

|  |  |  |
| --- | --- | --- |
| **Algoritmy umelej inteligencie (VAI)** | | |
| Meno: | | ID študenta: |
| Bc. Daniel Dlugoš | | 217099 |
| Dátum zadania: | Dátum odovzdania: | Študijná skupina: |
| 07.02.2024 | 12.06.2024 | 4pAIŘ/1 |
| Názov úlohy: | | |
| **Vizualizácia hľadania optimálnej cesty pomocou algoritmu A\* v 2D priestore.** | | |

# Algoritmus A\*

Je úplný a optimálny algoritmus usporiadaného prehľadávania na základe hodnotiacej funkcie f. Ak existuje riešenie, je zaručené ho nájde. Hlavným problémom algoritmu je pamäťová náročnosť, pretože v pamäti udržiava všetky generované stavy. Pri rozsiahlych problémoch môže algoritmus predčasne skončiť vyčerpaním pamäte.

## Informované metódy prehľadávania stavového priestoru

Informované metódy prehľadávania sú založené na využití hodnotiacej funkcie *f*.

* Určuje ohodnotenie pre každý uzol stromu stavového priestoru.
* Ohodnotenie sa používa pre výber ďalšieho uzlu k expanzii.
* Vždy sa expanduje ten uzol, ktorý mal doteraz najlepšie ohodnotenie a vyhodnocujú sa iba jeho následníci.
* Hľadanie je zastavené, keď je dosiahnutý stav, ktorý má lepšiu hodnotu funkcie f ako jeho následníci.

## Hodnotiaca funkcia A\*

*f (i) = g (i) + h (i)*

*g (i)* — cena optimálnej cesty z počiatočného stavu do stavu *i*.

*h (i)* — cena optimálnej cesty zo stavu *i* do niektorého z cieľových stavov.

### Funkcia *h (i)*

Kvantitatívne vyjadruje „odhad” ceny cesty z aktuálneho stavu do niektorého z cieľových stavov. „Odhad” predstavuje heuristickú znalosť o tom, aké sú šance nájsť riešenie, ak by sme pokračovali expanziou daného stavu. Funkcia *h (i)* je preto nositeľom heuristickej informácie a nazýva sa aj heuristickou funkciou.

# Zdrojový kód

V tejto kapitole bude predstavený zdrojový kód a funkcie, ktoré je naprogramovaný vykonávať.

## Predrekvizity

Importuje moduly pygame, math a PriorityQueue. Inicializuje rozmery zobrazovacieho okna s názvom "A\* Snake Hunting It's Prey".

Pygame je sada modulov programovacieho jazyka Python určených pre vitváranie videohier. Pygame má veľa vstavaných funkcií a nakoľko nie je potrebné využiť mnoho z nich, intuitívnejší prístup bude vytvoriť vlastnú hernú logiku a triedy.

## Globálne nastavenia

Medzi prvé kroky zaradíme deklaráciu konštantných hodnôt. Tie zahŕňajú nastavenia:

* Absolútnej veľkosti šírky hracieho poľa - hracie pole bude nastavené pre štvorcovú plochu.
* RGB kód pre farby uzlov tzv. spots. Uzly budú vyfarbené na základe funkcie, ktorú v programe vykonávajú.

## Trieda Spot

Bude reprezentovať každú bunku (uzol) v mriežke a všetky metódy týkajúce sa vlastností uzlov, aktualizácie ohodnotenia uzlov a vyfarbenia uzlov. Každý uzol má vlastnosti ako riadok, stĺpec, šírka, farba a susedia. Význam jednotlivých funkcií je popísaný v následujúcich kapitolách:

### \_\_init\_\_()

Metóda, ktorá slúži na inicializáciu vlastností uzlov a všetkých nastavení objektu.

### get\_pos()

Získava polohu daného uzla. Polohu reprezentuje index stĺpca a riadku mriežky v zobrazovacom okne.

