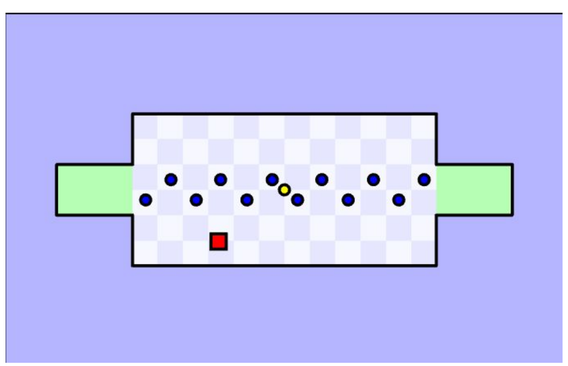
Vision-Driven RL for Legacy Games: A Case Study in The World's Hardest Game

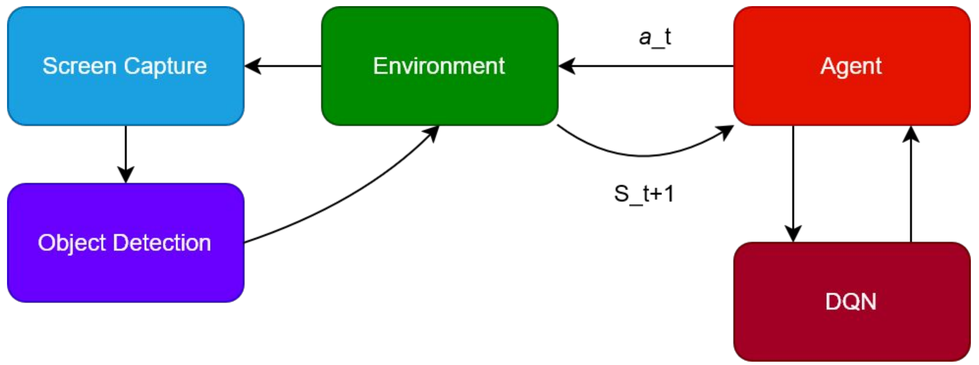
Flash Fanatics
[1]Jason Lee, [1]George Mularadelis, [1]Dhruv Prasanna, [1]Dr. Hao Wang
[1]Stevens Institute of Technology

Code

Introduction

- Most existing video game reinforcement learning solutions utilize full game cloning, due to the lack of source code access.
- Classic Flash-based games, such as *The World's Hardest Game*, pose unique challenges for Reinforcement Learning (RL) due to its lack of structured APIs or internal state information.
- This project uses computer vision techniques, like color masking, to provide input data for the learning model, allowing the learning process to utilize the native game.
- The results of this project can extend outside the space of legacy games into more complex video games, such as real-time multiplayer games, without the need for emulation.





Design

- The environment takes input screenshots and interprets them using color masking to find each type of object that needs to be accounted for.
- The objects are fed as a state vector into the agent's Deep-Q Network to obtain an action which will be taken
- The model then inputs the action to the environment, and the process loops until the goal state is reached a sufficient number of times
- The reward structure incentivises moving quickly outside of the starting zone, then to any coins, then to checkpoints, including the goal.
- Enemy proximity is tracked relative to the player and punishments are dispersed for getting too close and being reset to the start.
- A time-based punishment is also applied to ensure the model tries to reach the goal in a timely manner

Evaluation

- The model generally attempts to leave the starting area, but exercises great caution in the process
- The model always goes for the coin and then the checkpoint but struggles dodging the enemies
- The reward structure is quite complicated to make due to the overlapping colors specifically of the goals, checkpoints, and starting point

Conclusion

- We created a RL model that takes a screenshot every few frames and detects all the essential elements of the game and makes decisions based on their locations
- Our reward system requires more tuning and refinement to efficiently move to the goal
- We used Q-Learning exclusively, however, an Actor-Critic model with policy gradients may perform better due to its more explorative nature
- The inability to have multiple actors in a single game makes training more time consuming, unless multiple emulators are used in parallel
- The baseline idea of this project can be transferred to other games that have inaccessible APIs or are complicated to clone for RL purposes