Snake Player

Ethan Larkham

Problem: Snake Arcade Game

- The "snake" starts three units long
- Can be steered along the cardinal directions, if no input is provided it will follow it's last given trajectory
- When it "consumes" apples by touching them it's length increases by one
- After an apple is consumed, a new one is randomly generated in a different location
- If the snake touches itself or hits a wall, the game ends
- Goal: Consume as many apples as possible, if you manage to fill the entire grid then you win the game

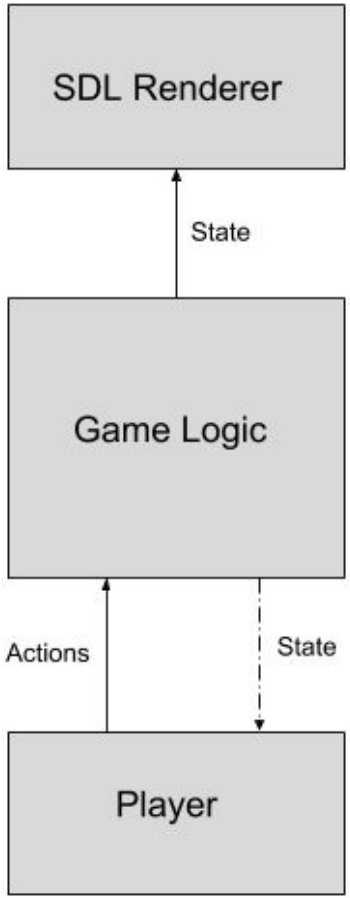
Gameplay

As an example of snake being played by a very skilled player, this is a popular gif demonstrating someone actually beating the game.

Since its unfortunately over 4 minutes long, game completion won't be demonstrated.

Implementation

- Main thread handles game logic, loops with a timed delay
- Sends game state through go-channels at the end of each loop
- Receives player actions via go-channels, if no action is received before timeout, snake trajectory is maintained



Algorithms

Modified A*

- After every apple is consumed, a new path is generated to the next one
- If no path is found, snake stalls by moving to random children until it creates an opening
- If algorithm is too slow, game logic thread will timeout and continue the snake on its current trajectory

