The A*-Algorithm

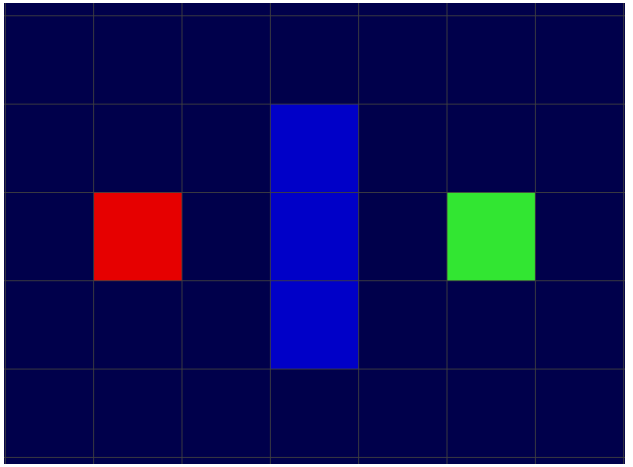
Samuel Arzt

(based on an article by Patrick Lester: *„A* Pathfinding for Beginners“*)

Essentials

- **Goal:** Find a fastest path from A to B in a field.
- **Starting point:**
  - Field (two dim. array) with different field-codes (e.g.: Path = 0, Obstacle = 1)
  - Start/Target field of the path

```
byte[,] fieldCodes = new byte[5, 7]
{
    { 0, 0, 0, 0, 0, 0, 0 },
    { 0, 0, 0, 1, 0, 0, 0 },
    { 0, 0, 0, 1, 0, 0, 0 },
    { 0, 0, 0, 1, 0, 0, 0 },
    { 0, 0, 0, 0, 0, 0, 0 },
};
```



A*-Basic structure

- Add the start field to the „*Open List*“.

1. Take the „cheapest“ element off the Open List and add it to the „*Closed List*“.
2. Calculate the cost of all neighbours and add them to the Open List and remember their „*Parent*“.
3. As long as the target is not in the Open List and the Open List is not empty, continue with step 1.

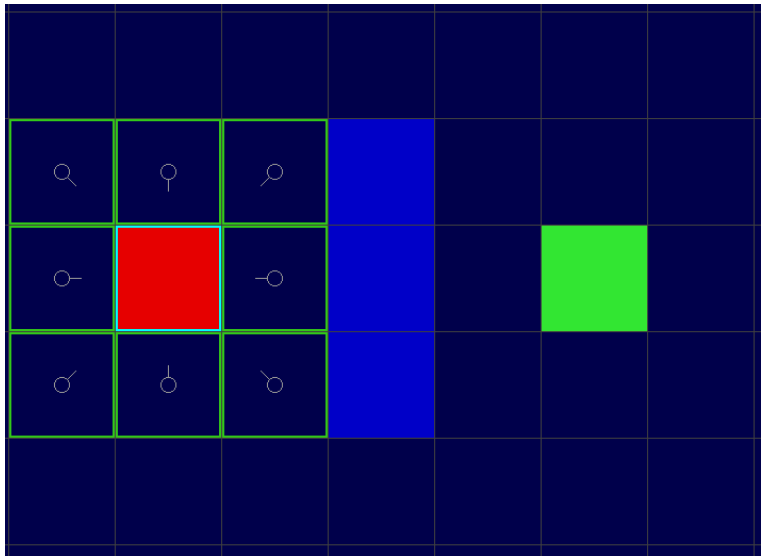
Open/Closed List

- Open List:

  Basically fields that are considered for the path.

- Closed List:

  Fields that do not need to be examined anymore.



Cost calculation

- Formula $\mathbf{F}_{Cost} = \mathbf{G}_{Cost} + \mathbf{H}_{Cost}$
  - G-Cost:

    The cost of walking from the start to the current field.

    E.g.: 10 for all straights, 14 for all diagonals.

  - H-Cost:

    „Heuristic“: Estimated cost from current field to target.

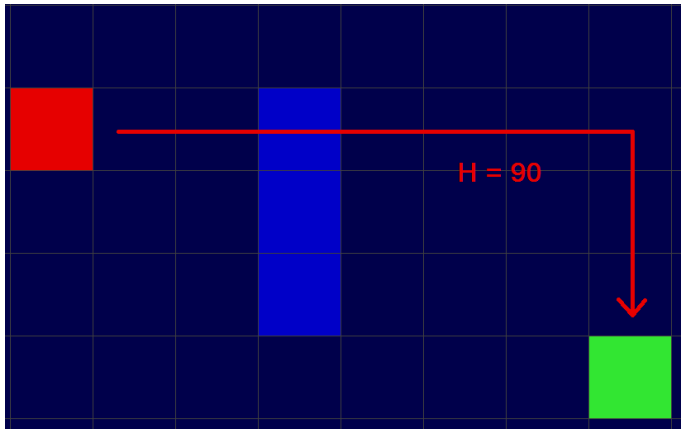
    Different methods: *Manhattan, Diagonal, Euclidean.*

