ChatGPT

Designing an Optimal Reward System for a Clash Royale Bot

Aligning Rewards with Game Objectives

A Clash Royale bot's reward system (for reinforcement learning) must reflect the game's win conditions. The ultimate goal is **winning the match**, which in Clash Royale means taking down more enemy towers than the opponent (or destroying the King's Tower outright). Therefore, the reward function should give the highest positive reward for winning and a large negative reward for losing [1]. This aligns the AI's incentives with long-term victory. At the same time, intermediate achievements in the game (like damaging towers or gaining an elixir advantage) can be rewarded to guide the bot's short-term decisions [1]. The key is to balance **short-term rewards** (for incremental progress) with the **long-term reward** of winning, so the bot learns strategies that consistently lead to victory rather than just momentary gains.

Rewarding Tower Damage vs. Tower Destruction

Yes – reward both tower damage and tower destruction. In designing the reward system, it's beneficial to give the bot points for **each bit of damage it deals to an enemy tower**, and an additional **bonus for completely taking (destroying) a tower**. Only rewarding final tower kills is very sparse (since a tower may only fall a few times per game at most), so the agent would learn slowly if it only gets feedback upon a tower's destruction. By also rewarding **partial damage**, the bot gets continuous feedback that guides it toward the ultimate goal of taking towers [1].

However, these rewards should be weighted so that **destroying a tower yields a much larger reward** than just a small amount of damage. This ensures the agent prioritizes actually finishing off towers, not just chipping damage. For example, one could grant a modest reward for every percentage of tower health removed, and then give a significant bonus when the tower's HP reaches zero. This way, the bot is encouraged to deal damage consistently and also to focus on fully destroying towers. If a tower has, say, 1000 HP, the bot might accumulate some reward as it brings the tower down to 0, *plus* a hefty reward at the moment of destruction as a "crown" bonus.

Existing research supports this approach. In a 2025 AI analysis, it's suggested that the reward signal include **intermediate rewards for damaging the opponent's towers**, not just the final win [1]. This helps the AI learn which actions lead to progress in taking down towers. In summary, awarding points for both tower damage and tower takedowns is ideal – **damage rewards** provide dense feedback, while **tower destruction rewards** emphasize achieving the main objectives.

Encouraging Defense by Penalizing Tower Loss

A Clash Royale bot must not only attack but also defend. To factor in everything that can happen, the reward system should **penalize the bot when its own towers take damage or get destroyed**. If the AI only gets positive points for offensive actions and no negatives for losing towers, it might adopt reckless strategies (trading towers or ignoring defense) to maximize damage. To prevent this, include negative rewards for events that favor the opponent: - **Own tower taking damage:** a small negative

reward proportional to the health lost (so the bot learns that allowing enemy damage is bad).
- **Own tower destroyed:** a large negative reward when the opponent takes one of your towers (this is essentially the inverse of the reward for destroying an enemy tower).

By incorporating these penalties, the bot's reward reflects the idea that **"the ultimate goal is to take down towers *while defending*"** [2]. In practice, some implementations simplify this by rewarding the *difference* in towers or health between players. For example, one open-source Clash Royale RL agent used a reward function that gives a small bonus if you have more crowns (towers taken) than the opponent and a penalty if you have fewer [3]. This effectively encourages taking towers and discourages losing them. The key is to balance offense and defense: the bot gains points for hurting the enemy's towers, but loses points when its own towers are hurt. This teaches the AI to value **positive tower health differential** – i.e., **maximize damage dealt and minimize damage received**.

Short-Term Rewards vs. Long-Term Success

Designing the reward system also means deciding how much to reward short-term gains versus the final outcome. You mentioned wanting long-term consistency – essentially, a bot that **constantly wins** rather than one that chases immediate points. In reinforcement learning terms, this means the agent should favor actions that lead to winning the game, even if they don't pay off instantly.

To achieve this, make sure the **final win/loss rewards dominate** the reward signal. Winning should yield a very high positive reward, and losing a large negative reward, outweighing smaller intermediate rewards. This ensures that any strategy which fails to win will result in a net negative total reward, no matter how many small points it accrued early. For example, if the bot gets some points for tower damage but eventually loses the match, the large loss penalty should outweigh those points – discouraging strategies that score early damage but lose overall. Conversely, even if the bot has to momentarily sacrifice damage (and hence forgo small rewards) in order to defend and win later, the huge win reward will make that worthwhile.

