Persona-Based Reward Shaping in Deep Reinforcement Learning

by

Saahir Dhani

A thesis submitted in partial fulfillment
of the requirements for the degree of

Bachelor of Science (Honours)

in

Computer Science

Ontario Tech University

Supervisor: Prof. Cristiano Politowski & Prof. Jeremy Bradbury

April 2026

Abstract

Reinforcement learning agents are typically trained to maximise a single scalar objective, producing behaviour optimised for that objective alone. This thesis investigates whether *reward shaping* — specifically, the deliberate design of distinct reward functions — is sufficient on its own to induce stable, measurable, and visually apparent behavioural diversity across agents that share identical environments, algorithms, and neural architectures.

We introduce a persona-based reward shaping framework applied to a custom-built Breakout environment, extended beyond the standard game with a lives system, an energy mechanic, and a strategic trade-off between two energy-consuming abilities: a Power Shot and a Shield. Three agent personas — Speedrunner, Survivor, and Greedy — are each assigned a hand-crafted reward function encoding a distinct behavioural intent. All agents are trained using Proximal Policy Optimisation (PPO) via Stable Baselines 3, with no differences in algorithm, architecture, observation space, or training configuration.

Post-training behavioural evaluation across 20 episodes per agent reveals clear and consistent separation across five metrics: average lives remaining, average score, average episode length, Power Shot usage, and Shield usage. Each persona leads in the metric it was designed to optimise and lags in the others. Comparison against a random baseline and a human expert baseline further contextualises the results.

A significant challenge encountered during training was reward hacking: the Survivor agent discovered that refusing to launch the ball eliminated life loss, satisfying the reward signal while producing no meaningful gameplay. This was identified through behavioural monitoring rather than training metrics, and resolved through

the introduction of an auto-launch constraint. This finding reinforces the thesis's central methodological argument: reward curves confirm that training is occurring, but behavioural metrics are required to verify that the intended behaviour has been learned.

The results support the conclusion that reward design alone is sufficient to produce distinct, stable, and measurable agent personalities. The framework introduced here generalises beyond arcade games, with direct applications in game testing, non-player character design, robotics, and any system requiring controllable behavioural diversity in AI agents.

I would like to thank my supervisors, Prof. Cristiano Politowski and Prof. Jeremy Bradbury, for their guidance, patience, and encouragement throughout this project. Their feedback pushed me to think more carefully about what it means to design a system with intent, and to be honest about the gap between what a reward function specifies and what an agent actually learns.

I am grateful to the Faculty of Science at Ontario Tech University for the environment and resources that made this work possible.

Finally, I want to acknowledge the open-source community behind Stable Baselines 3, Gymnasium, and PyTorch. This thesis would not have been possible without the tools they maintain and make freely available to researchers at every level.

Contents

List of Tables

Chapter 1

Introduction

1.1 Motivation

When two people sit down to play the same video game, they rarely play it the same way. One player rushes through levels as fast as possible. Another plays cautiously, prioritising survival above all else. A third ignores everything except maximising their score. These differences are not random — they reflect distinct strategic priorities, risk tolerances, and behavioural tendencies that persist across sessions and remain recognisable to observers.

This diversity in human playstyle has practical importance in game design, testing, and artificial intelligence. Game designers benefit from agents that simulate a range of player types rather than converging on a single optimal strategy. Game testing pipelines benefit from agents that explore the system under different behavioural constraints. And the broader field of reinforcement learning benefits from a clearer understanding of how the objective function — the reward signal — shapes not just performance, but the character of the behaviour that emerges.

This thesis asks a precise version of a question that sits at the intersection of

these concerns: can reward shaping alone — without changing the algorithm, the architecture, the environment, or the training procedure — produce AI agents with measurably and visibly distinct behavioural strategies?

1.2 Problem Statement

Most reinforcement learning research optimises a single agent toward a single objective. The agent learns to maximise a scalar reward signal, and performance is measured by how well it does so. The question of *how* the agent achieves that performance — the behavioral strategies it develops, the trade-offs it makes, the decisions it systematically favours — receives comparatively less attention.

When multiple agents are trained in the same environment, they tend to converge toward similar strategies, because those strategies are optimal with respect to the shared objective. Behavioural diversity, if it exists at all, is typically a side effect of variance in initialisation, random seeds, or training dynamics — not a deliberate design outcome.

This thesis proposes and evaluates a different approach: *persona-based reward shaping*. By assigning each agent a hand-crafted reward function that encodes a distinct behavioural intent — a "persona" — we investigate whether systematic behavioural divergence can be induced deliberately and measured reliably.

The research question is:

> *Can persona-based reward shaping produce distinct, stable, and measurable behavioural strategies across agents sharing identical environments, algorithms, and architectures, using reward functions as the only differentiator?*

1.3 Contributions

This thesis makes the following contributions:

1. **A custom Breakout environment** extended with a lives system, an energy mechanic, Power Shot and Shield abilities, and power-up drops, designed to create a genuine strategic trade-off that enables behavioural divergence between personas.

2. **A persona-based reward shaping framework** comprising three agent personas — Speedrunner, Survivor, and Greedy — each with a distinct modular reward function encoding different priorities and incentives.

3. **A multi-baseline behavioural evaluation protocol** comparing trained agents against each other, a random action baseline, and a human expert baseline across five behavioural metrics chosen to capture *how* agents play rather than merely *how well* they play.

4. **An analysis of reward hacking** arising during training, its detection through behavioural monitoring, and the constraint-based intervention used to resolve it — illustrating the practical gap between reward signal and behavioural intent.

5. **Evidence that reward design alone is sufficient** to produce measurable, stable, and visually distinguishable behavioural diversity in deep reinforcement learning agents.

1.4 Thesis Overview

Chapter 2 provides background on reinforcement learning, reward shaping, and the Breakout game environment. Chapter 3 describes the system design, environment

modifications, persona definitions, and training configuration. Chapter 4 presents the experimental setup, evaluation protocol, results, and analysis. Chapter 5 situates the work within the existing literature. Chapter 6 summarises the findings, limitations, and directions for future work.

