SEN2212 – Data Structures and Algorithms II

Project Report

Group No	Project Title	Lab Section No	Students
1	Pac-Man Game	902	Sasan Shafieımatanagh 2250390 Anıl Türkyılmaz 2202873

Introduction

This project recreates the classic Pac-Man arcade game in Java. Our implementation features grid-based movement, collision detection, score/life tracking, and breadth-first-search (BFS) path-finding for ghost AI. The objective is to demonstrate the practical use of core data-structure concepts—hash sets, queues and grids—within an interactive GUI application.

Purpose / Project Proposal

The purpose of the project is two-fold: (1) give users a playable Pac-Man clone, and (2) illustrate how data-structures such as Arraylist, HashSet, LinkedList (as Queue) and 2-D boolean grids can be integrated with event-driven programming (Swing) to solve real-time path-finding and collision problems.

Software Language / Project Environment

• Language: Java 21
• GUI Toolkit: Java Swing
• Build Tool: IntelliJ IDEA
• JDK: OpenJDK 17 or later

Data Structures

• HashSet<Block> — O(1) average membership test used for walls, foods and ghosts.
• LinkedList<Point> as java.util.Queue — underlying container for BFS frontier.
• boolean[][] grid — passability matrix for the maze; enables O(1) access during BFS.
These were selected because constant-time look-ups are critical for real-time gameplay, and linked-list queues simplify FIFO expansion for BFS.

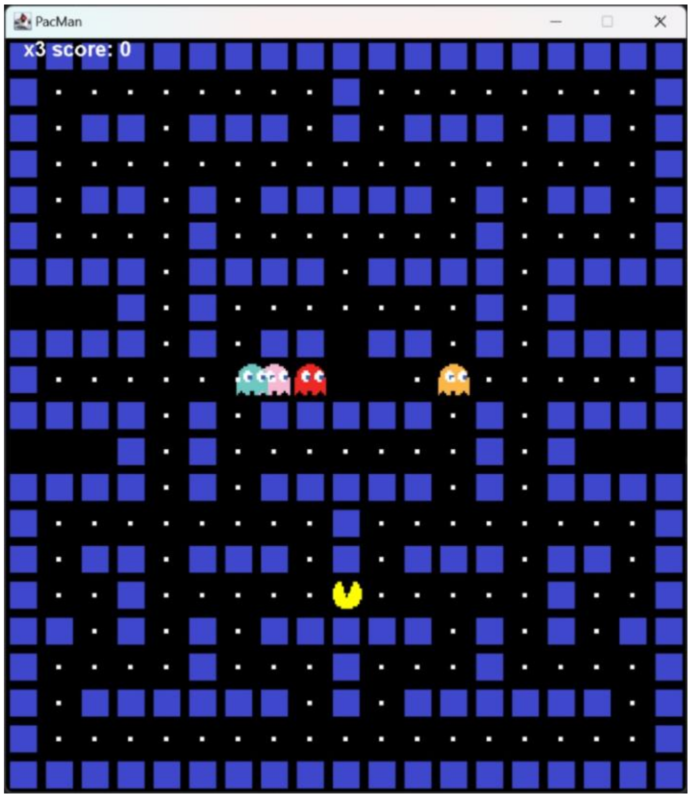
Work Partitioning

Name / ID	Role	Dates	Description
Sasan Shafıeımatanagh 2250390	Back-end & AI	10-04-2025 → 02-05-2025	Implemented BFS path-finding, collision logic, and timer loop.
Anıl Türkyılmaz 2202873	Back-end & AI	10-04-2025 → 02-05-2025	Collision logic, and timer loop.

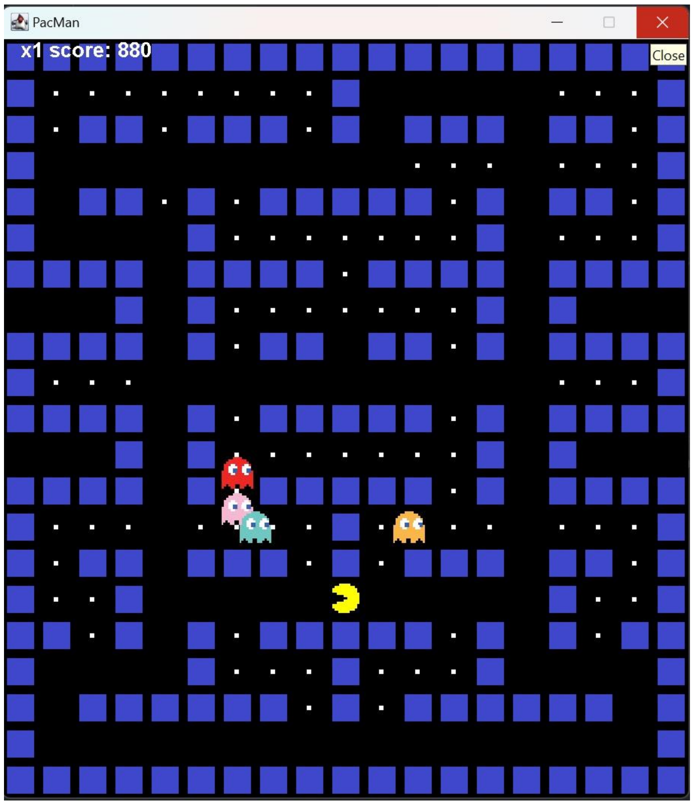
Architectural Representation

The architecture is presented through two UML diagrams.

