Algoritmy umělé inteligence (VAI 22/23L)

Semestrální projekt: Technická zpráva

Řešení hry Snake pomocí algoritmu A – star

Vypracoval: Jan Fiala (217141)

Datum: 1.5.2023

Zadání:

Vypracování programu využívající vámi vybranou metodu (metody) umělé inteligence pro řešení vybrané úlohy.

Úvod – Popis řešené problematiky:

První variací hry Snake (had) byla arkádová hra zvaná Blockade, která byla vytvořena v roce 1976. Ve stylu, kterém je známá v současnosti byla hra vytvořena na mobilní zařízení Nokia 6110, od té doby byla zpracována v mnohých formách a zpracováních. Cílem hry je navést hada k bodu zvaném jablko, bez nárazu do svého vlastního těla a okrajů hrací plochy. Při dosáhnutí tíženého bodu je těla hada zvětšeno o jeden blok, jablko se přesune na náhodně generované místo a proces se opakuje. [2] [3]

Relativně jednoduchá herní mechanika nabízí možnost implementace hladového algoritmu pro hledání nejkratší cesty v heuristickém prostoru. Jedním z nejpoužívanějším je algoritmus zvaný A* (A star).

A* využívá již zmíněného hladového principu pro nalezení optimální cesty z daného počátečního bodu do požadovaného koncového bodu. Byl vytvořen v roce 1968 a používá stejné principy jako Dijkstrův algoritmus, ale přidává navíc heuristický prvek. [1]

Popis algoritmu:

- Pro každý navštívený vrchol x se spočítá ohodnocení nejlepší cesty f(x):

$$f(x) = g(x) + h(x)$$

  g(x) je součet ohodnocení nejkratší cesty ze startu přes dosud navštívené (a probrané) vrcholy do x

  h(x) je heuristický odhad délky cesty z x do cílového stavu

  - pro cílový stav: $h(x) = 0$
  - pro každý stav: $h(x) \geq 0$

Využívá dvě datové struktury:

- Open, seznam navštívených vrcholů, které je potřeba probrat
- Closed, seznam již probraných vrcholů

Výhody a nevýhody:

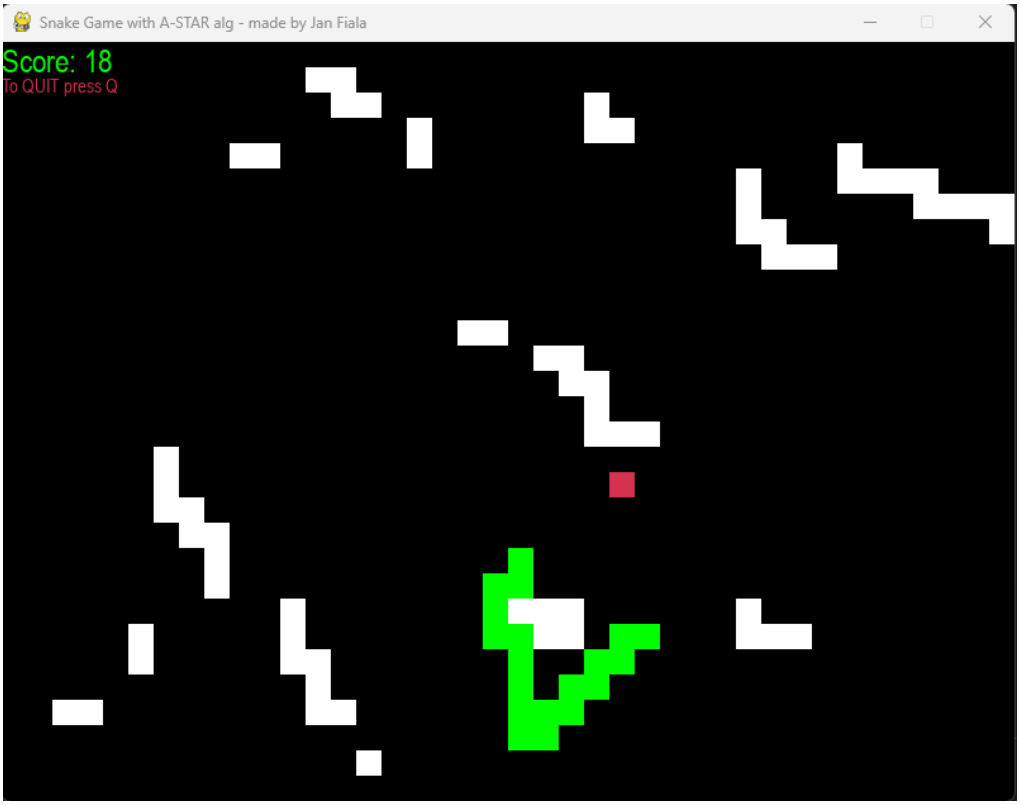
\+ cesta je vždy nalezená a je nejkratší možná
\+ volnost v ohodnocování
\+ použití s libovolnou reprezentací terénu
\+ jednoduchost implementace

\− počet vrcholů v Open nebo Closed listu může být v řádu stovek nebo tisíců
\− cesta se musí spočítat celá
\− najednou pokud cesta neexistuje, prohledá se celý prostor [1] [4]

Metoda řešení:

Pro danou problematikou byl na implementaci zvolen programovací jazyk python. Pro vytvoření vizualizace byla zvolena knihovna Pygame, která pomocí jednoduchých smyček dokáže vytvářet herní pole a vykreslovat objekty pro vybrané hry. Pro vytvoření algoritmu je nutné použití datové struktury pomocí prioritní fronty, zde implementované knihovnou heapq.

