups by a pre-defined count.
* Among the rows which has the first and the last grid as a path, randomly open up portal gates.



Maze Generated by Prim's Algorithm

Maze Generated by Depth First Search Algorithm

### Strategy-AI

This part aims to regulate the strategy of the game in a way to increases the winning probability of the player. In this regard, the primary objective of the Strategy AI is to strategically position and spawn ghosts on the game board, aiming to maximize the challenge for the player (Pac-Man) and increase the game's difficulty over time. The details on design and enhancement are given below:

#### Strategic Spawn Positioning:

The AI should determine the spawn positions of the ghost based on the pattern of the maze and the player's starting position. To realize this task **Breadth First Search (BFS) Algorithm** was used to scan each unit of the maze. BFS adds the starting node to the queue and searches all the nodes by marking nodes as “Visited” and “Not Visited”. In this regard, the algorithm is designed as below:

* All the nodes are searched, and the node is taken into consideration if the node is available (not wall, route).
* According to the direction of the node regarding the starting point, distance is increased by the node base. The position and the distance to Pac-Man are set to the list.
* The limit distance was set as 10, and the available nodes that are not close to 10 were added to the “Candidate Spawn Locations”.
* According to the output of the BFS algorithm, the candidate spawn location for each ghost is selected randomly.

#### New Ghost Generation:

The AI similarly works as Strategic Spawn Positioning. The only difference in spawn positioning is the location of the Pac-Man. The candidate spawn locations are regenerated by basing the current position of the Pac-Man, and the algorithm is revised as below:

* The timer is set from the starting time, and if the difference is bigger or equal to 60 seconds, the new ghost is generated.
* The current location of the Pac-Man and the generated maze is set and the BFS Algorithm is generated to obtain candidate spawn locations.
* The candidate spawn location is selected randomly from the list.
* The total number of ghosts is set in the code, and ghosts with different colors are selected respectively from the available colors.

#### Enhancement of the Strategy of AI Generation

In the base code, there was no ability in terms of strategy since the placement of ghosts for the generated maze was not available. Thus, the enhancement stands as defined below:

* The placement for spawn locations of the ghosts is generated for diverse locations of the players and mazes.
* Difficulty level can be arranged strategically by setting ghost spawn distance.
* A new ghost generation structure was created.
* Ghost generation time limit can be determined, and color variety is provided.

### Tactical-AI:

The AI should provide the sensing ability to the ghost to provide the ability to chase the Pac-Man. The player and ghost locations are used to check whether the sensing is available or not. A\* algorithm is used to realize the chasing task. The design flow is given below:

#### Sensing Mechanism:

The AI checks the player, and the ghost positions each time to determine whether the mode is **Chase** or **Random**. The mode selection is turned into an adaptive form as given below:

* The “Manhattan Distance” is calculated for each ghost.
* If the “Manhattan Distance” is lower or equal to 10, the mode is selected as “Chase”.
* If the distance score is bigger than 10, the path is selected randomly.

#### Generating Pathfinding Algorithm:

The AI is going to determine the path of the ghost for a given target location. The implemented A\* is preferable since the A\* algorithm is goal-oriented while the Dijkstra algorithm (evaluated for a possible change) finds the shortest path without considering all directions. Therefore, the usage of the pathfinding algorithm is revised as below:

* Chase Mode is activated each time by comparing the “Manhattan Distance”.
* In the base mode, the ghost moves through the given hero location while the pathfinding algorithm is triggered by the current location of the player.

#### Enhancement of the Tactical-AI Generation

In the base code, there was no ability to track the player constantly. The pathfinding algorithm is dynamically used. Thus, the enhancement stands as defined below:

* Chase mode is adaptively determined while in the base code, the ghost tracks the player in discrete time intervals.
* The sensing mechanism was added. The ghost chases the player by comparing the distance with “Manhattan Distance” and searches for the player while the distance is satisfied.

### Analytics

The analytics module is created to provide a log mechanism to track a wide array of movements and decisions made by each entity—be it the player's character or the AI-driven ghosts. We have the parameters about the generated game and setup of the game as a label at the start of each log:

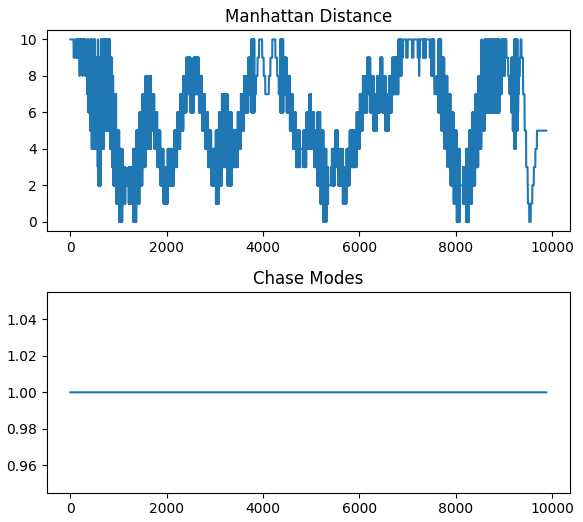
* Power Up Count: 7
* Portal Gate Count: 5
* Map Size: 29x26
* Ghost Spawn Frequency: 60
* Manhattan Distance Limit: 10

The log mechanism proves the enhancement of the game as follows:

* **Strategy-AI:** Log mechanisms show the distances at the starting time and time of new ghost generation. The logs below show that distances are bigger than the limit values of 10 where “Ghost Generation Time” is equal to 0, the initial time of ghost generated, and “Ghost Generation Time” is equal to 1, ghost generated at every 60 seconds. The output is given below:

{**'StrategyAIDist'**: [13, 12, 16.278820596099706], '**GhostGenType**': [0, 0, 1]}

* **Tactical-AI:** Log mechanisms add the information of each ghost turned into a “Chase Mode” which provides to observe the sensing mechanism of the ghosts by comparing the Manhattan Distance between ghosts and the players. For the case of Chase Mode is 1, the distance must be equal and lower than 10. The figure below confirms the design:



Manhattan Score for Ghost Mode is Chase

## Conclusion

In conclusion, the integration of dynamic maze generation, sophisticated Strategy-AI, Tactical-AI, and an in-depth Analytics module has significantly elevated the complexity and engagement level of our Pac-Man game. The base project has been developed for a single maze and deterministic behaviors of the ghosts. With our contribution, the diversity in the game was provided to the player to have a wider experience.

Together, these enhancements transform the classic Pac-Man game into a more challenging and engaging experience. We have pushed the boundaries of traditional Pac-Man by incorporating abilities of maze generation, dynamic pathfinding, artificial intelligence behaviors, and decision-making processes with running game flow. This project not only regulates the conventional game but also sets a state-of-the-art method for how classic games are reborn with modern technology.