Event Matching Algorithm

Situering

Dit bestand bevat de uitleg over hoe we testen of een punt op een route ligt die gedefinieerd is door waypoints. Het hele algoritme kunt u vinden de map "backend/src/main/java/be/ugent/backend/businesslayer/businnesworkflow/akka/algorithms".

Gegeven een lijst van events en users, gaan we per event na of het event relevant is voor een user. Vooraleer we kijken of het event op de route ligt, kijken we eerst of het event voldoet aan allerlei constraints (een event van het type "weer" zal bv. niet relevant zijn voor een route die enkel events van het type "file" wil ontvangen). Als het event voldoet aan al deze constraints, zal pas het echte algoritme opgeroepen worden.

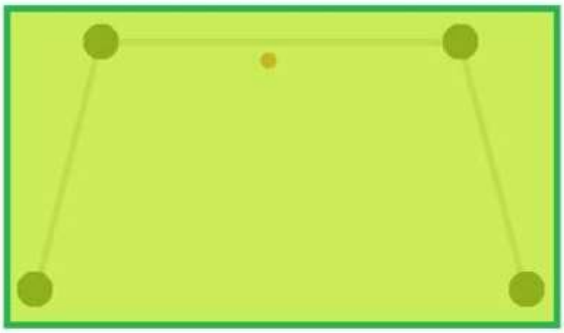
Omzetting coördinaten

Het algoritme zelf gebeurt in verschillende stappen, alvorens de stappen besproken worden, is het belangrijk te weten dat alle punten (zowel de waypoints als de events) in lengte- en breedtegraden beschreven worden. Aangezien het rekenen hiermee enorm moeilijk is, worden deze eerst omgezet naar cartesische coördinaten. Hierdoor zetten we alle coördinaten, uitgedrukt in graden, om naar coördinaten uitgedrukt in meter. Om dit te doen gebruiken we de formule beschreven op deze link: https://en.wikipedia.org/wiki/Geographic_coordinate_system#Expressing_latitude_and_longitude_as_linear_units. Deze formule houdt met veel meer rekening dan de meeste formules, het is aangeraden dit eens door te nemen als u er meer uitleg over wilt. Deze omzetting gebruikt een referentiepunt, hiervoor hebben wij de coördinaten van de Kouter in gent genomen. Dit stelt het nulpunt in ons nieuw assenstelsel voor, hierdoor zullen de x- en y-coördinaten nooit te hoog worden waardoor er minder afrondingsfouten kunnen voorkomen.
De onnauwkeurigheid van dit algoritme is te wijten aan het feit dat de omzetting van de coördinaten niet gepaard gaat zonder fouten. Dit komt omdat er met enorm veel factoren rekening gehouden moet worden (zoals de hellingsgraad van de ondergrond, de positie ten opzichte van de horizon…).

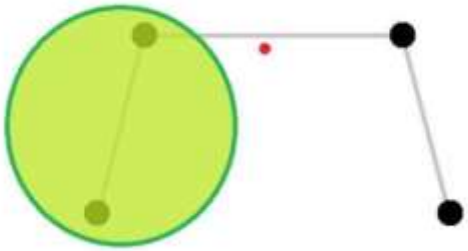
Uitleg algoritme

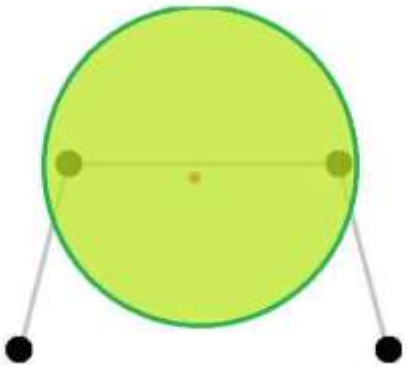
Om deze uitleg wat meer visueel voor te stellen, zullen we een versimpeld model gebruiken (zie figuur hieronder), de zwarte punten stellen waypoints voor en het rode punt is het event dat we willen matchen. De grijze lijnen zijn denkbeeldige lijnen die de route wat duidelijker zichtbaar maken.

In de eerste stap van het algoritme wordt er op grote schaal gefilterd. We doen dit door alle punten van een route te overlopen en te kijken welk rechthoek al deze punten bevat (door het maximum en minimum van deze punten bij te houden en een kleine marge toe te voegen). Als de coördinaten van het event niet in het vierkant liggen, zijn we al zeker dat dit event niet gematcht kan worden voor deze route. Zo voorkomen we, al vroeg in het algoritme, veel overbodig rekenwerk. De figuur hieronder visualiseert dit proces: het rode punt zit in de rechthoek dus wordt het algoritme niet vroegtijdig gestopt.



In de volgende stap worden waypoints opeenvolgend per koppel overlopen. Per koppel wordt het middelpunt bepaald om vervolgens te kijken of het punt in de cirkel ligt die beschreven wordt door dit middelpunt en de halve afstand van de 2 punten + een kleine marge (de marge die we telkens gebruiken is 10m). Al deze berekeningen zijn exact en maken gebruik van de wiskunde (formule voor afstand tussen 2 punten). Dit alles is mogelijk aangezien we de coördinaten hebben omgezet in de vorige stap. De bedoeling hiervan is om na te gaan of het punt in de buurt van 2 waypoints ligt. Als dit het geval is kan naar de 3^{e} en laatste stap van het algoritme overgegaan worden, anders worden de volgende 2 waypoints overlopen.
De volgende 2 afbeeldingen tonen dit proces, het event ligt niet in de buurt van de eerste 2 waypoints dus worden de volgende 2 in achting genomen. Hier zien we dat het event wel in de cirkel ligt, dus zal er naar de volgende en laatste stap overgegaan worden.





In de laatste stap wordt er gekeken of de afstand tussen het punt en de rechte, bepaald door te 2 waypoints, kleiner is dan onze marge (10m). Als dit zo is, dan weten we dat het punt dichter dan 10 meter van de 2 waypoints ligt en beslist het algoritme dat het punt op de route ligt. Moesten we de cirkels in de vorige stap niet gebruikt hebben, dan kon het zijn dat het punt wel in de buurt van de rechte zou liggen, maar niet in de buurt van de punten aangezien de rechte tot het oneindige wordt doorgetrokken in de wiskundige formule. De volgende stap visualiseert dit proces: zoals u ziet, wordt het groene gebied doorgetrokken tot het oneindige, maar aangezien wij weten dat het punt in de cirkel ligt weten we dat het punt weldegelijk in de buurt ligt.

Complexiteit

Als *n* het aantal waypoints van een route is, dan zal het testen van 1 event een complexiteit hebben van $\Theta(n)$. In zowel het beste als slechtste geval is deze hetzelfde, er is wel een constante waar rekening mee gehouden moet worden. In het beste geval geraakt het algoritme niet in de 2^e stap en zal de constante zeer laag zijn (punten omzetten gebeurt in constante snelheid + overlopen van alle punten zijn slechts enkele vergelijkingen). De 2^e en 3^e stap zijn wel wat zwaarder maar ook hier gebeuren er *n* berekeningen van constante tijd, maar hier zijn er wel vierkantswortels/machtsverheffingen en die zijn een pak intensiever dan slechts enkele vergelijkingen.