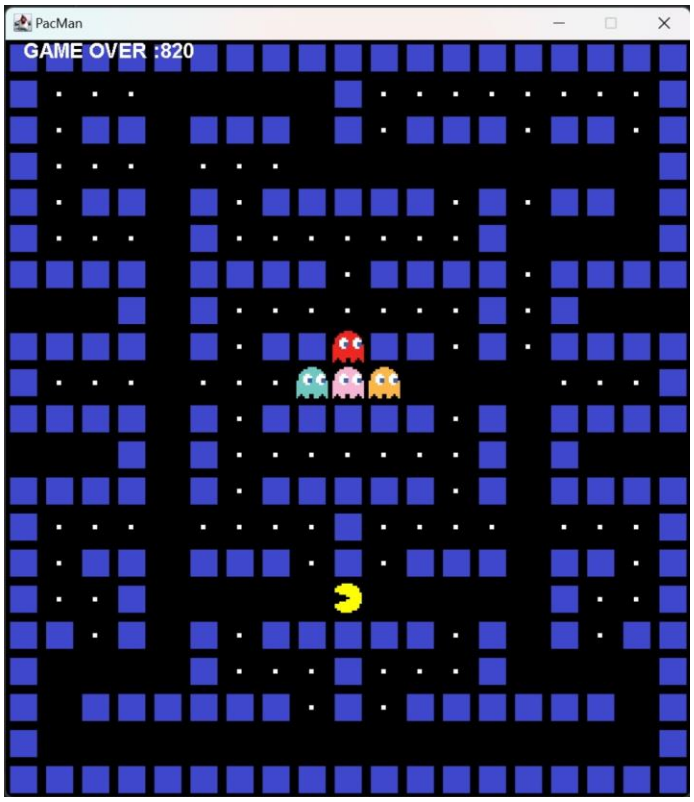
Use Case Diagram



START THE GAME



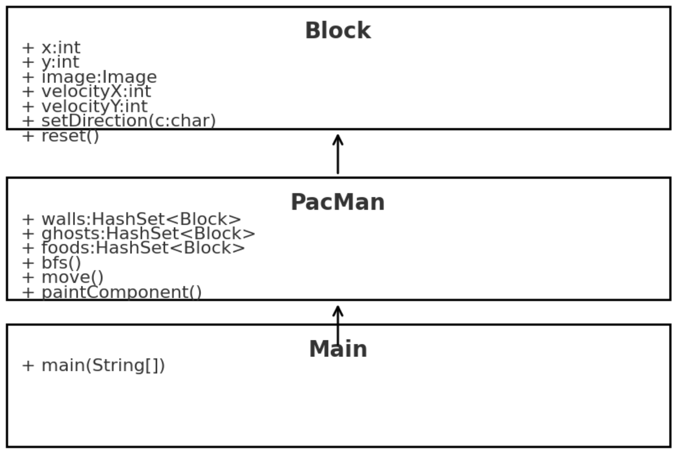
QUİT THE GAME



GAME OVER

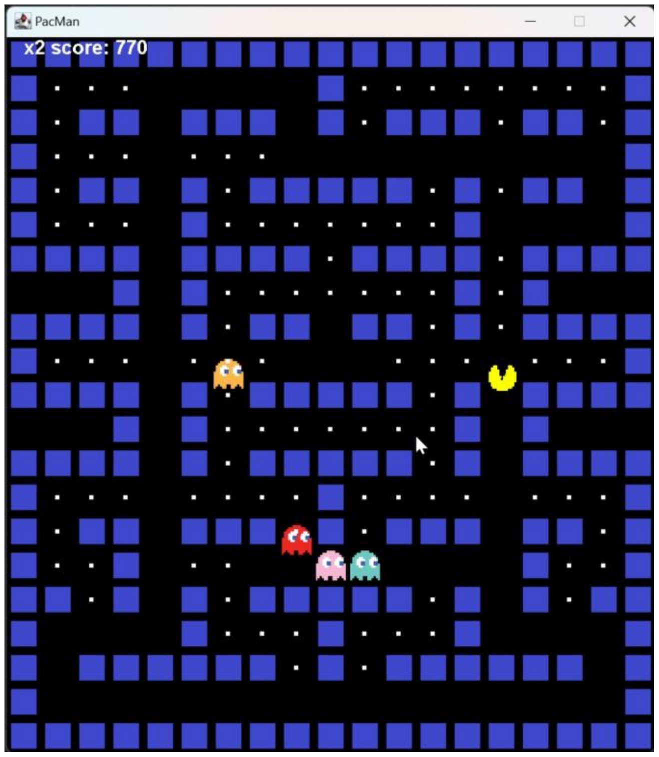
Class Diagram

Illustrates the main classes and their relationships. PacMan extends JPanel and implements ActionListener and KeyListener. Block encapsulates shared sprite state, while Main boots the Swing frame.



Application

Below is a representative game screen. The player controls Pac-Man using the arrow keys while ghosts pursue him using BFS over the passability grid. Pellets disappear when eaten, adding 10 points each. Colliding with a ghost decrements one life; losing all lives triggers the "GAME OVER" overlay.



Conclusion / Summary

The project demonstrates how classical data-structures integrate with event-driven GUI programming to solve a real-time game problem. Hash sets efficiently store static obstacles, linked-list queues power BFS path-finding, and a simple timer loop orchestrates rendering and state updates at ~18 FPS.

References

[1] Oracle. "How to Use Key Bindings." Java Tutorials.
[2] Red Blob Games. "Breadth-First Search Visualization."
[3] Pac-Man Wiki. "Original Maze Layout."
[4] Stack Overflow threads consulted January–April 2025 for Java Swing event handling and repaint best-practices.