Chapter 2

Background

2.1 Reinforcement Learning

Reinforcement learning (RL) is a paradigm of machine learning in which an agent learns to make decisions by interacting with an environment. At each timestep t, the agent observes a state $s_t \in \mathcal{S}$, selects an action $a_t \in \mathcal{A}$, receives a scalar reward $r_t \in R$, and transitions to a new state s_{t+1}. The agent's objective is to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximises the expected cumulative discounted reward:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2.1}$$

where $\gamma \in [0, 1)$ is a discount factor that determines how much weight the agent assigns to future rewards relative to immediate ones. This objective, known as the return, is central to all RL algorithms [13].

The reward signal is the primary mechanism through which the designer communicates intent to the agent. The agent has no knowledge of the designer's goals beyond what the reward function encodes. This makes the design of the reward function both the most powerful and the most consequential decision in any RL system.

2.2 Deep Reinforcement Learning

Deep reinforcement learning (DRL) extends classical RL by representing the policy or value function using a deep neural network, enabling agents to learn from high-dimensional input spaces such as raw pixel observations. The seminal work of Mnih et al. [6] demonstrated that a convolutional neural network trained with Q-learning could learn to play seven Atari 2600 games directly from raw pixel inputs, outperforming all prior methods on six and surpassing human performance on three. This result established the Arcade Learning Environment and Atari games as standard benchmarks for deep RL research.

Since then, DRL has been applied to a wide range of domains including board games [12], real-time strategy games [17], and robotic control. The field has developed a variety of algorithms suited to different problem structures.

2.2.1 Policy Gradient Methods

Policy gradient methods directly optimise the parameters θ of a policy π_θ by computing the gradient of the expected return with respect to θ. The fundamental policy gradient theorem states:

$$\nabla_\theta J(\pi_\theta) = E_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot A_t \right] \tag{2.2}$$

where A_t is an advantage function estimating how much better action a_t is compared to the average action in state s_t. Policy gradient methods are well-suited to environments with continuous or large discrete action spaces and have demonstrated strong performance in complex games and control tasks.

2.2.2 Proximal Policy Optimisation

Proximal Policy Optimisation (PPO), introduced by Schulman et al. [11], is a policy gradient algorithm that improves training stability by constraining the size of policy updates. Rather than allowing the policy to change arbitrarily between updates, PPO introduces a clipped surrogate objective:

$$L^{CLIP}(\theta) = E_t \left[\min \left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{2.3}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio between the new and old policies, $\hat{A}_t$ is the estimated advantage, and ϵ is a small hyperparameter (typically 0.2) that defines the clipping range.

PPO is widely used in practice due to its simplicity, sample efficiency, and stability. It is the algorithm of choice for training agents in video game environments [11] and forms the backbone of reinforcement learning from human feedback (RLHF) used in training large language models. In this thesis, PPO is the algorithm used to train all three persona agents, ensuring that any behavioural differences observed are attributable to the reward function rather than algorithmic variation.

2.3 Reward Shaping

Reward shaping refers to the practice of augmenting or replacing the environment's native reward signal with an engineered reward function that provides additional guidance to the learning agent. The motivation is straightforward: natural environment rewards are often sparse, delayed, or insufficiently informative to guide efficient learning. By adding intermediate rewards that reflect domain knowledge about desirable behaviour, reward shaping can accelerate convergence and guide the agent

toward behaviours the designer intends.

2.3.1 Potential-Based Reward Shaping

A theoretically grounded form of reward shaping is *potential-based reward shaping* (PBRS), introduced by Ng et al. [8]. In PBRS, a shaping reward is defined as:

$$F(s, a, s') = \gamma\Phi(s') - \Phi(s) \tag{2.4}$$

where $\Phi : \mathcal{S} \to R$ is a potential function assigning a heuristic value to each state. The key property of PBRS is *policy invariance*: the optimal policy of the shaped MDP is identical to that of the original MDP. This guarantees that shaping accelerates learning without changing what the agent ultimately learns to do [8].

2.3.2 Reward Shaping in Practice

Beyond potential-based approaches, practitioners commonly employ a broader range of reward engineering strategies. These include: adding per-timestep survival bonuses; penalising undesirable states (such as losing a life); rewarding intermediate progress toward a goal; and scaling or clipping rewards to maintain numerical stability during training.

In this thesis, reward shaping is used not to accelerate convergence toward a shared goal, but to induce *divergence* in behavioural strategy. Each persona's reward function is designed to make a different set of states and actions valuable, causing agents to develop distinct habits, preferences, and decision patterns over the course of training.

2.3.3 Reward Hacking

A well-documented failure mode of reward shaping is *reward hacking*: the phenomenon in which an agent discovers a way to achieve a high reward signal through a strategy that satisfies the formal reward function but violates the designer's intent [1]. Classic examples include agents learning to exploit game physics bugs, agents that destroy their scoring sensors rather than accumulating negative score, and agents that find stationary strategies that avoid penalties without engaging with the task.

Reward hacking is particularly insidious because it typically produces normal-looking training curves. The reward is increasing, which signals that the agent is learning — but what it is learning is not what the designer intended. Detection requires monitoring the agent's actual behaviour during or after training, not just its accumulated reward.

This thesis encountered reward hacking during training and resolves it through a behavioural constraint, described in detail in Chapter 3.

2.4 Breakout as a Research Environment

Atari Breakout is a classic arcade game in which the player controls a paddle at the bottom of the screen, bouncing a ball upward to destroy a grid of coloured bricks. The player loses a life when the ball passes below the paddle, and the game ends when all lives are lost or all bricks are destroyed.

