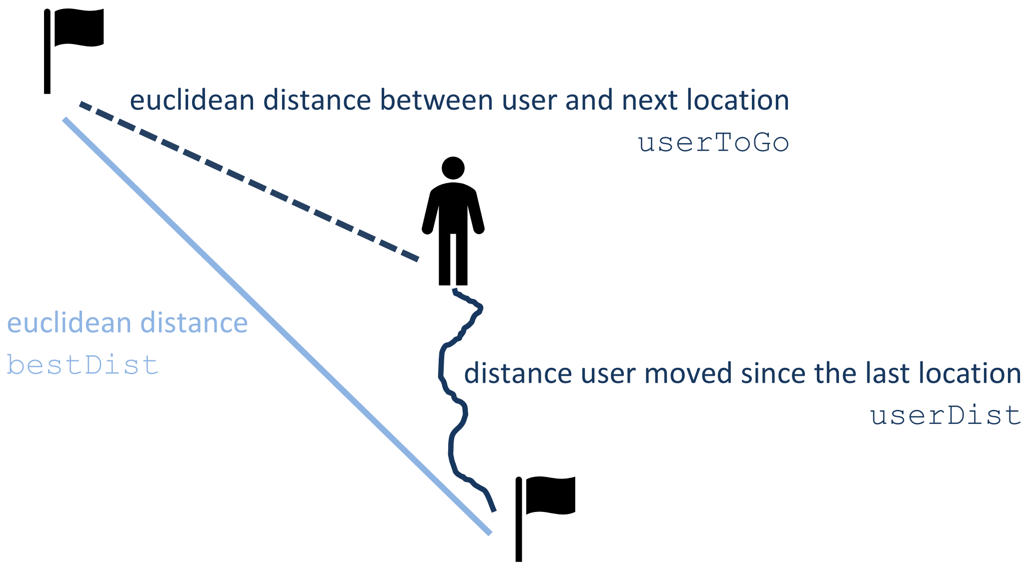
Detecting a wrong direction

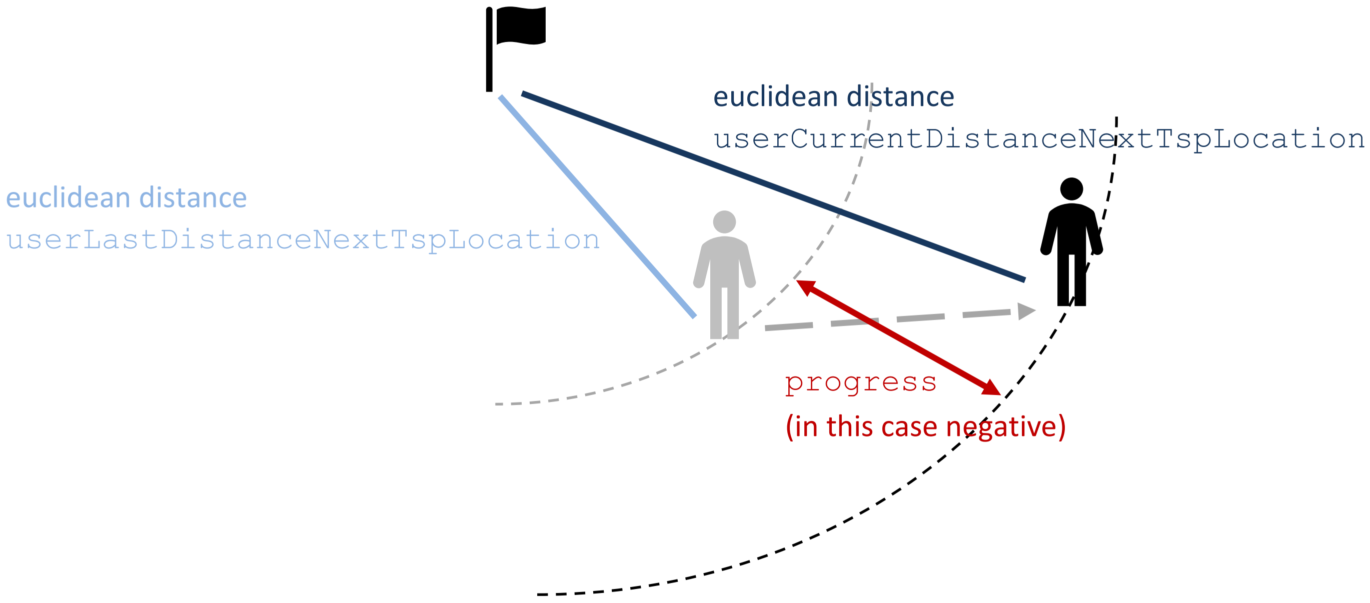
1. rate criterion


euclidean distance between user and next location
userToGo
euclidean distance
bestDist
distance user moved since the last location
userDist

```
userBestDist = userDist + userToGo
rate = bestDist / userBestDist
```

If **rate** < VARIATION_OF_PATH_ACCEPTED, then the **rate criterion** will be set to true.