### is\_closed(), ..., is\_end()

Metódy kontrolujú stav uzlu:

* Jeho výskyt v zozname OPEN alebo CLOSED.
* Či sa jedná o uzol reprezentujúci stenu.
* Či sa nejedná o počiatočný alebo koncový uzol.

### reset() ... make\_path()

Metódy menia farbu uzlu na základe jeho stavu.

### draw()

Metóda pre zakreslenie uzlu do zobrazovacieho okna.

### update\_neighbors

Funkcia pre aktualizáciu susedov. Pozerá sa na susedov aktuálneho uzlu v každom smere. do zoznamu OPEN zapíše uzol, ktorý je najviac optimálny pre priblíženie sa ku konečnému uzlu. Uzly sa preskúmavajú v smere od ľavého horného uzlu (počiatok súradnicového systému) po pravý dolný uzol. Funkcia si vyberá susedov, ktorých expanduje následovne:

* Smerom dole, ak sa nejedná o posledný riadok v mriežke a zároveň sa nejedná o uzol, ktorý reprezentuje stenu.
* Smerom hore, ak sa nejedná o prvý riadok v mriežke a zároveň sa nejedná o uzol, ktorý reprezentuje stenu.
* Doľava, ak sa nejedná o prvý stĺpec v mriežke a zároveň sa nejedná o uzol, ktorý reprezentuje stenu.
* Doprava, ak sa nejedná o posledný stĺpec v mriežke a zároveň sa nejedná o uzol, ktorý reprezentuje stenu.

## Heuristika *h()*

Definuje heuristickú funkciu *h*. Heuristika používa Manhattanskú vzdialenosť na odhad vzdialenosti medzi dvoma bodmi. Obmedzenie pohybu iba na horizontálny a vertikálny smer. Vracia súčet absolútnych vzdialeností v oboch smeroch.

## reconstruct\_path()

Funkcia, ktorá spätne vykresľuje respektíve rekonštruuje cestu z koncového bodu k počiatočnému.

## Raw A\*

Funkcia, ktorá programovo interpretuje algoritmus A\* začína:

* open\_set – Inicializáciou zoznamu OPEN pomocou modulu Queue.
* count – premenná udržiavacia poradie uzlov, podľa toho ako vstupovali do zoznamu. Počítadlo nadobúda dôležitosť v prípade že algoritmus objaví viacero uzlov s rovnakým ohodnotením. V tomto prípade si podľa premennej count algoritmus zvolí uzol, ktorý sa v zozname objavil ako prvý.
* came\_from – je zoznam, do ktorého sa zapisujú uzly cesty. Spätne sa využíva pre zvýraznenie optimálnej cesty vo funkcii recostruct\_path().
* g\_val – deklarácia vzdialenosti od počiatočného uzlu do aktuálneho uzlu [*g(i)*].
* f \_val – deklarácia ohodnocovaciej funkcie (súčet g\_val a heuristickej funkcie h()).
* open\_set\_hash – Zoznam, ktorý kontroluje obsah zoznamu queue a odstraňuje duplikáty.

Po deklarácií počiatočných premenných algoritmus prebieha v slučke a je ukončený v prípade, že sa prehľadal každý uzol alebo neexistuje riešenie (cieľový uzol je nedostupný). Slučka obsahuje podmienku, v ktorej kontroluje či aktuálny uzol nie je koncový. V opačnom prípade prehľadáva susedné uzly.

Pri výbere susedného uzlu s najlepším ohodnotením sa pracuje s premennou temp\_g\_val, ktorá predpokladá doposiaľ najlepšiu cestu od počiatočného uzlu k aktuálnemu. Ak existuje lepšia cesta, algoritmus si aktualizuje jej ohodnocovaciu funkciu.

Algoritmus pokračuje vyfarbovaním uzolov, pričom kontroluje podmienku či je aktuálny uzol zároveň počiatočný. V tomto prípade prefarbí uzol na červeno.

## Visualization Tool

Súbor funkcií, ktoré vykonávajú všetky dôležité aspekty pre vizualizáciu princípu fungovania algoritmu A\*.

### make\_grid()

Funkcia, ktorá vytvára mriežku.

