Protokol k semestrální práci z BI-ZUM.21

FIT ČVUT, LS 2022/2023

Jméno studenta: **Anopa Denys**
Username: **anopaden**
Název semestrální práce: **Dron 'Opatrník' v krápníkové jeskyni**

1. Zadání semestrální práce.

Máme jednoho létajícho drona, který je velmi opatrný a nechce si rozbít vrtulky. Jeho úkol ovšem je proletět krápníkovou jeskyní s velkými a ostrými krápníky. Vymyslete a implementujte algoritmus pro plánování pohybu takového drona. Musíme předpokládat, že dron se pohybuje ve velmi členitém trojrozměrném prostředí, kde nikde nesmí narazit, či lépe se ani přiblížit k překážce. Senzorickou stránku úlohy zanedbáváme, dron prostředí dokonale zná a vždy přesně zná svoji polohu.

- Jedná se o hledání bezpečné cesty pro drona, což není úplně libovolná cesta.
- Můžeme uvažovat dvě kritéria, a sice pohybovat se po bezpečné cestě, ale zároveň dorazit do cíle co nejrychleji.

2. Stručný rozbor, analýza problému/ zadání.

Podle popisu zadání musíme vytvořit 3d prostor ve kterém bude pochybovat náš robot. Prostředí musí obsahovat překážky a robot nesmí se k nim přiblížit nebo narazit se na ně. Budeme předpokládat, že pokud dron se pochybuje v trojrozměrném prostředí, může se letět celkem v šesti směrech. Tedy nahoru, dolu, doleva, doprava, dozadu, dopředu. Ze zadání taky plyne, že robot musí se dostat do cíle co nejrychleji. Můžeme tedy použit nebo BFS nebo A star. Pro splnění požadavků na bezpečnou cestu, budeme uvažovat, že dron nesmí se pohybovat přímo před překážkou (příští blok nemá jako souseda překážku)

3. Výběr metody.

BFS jak i A star nalezne nejrychlejší cestu ale bude potřebovat víc času a větší počet stavu, což v případě 3d prostředí, ve kterém oproti 2d je mnohem víc stavu nebude docela efektivně. Z tohoto důvodu zvolíme A star. Pro trojrozměrné prostředí A star bude fungovat stejně jak i pro 2d, přibude ještě jedna souřadnice. A star je v podstatě spojení dvou algoritmu prohledávání Dijkstrůvá algoritmu a Greedy search. Jako heuristiku lze zvolit Manhattanskou nebo Euklidovskou vzdálenost. Implementoval jsem obě ale jako výslednou použil jsem Manhattanskou vzdálenost.

4. Popis aplikace metody na daný problém.

Pro 3d prostředí algoritmus je docela podobný jak i pro 2d případ. V každém stavu musíme ověřit šest sousedů(levý, pravý, dolní, horní, přední, zadní). Pro každého výpočtem vzdálenost, která se skládá ze vzdálenosti od začátku(Dijlstra) a vzdušné vzdálenosti (Manhattanská vzdálenost) do konce. Sečteme je a pokud je menší než existující a není nalezený(při inicializaci nastavíme obrovské číslo) zařadíme do fronty, změníme předchůdce

na aktuální stav, označíme ho jak nalezený a zvětšíme vzdálenost od počátku na jedničku, protože cena přechodu mezi stavy v našem případě je 1. Opakujeme to dokud se ne skončí fronta(cesta neexistuje) nebo dokud nenalezneme cestu do cíle. Cestu získáme snadno, protože každý stav pamatuje svého předchůdce, proto jenom projdeme od cíle do startu a tím získáme cestu. Heuristika (tj. Manhattanská vzdálenost) bude vypadat tak

$$|x_{end} - x_{curent}| + |y_{end} - y_{curent}| + |z_{end} - z_{curent}|$$

5. Implementace.

Pro ukládání stavů, jsem použil slovník, kde jako klíč byl tuple pro souřadnice a jako hodnota list ve který jsem ukládal souřadnice předchůdce, obsazený přehradkou nebo ne, nalezeny nebo ne, vzdálenost. Pro nalezení sousedů jsem vytvořil list tuplů a přičítal nebo odečítal jedničku od souřadnice aktuálního stavu. Taky jsem použil **PriorityQueue**, aby z fronty vždy odebírat prvek s největší prioritou. Taky jsem vytvořil funkci **checkNeighbors**, která ověřuje, zda nová poloha bude bezpečná pro drona. Na vizualizaci jsem použil knihovnu ursina, pomoci které se dá snadno vytvořit 3d prostředí z bloků. Z této knihovny jsem taky použil modul **FirstPersonController**, pomoci kterého se dá snadno pohybovat v vytvořeném prostředí. **Červené** bloky jsou bariéry, **modré** - začátek a cíl, **zelené** - cesta. Zmáčknutím klávesy enter bude nalezena(nebo nenalezena, závisí na prostředí) cesta od startu do cíle. Prostředí vždy různé, poloha překážek vždy generovaná náhodně. Start a cíl vždy stejný jak i velikost mapy(větší mapa bude potřebovat výkonnější hardware) ale jak i počet překážek je docela snadno měnitelné v souboru location.py. Uzavřít program se dá pomoci klávesy escape.

6. Reference.

https://www.youtube.com/watch?v=qns1vvi0WVo - Tutorial na použití grafické knihovny