Touhou Reinforcement Learning with PPO and Curriculum Training

Nathan Chen

August 11th, 2025

1 Abstract

This project explores the application of reinforcement learning to automate gameplay in Touhou Fuujinroku ~ Mountain of Faith, a Bullet-hell (Danmaku) game. I developed a custom Gymnasium-compatible environment that interfaces with the game via screen capture and direct inputs. The training process uses Proximal Policy Optimization (PPO) with a convolutional policy. Additionally, learning was made more efficient using curriculum learning. The reward function was designed to encourage survival and resource collection while penalizing deaths. The trained agent successfully learned fixed-pattern navigation and partial evasion of randomized bullet types, though performance plateaued on sections with attack patterns with high variance. Limitations include possible resolution constraints on recognizing fine bullet details and the computational demands of vision-based RL.

2 Introduction

Bullet-hell games present a uniquely challenging environment for reinforcement learning due to their high-speed gameplay and dense projectile patterns. My objective in this project is to train a reinforcement learning model that could learn routes, discover safe-spots, and find ways to avoid bullet collisions in real time. In doing so, this model would be able to beat Touhou games without losing all of their lives, also known as a one credit clear (1cc). This was a bit ambitious, so I focused on the 10th mainline game Touhou Fuujinroku ~ Mountain of Faith. This title is a bit long so I will abbreviate it to Touhou 10.

Touhou 10 was selected because it features a mix of predictable “fixed” enemy bullet patterns and unpredictable “random” bullet behaviors, requiring the agent to memorize routes and adapt to varying bullet patterns.

This project explores whether reinforcement learning can handle high-precision environments and aims to provide insight into the potential limitations and improvements needed to apply RL to real-time, reaction-intensive tasks.

3 Methodology

3.1 Environment Setup

Originally, the plan was to train the model across all mainline Touhou games. However, after some preliminary testing, it became clear that the variability and visual noise present in the screenshots across different titles significantly complicated the learning process. To maintain consistency and focus on achieving meaningful progress, this project concentrated exclusively on Touhou 10. All training was done using ReimuA shot type and Normal Difficulty.