Cost calculation: Manhattan-Method

- Most simple/naive heuristic:

  Ignore field-type (whether path or obstacle) and simply calculate the cost as if you could walk straight to the target (no diagonals).

```
//Calculate hCost with manhattan method
int hCost = STRAIGHT_COST * (Math.Abs(nbX - targetPosition.X) +
                             Math.Abs(nbY - targetPosition.Y));
```



Other Heuristics: http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html

Example cost calculation:

- Example: first „Neighbour“ right/lower right:

  G = 10 (Straight)

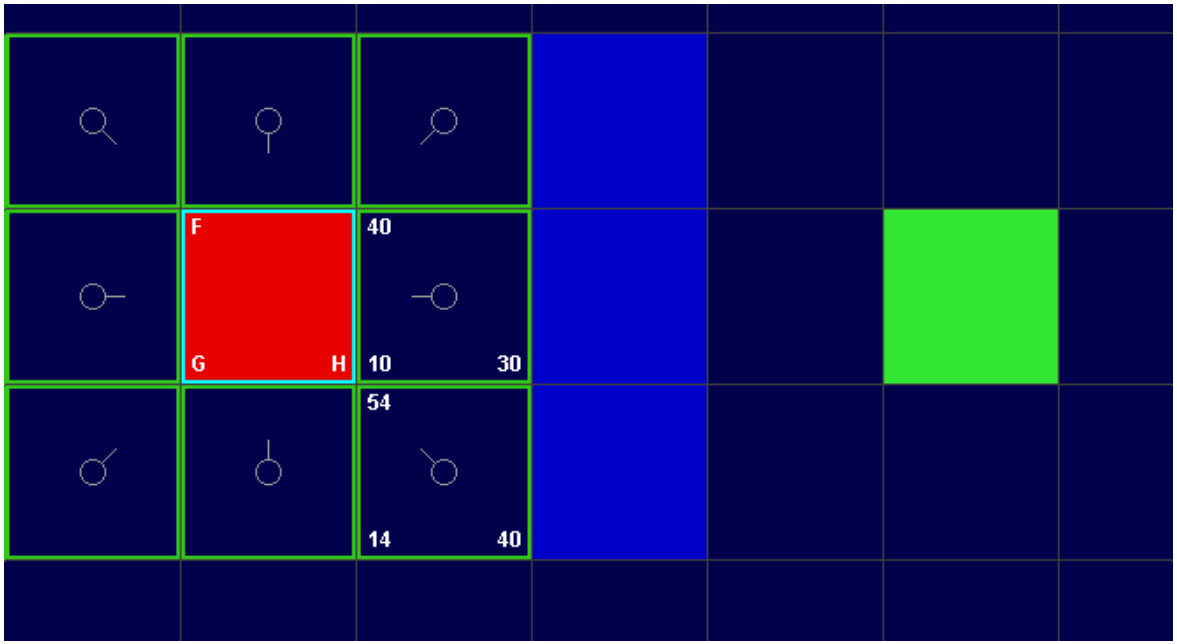
  H = 30

  F = G + H = 40

  H = 40

  G = 14 (Diag.)

  F = G + H = 54



A*-Basic structure

- Add the start field to the „*Open List*“.

1. Take the „cheapest“ element off the Open List and add it to the „*Closed List*“.
2. Calculate the cost of all neighbours and add them to the Open List and remember their „*Parent*“.
3. As long as the target is not in the Open List and the Open List is not empty, continue with step 1.

A*-Structure in detail: Open List

- Open List: Element with lowest costs is required in every iteration (=> Priority-Queue)
- Best implementation: Binary-Min-Heap (Dequeue/Enqueue in O(log(n))!)

A*-Basic structure

- Add the start field to the „*Open List*“.

1. Take the „cheapest“ element off the Open List and add it to the „*Closed List*“.
2. Calculate the cost of all neighbours and add them to the Open List and remember their „*Parent*“.
3. As long as the target is not in the Open List and the Open List is not empty, continue with step 1.

A*-Structure in detail: Neighbours

- Examine all 8 neighbours of the current field, as long as they are **walkable** and **not on the Closed List** yet:
  - Calculate costs.
  - Add to Open List and remember parent.
  - If already on Open List recalculate the costs and update them and parent if cheaper.

A*-Basic structure

- Add the start field to the „*Open List*“.