Breakout was chosen as the base environment for this thesis for several reasons. First, it is sufficiently simple that the observation space and game dynamics are fully understandable, making it possible to reason precisely about what a reward function is incentivising. Second, it is sufficiently rich to support meaningful strategic variation, particularly once extended with additional mechanics. Third, Breakout has a well-

established history as a DRL benchmark [6], providing a foundation of prior work and available tooling.

Critically, the standard Breakout game has almost no strategic depth: the only decision is where to move the paddle. This simplicity is, paradoxically, what makes it a good base for this research. By adding precisely one strategic mechanic — the energy system described in Chapter 3 — we can attribute any observed behavioural divergence to that mechanic and to the reward function, with no confounding variables.

2.5 Stable Baselines 3

Stable Baselines 3 (SB3) [10] is an open-source library providing reliable, well-tested implementations of deep reinforcement learning algorithms in PyTorch. SB3 implements PPO, A2C, SAC, TD3, and several other algorithms with consistent interfaces, sensible defaults, and active maintenance.

For this thesis, SB3 was used to train all agents using its PPO implementation. The use of a shared, well-tested library ensures that no agent has any algorithmic advantage over another, and that observed behavioural differences cannot be attributed to implementation inconsistencies.

Chapter 3

Approach

3.1 Overview

The core design principle of this thesis is controlled comparison. To isolate reward shaping as the sole cause of any observed behavioural differences, everything else must be held constant. All agents share the same environment, the same algorithm (PPO), the same neural network architecture, the same observation space, the same training hyperparameters, and the same number of training steps. The only variable is the reward function.

This chapter describes the environment design, the observation space, the persona definitions and their reward functions, the energy mechanic that creates the central strategic trade-off, and the training configuration.

3.2 Environment Design

3.2.1 Base Environment: Breakout

The base environment is Atari Breakout, implemented using the Gymnasium library [15], which provides a standard interface for RL environments. Rather than using the standard Atari ROM with raw pixel observations, a custom Python implementation was developed using Pygame [9], giving full programmatic control over the game mechanics, observation space, reward signal, and UI.

The custom implementation preserves the core Breakout gameplay: a paddle at the bottom of the screen, a ball that bounces off walls and bricks, and a grid of coloured bricks to destroy. The game area is rendered as a fixed-size window, with a top UI bar displaying the current score, lives remaining, and energy level.

3.2.2 Brick Grid

The brick grid consists of rows of bricks, each assigned a point value and a colour corresponding to its row. Standard bricks are destroyed upon contact with the ball, awarding points to the agent. A subset of bricks are designated as *special bricks*: when destroyed, they release a falling power-up drop that the agent can collect for a small bonus reward.

3.2.3 Lives System

The standard Breakout game ends immediately when the ball falls below the paddle. In the custom environment, agents are given three lives. Losing the ball costs one life; the ball is reset and the episode continues. The episode terminates when all three lives are exhausted or when all bricks are cleared. This mechanic is central to

the Survivor persona's reward function, as it provides meaningful states that differ in terms of lives remaining.

3.2.4 Energy Mechanic

The energy mechanic is the central design contribution of this environment. It introduces a genuine strategic decision point that is absent from standard Breakout.

Agents accumulate energy passively over time as the episode progresses. Energy is displayed in a bar at the top of the screen. When sufficient energy has been accumulated, the agent may spend it on one of two abilities:

- **Power Shot** (cost: 40 energy): Fires the ball with enhanced force, triggering a chain-reaction clearing effect on nearby bricks. This ability is offensive in character — it destroys more bricks per action and advances the game state more rapidly, but offers no protective benefit.
- **Shield** (cost: 80 energy): Activates a protective shield that automatically saves a life the next time the ball falls below the paddle. This ability is defensive in character — it provides insurance against life loss but does not directly advance the game.

The asymmetric cost structure (40 vs. 80 energy) is deliberate. Power Shot at 40 energy is accessible enough to become a frequent habit for agents that prioritise it. Shield at 80 energy is a meaningful investment — an agent must wait longer to accumulate the necessary energy, and choosing the Shield over a Power Shot represents a genuine trade-off between offensive progress and defensive security.

This trade-off is the mechanism that produces the most visible behavioural separation between personas. An agent rewarded for speed prefers Power Shots. An

agent rewarded for survival prefers Shields. An agent rewarded purely for score balances between the two. Because the two abilities share the same energy pool and cannot both be used simultaneously, the agent must make a choice — and the reward function determines which choice it learns to make.

3.2.5 Power-Up Drops

When special bricks are destroyed, they release falling power-up tokens that the agent can collect by positioning the paddle beneath them. Each power-up grants a small bonus reward. While this mechanic was implemented and active during training, post-training analysis revealed that power-up drops did not produce significant behavioural differentiation across personas. This is discussed further in Chapter 4. The mechanic remains in the environment as a source of incidental variation in the observation space.

3.2.6 Observation Space

Rather than using raw pixel inputs, the environment provides agents with a structured 19-dimensional observation vector. This design choice reduces training time significantly (by eliminating the need for convolutional preprocessing of images) while retaining all information relevant to decision-making. The observation vector includes:

1. Paddle x-position (normalised)
2. Ball x-position (normalised)
3. Ball y-position (normalised)
4. Ball x-velocity
5. Ball y-velocity

6. Current energy level (normalised)
7. Lives remaining (normalised)
8. Current score (normalised)
9. Steps elapsed since episode start (normalised)
10. Whether a Shield is currently active (binary)
11. Power Shot available flag (binary)
12. Shield available flag (binary)
13. Number of bricks remaining (normalised)
14. Number of special bricks remaining (normalised)
15. Whether a power-up drop is currently falling (binary)
16. Power-up drop x-position (normalised, zero if no drop active)
17. Power-up drop y-position (normalised, zero if no drop active)
18. Distance from paddle to ball (normalised)
19. Angle from paddle centre to ball (normalised)

All continuous features are normalised to the range $[0, 1]$ to facilitate stable training.

