Tron AI Proposal

Summary: In the interest of incorporating multiple AI concepts, we’re going to want to build a very dynamic AI with multiple states that has an overarching Intelligence that can “read” the other players actions and react appropriately. Each state will be a sort of technique in or action common to the game Tron and I will list suggested ones below, as well as a concept for our overall intelligence, and some utility functions we’ll need to use.

Utility Functions

* Find – Find should only need to be run once. However, because Tronament is designed to run the AI’s JS files every instance we will need to build in an if statement with a Boolean to determine if we have already run its initial course or not. Find’s core functionality is to find the opponents current position. The initial run will scan the board until the opponent is found, and from there the player can only move one direction per turn, so we will only need to check those 3 positions after the initial run. This is obviously important because we will be using the opponents position to determine patterns, and ultimately what state to enter
* Scan – Scan will query a certain dimension (maybe 5x5) square around the AI to determine what’s surrounding it. This will run every time to make sure there is nothing we’re about to run into.
* Dodge – if scan pulls up any lines 2 or 3 blocks away the we deem dangerous, Dodge will be activated and will override any state functions in the interest of keeping the AI alive (unless Trapped is our current state)

State Functions – State functions will be determined by the master AI by a string return.

* Chase(curr, dest) – chase will take our AI’s current position and finds the shortest distance to that point, and makes a move accordingly.
* Cut() – Cut will attempt to cut off a certain area of the grid cutting down the size of the playing field by traveling a line down the spot you wish to cut. This one may be tough to develop, but could prove useful in stopping the opponent from escaping.
* Surround\_loose(point) – surrounds this point in a 12x12 square in attempt to trap the player. Costs time, but takes longer to escape
* Surround\_tight(point) – same as Surround\_loose but with an 8x8 square, less time consuming, but easier to escape from
* Evade() – this will basically set the AI to run from the other player while making some sporadic turns to fool them
* Trapped() – AI will move in an outward spiral to survive as long as possible as it is trapped in a box.

Master AI – The Master AI will analyze players position and patterns in relation to its own position to determine what state to enter. The only function that can override the Master AI is dodge, which is essentially our emergency escape function.

* Master AI will begin by scanning and initiating dodge if necessary
* If dodge is not initiated, Master AI will look at the opponents position, previous moves, and its own current position to determine a state (this is where the fun theory comes in  )
* Once a state is chosen, Master AI returns it in string form, or just calls the function. Not sure how we will structure it yet, but basically at this point it lets the function take care of the rest
* Tronament has a message screen. We can utilize this for testing by printing the state chosen

A logical example of how our AI will work:

1. AI and OPP enter game, let’s say OPP has the first move, they move one block closer to AI
2. AI runs it’s initial scan, but takes 2 turns to actually determine current position, so we’re in a static state
3. OPP moves closer once again
4. AI’s second scan reveals OPP’s current position, determines we’re in no immediate danger so dodge is not initiated, notices It’s a significant distance from OPP’s position, enters chase state.
5. 10ish turns go by as both players approach each other, they’re about 10 blocks away, AI notices and enters cut state in an attempt to close off space, limiting OPP’s escape routes
6. OPP responds by maneuvering down the same path but is 3 squares in before reacting. They continue for 3 more turns
7. AI re-enters chase state to close in on OPP
8. OPP is aggressive and follows suit moving closer to AI in an attempt to cut it off last minute
9. Noticing OPP is now in range, AI enters Surround\_tight state wrapping OPP in a box. OPP reacts a few turns later, but it is too late. AI surrounds OPP, then enters evade state to survive as OPP attempts to survive as long as possible.

Granted I provided no logic as to how the Master AI will make this decision, and the Opponent in this example isn’t the brightest, it’s sort of the logic we want to bring into this AI