# Hra Racetrack

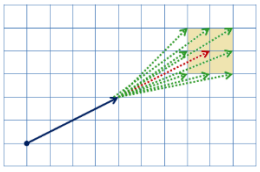
Ondřej Dušek – duseko10

duseko10@fel.cvut.cz

15.5.2022

## Shrnutí pravidel

Tato hra je inspirována hrou popsanou v [1]. Hra se hraje na čtverečkovaném papíru, na který se nakreslí obrysy závodní dráhy, včetně několika uzlů (políček) ozačených jako start a několika uzlů označených jako cíl. Oba hráči začnou na náhodném políčku na startovací dráze. Na počátku se mohou pohnout na libovolné z osmi sousedících políček (počítáme i ta diagonálně sousedící). Jakmile se pohnou, tak na mapě označíme vektor, který přičteme k nové pozici a je daný rozdílem nové pozice na mapě a původní pozicí na mapě. Hráč se může pohnout do osmi sousedních uzlů, ale ne do pozice dané těmito vektory uprostřed (zde se liší pravidla v [1] od toho, jak je znám já). Ilustrováno to je na:



Obrázek 1: Ilustrace osmi nových možných pohybů (zelená šipka) určených vektorem rozdílů pozicí (červená šipka). Převzato z [1].

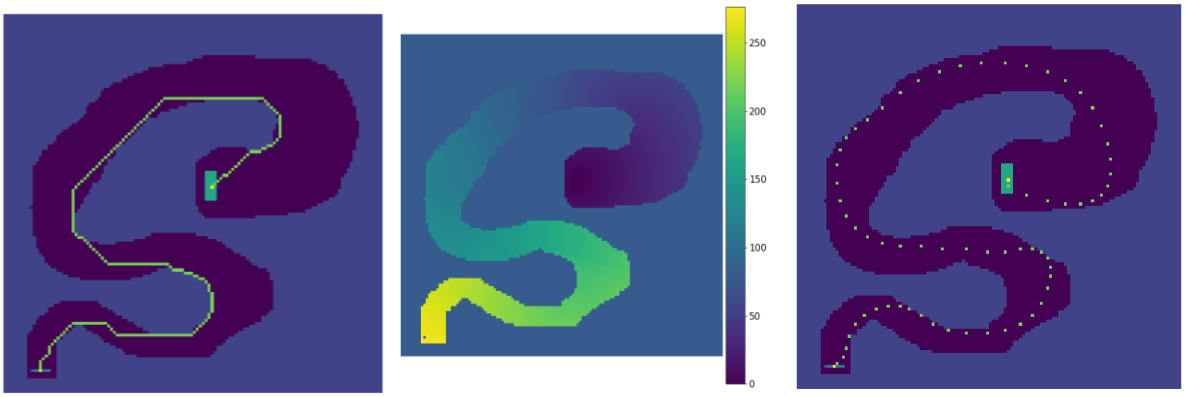
Tímto způsobem pohybu má hráč za cíl se dostat až do cíle (nemusí skončit přesně na cílovém políčku, stačí, když jím projede). Jakmile hráč vyjede z dráhy (či by se nám v našem případě stalo, že bude mít takovou rychlost, že by v příštím tahu skončila formule mimo hrací plán), hra končí. Hráč se nemůže pohybovat větší rychlostí, než takovou, pro kterou má červený vektor na obrázku 1 normu 10 (dále budu tento vektor označovat jako vektor rychlosti).

