2048 – Development Report

Name: Smit Patel

Objective:

The goal of this project was to improve the performance of a 2048 AI agent using only the Rule.py file provided in the base repository. The agent had to be rule-based (not learning-based) and perform well against the leaderboard benchmark, where the professor's best score was around **255,000**.

Development:

1st Code: Basic Rule-Based

Approach:
This version follows a simple greedy rule-based logic that prioritizes moves in a fixed order: Right > Up > Down > Left. It skips moves that don't change the board (stale moves), ensuring better stability. A basic evaluation function was later added to consider factors like the number of empty tiles and whether the highest tile is in the corner.

Performance Summary:

- Highest Score: 6,116
- Average Score (100 runs): ~2,000–3,000
- Typical Range: 1,000 – 4,000
- 2048 Tile Reached: Rarely

Observations:
While extremely simple, this version establishes a reliable baseline. The addition of board-state checks and heuristics made it a more thoughtful version of a static agent, but it still lacks adaptability and long-term strategy.

2nd Code: Simple Greedy

Approach:
This agent evaluates each possible move by simulating it and scoring the resulting board using a weighted heuristic. It chooses the move with the highest score. The depth is fixed at 1 (no recursion or future planning). This one-step lookahead makes the agent responsive and quick, trading complexity for execution speed.

Performance Summary:

- Highest Score: 11,316
- Lowest Score: 776
- Average Score (100 runs): ~4,800–5,400
- Typical Range: 3,000 – 6,500
- 2048 Tile Reached: Occasionally (~8%)

Observations:
This version consistently outperforms static agents due to better move evaluation. It runs fast and is suitable for limited-compute environments but lacks foresight in late-game situations, sometimes leading to premature stagnation.

3rd Code: Greedy with Improved Weights

Approach:
Still greedy and single-step (depth=1), but uses a more refined evaluation function with better-tuned weights. It factors in empty tiles, tile smoothness, directional monotonicity, and whether the maximum tile is in a corner — all with sharper tuning for more intelligent decision-making.

Performance Summary:

- Highest Score: 14,216
- Lowest Score: 776
- Average Score (100 runs): ~4,500 – 5,200
- Stable Range: 6,000 – 8,000+

- 2048 Tile Reached: Occasionally (~5–10%)

Observations:
This Offers a big jump in quality compared to simpler agents. Although it doesn't plan multiple steps ahead, its improved scoring logic allows it to hit higher scores with better consistency. Very effective when the early board is favorable and merges are abundant.

4th Code: Expectimax-Based Agent

Approach:
This version uses a depth-limited Expectimax algorithm (depth=3) that models both player actions (max nodes) and random tile spawns (chance nodes). It combines this with a strong evaluation function that includes gradient-weighted scoring, monotonicity, smoothness, center penalties, and empty tile reward — resembling human-like play patterns.

Performance Summary:

- Highest Score: 34,856
- Lowest Score: 0 (rare, in bad board states)
- Average Score (100 runs): ~13,800
- Typical Range: 14,000 – 17,000
- Frequent High Scores: 25,000 – 33,000+
- 2048 Tile Reached: Frequently

Observations:
Far superior to greedy models, especially in mid-to-late game. Expectimax allows for long-term planning, resulting in more intelligent merges and fewer dead-ends. This comes at the cost of computation, but it's well worth it in most scenarios for a substantial performance boost.

5th Code: Expectimax with Tuned Heuristics

Approach:
An Expectimax agent (depth=3) using a highly tuned evaluation function. Prioritizes empty

tiles, smoothness, monotonicity, gradient placement, and penalizes central clutter. The weights were carefully adjusted to improve board control and long-term strategy.

Changed values for Higher weights on empties (150 to 210), smoothness (1.2 to 2.8), mono (4 to 6.5), gradScore (0.01 to 0.025), Stronger penalty on center tiles (-0.3 to -0.45).

Performance Summary:

- Highest Score: 36,268
- Median Score: ~15,080
- Average (100 runs): ~14,653
- Range: 13,500 – 17,000
- 2048 Tile: Very frequently reached

Observations:
This version consistently outperforms previous agents. It's highly stable, with fewer crashes and smarter merges, making it leaderboard-competitive and ideal for mid-to-late game success.

Conclusion:

This dance describes the gradual evolution of a 2048 agent, beginning from very simple rule-based logic and transforming into a strong Expectimax model. The agent operated under fixed move ordering and employed very simple heuristics and was not very successful. Each revision was aimed to improve upon the evaluation schemes considering smoothness, tile monotonicity, gradient scoring, and penalties at the center. The final Expectimax version at depth 3 and with fine-tuned weights would always reach the 2048 tile and have a median score over 13000+, This is my best among all code that is well-designed heuristics combined with some depth of planning can achieve impressive results even without the need for learning-based methods.