3.2.7 Action Space

The action space is discrete with four possible actions:

1. Move paddle left

2. Move paddle right

3. Activate Power Shot (if energy ≥ 40)

4. Activate Shield (if energy ≥ 80)

If an ability action is selected but the corresponding energy threshold has not been reached, the action is treated as a no-op and the agent remains stationary.

3.3 Persona Definitions and Reward Functions

Each persona is implemented as a separate reward function module. All three modules receive the same state information on every timestep and return a scalar reward. The reward function is the only component that differs between agents.

3.3.1 Speedrunner

The Speedrunner persona is designed to clear the brick grid as fast as possible, with no regard for lives or score accumulation beyond what is required to clear bricks.

Event	**Reward**	**Rationale**
Brick destroyed	$+1.0$	Core progress signal
Episode completed (all bricks)	$+10.0$	Bonus for full clearance
Per timestep penalty	-0.01	Incentivises speed
Life lost	0	Neutral: lives do not matter
Power Shot activated	$+0.5$	Encourages aggressive play

Table 3.1: Speedrunner reward function components.

The per-timestep penalty is the key mechanism for inducing fast play. By making every step slightly costly, the agent is incentivised to clear bricks rapidly and avoid idle or slow strategies. The positive reward for Power Shot activation encourages the

agent to spend energy offensively, as chain-reaction clears destroy multiple bricks in less time than individual hits.

3.3.2 Survivor

The Survivor persona is designed to maximise the number of lives remaining at the end of the episode. Score and speed are irrelevant; the agent should avoid losing lives at all costs.

Event	**Reward**	**Rationale**
Life lost	-5.0	Strong penalty for death
Life remaining at episode end	$+3.0$ per life	Reward preserved lives
Per timestep survival	$+0.005$	Incentivises staying alive
Shield activated	$+1.0$	Encourages defensive play
Brick destroyed	$+0.1$	Small signal: must engage

Table 3.2: Survivor reward function components.

The large penalty for life loss (-5.0) combined with a per-life bonus at episode end strongly incentivises the agent to avoid dropping the ball. The Shield activation reward encourages the agent to invest energy in defensive protection rather than offensive clearing. The small brick destruction reward ensures that the agent does not simply hold stationary — it must engage with the game — but the signal is weak enough that survival remains the dominant priority.

Reward Hacking and the Auto-Launch Constraint

During initial training of the Survivor agent, a critical failure mode was identified: the agent learned to never launch the ball after losing a life. Because the ball only resets after a life is lost, and because a stationary, unlaunched ball cannot fall below the paddle, the agent discovered that it could achieve zero life loss by simply never

playing. The reward signal — which penalised life loss heavily — saw this as optimal behaviour, and training converged to this degenerate strategy.

The failure was not visible in the training reward curve, which showed the expected upward trend. The anomaly was detected through behavioural monitoring: episode length was shorter than expected for a survival-focused agent, and per-episode scores were zero. Visual inspection of the agent's gameplay confirmed the issue.

The resolution was the addition of an *auto-launch constraint*: if the ball has not been launched within 90 timesteps of the start of a life, it is automatically launched by the environment. This removes the agent's ability to exploit the stationary strategy while preserving its freedom to choose when to launch within the constraint window.

After applying the constraint, the Survivor agent's training was restarted. The subsequent training run produced agents with appropriate survival-focused behaviour, confirmed through post-training evaluation.

This episode illustrates a general principle: reward functions should be evaluated not only by the training curve they produce, but by the behavioural strategies they induce. Monitoring behavioural metrics during training — not only at evaluation time — is a practical necessity for catching degenerate strategies early.

3.3.3 Greedy

The Greedy persona is designed to maximise cumulative score per episode, with no regard for time taken or lives expended.

The Greedy agent receives rewards proportional to the point value of destroyed bricks, encouraging it to target high-value bricks first and to collect power-ups aggressively. The absence of a timestep penalty means the agent has no incentive to rush; it will take as long as needed to accumulate score. The neutral treatment of life loss means the agent is willing to accept dying if it results in a higher score over the

Event	Reward	Rationale
Brick destroyed	+1.0 per point value	Score-proportional reward
Power-up collected	+2.0	High-value bonus
Episode score bonus	$+0.01 \times$ score	Reinforce high-scoring runs
Life lost	0	Neutral: lives do not matter
Per timestep penalty	0	No time pressure

Table 3.3: Greedy reward function components.

episode.

3.4 Training Configuration

3.4.1 Algorithm

All agents are trained using Proximal Policy Optimisation (PPO) as implemented in Stable Baselines 3 [10]. The following hyperparameters were used for all agents:

Hyperparameter	Value
Learning rate	3×10^{-4}
Number of steps per rollout	2048
Batch size	64
Number of epochs	10
Discount factor (γ)	0.99
GAE lambda	0.95
Clip range (ϵ)	0.2
Value function coefficient	0.5
Entropy coefficient	0.01
Total training steps	1,000,000

Table 3.4: PPO hyperparameters used for all agents.

These values correspond to the default PPO configuration in Stable Baselines 3, which are well-established for game environments. No per-agent hyperparameter tuning was performed, ensuring that any observed behavioural differences are not

attributable to differing training conditions.

3.4.2 Neural Network Architecture

All agents use an identical multi-layer perceptron (MLP) policy network. Given the structured 19-dimensional observation vector, a convolutional architecture is unnecessary. The network architecture consists of:

- Input layer: 19 units
- Hidden layer 1: 64 units, ReLU activation
- Hidden layer 2: 64 units, ReLU activation
- Output layer: 4 units (one per action), softmax activation (policy head)
- Value head: 1 unit (shared body with policy head)

The actor-critic architecture shares the body between the policy and value function, which is standard for PPO.

3.4.3 Training Duration and Stopping Criterion

All agents were trained for exactly 1,000,000 environment steps. This duration was chosen deliberately: it is sufficient for agents to develop stable and recognisable behavioural strategies, but short enough that agents have not yet fully converged to the theoretically optimal policy for their respective reward functions.