### Počítání trasy umělé inteligence

Hlavním cílem bylo vytvořit ve hře umělou inteligenci, proti které by hráč mohl hrát. To se úspěšně podařilo. Vytváření umělé inteligence pro tuto úlohu má několik výzev. První výzvou je, že proti obyčejnému prohledávání bludiště máme pro každé políčko na mapě příslušných skoro 400 možných vektorů rychlosti, přičemž tyto možnosti si nejsou ekvivalentní. Pro některé vektory rychlosti bychom po několika tazích nevyhnutelně vyjeli z dráhy, pro jiné vektory rychlosti ne. Druhou výzvou je, že nejde použít algoritmus A*, a to z toho důvodu, že ačkoliv je nejpřirozenější dělat algoritmus, který se snaží vždy co nejvíce přiblížit k cíli, tak ve skutečnosti nám jde o to, dostat se do cíle v co nejmenším počtu kroků, což neodpovídá nejkratší cestě. U algoritmu A* hodnotu kriteriální funkce, podle které otevírám nové možnosti cesty, mám jako vážený součet heuristiky (ideálně vzdálenost do cíle) a již uražené vzdálenosti (či počtu uražených kroků). Problém je, že uraženou vzálenost použít nelze, neboť ta jde proti počtu uražených kroků (ideální řešení by bylo krokem o velikosti 1 se pohybovat přesně podél obrysu zatáček dráhy), a pakliže bychom použili počet uražených kroků, tak bychom spolu se vzdáleností do cíle míchali dvě nesouvisející věci. Museli bychom daný součet rozumně navážit, jenže toto vážení by nemuselo pokrýt

situace, kdy mám například příliš rovnou, dlouhou, krátkou, či klikatou dráhu (průměrná velikost kroku se pro různě klikaté dráhy může výrazně lišit).

Řešením tedy bylo použít Greedy search – v kriteriální funkci mám pouze heuristiku, už nepřičítám uraženou vzdálenost či počet kroků. Heuristika byla udělána následovně. Nejprve jsem prošel dráhu algoritmem A* jako při prohledávání bludiště (krok vždy jen na osm sousedících políček, najdu nekratší cestu). Tím jsem získal trasu vyznačenou na obrázku 2.




Obrázek 2: Trasa hledaná pomocí A* (vlevo), evaluační matice (uprostřed), trasa nalezená Greedy search (vpravo)

Dále jsem si udělal novou mapu, ve které jsem každému bodu dráhy přiřadil hodnotu danou nejmenším možným součtem vzdálenosti k nějakému bodu trasy hledané pomocí A* a délce A* trasy od tohoto bodu do cíle. Ta je též vyznačena na obrázku 2, nazveme ji zkráceně evaluační matice. Pro zřetelnost mají políčka mimo trasu hodnotu nastavenou na 140, v samotném algoritmu to je $10^9$. Heuristika Greedy search pak byla pro příslušné políčko vždy hodnota evaluační matice v daném bodě. Pomocí Greedy search jsem pak byl schopen nalézt trasu jako je například ta zobrazená na obrázku 2. Greedy search zohledňuje to, že je třeba políčko $(x, y)$ zavírat pouze pro daný vektor rychlosti $(v_x, v_y)$. Pro ušetření času jsou ovšem otevírána políčka $(x, y)$ pro všechny vektory rychlosti $(v_x, v_y)$ současně. Samotný Greedy search běží rychle (pro obrázek o velikosti 128x128 jako na obrázku 2 okolo 0.1 s – „klikatější“ dráhy ale běží déle), co běží dlouho je vytváření evaluační matice (cca 15 vteřin), jelikož to bohužel nelze počítat v numpy maticově. To by šlo odhaduji o polovinu zrychlit tím, bych z trasy hledané pomocí A* nechal pouze body, co se dotýkají okrajů dráhy, na druhou stranu, pak by se mohlo stát, že v případě členitější dráhy bych pro nějaké body dráhy nemohl určit evaluační hodnotu, protože by z toho bodu nebyl žádný v bod dráhy A* v přímé linii pohledu tak, abych neprotínal region mimo dráhu.

Co se A* a Greedy search týče, tam jsem vycházel z přednášky doc. RNDr. Pavla Suryneka, Ph.D., které jsou dostupné na [2], jinak už jen z dokumentací použitých knihoven. Zvuky a obrázek formule na ikonu hry jsem stáhl z [3].

## Reference:

1. Racetrack (game). Wikipedia. [Online] 23. 4 2022. [Citace: 15. 5 2022.] https://en.wikipedia.org/wiki/Racetrack_(game).

2. doc. RNDr. Pavel Sudynek, Ph.D. BI-ZUM: Základy umělé inteligence. [Online] 16. 2 2021. https://courses.fit.cvut.cz/BI-ZUM/lectures/index.html.

3. opengameart.org. [Online] https://opengameart.org/