### draw\_gridlines()

Funkcia, ktorá vytvára zvislé a vodorovné čiary mriežky.

### draw()

Funkcia, ktorá vykresľuje zobrazovacie okno, mriežku a čiary mriežky.

### get\_clicked\_pos()

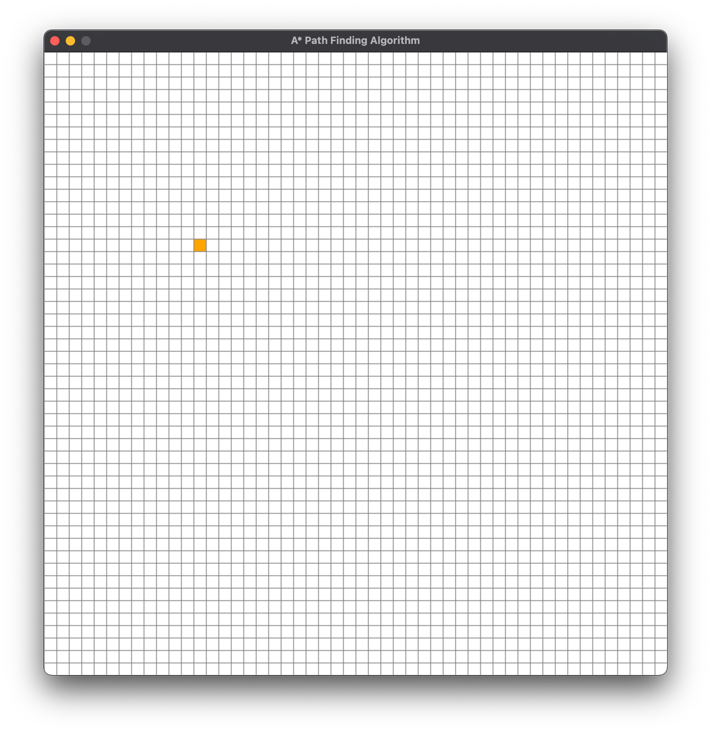
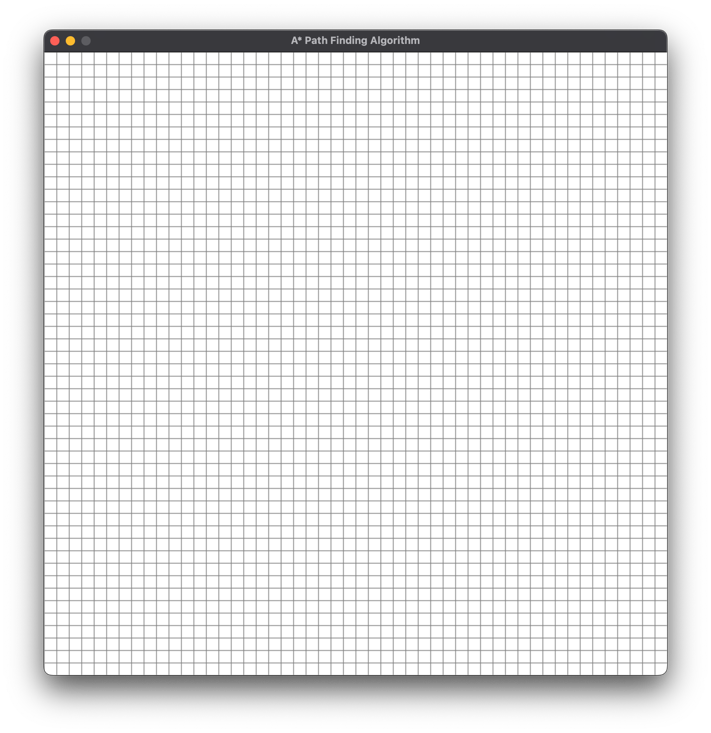
Funkcia, ktorá získava polohu mriežke kde došlo ku kliku myšou.