2. negative user movement criterion



```
progress =
userLastDistanceNextTspLocation - userCurrentDistanceNextTspLocation
```

Initially, the **`negativeUserMovement`** is zero (and it is always clamped to zero if it is greater than zero). If `progress` < 0 or **`negativeUserMovement`** < 0, then the value of `progress` is added to the **`negativeUserMovement`**:

```
negativeUserMovement += progress
```

Thus, as soon as the user's distance to the next location increases, we keep track of this increase until

- either the user reaches his original distance, then **`negativeUserMovement`** `= 0`
- or the increase reaches a threshold:
  `negativeUserMovement` < `-NEGATIVE_USER_MOVEMENT_THRESHOLD`,
  then the **negative user movement criterion** is set to true.

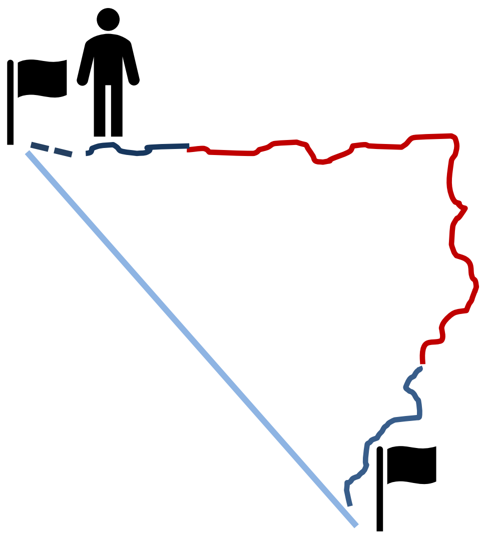
Combine the criterions

If either the **rate criterion** or the **negative user movement criterion** is set to true, **wrongDirection** is set and the following path segments are drawn in red color. If both criterions are false again, the following path segments are drawn in dark blue again.

Details

These are the basic thoughts. We added a few more rules to make the color changing more intuitive for the user.

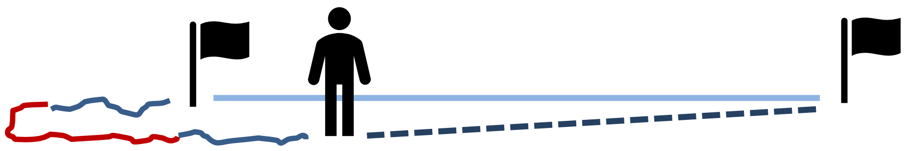
a) If the rate criterion is set to true and the user does not react to it immediately, the user might need to go a long distance in the direction of the location until the path turns blue again.



It would be easier for the user if the path turns blue again as soon as he moves in the direction of the next location. Therefore, we trigger the **negative user movement criterion** artificially as soon as the **rate criterion** is set to true by defining
`negativeUserMovement` = `-NEGATIVE_USER_MOVEMENT_THRESHOLD - 1`

and "ignore" the **rate criterion** while **wrongDirection** is set. This has the consequence that only the **negative user movement criterion** can set the color to blue again. If this happens, we set the **rate criterion** to false by setting the current user location as the initial point for the following rate computations (in the code this is done via the variables `startUserPosition` for the location and `userPathLengthSegment` for the distance the user moved since this location).

b) A similar problem appears considering the **negative user movement criterion**. If it is set to true, the user has to move `NEGATIVE_USER_MOVEMENT_THRESHOLD` meters in the direction of the next location until the path turns blue again.



It would again be easier for the user if the path turns blue earlier, for example if the user moves just 10 meters in the direction of the next location. For this reason, we added another threshold, namely `POSITIVE_USER_MOVEMENT_THRESHOLD`. If **wrongDirection** is set, we clamp the **`negativeUserMovement`** to the negative value of this threshold, thus it always holds that

`negativeUserMovement` `>= -POSITIVE_USER_MOVEMENT_THRESHOLD.`