3.2 Observation Pipeline

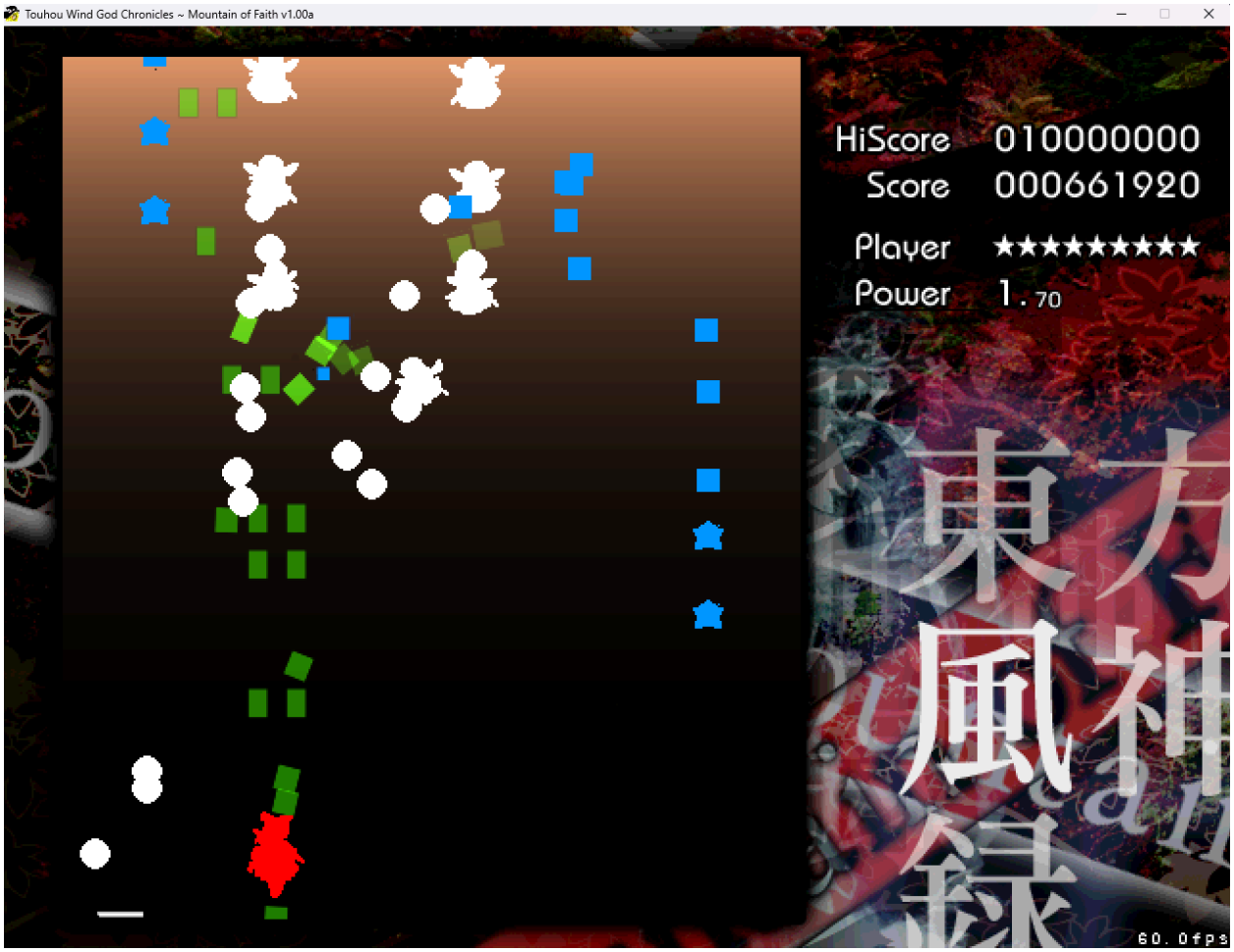
Observations were obtained via real-time screen capture using the mss library. The captured region corresponds to the game’s active play area, cropped and resized to 84x84 RGB frames. In the beginning, these frames were preprocessed using OpenCV into grayscale images, but background elements and HUD overlays resulted in noisy observations. This noise reduced the agent’s ability to detect critical game elements needed to progress in the game.

3.3 Reward Signal Extraction

Optical Character Recognition (OCR) and template matching were used to locate the location of player lives, which I used to shape my reward function. These methods required a lot of testing and preprocessing parameters, mainly thresholding, and was difficult to tune across the different mainline games due to differences in fonts, backgrounds, and UI layouts. This was the point where I decided to focus on Touhou 10.

I ended up patching the game with my own custom shaders to reduce noise in the screenshots. Using an application called Touhou Toolkit to extract relevant files, I applied the following changes:

- Enemy bullets and hitboxes recolored to white
- Backgrounds recolored to black or made transparent
- Player sprite recolored to red
- Player projectiles recolored to green
- Collectable items recolored to blue



One detail I overlooked on my first attempt to create custom sprites was that the laser projectile uses the background pixel values to calculate the color of the laser. A white background would result in invisible laser projectiles. These shaders were applied using a tool called Touhou Community Reliant Automatic Patcher. Additionally, I used CheatEngine

to find memory addresses of player lives and player power to make a more reliable and efficient way to shape the reward function.

3.4 Curriculum Learning Strategy

Up until now, training has been done using story mode, which starts the game with two extra lives. To improve efficiency of learning and encourage stable policy learning, a curriculum learning approach was adopted. The agent would be initially trained on the six stages of Touhou 10 independently. Upon achieving a performance threshold, the environment would progress to the next stage. As the environment progresses, the stages would get more challenging. Additionally, separating the stages using practice mode would give the model more leniency in training because the session starts with nine extra lives. Stage transitions were implemented by detecting the game-over screen using template matching.

