Monte Carlo Tree Search (MCTS)

One of the main challenges in developing a Swing Copters player is the ability to control the avatar's movement. All actions produce changes in the avatar's position, which can influence the outcome of the game. To be successful, a player must plan ahead.

We wanted to factor in these delayed actions in our model by using swing copter actions. We trained MCTS to consider the current state of the game and make decisions based on the expected outcomes of various actions. This approach allowed us to improve the swing copters' performance and achieve better results.

Another aspect of Swing Copters that makes MCTS a worthwhile approach is the combination of the player's success in achieving the final goal. A player using MCTS can improve their performance by avoiding repeatedly playing games. In Swing Copters, although random play will not lead to long-term success, it can give a good idea of the likelihood of short-term survival.

The MCTS algorithm produced a Swing Copter player. The player regularly improved through the levels of the game, but was not consistently successful and needed more work than a benchmark bot player. Scoring improved slightly with more simulated games. However, the computation time increased drastically, making it impractical as a way to play the game in real-time.

Avatar Detection

Originally, we thought to make a player that could play Swing Copters on an improved algorithm using a series of heuristics from the game's code. This was not effective and the bot could not learn to play well. However, we developed a novel method of learning the game's states, and the bot was able to learn to play well.

Q-Learning

Another approach we took to make Swing Copters playable was to use reinforcement learning. The biggest challenge in this approach was coming up with useful features for the player.

Features

- Rewarding terminal states
- Choosing a distance to gap
- Avoiding obstacles

The player learns to choose actions that maximize the expected reward at each step. This approach allowed us to improve the Swing Copters' performance and achieve better results.

Our Version vs Original Game

In our version of Swing Copters, we improved the gameplay by making the levels more challenging and introducing new obstacles. The result was a more engaging and exciting game experience for players.

Key Questions to Develop Useful Features:

- Do we have enough time to stop before hitting the wall?
- Do we have enough time to stop before passing the platform gap?
- Do we have enough time to avoid the hammer?

The features that improved our results significantly were indicators on the avatar's 'stopping distance' from a wall: how close to a wall the avatar would be if the player tried to stop immediately. With the combination of these features, our player's performance was comparable to that of the MCTS player.

Next, we added a feature indicating whether the avatar could stop before making contact with the gap without hitting the wall. This feature helps the player to avoid getting stuck in the walls and allows them to progress through the game more smoothly.

Finally, we implemented a set of additional features, which allowed our Swing Copters player to achieve higher scores and beat the benchmark.

Features

- Stopping distance from wall
- Stopping distance from platform

We tested these features using different versions of Swing Copters and found that they were effective in improving the performance of the player.

Once we implemented a set of appropriate features, we had great success using Q-Learning. Our player now learns better ways to play and to survive indefinitely.

Monisha White and Nolan Walsh