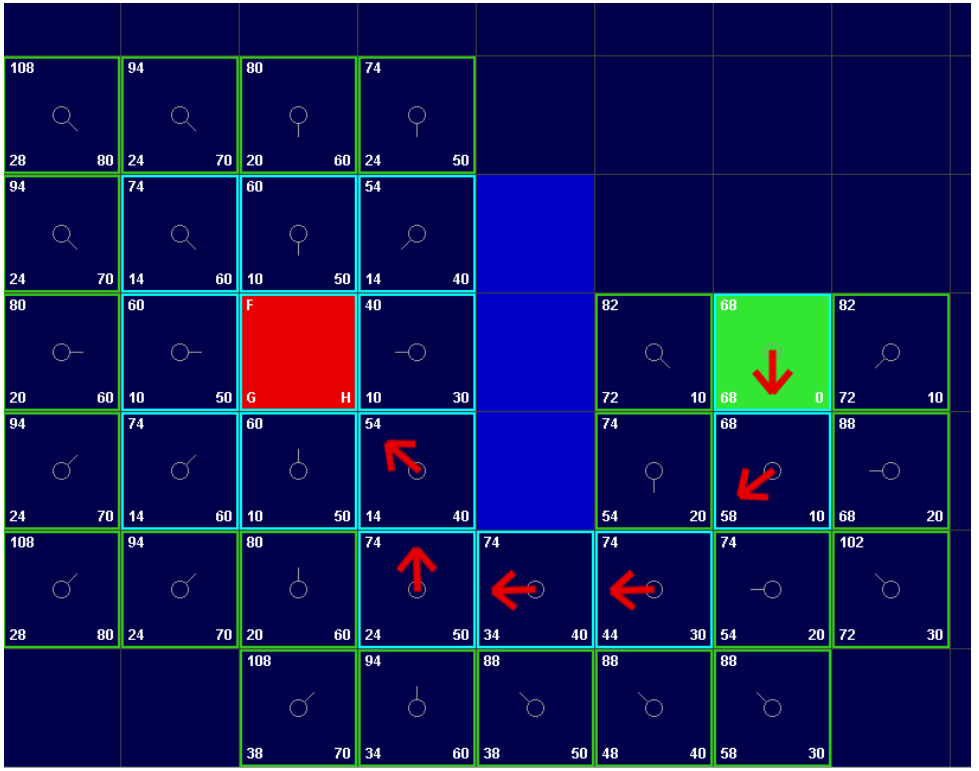
1. Take the „cheapest“ element off the Open List and add it to the „*Closed List*“.
2. Calculate the cost of all neighbours and add them to the Open List and remember their „*Parent*“.
3. As long as the target is not in the Open List and the Open List is not empty, continue with step 1.

A*-Structure in detail: Break conditions

- If the target is on the Open List, the path has been found:

  What is the path?

  Trace parents backwards:

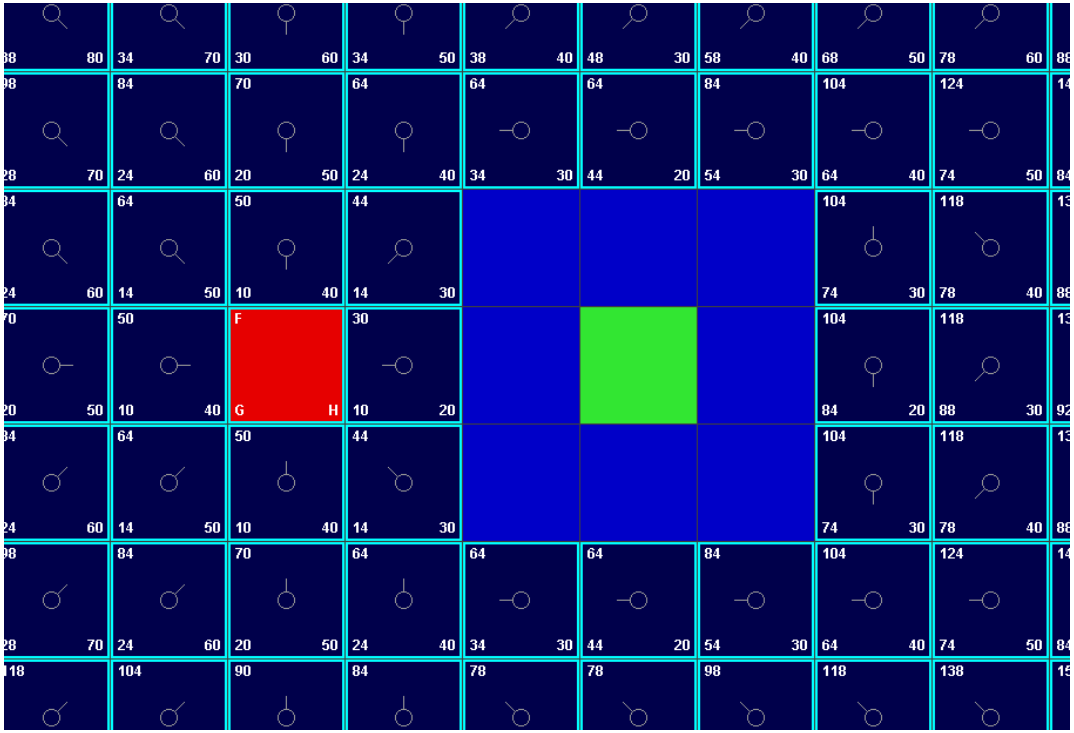


A*-Structure in detail: Break conditions

- If the target is not on Open List, but the List is empty, then there is no path:

  Target unreachable:

  (= Worst Case,
  all reachable fields
  will be examined.)



A*-Demonstration

- Source-Code
- Java Demo
- „Blocked“