## Funkcia main()

Hlavná funkcia, ktorá inicializuje mriežku a spracováva všetky interakcie užívateľa. Po vykreslení mriežky je užívateľ vyzývaný aby si ľubovoľne zvolil počiatočný bod, koncový bod a zakreslil steny.

# Pokyny k používaniu

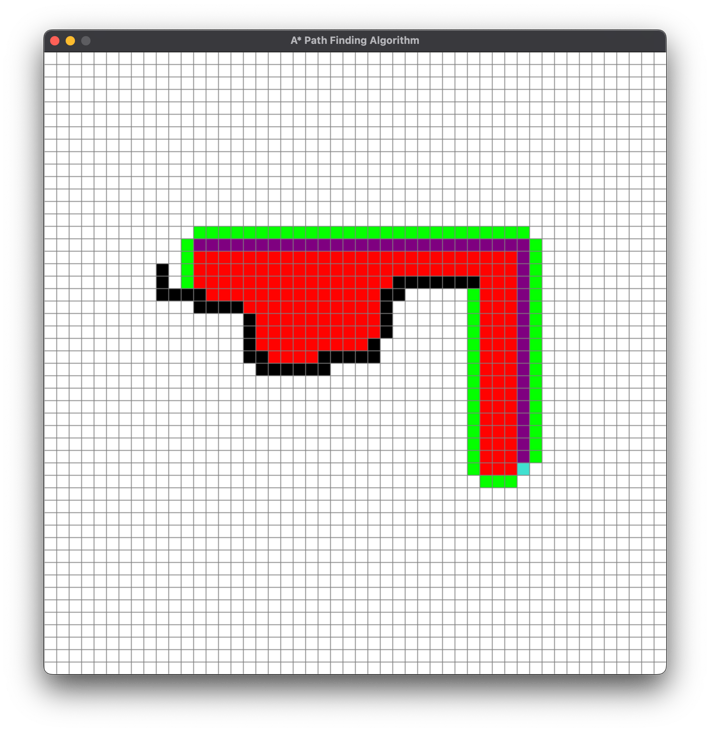
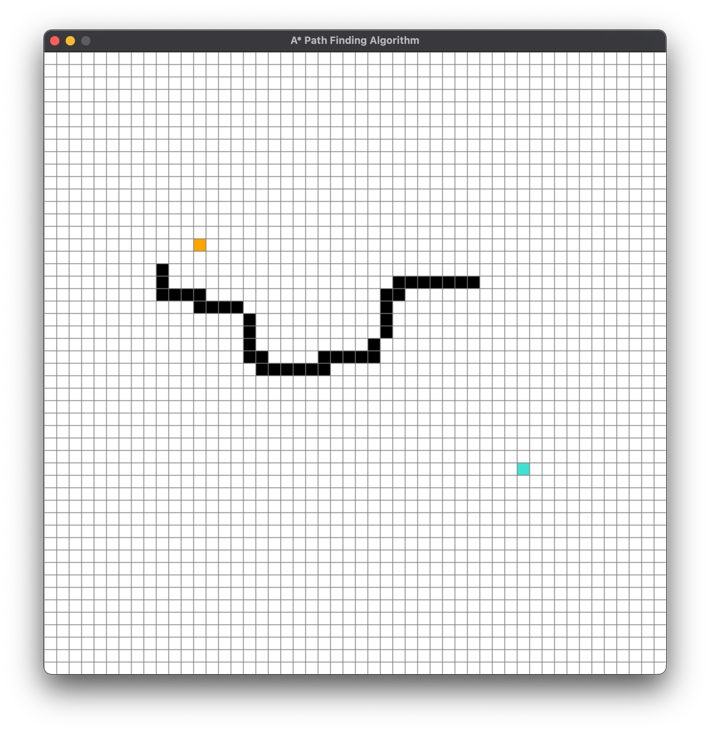
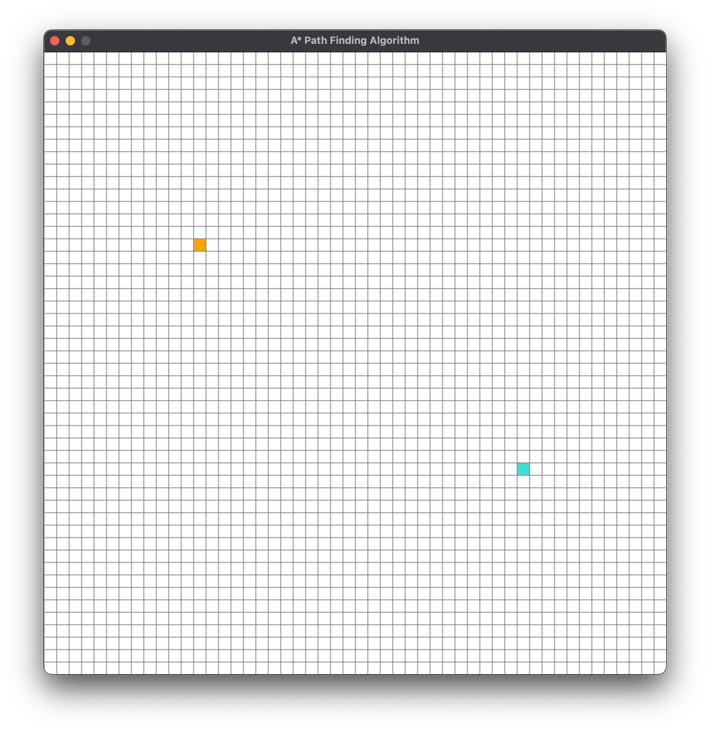
* Program začína s vykreslením prázdnej mriežky vo fixnej veľkosti.
* Nastavenie počiatočného bodu, cieľového bodu a steny sa vykonáva kliknutím ľavého tlačidla myši na bunky mriežky (pri kreslení steny možno tlačidlo aj držať).
* Uzly sa resetujú (späť do bielej farby) prvým klikom myši.
* Stlačenie medzerníka spustí algoritmus A\*.
* Mriežku možno po skončení algoritmu vrátiť do pôvodného stavu stlačením klávesy „C“.



