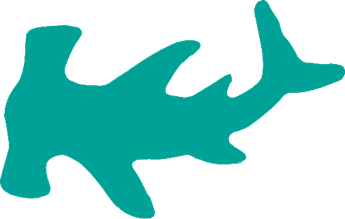
Alex Laurnet, Nick Bolles, David Dowling, Jon Scott | Introduction to Artificial Intelligence | December 12, 2015

Hammerhead Tron AI

[Report Subtitle]



# Introduction:

Motivation:

The motivation behind this project was to get a better understanding behind gaming AI’s. Gaming is an ever-growing industry gaining a more and more competitive job market. As programmers, we must understand the basics of gaming AI programs to open the paths to more advanced and complex implementations of enemy, friendly, and cooperative AI development. This, of course, makes us more desirable and marketable as future employment seekers. The motivation behind the specific Tron game was that the concept of this game was easy to conceptualize on a macro level. This is, the idea and rules of the game were simple. This would allow us to build a more complex AI around the simple concept.

Define the problem:

On a finite grid, two players move forward constantly and are able two change their direction (north, east, south, and west) to avoid or engaged their opponent. No space may be traversed twice (so players leave a wall or trail as they go) and the game is over when one player crashes into either a wall or a previously traversed space. Our goal was to implement an AI for a user to play against. The point of the AI would be to give an opponent that does not simply move at random. So our questions were as follows:

1. What strategies can we use to implement an AI?
2. How can we turn those conceptual strategies into working code?
3. How can we determine a “good” AI from a “bad” AI in this particular case?

# Approach:

General Approach:

Before any coding could happen, we needed to clearly understand strategies of the game (question 1. from above). Many games were played in order to come up with a few plausible strategies. The two mains ones are to either box your opponent in and force them to crash into your trail, or block them off in a section smaller than your own section, and force them to eventually crash into their own trail before you run out of spaces yourself. For this project, we modified a version of Tronament, which is a Tron game that allows for AI module plugins. All AI was coded in JavaScript.

The result of answering our first question resulted in the idea of several states for the AI, which would be controlled by the master AI. Those states are chase, bail, and attack. The idea was for the AI to start of defensively, and switch to become more aggressive when the conditions were right. The chase state is more defensive as it will have the computer player find and mimic user player movement until an opportunity to switch to be more aggressive. The bail state has the computer player move away from the user player and towards the least occupied area (least occupied mean most safety). The attack state is more aggressive. In this state, the computer player will attempt to box in the player (described above) or cut them off (a strategy dependent on human error). All states use a modified A\* search algorithm with an input starting location and goal location.

Computational model:

Implementation:

Evaluation:

A strong AI will win often and quickly. We used this notion to define a successful Tron AI player. That is, a successful AI will have many wins numerically (those wins will be despite an attempt to win by the user) and those wins will be as short in length as possible.

# Results:

# Conclusion: