

Distributed Multiplayer Pacman

Simone Carpi, Alessandro Ronchi, Jiekai Sun

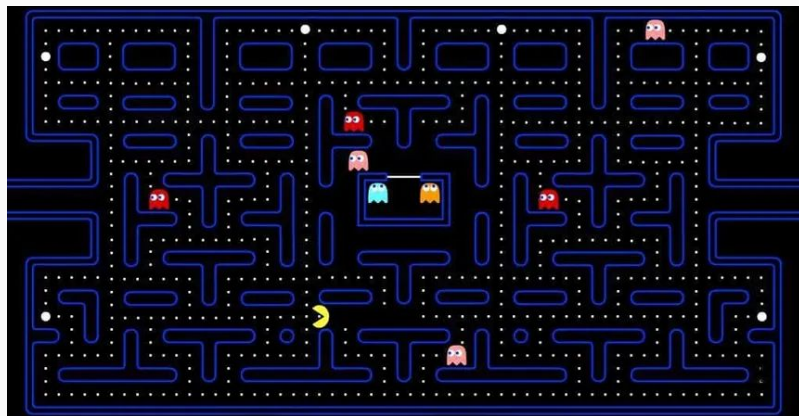
A.Y 2025-2026



Gameplay & Rules

We adapted the traditional Pacman rules to accommodate a multiplayer format:

- Player-controlled Pacmans navigate a 2D maze to eat Dots while avoiding Ghosts. Each player starts with 3 lives.
- Consuming special dots temporarily disables Ghosts on impact.
- A match ends after 3 minutes or when only 1 player with lives left remains.



Key Features

Account Management & Statistics

User registration and login, alongside tracking of personal performance metrics, including total matches, best scores, and win rate.



Matchmaking & Real-Time Gameplay

Matchmaking system that matches 4 players into a 3-minute game where they compete by eating dots and avoiding ghosts.



Disconnection Handling

Instant bot substitution when players disconnect mid-game, with match reconnection measures upon client or server recovery.



Cross-Platform Compatibility

Client application support for Windows, macOS, and Linux, enabling cross-platform play across a wide range of devices.



Authoritative State & Client Prediction

Authoritative game server execution paired with local client speculative execution for a smooth user experience.



Dynamic Scalability & Crash Recovery

On-demand game server provisioning with game state restoration to recover active matches upon crash.



Why Distribution Is Needed

Intrinsically Distributed

Supporting real-time matches across multiple client devices requires a distributed backend infrastructure.

Performance Consistency

To guarantee consistent real-time performance, game server logic should be isolated from secondary tasks. Separating CPU-bound match logic from I/O-heavy operations like authentication avoids resource competition during live matches.