Modern RL algorithms inherently account for long-term rewards by using a **discount factor** and learning the expected future reward (the value of states) [4]. A well-tuned agent will "know its possibility to win at each time" and take actions that maximize its chance of eventual victory [5]. In other words, as one expert noted, **agents consider the long-term cumulative reward** – a move that isn't immediately rewarding but leads to a win later is still seen as "good" by the AI [4]. We should support this by not over-incentivizing any short-term metric at the expense of the final result. Maintaining a high discount factor (close to 1) will make future rewards (like winning) nearly as important as immediate ones, reinforcing long-term thinking.

"Long-term consistency" in winning also comes from the bot learning effective trade-offs. That's where carefully shaped intermediate rewards help: they act as breadcrumbs that guide the bot toward victory, without replacing the importance of victory. The bot should indeed *figure out the strategy by itself*, but our reward design defines what "success" looks like. By emphasizing the final outcome and giving supportive feedback for the right intermediate steps, we let the AI learn *how* to win consistently on its own.

Incorporating Elixir Trades and Other Factors

Clash Royale has more nuances than just tower health – notably **elixir management and troop exchanges**. An advanced reward system can account for **positive elixir trades** and efficient combat. The idea of "value trading" is that the bot should be rewarded for using fewer resources to counter the

opponent's moves. In practice, this could mean giving a bonus when the bot destroys enemy troops or buildings using a smaller total elixir than the opponent spent. For example, if the bot counters a 5-elixir unit with a 3-elixir card and successfully neutralizes it, that's a +2 elixir trade and could be rewarded.

Online discussions about a Clash Royale AI suggest that adding **short-term rewards for positive elixir trades** can improve learning [2]. This means the bot gets some points for making efficient plays (trading favorably), not just for dealing damage. One way to implement this is to track the **elixir value** of destroyed enemy units minus the elixir value of your own units lost. A positive difference indicates a good trade, which can be rewarded. This encourages the AI to fight efficiently, preserve its troops, and waste the opponent's elixir – behaviors that indirectly lead to long-term success (since an elixir advantage often translates to a successful push or defense).

It's important not to over-complicate the reward with too many variables, but **factoring in elixir trades and troop outcomes** can be helpful. For instance, the bot could receive a small reward for every enemy troop eliminated (especially high-cost troops) and a small penalty when it loses a troop, weighted by elixir cost. This is effectively a proxy for value trading. By "keeping records of total elixir on the battlefield and in hand" during exchanges, the bot can be guided to favorable trades [2].

In addition to elixir, other possible reward factors might include: - **Time/Survival:** A mild step reward for surviving each time step, which encourages the bot to stay in the fight (though be careful – a flat per-step reward can inadvertently encourage stalling. It was used in one project as +1 per step [6], but such a reward should be much smaller than objective-based rewards). - **Special events:** You generally don't need extra rewards for specific events like using a certain card or combo – the bot should learn those through the effect on tower damage or elixir trade. Keep the reward focused on outcomes, to let the bot figure out the tactics by itself (as you said, the bot should learn how to achieve these rewards on its own).

The overarching principle is to include **everything important that affects winning**: dealing damage, protecting your towers, efficient use of resources, and ultimately the win/loss result. This comprehensive reward signal will cover all facets of the game that the AI needs to consider.

Example: Reward Point Distribution Scheme

Bringing it all together, here's a sample reward distribution that factors in all key aspects of Clash Royale gameplay. The numeric values can be tuned, but the relative magnitudes illustrate their importance:

- **Win the match: +500 points** (huge positive reward for victory)
- **Lose the match: −500 points** (large penalty for defeat)
- **Destroy an enemy tower: +100 points** (bonus for each tower taken, on top of damage points earned while bringing it down)
- **Enemy tower damage:** small continuous reward, e.g. **+0.1 point per 1% of tower HP** dealt in damage. (This would sum to +10 points for fully taking a tower's 100% HP, ensuring most reward comes at completion along with the +100 bonus. The bot gains some points as it chips away, but the big payoff is finishing the tower.)
- **Your tower destroyed: −100 points** (penalty each time you lose a tower)
- **Damage to your tower:** proportional negative reward, e.g. **−0.1 point per 1% of your tower HP** lost. (For consistency with the above, losing an entire tower's worth of HP (100%) would total −10 points; the additional −100 is applied if the tower actually falls.)