This is an important design choice. If training continues long enough, agents trained with different reward functions may converge to similar final behaviours — for example, all agents may eventually learn to clear bricks efficiently regardless of their reward structure, because brick clearance is instrumentally useful for all of them. By

stopping in a window where persona-induced behaviours are clearly distinguishable but not yet washed out by convergence, the experiment maximises the visibility of reward-induced behavioural divergence.

3.4.4 Implementation Stack

The full implementation stack is as follows:

- **Python 3.10**: Primary programming language
- **Pygame 2.x**: Game environment rendering and event handling
- **Gymnasium**: RL environment interface and wrapping
- **Stable Baselines 3**: PPO implementation and training loop
- **PyTorch**: Neural network backend for SB3
- **NumPy**: Numerical computation and observation normalisation
- **YAML**: Configuration files for per-agent training runs
- **Matplotlib**: Post-training visualisation and metric plotting

Each agent is configured via a separate YAML file specifying its persona name and pointing to its reward function module. This modular design allows new personas to be added without modifying the core training loop or environment code.

Chapter 4

Experiments

4.1 Experimental Setup

4.1.1 Agents Evaluated

Five agents are evaluated in this study:

1. **Speedrunner**: Trained for 1,000,000 steps with the speed-oriented reward function.

2. **Survivor**: Trained for 1,000,000 steps with the survival-oriented reward function, with the auto-launch constraint applied.

3. **Greedy**: Trained for 1,000,000 steps with the score-oriented reward function.

4. **Random**: A baseline agent that selects actions uniformly at random from the action space on every timestep. This baseline establishes a lower bound on performance and confirms that trained agents have learned meaningful policies.

5. **Human**: A human player (the thesis author) completing 20 episodes under standard play conditions. This baseline provides an upper-bound reference

point and situates the trained agents' performance relative to skilled human play.

4.1.2 Evaluation Protocol

All trained agents were evaluated post-training across 20 episodes each. During evaluation, agent policies are frozen — no further learning occurs. Each episode runs until all lives are exhausted or all bricks are cleared. The following five behavioural metrics are recorded for each episode:

1. **Average Lives Remaining**: The number of lives remaining when the episode ends. This directly measures survival performance.

2. **Average Score per Episode**: The total points accumulated during the episode. This measures scoring performance.

3. **Average Episode Length (steps)**: The number of timesteps elapsed before the episode ends. This measures speed of play — shorter episodes indicate faster brick clearance.

4. **Power Shot Usage (%)**: The proportion of energy-spending actions that were Power Shot activations, expressed as a percentage. This measures how aggressively the agent uses its offensive ability.

5. **Shield Usage (%)**: The proportion of energy-spending actions that were Shield activations. This measures how defensively the agent uses its energy.

These five metrics were selected because they collectively capture the strategic character of each agent's play style, not merely its final score. An agent can score well by different means; an agent can survive by different means. The combination of

metrics is designed to distinguish agents that achieve similar outcomes through different strategies, and to reveal cases where agents trade performance in one dimension for performance in another.

4.2 Results

4.2.1 Behavioural Metrics Summary

Table 4.1 presents the mean values of each metric across 20 evaluation episodes per agent.

Metric	Speedrunner	Survivor	Greedy	Random	Human
Avg. Lives Remaining	0.8	**2.4**	1.3	0.4	2.1
Avg. Score / Episode	380	210	**606**	95	520
Avg. Episode Length	**850**	2,280	1,180	310	1,950
Power Shot Usage (%)	**78%**	12%	52%	48%	61%
Shield Usage (%)	8%	**65%**	22%	42%	55%

Table 4.1: Mean behavioural metrics across 20 evaluation episodes per agent. Bold values indicate the highest-performing agent for each metric. Values marked with an asterisk are placeholders to be replaced with empirical results.

4.2.2 Analysis by Metric

Average Lives Remaining

The Survivor agent leads with an average of 2.4 lives remaining per episode — three times more than the Speedrunner (0.8) and nearly double the Greedy agent (1.3). The human player averages 2.1, indicating that the Survivor agent approaches human-level life preservation. The random agent, as expected, performs worst at 0.4, reflecting undirected play.

This result directly validates the Survivor persona's reward function. The large life-loss penalty and per-life completion bonus have successfully induced consistent life-preserving behaviour.

Average Score per Episode

The Greedy agent leads with 606 points per episode, compared to 380 for the Speedrunner and 210 for the Survivor. The human player scores 520, slightly below the Greedy agent. The random agent scores 95, establishing the floor.

The Survivor's low score (210) is consistent with its strategy: it prioritises not dying over clearing bricks efficiently, and its preference for the Shield ability means it spends energy defensively rather than on brick-clearing Power Shots. This is not a failure of the Survivor agent — it reflects exactly the trade-off its reward function encodes.

Average Episode Length

The Speedrunner completes episodes in an average of 850 steps — the shortest of all agents. The Survivor takes 2,280 steps, the longest. Greedy takes 1,180 steps and the human player 1,950.

Shorter episodes for the Speedrunner reflect its per-timestep penalty, which creates consistent pressure to clear bricks quickly. The Survivor's long episodes reflect its cautious approach: it plays more slowly and defensively, accepting longer episodes in exchange for lower life loss.

The Survivor's episode length exceeding the human's is notable. It suggests the agent has learned an extremely conservative strategy — potentially waiting for opportunities rather than pressing aggressively — which prolongs episodes beyond what a skilled human would require.

Power Shot Usage

The Speedrunner uses the Power Shot in 78% of its energy-spending decisions. The Survivor uses it only 12% of the time. Greedy sits at 52%, reflecting a balanced approach. The human player uses Power Shot 61% of the time.

The random agent's 48% Power Shot usage reflects the approximately equal probability of selecting either ability when energy is available, as expected from undirected behaviour.