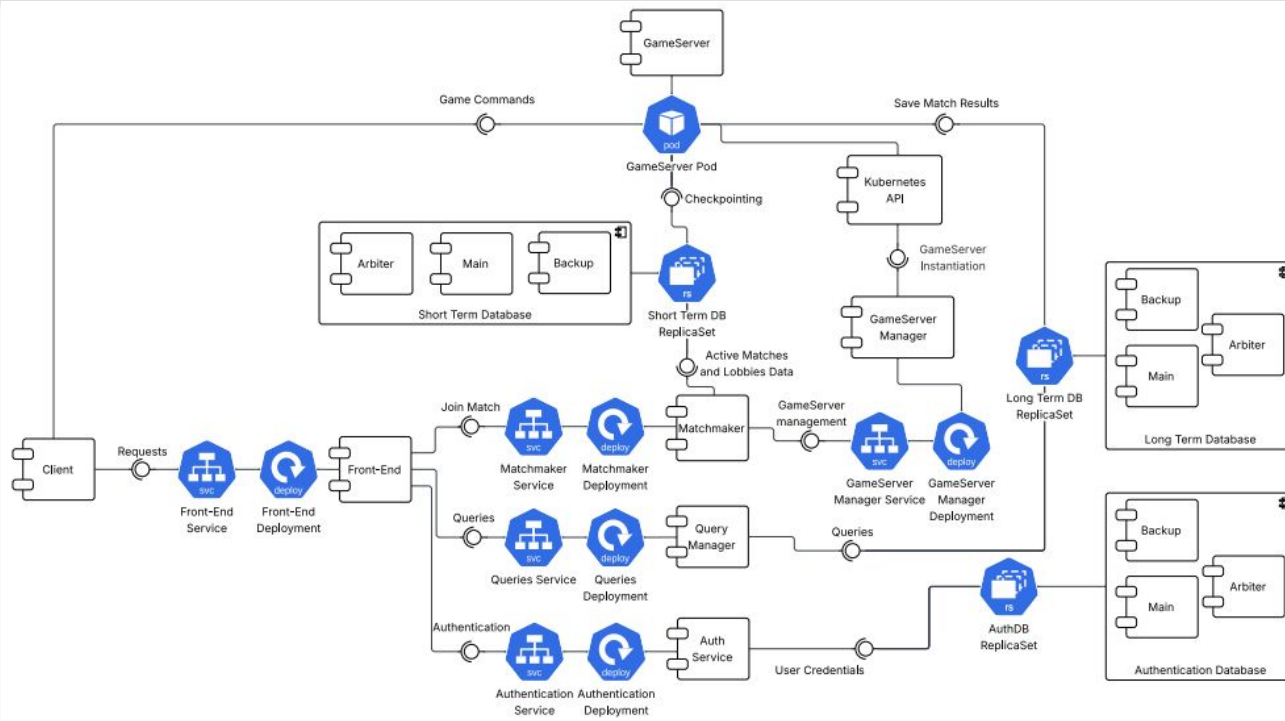
Fault Tolerance

In a monolithic architecture, a single process failure affects the entire infrastructure. Distributing live matches across separate nodes isolates such failures.

Horizontal Scalability

Isolating components allows them to be independently scaled. For example, as active player demand grows, game server nodes can be dynamically provisioned on the fly.

Distributed System Architecture



Balancing Consistency & Availability



Database Replica Sets

Prioritizes Consistency

- Use one primary node for writes and two secondary ones for data replication
- Consistency is prioritized since the only entry point for read and write is the primary one. Secondary ones serve as backup nodes in case of a primary node failure.



Authoritative Game Server

Prioritizes Consistency

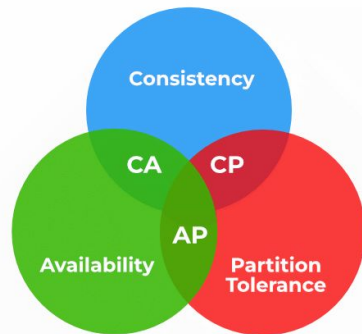
- Serves as the single source of truth for all non-deterministic game logic, including collisions, scoring and AI pathfinding.
- Clients are constantly updated with such information



Speculative Execution

Prioritizes Availability

- Clients locally predict inertial player movement and side-effect-free state changes to maintain visual responsiveness under adverse network conditions.
- Any local drift is reconciled once authoritative server updates arrive



Fault Tolerance



Failure Detection

- Client and Game Server constantly exchange heartbeats
- The Game Server emits heartbeats to the Game Server Manager



Disconnection Handling

- Disconnected players are replaced by bots to keep the match running uninterrupted
- Players can reconnect at any time to the ongoing match



Match Recovery

- Game Servers take state snapshots every 10 seconds (checkpointing)
- Upon a Game Server crash, the match can be resumed on a recovery one

Results

Pacman

Login

Username
player1

Password
...

Login

Register

Pacman

Register

Username
player1

Password
...