- **Positive elixir trade: +X points** per elixir advantage in an exchange (for example, if the bot kills a 5-elixir unit with a 3-elixir card, +2 points). Conversely, a **negative elixir trade** could incur a small penalty. This encourages efficient trades [2].
- **Enemy troop eliminated:** optionally, **+1 point per enemy troop destroyed**, especially if measuring elixir directly is hard. Higher-cost troops might be weighted more (or simply rely on the elixir trade calculation above).
- **Own troop lost:** optionally, **−1 point per friendly troop lost** (again weighted by cost). This is indirectly covered by tower damage and elixir loss, but can reinforce that wasting troops is bad.

These values are illustrative. In practice, one would tune them so that a typical match outcome yields a total reward that clearly encourages winning. Notice that winning gives a **far larger reward** than any single tower or damage event – as it should. For instance, even if the bot dealt a lot of damage (say it took two towers, gaining damage rewards and tower bonuses) but then lost the match (opponent took three towers), the −500 for losing plus penalties for lost towers should outweigh its gains, resulting in a net negative reward. On the other hand, a strategy that maybe trades some damage early but wins in the end will net a large positive reward.

For reference, a public Clash Royale RL project used a similar approach: it gave +300 for a win, −200 for a loss, and then additional reward up to +15 or −15 based on the **crown difference** (number of towers taken vs lost) [6] [7]. That means the agent got a moderate bonus for each tower advantage and a penalty for each tower behind, on top of the big win/loss reward. This ensured that taking more towers than the opponent was always rewarded, but ultimately the win reward dominated. Our suggested scheme above expands on that with continuous health-based rewards and elixir trade considerations for an even denser feedback signal.

Conclusion

In summary, the **best reward system for a Clash Royale bot** is one that is **comprehensive and aligned with the game's objectives**: - **Emphasize winning**: give a very high reward for victory and a big penalty for defeat so the bot focuses on long-term success [1].
- **Reward progress**: award points for damaging opponent towers and taking them down, to provide guidance at each step [1]. Similarly, penalize letting your own towers get damaged or destroyed, encouraging the bot to defend.
- **Include efficiency**: if possible, reward positive elixir trades and smart exchanges, since efficient play leads to advantage [2].
- **Balance short-term and long-term**: ensure intermediate rewards (damage, trades, etc.) are not so large that the bot can "game" them without winning. The largest rewards should still come from achieving the win conditions, which fosters long-term consistent winning behavior. As reinforcement learning theory points out, the agent will learn to maximize the **cumulative reward**, so the reward design must make the cumulative reward highest when the bot actually wins the game [4].

By following these principles, the bot's reward system will naturally guide it to figure out the optimal strategy by itself through trial and error. The bot will learn that **damaging towers, protecting its own towers, and efficient card usage all contribute to winning**, because those actions are consistently rewarded in accordance with their true value towards victory. Over time, this should produce a Clash Royale agent that plays with long-term strategy in mind and "constantly wins" as you desire, since every aspect of the reward function drives it toward that outcome.

Sources:

- ReelMind AI Blog – *Clash Royale Gaming: AI Analyzes Mobile Game Strategy* [1] (on using win/loss and intermediate tower damage rewards in RL).
- GitHub (Jaso1024) – *ClashRoyaleRL (PPO Bot)* [6] [7] (example reward function with win, loss, and tower-based rewards).
- Reddit discussion – *"AI learns to play Clash Royale"* [2] [4] (advice on rewarding tower takedowns, defense, and positive elixir trades, and focusing on long-term rewards).

---

[1] Clash Royale Gaming: AI Analyzes Mobile Game Strategy | ReelMind
https://reelmind.ai/blog/clash-royale-gaming-ai-analyzes-mobile-game-strategy

[2] [4] [5] AI learns to play Clash Royale : r/ClashRoyale
https://www.reddit.com/r/ClashRoyale/comments/a6hwlw/ai_learns_to_play_clash_royale/

[3] [6] [7] GitHub - Jaso1024/Real-Time-Strategy-RL-Clash-Royale: A Reinforcement Learning agent for Clash Royale created using Proximal Policy Optimization
https://github.com/Jaso1024/Real-Time-Strategy-RL-Clash-Royale