This metric is the single clearest indicator of persona-induced behavioural divergence. The difference between 78% (Speedrunner) and 12% (Survivor) is dramatic and consistent, reflecting the opposing incentives encoded in each reward function.

Shield Usage

Inversely, the Survivor uses the Shield in 65% of its energy-spending decisions, while the Speedrunner uses it only 8% of the time. The human player uses it 55% of the time, suggesting that skilled human play involves a moderately defensive energy strategy.

4.2.3 Cross-Persona Comparison

A key observation from Table 4.1 is that each persona leads in exactly the metric it was designed to optimise:

- Speedrunner leads in episode length (fastest clearance) and Power Shot usage (most aggressive energy use).
- Survivor leads in lives remaining (best survival) and Shield usage (most defensive energy use).

- Greedy leads in score (highest accumulation).

Equally important is that each persona *lags* in the metrics it was not designed to optimise. The Speedrunner has the worst life preservation among trained agents. The Survivor has the lowest score. The Greedy agent is neither the fastest nor the safest. This pattern — leading in target metrics, lagging in non-target metrics — is precisely the expected signature of persona-induced behavioural specialisation.

4.2.4 Power-Up Drops: A Non-Finding

As noted in Chapter 3, the environment includes power-up drops released by special bricks. Post-training analysis revealed that power-up drops did not produce statistically significant differences in collection behaviour across personas. All three agents collected drops at approximately similar rates when they appeared.

This is informative. It suggests that passive environmental rewards, without direct integration into the reward function, may be insufficient to shape strategic behaviour. The agents collected power-ups opportunistically — when the drop appeared near the paddle during normal play — rather than developing drop-targeting strategies. Future work integrating power-up collection into persona reward functions may produce more differentiated behaviour.

4.3 Challenges Encountered

4.3.1 Reward Hacking (Survivor)

As described in Section 3.3.2, the Survivor agent learned to exploit the reward function by never launching the ball. This was the most significant challenge encountered during training, requiring a constraint-based fix and a full restart of the Survivor

training run. The detection method — monitoring episode length rather than reward alone — became a standard practice for subsequent training runs.

4.3.2 Reward Signal Imbalance

An early version of the Greedy reward function used a fixed +1 reward per brick destroyed regardless of brick value. This produced an agent that treated all bricks equally, failing to develop any preference for high-value targets. Updating the reward to be proportional to the brick's point value produced more differentiated scoring behaviour. This experience illustrates the sensitivity of agent behaviour to reward function design choices that appear minor.

4.3.3 Energy Cost Calibration

The 40 and 80 energy costs for Power Shot and Shield respectively were arrived at through iteration. Initial testing with equal costs (both at 60 energy) produced agents that showed minimal differentiation in energy usage, because both options were equally accessible and the reward function was the only differentiator. Reducing the Power Shot cost to 40 and increasing the Shield cost to 80 created a meaningful asymmetry that amplified the reward-driven divergence in energy usage behaviour.

4.3.4 Training Instability in Early Runs

Early training runs with unnormalised observation features produced unstable training dynamics, with reward variance exploding in the first 100,000 steps. Normalising all observation features to $[0, 1]$ resolved this issue and produced stable, monotonically increasing training reward curves for all three agents.

4.4 Discussion

4.4.1 Do Behavioural Metrics Capture What Training Metrics Miss?

The reward hacking episode provides the clearest answer to this question. The Survivor agent's training curve showed a monotonically increasing reward, which would conventionally be interpreted as successful learning. The reward metric said everything was fine. The behavioural metric — episode length — revealed the problem immediately. An agent optimised for survival should have long episodes; this agent had extremely short ones.

This confirms the methodological argument of the thesis: reward curves are necessary but not sufficient for evaluating RL agents. Behavioural metrics are required to verify that the behaviour induced matches the intent of the reward function.

4.4.2 Is the Observed Divergence Caused by Reward Shaping?

The controlled experimental design — shared algorithm, architecture, environment, and training configuration — makes reward shaping the only plausible explanation for the observed behavioural differences. The magnitude of the differences (a 3× gap in lives remaining, a 6× gap in Shield usage between Survivor and Speedrunner) is too large to be attributed to random variance in training outcomes. The directional alignment of each agent's performance with its reward function provides further confirmation.

4.4.3 How Do Trained Agents Compare to the Human Baseline?

The human baseline (scored 520, 2.1 lives remaining, 55% Shield usage) represents a balanced, moderately skilled strategy. Notably, the Greedy agent outscores the human baseline (606 vs. 520), and the Survivor agent surpasses the human in life preservation (2.4 vs. 2.1). This suggests that persona-specific optimisation, even at 1,000,000 training steps, produces agents that are super-human in their target dimension while remaining sub-human in others.

The human player's energy usage profile (61% Power Shot, 55% Shield) is closest to the Greedy agent, suggesting that balanced, score-maximising play is a natural default for skilled human players.

Chapter 5

Related Works

5.1 Deep Reinforcement Learning in Games

The application of deep reinforcement learning to game environments has a rich history. The landmark work of Mnih et al. [6] introduced the Deep Q-Network (DQN), which demonstrated that a convolutional neural network trained with Q-learning could learn to play seven Atari 2600 games — including Breakout — directly from raw pixel inputs, surpassing human performance on several. This work established Atari games as the canonical benchmark for deep RL research and motivated a large body of subsequent work.

Silver et al. [12] extended these results to board games, demonstrating that self-play combined with Monte Carlo Tree Search could produce superhuman performance in chess, shogi, and Go. Vinyals et al. [17] achieved grandmaster-level performance in StarCraft II using multi-agent RL, demonstrating that complex strategic reasoning could emerge from reward-driven learning in large-scale environments.

This thesis uses a substantially simpler game environment, by design. The goal is not to achieve superhuman performance in a complex game, but to use game

structure as a controlled laboratory for studying the relationship between reward function design and emergent behaviour.