Register

Home

Results

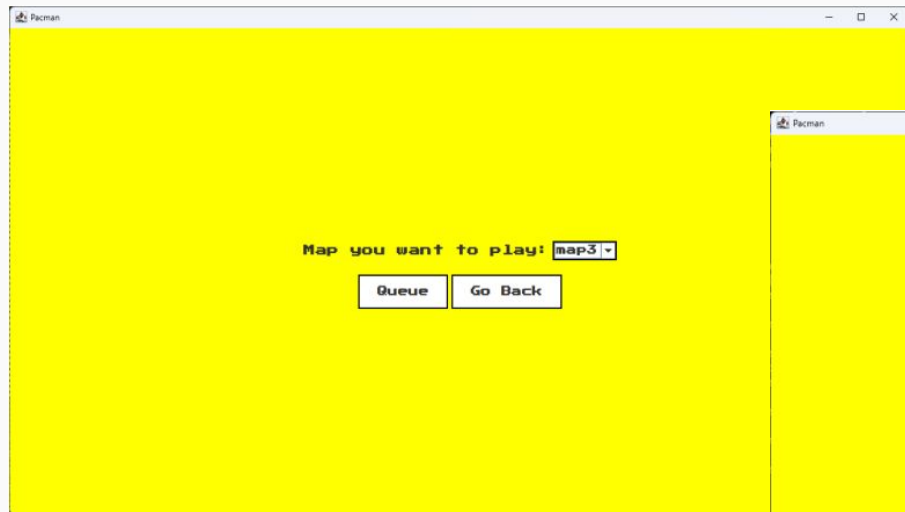
The image shows two overlapping browser windows from a web application titled "Pacman".

The background window displays a menu on a red background with the title "Pacman" in a pixelated font. It contains three white buttons with black text: "Find Match", "My Stats", and "Logout".

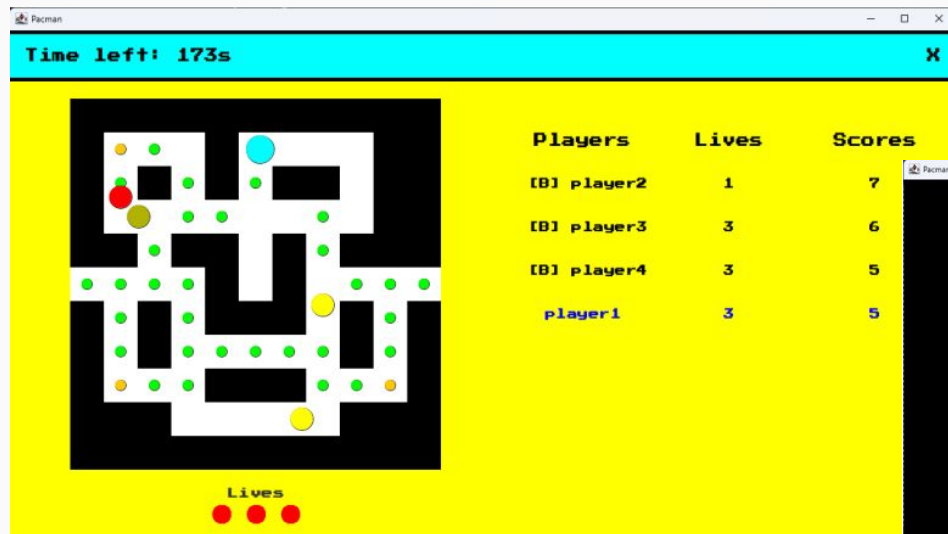
The foreground window shows the profile for "player1". It features a table with statistics on a red background. The table has four columns: "N. Matches", "N. Wins", "Win Rate", and "Best Score". The values are 12, 5, 0.1, and 1250 respectively. Below the table is a yellow "Home" button.

<u>N. Matches</u>	<u>N. Wins</u>	<u>Win Rate</u>	<u>Best Score</u>
12	5	0.1	1250

Results



Results



Time left: 173s

Players	Lives	Scores
[B] player2	1	7
[B] player3	3	6
[B] player4	3	5
player1	3	5

Lives

The screenshot shows a Pacman game in progress. On the left is a maze with a yellow background and black walls. Pacman (a blue circle) is at the top center. There are several green dots (pellets) and a few colored dots (red, yellow, and orange) scattered throughout the maze. On the right is a table showing the status of four players. At the bottom, there are three red dots representing lives.



Game Over

Winner: player3

player3 : 30 pts
player4 : 15 pts
player2 : 11 pts
player1 : 5 pts

Go Back

The screenshot shows a black screen with white text. At the top, it says 'Game Over'. Below that, it says 'Winner: player3'. Then, it lists the scores for each player: player3 has 30 points, player4 has 15 points, player2 has 11 points, and player1 has 5 points. At the bottom, there is a white button with the text 'Go Back'.

[Link to the demo](#)

Thank you for your
attention!