Introduction

This project aims to develop a terminal-based version for Mac of the classic Pacman game using C++ and Object-Oriented Programming (OOP) principles. The goal is to recreate core game mechanics such as player movement, ghosts AI behavior, map structure, win tracking, and game-over conditions, all within a text-based environment.

The program involves multiple classes to represent the game logic, characters (Pacman and ghosts), and interaction with the console.

This implementation showcases:

- Object-Oriented Programming (OOP): via encapsulated classes like PM_PacMan, Ghost, PM_Map, and their derived types.
- Polymorphism: used to define unique ghost behaviors through inheritance.
- Dynamic memory management
- File I/O (to save/load game state or scores)
- Basic pathfinding algorithms (Dijkstra)
- exception handling (try/catch)

Program Interface

To run the program, compile the project using CLion compiler or a compatible C++ compiler. As program is compiled, the executable file can be found in the cmake-build-debug directory.

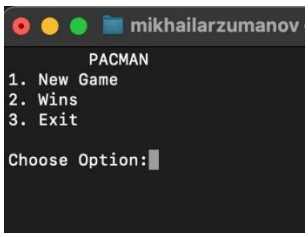
To run the program, write this command in the terminal in the directory of the executable file:
./PacMan_Game.exe

Ensure that the hall_of_fame.txt file is located in the same directory as the executable file.

To exit the game at any time, select the third option "Exit" from the main menu and close the terminal.

Program Execution

Upon starting, the user is presented with a menu:



1. New Game starts the new game from a beginning 2.
Wins show the number of wins on this current device.
3. Exit closes the whole game In-game

features:

- Use WASD keys to move the Pacman.

- Collect all dots (.) to win the game.

- Power Pellets ([]) temporarily make ghost slowly and allow Pac-Man to eat ghosts.

- Ghosts chase the player unless scared.

-Win is saved to hall_of_fame.txt when all dots are collected.

Input and Output

Input:

- Keyboard input (WASD) for Pacman movement.

- 'Q' to quit the current game.

- Number selection for menu options.

Output:

- menu section.

- Console-based rendered map and characters.

- Victory is displayed when all dots are collected.

- Wins section displayed hall_of_fame.txt file, which contains a number of past victories as integers.

Program Structure

The project consists of several source files divided into classes, which together implement the logic, interface, and game mechanics of a text-based Pac-Man game.

Below is a breakdown of the most important modules and their responsibilities:

- PM_Map (PM_Map.h/.cpp): Stores and updates the map of the game using a 2D integer array. Handles drawing the map, player and ghost positions, pellets, and power pellets. Implements ClearMap() and ResetMap() to restore game state for new games. Contains logic to check for collision, determine victory condition, and manage file I/O with hall_of_fame.txt.

- PM_PacMan (PM_PacMan.h/.cpp): Manages the Pac-Man character, including coordinates and movement logic. Contains collision logic with power pellets. Reacts to keyboard input and updates Pac-Man's position accordingly.

- Ghost (Abstract Class) (Ghost.h/.cpp): Base class for all ghosts. Stores common attributes like position, direction, mode (CHASE, SCATTER, FRIGHTENED), and timers. Provides virtual methods like MoveStep(), ChaseMode(), ScatterMode(), and FrightenedMode() for every ghost. Handles speed control through ghostSpeedFactor and transition logic between modes.

- RedGhost, PM_PinkGhost, PM_BlueGhost: Each inherits from Ghost and implements ghost-specific AI behavior in the SwitchMode() method. Use their own logic to switch between CHASE and SCATTER modes based on proximity to Pac-Man. React to FRIGHTENED state triggered by power pellet consumption.

- PowerPellet (PM_PowerPellet.h): A simple class storing x and y coordinates of a pellet. Used for initializing and resetting the position of power pellets via dynamic memory.

Map Representation

Ais a std::vector<std::vector<int>> is used to represent the game board. Special integers represent different map elements:

- ❖ 0 = empty space
- ❖ 1 = dot
- ❖ 2 = power pellet
- ❖ -1, -3 = eaten items

File I/O

File hall_of_fame.txt is used to store the number of victories. File is accessed using fopen, fscanf, and fprintf(from BoP 1).

Menu Logic

The Menu() function inside PM_Map.cpp handles the main game loop:
-Displays start screen and reads user option.
-Starts a new game loop or shows saved scores depending on the input.

Testing and Verification

The program was tested manually through multiple full playthroughs, simulating both successful and unsuccessful game sessions. Particular focus was placed on the following points:

- Ghost Behavior Validation: Ghosts were observed in all three modes: CHASE, SCATTER, and FRIGHTENED. The transition to FRIGHTENED mode after eating a power pellet was verified. Ghosts revert to SCATTER mode correctly after the timer ends. After being eaten, ghosts return to their starting positions and remain in FRIGHTENED mode until the timer expires.
- Power Pellet Logic: Confirmed that eating a power pellet triggers FRIGHTENED mode in all ghosts. Ensured that power pellets disappear only when eaten and reappear properly after restarting the game. Verified that multiple restarts during one program session still reset pellet positions correctly without memory leaks.

- Victory and Game Over: Tested that collecting all dots triggers a victory message and logs a new score to the file. Ensured that contact with a ghost outside of FRIGHTENED mode results in a game over.
- Map Integrity and Collision Detection: Checked that Pac-Man cannot move through walls. Validated that map boundaries are respected and there is no accidental access beyond array limits.
- Menu and File I/O Testing: Tested the 'Wins' menu option, ensuring that it reads and prints the contents of hall_of_fame.txt correctly. Ensured that saveScore() appends new entries to the file without overwriting previous ones.

Improvements and Extensions

We can continue to improve game experience with the following features:

- Add scoring system with bonus points.
- High score screen and player names.
- More complex maze generation.
- Additional ghost types with smarter AI.
- Save/load game metrics.

Difficulties Encountered

Several non-trivial problems emerged during development and testing:

- Memory Management of Power Pellets

Since power pellets were allocated dynamically using new, it was essential to properly delete them and avoid memory leaks. Ensuring that each new game deletes the old pellets and allocates fresh ones required careful placement of delete and new within ClearMap().

- State Consistency After Ghost Reset

One of the hardest bugs occurred when ghosts were eaten by Pac-Man while in FRIGHTENED mode. After calling ResetStart(), they incorrectly exited frightened state. This was resolved by ensuring ResetStart() does not modify the ghost mode or timer unless explicitly needed.

- Power Pellet Side Effects on the Map

When placing two adjacent cells for each power pellet, improper boundary checks allowed PacMan to access out-of-bounds cells. The logic was rewritten to place only one pellet cell and ensure all map writes respect width boundaries.

Conclusion

The project successfully demonstrates core programming concepts including classes, dynamic memory, file I/O, and state-based game logic. It replicates a simple, playable version of Pac-Man in the console.

References

- https://youtu.be/jB2E-RisUtI?si=bRajAcGQ90Rar0rf
- https://youtu.be/J-7MzbEtTR0?si=5Zu1HLaU5AWShR2U
- https://youtu.be/GXlckaGr0Eo?si=di70EYUIB5v2_1-r
- https://youtu.be/vC0d1rDmPBs?si=VpbZFajfcWyCBjvX
- https://habr.com/ru/articles/109406/