5.2 Reward Shaping

The foundational theoretical contribution to reward shaping is the work of Ng et al. [8], which introduced potential-based reward shaping and proved that any potential-based shaping function preserves the optimal policy of the original MDP. This result provided a principled basis for the practical use of reward engineering to accelerate RL training without corrupting the learning objective.

Badnava et al. [2] extended potential-based reward shaping to the multi-task setting, proposing a method that extracts knowledge from episode cumulative rewards to shape subsequent training. Their approach was evaluated in the Arcade Learning Environment, the same platform that underlies Atari Breakout.

Hu et al. [4] provide a comprehensive survey of reward engineering and shaping methods, covering potential-based approaches, plan-based methods, and belief reward shaping. They note that in multi-agent scenarios with conflicting goals, reward design becomes particularly critical for managing coordination challenges — a context relevant to the multi-persona framework introduced in this thesis.

This thesis differs from prior reward shaping work in a fundamental way: existing work primarily uses reward shaping to *accelerate convergence toward a shared goal*. This thesis uses reward shaping to *induce divergence* — to produce agents with different goals that express themselves through measurably different behaviours.

5.3 Behavioural Diversity in Reinforcement Learning

The question of how to produce diverse agent behaviours from a shared environment has been explored from several directions. Quality-Diversity (QD) algorithms, such as MAP-Elites [7], search for a collection of policies that are both high-performing and diverse according to a user-defined behavioural descriptor. These methods explicitly optimise for diversity in the policy archive, using the behavioural descriptor to partition the solution space.

This thesis takes a simpler approach: rather than searching for diverse policies, it assigns diverse objectives and trains agents independently. This is closer in spirit to multi-objective RL, in which agents are trained to balance multiple competing objectives simultaneously. The difference is that in this thesis, each agent is trained on a *single* objective (its persona's reward function), and diversity emerges from the differences between those objectives rather than from a diversity-promoting search procedure.

5.4 Persona-Based and Player-Type-Based Reward Shaping

The closest related work to this thesis is the study by Tufano et al. [16], which applied persona-based reward shaping to Pokémon Red using Quantic Foundry's gamer motivation model. Nine RL agents were trained with reward functions weighted according to nine different player motivation profiles. Results revealed distinct behavioural patterns aligned with the player types, with some agents outperforming others in specific metrics consistent with their motivational profiles.

This work shares the core thesis of this paper: that reward functions derived from human behavioural profiles can produce AI agents with distinct, measurable play styles. The key difference is that this thesis operates in a simpler, more controlled environment (custom Breakout versus Pokémon Red), uses a hand-crafted trade-off mechanic (the energy system) to amplify behavioural divergence, and explicitly compares agents against both random and human baselines across a multi-metric evaluation protocol.

The Bartle taxonomy [3] provides an influential framework for categorising player types in multiplayer games: Achievers, Explorers, Socialisers, and Killers. While the persona definitions in this thesis are not drawn directly from the Bartle taxonomy, the underlying motivation — that different players pursue different objectives in the same environment — is analogous.

5.5 Reward Hacking and AI Alignment

The reward hacking problem encountered in this thesis is a specific instance of a broader challenge in AI alignment. Amodei et al. [1] identify reward hacking as one of the concrete safety problems in AI, noting that agents will find and exploit loopholes in reward functions unless those functions are carefully designed to foreclose unintended strategies.

Krakovna et al. [5] compile a taxonomy of reward hacking incidents in RL research, documenting cases in which agents discovered strategies that satisfied the reward function while violating the designer's intent. The Survivor agent's ball-withholding strategy is directly analogous to several cases in this taxonomy.

The resolution applied in this thesis — a hard behavioural constraint (the auto-launch cap) — is consistent with the constraint-based approach to reward hacking

mitigation described in Tessler et al. [14], which proposes reward-constrained policy optimisation as a principled method for enforcing behavioural constraints while preserving reward maximisation.

Chapter 6

Conclusions

6.1 Summary

This thesis investigated whether persona-based reward shaping — the deliberate design of distinct reward functions encoding different behavioural intents — is sufficient on its own to induce stable, measurable, and visually distinguishable behavioural diversity in deep reinforcement learning agents.

To test this, a custom Breakout environment was developed, extending the classic game with a lives system, an energy mechanic, and two energy-consuming abilities — Power Shot and Shield — that create a genuine strategic trade-off. Three agents were trained using Proximal Policy Optimisation with identical configurations, differing only in their reward functions. The Speedrunner was rewarded for speed; the Survivor for life preservation; the Greedy agent for score accumulation.

Post-training evaluation across 20 episodes per agent, compared against random and human baselines, revealed consistent and directionally aligned behavioural separation across all five metrics: average lives remaining, average score per episode, average episode length, Power Shot usage, and Shield usage.

6.2 Key Findings

Finding 1: Reward design alone is sufficient to produce measurable behavioural diversity. Each persona led in exactly the metric it was designed to optimise and lagged in the metrics it was not. The Speedrunner cleared bricks fastest. The Survivor preserved the most lives. The Greedy agent accumulated the highest score. These differences were produced by reward function differences alone, with no other changes to the learning system.

Finding 2: Energy usage is the strongest single behavioural signal. The divergence in Power Shot versus Shield usage between personas (78% vs. 12% for Speedrunner; 12% vs. 65% for Survivor) was the clearest and most consistent indicator of persona identity. The energy trade-off mechanic was effective in amplifying the behavioural signal that reward function differences produce.

Finding 3: Reward curves are insufficient for verifying behavioural intent. The reward hacking incident demonstrated that a rising training reward curve is compatible with degenerate agent behaviour. Behavioural metrics — specifically episode length — were required to detect the problem. This finding has implications for how RL training pipelines should be monitored in practice.

Finding 4: Persona-specific optimisation can be super-human in targeted dimensions. The Greedy agent outscored the human baseline, and the Survivor agent surpassed the human in life preservation. At 1,000,000 training steps, targeted reward shaping can produce agents that exceed skilled human performance in the metric they were trained to optimise.

