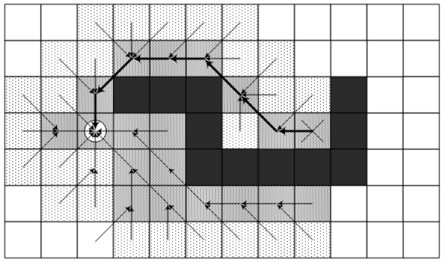
A*

- Use best of Djikstra and Best-First
- Both heuristic cost (best first - estimate) and given cost (Djikstra - actual) to pick next node from open list

Estimated Final Cost = Given Cost + (Heuristic Cost * Heuristic Weight)



(Avoids Best-First trap!)

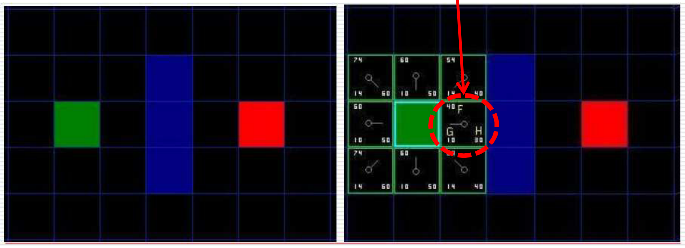
A* Internals (1 of 3)

- Green: start
- Red: goal
- Blue: barrier

G: 10 for ver/horiz, 14 for diag

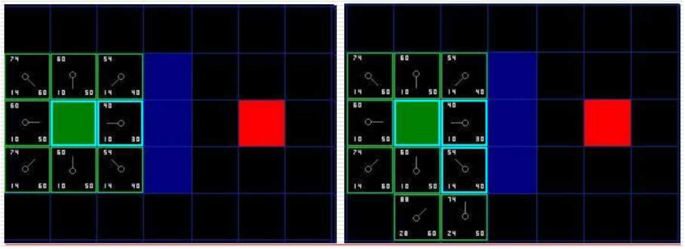
H: “manhattan distance” to dest * 10

F: Estimated “cost” (G+H)



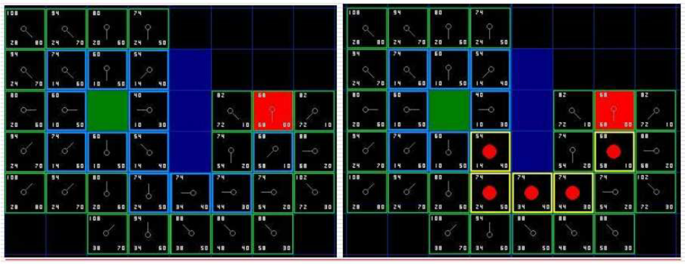
A* Internals (2 of 3)

- Now check for the lowest **F** value in OPEN
  - In this case NE, SE both 54, so randomly choose SE
- Going directly to SE is cheaper than E->SE
  - Leave start as the parent of SE, and iterate



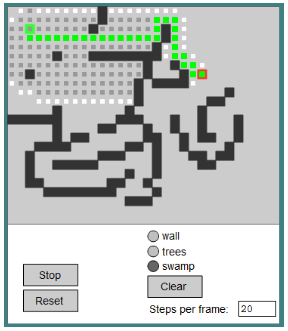
A* Internals (3 of 3)

- Keep iterating until reach goal and OPEN is empty
- Follow parent links to get short path



(Note: crossing corners of objects not allowed)

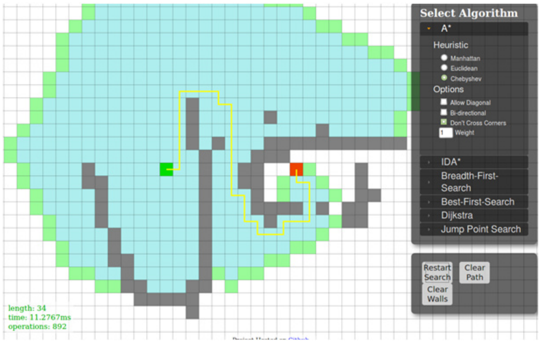
A* Demo (1 of 2)



Steps per frame: 20

http://www.antimodal.com/astar/

A* Demo (2 of 2)



length: 34
time: 11.2767ms
operations: 892

http://qiao.github.io/PathFinding.js/visual/

A* Characteristics

- On average, uses fewer resources than Dijkstra and Breadth-First
- Heuristic search
  - "Weight" can control how much to value heuristic (H)
    - If 0 then like Dijkstra
    - If large then like Best-First
- "Good" heuristic guarantees it will find optimal path
  - "Good" as long as doesn't overestimate actual cost
  - For maps, "good" is "as a bird flies" distance (best-case)
- *Complete* algorithm