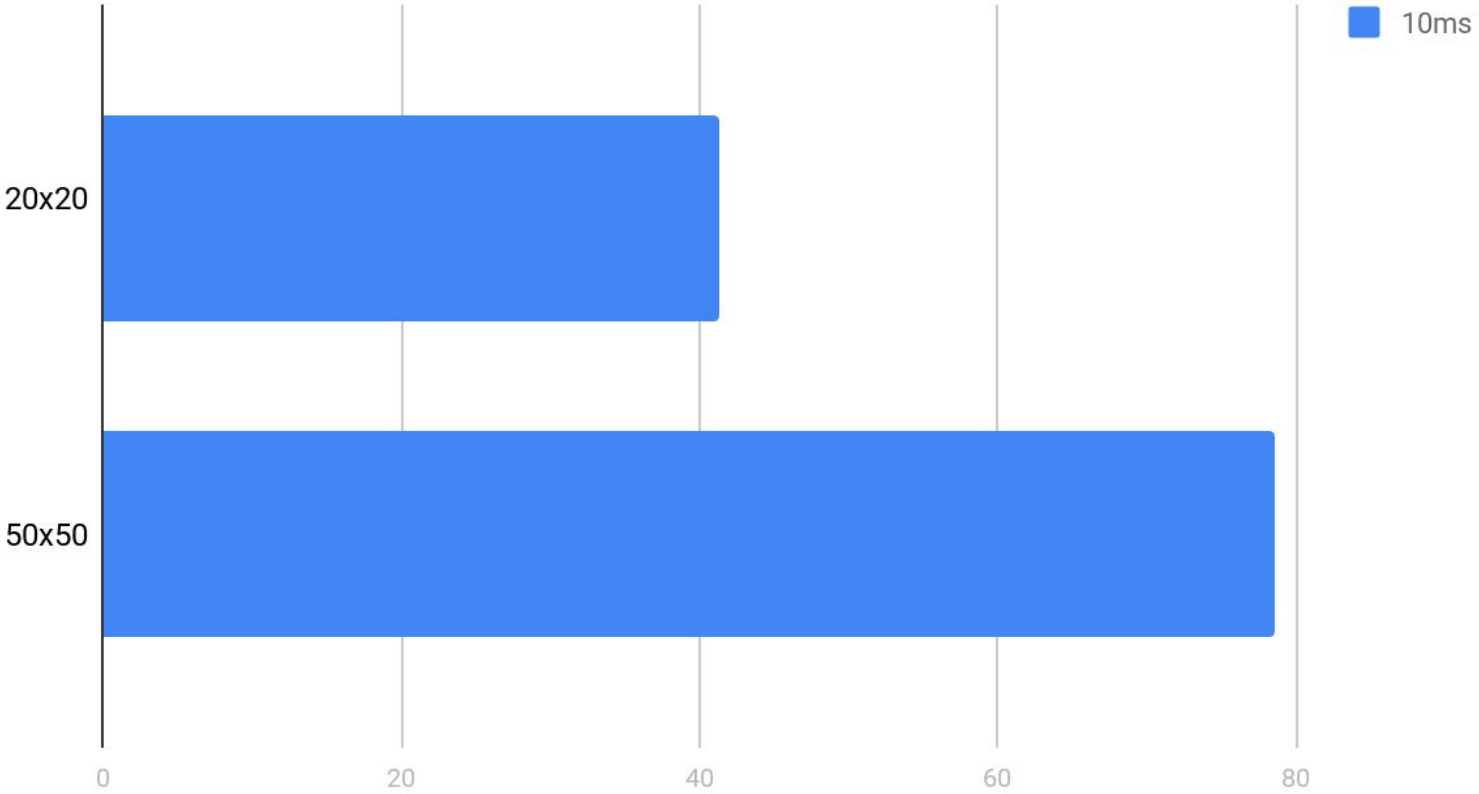
Greedy Real Time A*

- Gets the depth-1 best scoring child
- Learning techniques were impractical since the snakes every moment changes board state
- Can work at any speed and never times out

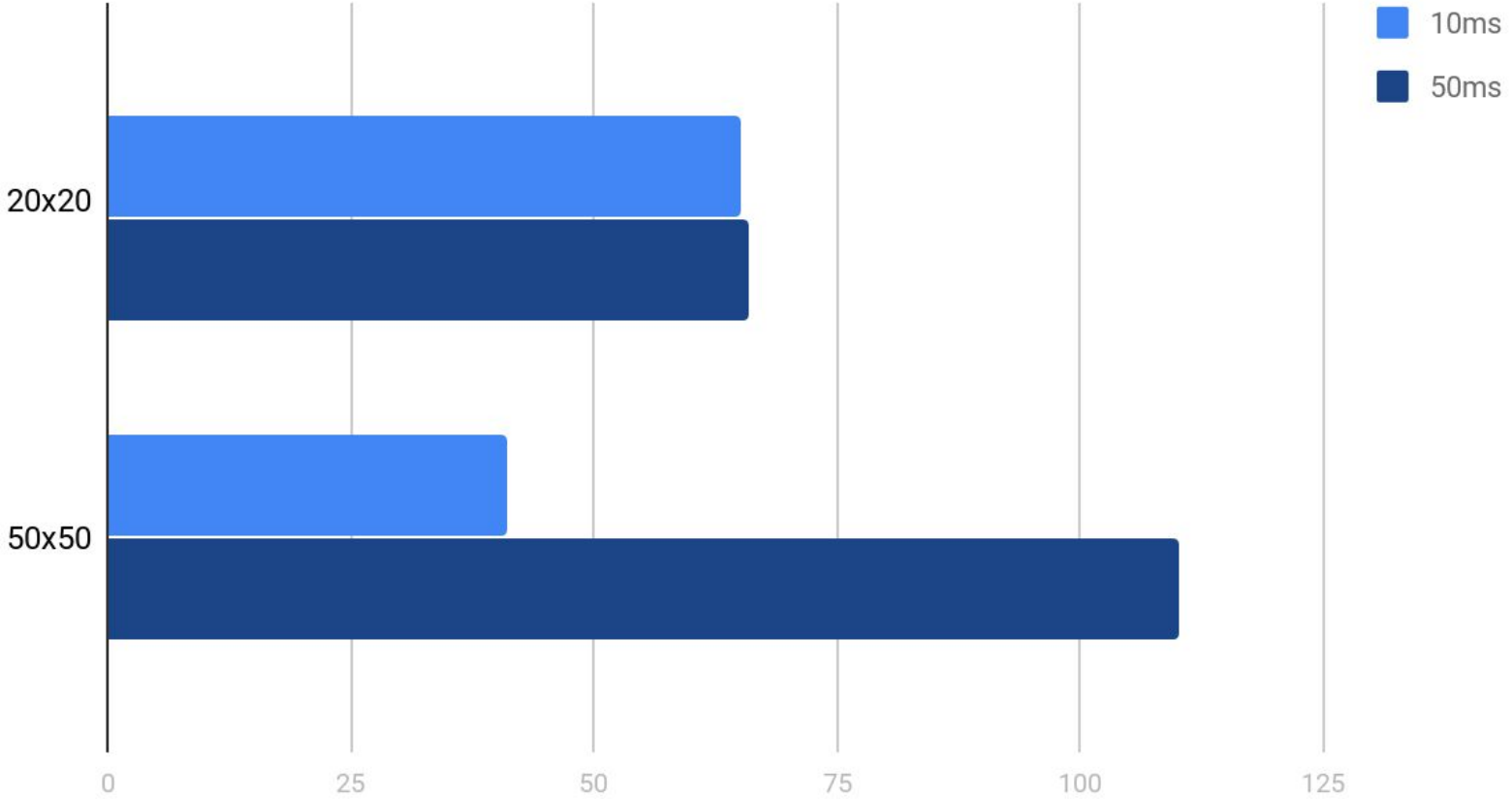
A* Shared Heuristics

- Manhattan distance to apple
- Manhattan distance to tail, weighted less than apple
- Distance from all four grid walls, negatively weighted with low scale

RTA* Performance



A* Performance



Possible Extensions

- Hamilton Cycles
- RTA* Tweaks
- D*
- JSON output