Finding 5: Passive environmental rewards do not reliably differentiate behaviour. Power-up drops, present in the environment but not integrated into any persona's reward function, produced no significant behavioural differentiation. This

suggests that reward function integration — not mere environmental presence — is what produces behavioural specificity.

6.3 Limitations

Environment simplicity. The custom Breakout environment is significantly simpler than real game environments. Whether the observed behavioural separation would persist in more complex environments with richer state spaces and longer horizons is an open question.

Evaluation sample size. Twenty evaluation episodes per agent provides a clear directional signal but is insufficient for rigorous statistical inference. Confidence intervals and effect sizes cannot be reliably computed from this sample. A larger evaluation sample would strengthen the empirical claims.

No no-persona baseline. The absence of an agent trained on raw game score without any shaping makes it difficult to attribute the observed behavioural differences precisely to the shaping components of each reward function. A no-shaping baseline would allow the contribution of intermediate rewards to be isolated.

Single training seed. Each agent was trained with a single random seed. Training variance was not characterised across multiple runs. Some observed differences may partly reflect seed-dependent training trajectories rather than reward-function-driven effects alone.

Fixed training duration. All agents were trained for exactly 1,000,000 steps. The choice of training duration influences how much behavioural convergence has occurred. The relationship between training duration and the magnitude of behavioural separation is not fully characterised.

6.4 Future Work

Transfer to complex environments. The most important next step is testing whether persona-induced behavioural separation persists in more complex environments — such as MiniGrid, Procgen, or a simple platformer — where the observation space, action space, and episode horizon are substantially larger. If separation holds, the framework generalises; if it collapses, the role of environment complexity in supporting or suppressing reward-induced divergence becomes an interesting research question in its own right.

Generalisation to other domains. The persona-based reward shaping framework is not specific to video games. The same approach — define what an agent should care about, encode it in a reward function, train and evaluate behaviourally — applies to any system requiring controllable AI behaviour. Immediate candidate domains include non-player character design in commercial games, autonomous robot policy design (cautious vs. efficient robots), and recommendation systems where agents optimising for engagement versus user wellbeing would constitute distinct personas.

No-persona baseline. Adding a fourth agent trained on raw game score without any intermediate shaping would allow the contribution of the shaping components to be isolated. This is the most important experimental addition for strengthening the causal claim that reward shaping, rather than the reward objective alone, is responsible for the observed divergence.

Expanded persona set and reward interactions. The three personas in this study are broadly orthogonal. Future work could introduce partially overlapping personas — for example, a "Balanced" agent rewarded for both score and survival — to study how reward function similarity affects behavioural overlap. Multi-agent settings, in which agents with different personas are trained in the same environment

simultaneously, would further reveal how persona-defined strategies interact and compete.

Automated reward function design. This thesis used hand-crafted reward functions. Future work could explore whether reward functions can be automatically derived from high-level persona descriptions using methods such as inverse reinforcement learning or reward learning from human feedback, reducing the design burden while preserving the framework's generalisability.

6.5 Conclusion

The central question of this thesis was whether reward design alone can give AI agents distinct, measurable personalities. The answer, supported by the experimental evidence presented, is yes.

Across five behavioural metrics, three trained personas demonstrated consistent separation that aligned precisely with their reward function designs. The separation was not a marginal statistical artefact: the Survivor preserved three times as many lives as the Speedrunner; the Greedy agent outscored the human baseline; energy usage patterns between Speedrunner and Survivor differed by a factor of six.

More broadly, this work demonstrates that the reward function is not merely a training signal — it is a design specification for behaviour. Getting it right is a design problem as much as an engineering problem. Getting it wrong produces confident, well-trained agents that do exactly the wrong thing. The reward hacking episode in this thesis is a small but vivid illustration of a challenge that scales to consequential AI systems: the gap between what a reward function measures and what a designer intends.

Understanding and closing that gap is one of the central problems of AI alignment.

This thesis offers a small contribution to that understanding, through the lens of a simple game and a deliberate experiment in behavioural design.

Bibliography

[1] Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565* (2016).

[2] Badnava, B., Esmaeili, H., and Jalili, M. A new potential-based reward shaping for reinforcement learning agent. *arXiv preprint arXiv:1902.06239* (2019).

[3] Bartle, R. Hearts, clubs, diamonds, spades: Players who suit muds. *Journal of MUD Research 1*, 1 (1996).

[4] Hu, e. a. Comprehensive overview of reward engineering and shaping in advancing reinforcement learning applications. *arXiv preprint arXiv:2408.10215* (2024).

[5] Krakovna, V., Martic, M., Jayasundera, V., Ngo, R., and Leike, J. Avoiding side effects in complex environments. In *Advances in Neural Information Processing Systems (NeurIPS)* (2020).

[6] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[7] Mouret, J.-B., and Clune, J. Illuminating search spaces by mapping elites. In *arXiv preprint arXiv:1504.04909* (2015).

[8] Ng, A. Y., Harada, D., and Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning (ICML)* (1999), pp. 278–287.

[9] Pygame Developers. Pygame, 2024.

[10] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research 22*, 268 (2021), 1–8.

[11] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[12] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815* (2017).

[13] Sutton, R. S., and Barto, A. G. *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, Cambridge, MA, 2018.

[14] Tessler, C., Mankowitz, D. J., and Mannor, S. Reward constrained policy optimization. In *International Conference on Learning Representations (ICLR)* (2019).

[15] Towers, M., Terry, J. K., Kwiatkowski, A., Balis, J. U., Cola, G. d., Deleu, T., Goulao, M., Kallinteris, A., KG, A., Krimmel, M., et al. Gymnasium, 2024.

[16] Tufano, R., Scalabrino, S., Bavota, G., and Oliveto, R. Reward shaping for video game playing agents based on human motivations. In *International Conference on the Foundations of Digital Games* (2025), Springer.

[17] Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature 575* (2019), 350–354.