Obr. 1 – Prázdna mriežka

Obr. 2 – V mriežke je zakreslený počiatočný bod

# Záver



Obr. 3 – V mriežke je zakreslený počiatočný a cieľový bod

Obr. 4 – Počiatočný bod, cieľový bod a prekážka (stena)

Obr. 5 – Vizalizácia optimálnej cesty nájdenej algoritmom A\*

V tejto práci bol implementovaný a vizualizovaný algoritmus informovaného prehľadávania stavového priestoru A\*. Grafický modul Pygame umožňuje simulovať správanie a schopnosti algoritmu, ktorý je schopný nájsť cestu najoptimálnejšej cesty, pokiaľ existuje. Vylepšenie programu by priniesla úprava heuristickej funkcie z Manhattanskej na Euklidovksú.

# Zdroje

# *A\* Algorithm Concepts and Implementation: How to Implement the A\* Algorithm in Python?* Online. 2024. Dostupné z: https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/a-star-algorithm#how\_to\_implement\_the\_a\_algorithm\_in\_python. [cit. 2024-06-12].

# *A\* Search Algorithm.* Online. 2024. Dostupné z: https://www.geeksforgeeks.org/a-search-algorithm/?ref=ml\_lbp. [cit. 2024-06-12].

# *Implementation of A\*.* Online. 2020. Dostupné z: https://www.redblobgames.com/pathfinding/a-star/implementation.html. [cit. 2024-06-12].

# *A\* Pathfinding (E01: algorithm explanation).* Online. 2014. Dostupné z: https://www.youtube.com/watch?v=-L-WgKMFuhE. [cit. 2024-06-12].

# *Pygame: About - Wiki.* Online. Dostupné z: https://www.pygame.org/wiki/about. [cit. 2024-06-12].

# *Python PyGame Tutorial: The Complete Guide.* Online. Dostupné z: https://coderslegacy.com/python/python-pygame-tutorial/?utm\_content=cmp-true. [cit. 2024-06-12].