Technische beschrijving individueel onderdeel
Algoritmen en Heuristieken
Amy Koelman (13977784)
case: Chips & Circuits
17-01-2025

Ons groepje is op dit moment bezig met meerdere algoritmes uit te werken en functies toe te voegen om de output van onze oplossingen te verbeteren. Voor mijn individuele onderdeel ga ik een A* algoritme uitwerken. Dit algoritme kan verzekeren dat voor elke connectie de kortste route wordt genomen, vandaar dat dit een goede oplossing kan zijn voor onze case.

Om dit te implementeren ga ik een nieuwe class aanmaken (hieronder in een UML te zien) die de make_connection() method zal gebruiken om vanaf een start gate stappen te zetten totdat de eind gate is bereikt. Dit zal voor elke connectie worden gedaan, waarbij rekening wordt gehouden dat er maar één draad per gridsegment mag bestaan, maar dat de connecties wel zo kort mogelijk blijven.

In de UML is te zien wat er in de __init__ method aangemaakt zal moeten worden. Dit bestaat uit een open_list (nodes die nog verkend moeten worden), een closed_list (nodes die al verkend zijn en waar het algoritme niet meer naar om kijkt), een start en end node (coordinaten van de start en eind gate van de connectie), en als laatst een lijst met occupied segments (gridsegmenten die al worden gebruikt door andere connecties en die dus niet meer belopen kunnen worden).

De make_connection() method zal het A* algoritme implementeren. Hiervoor is een Pseudocode onderaan te zien. In dit algoritme wordt elke keer de beste volgende node geselecteerd volgens een cost functie. Deze is in de vorm van f = g + h, waarin h een heuristiek is die de een schatting geeft van de afstand tussen de huidige node en de eind node, en g de afstand is tussen de huidige node en de start node (N.T. een node is in de context van ons grid een coordinaat (x, y, z)). De node met de laagste cost word geselecteerd om de volgende stap naartoe te zetten (tenzij het gridsegment van deze stap al bezet is door een andere draad).
Om de cost waarde van een node te berekenen worden de methodes distance_h(), distance_g() en cost_f() gebruikt. Om g en h uit te rekenen is het nodig om twee functies (een heuristiek en een manier om de kortste afstand te berekenen) te gebruiken die het beste bij onze case passen en te schrijven in de distance_g() en distance_h() methods.

Deze week ga ik dus een A* class aanmaken, de Pseudocode in python code schrijven om de make_connection() method te maken, een heuristiek en kortste afstand functie vinden en programmeren om de distance_h() en distance_g() methods te maken, en als laatste de waardes van deze functie gebruiken in de cost_f() method (die in de make_connection() method wordt gebruikt).

*A*Algorithm*
closed_list: list start_node: coordinate (x,y,z) end_node: coordinate (x,y,z) end_node: coordinate (x,y,z) open_list: list occupied_segments(): list
distance_g(): int distance_h(): int cost_f(g, h): int make_connection(open_list, closed_list, start_coor, end_coor): list

Pseudocode A* pathfinding algorithm[1]:

1. Initialize an empty path list
2. add the start node to the open list
    a. f value of this node can be equal to 0
3. loop until end node/location is reached → while open list is not empty:
    a. get the current node (first node in open list)
        i. Equal to the node with the lowest f value)
        ii. remove current node from open list and add to closed list
        iii. add node to path list
    b. if current node equals the end node: found path!
        i. return path list
    c. generate list of child nodes
        i. the adjoining nodes (nodes directly besides the current node)
    d. for each child in the children list:
        i. if child on the closed list or not walkable from current node (already occupied by other connection)
            - continue to the beginning of the loop (next child)
        ii. Create the f, g, and h values for the child and calculate f.
        iii. if childs' position is in the open list nodes positions:
            - if child's g value is higher than the open list' node g value
                a. continue to beginning of the loop.
        iv. if not in open list: add child to open list

[1] Swift, N. (2020, 29 mei). Easy A* (star) Pathfinding - Nicholas Swift - Medium. *Medium*. https://medium.com/@nicholas.w.swift/easy-a-star-pathfinding-7e6689c7f7b2