3.5 Reward Design

Here is the reward function that resulted in the best survival time for my experiments:

- Small positive reward for each timestep survived (+0.1)
- Moderate positive reward for collecting power-ups (+0.3)
- Significant penalty for losing a life (-10)
- Significant positive reward for gaining a life (+10)

3.6 Algorithm and Hyperparameters

The model was trained using the Proximal Policy Optimization (PPO) algorithm implemented in Stable Baselines3, with a convolutional neural network (CNN) policy to process image-based observations. The final hyperparameters that were experimented with were:

- learning_rate - step size for gradient updates: 5.0×10^5
- n_steps - number of environment steps before an update: 512
- batch_size - mini-batch size for gradient descent: 256
- n_epochs - number of passes over the rollout buffer per update: 20

- gamma - discount factor for future rewards: 0.99
- gae_lambda - smoothing parameter for Generalized Advantage Estimation: 0.95
- ent_coef - entropy regularization coefficient to encourage exploration: 0.01

3.7 Training Infrastructures

Touhou 10 accepts only a single source of DirectInput, preventing the use of parallel environments. Consequently, all training was conducted in a single-environment configuration. Additionally, since Touhou games are not readily available on macOS or Linux machines, these experiments may be difficult to reproduce on such devices.

To capture bullet projectiles and player movement, frame stacking was implemented using Stable Baseline3's VecFrameStack wrapper. Model checkpoints were saved at fixed intervals, and an automated crash detection system was implemented to restart the game and resume training without manual intervention. Since Touhou titles are natively Windows-based and lack official support for macOS or Linux, the experiments were conducted exclusively on a Windows machine.

4 Results and Observations

Training the PPO agent on the modified Touhou 10 environment produced noticeable improvements in survival-orientated behaviors over time. After sufficient training, the agent began demonstrating a learned route through stage 1 of Touhou 10. The agent demonstrated deliberate positioning, such as moving to specific safe spots during certain attacks.

The custom visual modifications significantly improved the clarity of significant visuals for the agent to learn from. However, there were still some visuals that may have caused some difficulty for the agent to learn. There were some gradients in the background that may have not been included in the sprite and background sheets that I could not find a way to remove. These gradients occasionally obscured the player's hitbox, likely impairing the agent's ability to accurately detect the player's hitbox.

Additionally, performance declined during boss attack patterns that featured random bullet trajectories. In these cases, the agent often remained stationary in either the

bottom left or bottom right corner in an attempt to stay alive. This behavior was likely due to two factors, the unpredictable nature of these patterns, which prevented memorization of a fixed route, and reduced visual clarity from downscaling screenshots to 84x84 pixels for processing efficiency.

While the model did not achieve the ultimate goal of completing the game without losing all lives, it successfully demonstrated a learned strategy for predictable enemy patterns.

5 Conclusion

This project explored the application of reinforcement learning to the bullet-hell game Touhou 10, focusing on training an agent to navigate complex projectile patterns and survive through early stages. By customizing the game's visuals and employing a PPO algorithm, the agent demonstrated the ability to learn effective routes for predictable bullet patterns and improve survival behavior over time.

However, limitations in handling random boss attack patterns and visual noise, partly due to downscaled input resolution, prevented the agent from achieving a one credit clear. These findings highlight the challenges of applying reinforcement learning in high-speed, visually complex environments.

Future work could explore higher-resolution inputs or improvements to curriculum learning techniques (maybe using thprac to skip directly to certain boss attack sequences) to better handle unpredictable patterns. Despite the challenges, this project demonstrates the feasibility of reinforcement learning approaches for navigating and surviving in bullet-hell games.

6 References

1. Li, Raymond C.; Ahn, Jun Min; Esteron, Zachary Tyler; and Hong, Qiyin, "Collision Avoidance with Deep Reinforcement Learning" (2019). Purdue Undergraduate Research Conference. 59. https://docs.lib.purdue.edu/purc/2019/Posters/59

2. Johnny Code. (2025, April 10). *Simply Explaining Proximal Policy Optimization (PPO): Full Whiteboard Walkthrough*. Youtube. https://www.youtube.com/watch?v=5VHLd9eCZ-w