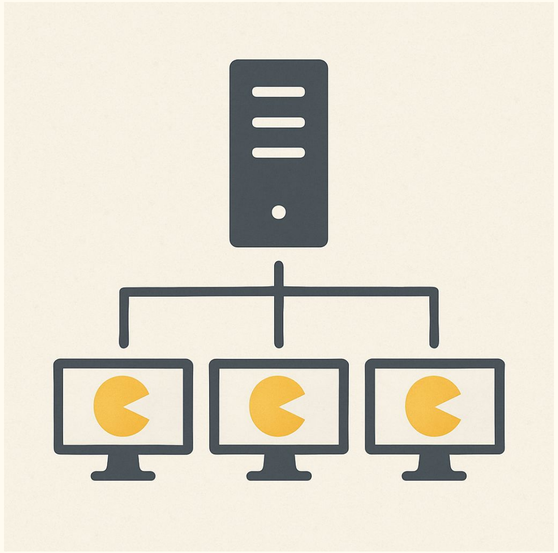
PAIcman CS262 Final Project

By Matthew Vu, Pedro Garcia, Samuel Kim

Introduction

- Distributed multiplayer reinterpretation of classic Pac-Man game
- Multiple game modes ❗❗
    - Players play against each other as Ghosts, or Pacman
    - Players control the Ghosts, chasing an AI Pac-Man
- Fun way to destroy AI as they take your job



Plan

- Distributed systems principles can enhance classic gameplay mechanics
- Team Ghost Mode
    - One AI-controlled Pac-Man
    - Three human-controlled Ghosts
- Full Multiplayer Mode
    - One human-controlled Pac-Man
    - Three human-controlled Ghosts
- Persistent scores

Challenges

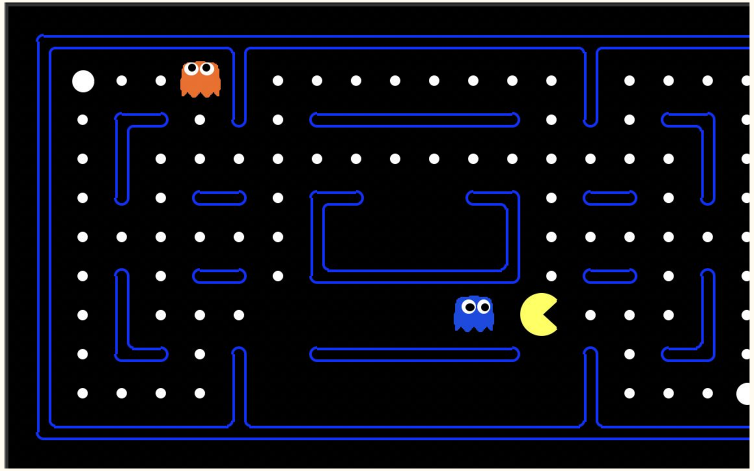
- Network latency affecting real-time gameplay experience
- Maintaining consistent game state across multiple clients
- Handling node failures without disrupting active gameplay
- Balancing system complexity with gameplay responsiveness
    - Performance degradation with increasing player count
    - Recovery mechanisms after network partitions

Details

- gRPC abstraction for simplified client-server interaction
- Hybrid logical clock that combines wall-clock time and event counting
    - Ensures consistent game state across all players and server nodes
    - Prevents race conditions in game actions
- State replication with consensus algorithm (Raft)
    - Fault tolerance for player disconnections
    - Consistent game state across all nodes
    - 2-fault tolerance (1 primary node and 2 secondary nodes)

Progress

- Created gRPC protocol buffer files
- Generated the GUI for the Game
- Basic Client-Server architecture
    - Connected client to server



Future Steps

- Collision system with real-time scoring calculation
- Logical clock implementation for event ordering
- Consensus algorithm integration in progress
    - Solutions will require fault tolerance testing
- Replication protocols under development
- Multiple game sessions concurrently
    - Users can join current game or create a new one

Questions?