





Graphic Era
deemed to be **University**
DEHRADUN

PROJECT AND TEAM INFORMATION

Project Title

Pac-RL: An Artificially Intelligent Pac-Man Player

Student/Team Information

Team Name: UG_ML_AIDS_30	Model Masters
Team member 1 (Team Lead)	Nirbhay Singh-23021166 nirbhay5813@gmail.com 
Team member 2	Neha Dhondiyal-23462000 nehasharma67ak@gmail.com 

Team member 3

Harshita Dobhal-2316022111
harshitadobhal24@gmail.com



PROJECT PROGRESS DESCRIPTION

Project Abstract

This project focuses on the integration of a Reinforcement Learning (RL) agent with a fully developed, tile-based Pac-Man game. The primary goal is to leverage this existing game environment to train an AI agent to play autonomously. The agent will be implemented using a Q-learning algorithm, learning to navigate the maze, consume pellets, and evade ghosts by maximizing a cumulative reward. The state of the agent will be defined by its immediate surroundings in the tile-based grid, actions will be the four cardinal movements, and rewards will be structured to encourage pellet consumption and survival. The final deliverable will be a trained AI agent that can demonstrate intelligent decision-making within the completed game, showcasing the practical application of reinforcement learning.

Updated Project Approach and Architecture

We will build a 2D Pac-Man game in Python using Pygame and integrate a Reinforcement Learning agent trained with Q-learning.

1. Game Development:

- **Engine:** Python with Pygame.
- **Architecture:** A tile-based map represented by a 2D array, managing game entities (Pac-Man, ghosts) and items (pellets, power-ups).
- **Core Mechanics:** Player movement, collision detection, scoring, and basic ghost AI.

2. Reinforcement Learning (RL) Integration:

- **Algorithm:** Q-learning.
- **State (S):** The status of the agent's immediately adjacent tiles.
- **Actions (A):** {Up, Down, Left, Right}.
- **Reward (R) Function:**
 - +10 for a pellet.
 - +50 for a power-up.
 - -500 for ghost collision.
 - -1 per step to encourage efficiency.
- **Integration:** The RL agent will select an action from its Q-table each frame within the main game loop.

Tasks Completed

Task Completed	Team Member
Project ideation and finalization of scope	All members
Developed the core game engine and map rendering	Neha Sharma
Implemented Pac-Man movement and pellet consumption logic	NirbhaySingh
Programmed basic ghost AI and movement patterns	Harshita Dobhal
Completed initial unit testing for core game mechanics	All members
Foundational study of Q-learning principles	Nir bhay Singh

Challenges/Roadblocks

With the game now built, our challenges have shifted entirely to the AI integration:

1. **Agent-Environment Integration:** The immediate challenge is to design a seamless interface between the RL agent and the existing game loop. This API must efficiently extract the current game state, pass the agent's chosen action to the game controls, and calculate the correct reward without introducing latency or bugs.
2. **State Representation:** We need to convert the game's visual information into a concise state representation for the Q-table. Deciding whether to use the entire grid or just a localized view around Pac-Man is a critical decision that will impact learning efficiency and performance.
3. **Hyperparameter Tuning:** The Q-learning algorithm's performance is sensitive to its learning rate (α), discount factor (γ), and exploration rate (ϵ). Finding the optimal combination for our specific game environment will require extensive experimentation and iterative testing.
4. **Training Efficiency:** Training the agent to a proficient level can be computationally expensive and time-consuming. We need to ensure our training loop is optimized and can run faster than real-time to allow for rapid iteration.

Tasks Pending

Task Pending	Team Member (to complete the task)
Design and implement the RL environment interface	Neha Sharma
Implement the Q-learning algorithm and data structures (Q-table)	Nirbhay Singh
Integrate the RL agent to take control of Pac-Man in the game	Neha Sharma and Harshita Dobhal
Train the agent over thousands of episodes to optimize its policy	Nirbhay Singh
Test and evaluate the trained agent's performance (score, survival)	Harshita Dobha;

Project Outcome/Deliverables

The key deliverables for this project are:

- 1) A fully functional and playable tile-based Pac-Man game, developed in Python with Pygame.
- 2) A trained Reinforcement Learning agent that can autonomously play the game with the goal of maximizing its score.
- 3) The complete source code for both the game and the RL agent, well-documented and hosted on a public Git repository (e.g., GitHub).
- 4) A final project report detailing the methodology, architecture, challenges faced, and the results of the agent's performance.
- 5) A short video demonstrating the trained agent playing the game.

Progress Overview

The project is approximately 30% complete. The entire game development phase, which constitutes a major project deliverable, has been successfully concluded. We have a stable, playable Pac-Man game that is ready to serve as the environment for our AI agent. The project is now transitioning into the reinforcement learning implementation phase. All tasks are currently on schedule, with a clear focus on the pending AI-related milestones.

Codebase Information

Repository Link: <https://github.com/SpringOnion4985/Pac-RL.git>

Main Branch: main

Important Commits: The repository is up-to-date with the complete, stable codebase for the playable Pac-Man game. Key commits include the implementation of the game engine, player controls, collision physics, and ghost AI .

Testing and Validation Status

Test Type	Status(Pass/Fail)	Notes
Unit Test Logic	Pass	Core mechanics such as movement, collision, and scoring are stable. To be conducted after the RL interface is built. Will begin once the agent starts its training phase.
Integration Test (Agent-Environment)	Pending	
Agent Performance Evaluation	Pending	

Deliverables Progress

The key outcomes and deliverables of the project are as follows:

- A fully functional and playable tile-based Pac-Man game.
- A trained Reinforcement Learning agent capable of playing the game autonomously.
- The complete, well-documented source code for both the game and the RL agent, hosted on a Git repository.
- A final project report detailing the design, implementation, challenges, and results.
- A video demonstration of the trained agent playing the game.