Nejdříve je definována hrací plocha, rozdělená do jednotlivých bloků, přes jsou následně generovány objekty ve hře. Následně jsou nastaveny základní uživatelské možnosti herního okna, s možností spuštění hry, ukončení a případném restartování.



Obrázek 1 Vytvořená hra Snake

V první fázi spuštění hry je vygenerován náhodný blok červené barvy představující cílový bod, jablko. Dále je vygenerován samotný had, umístěný přesně ve středu hracího pole. Pro lepší vizualizaci a stižení hledání nejkratší cesty jsou na rozdíl od originální verze hry přidány náhodně generující se objekty, překážky. Při narazení hada do těchto polí, svého těla, nebo okrajů okna dojde k ukončení hry. Po inicializačním vygenerování všech bodů se hodnoty přesouvají do iterace A* algoritmu. Vstupem je počáteční bod (hlava hada), cílová pozice jablka a seznam všech vygenerovaných překážek včetně polí těla hada kromě hlavy.

```
# Function to generate obstacles
def generate_obstacles(num_obstacles):
    obstacles = []

    # Random obstacles based on number
    for i in range(1, num_obstacles + 1):
        lo = random.randrange(1, h_number - 1)*blocksize, random.randrange(
            1, w_number - 1)*blocksize  # last obstacle
        obstacles.append(lo)
        for j in range(1, random.randint(1, int(num_obstacles / 2))):
            if random.randint(1, 2) == 1:
                lo = (lo[0] + blocksize, lo[1])
            else:
                lo = (lo[0], lo[1] + blocksize)
            if 0 < lo[0] <= h_number*blocksize and 0 < lo[1] <= w_number*blocksize:
                obstacles.append(lo)
    return obstacles
```

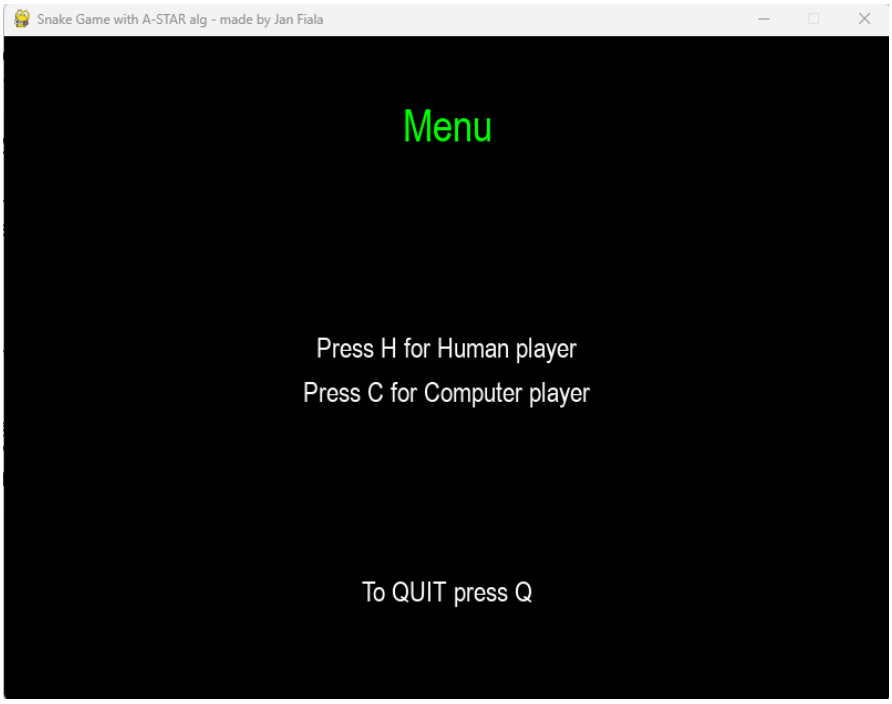
Obrázek 2 Část programu na generování náhodných překážek

Následnými iteracemi je vždy zvolen první soused ležící v blízkosti startovního bodu. Tento bod je kontrolován, aby nenáležel žádnému bodu z překážek a ležel v hrací ploše. Při splnění této podmínky je následně vypočtena teoretická heuristická vzdálenost vedoucímu. Připočte se hodnotě dosud navštívených bodů. Následnou iterací se kontrolují ohodnocení všech doposud navštívených bodů a v případě nalezení nejmenšího ohodnocení, je zařazen do cílové trasy ve formě prioritní fronty. Proces se opakuje do nalezení cíle přes body s nejmenším ohodnocením, nejkratší cestou. Pokud algoritmus po prohledání celé plochy není schopný nalézt cestu k vybranému bodu, dokud se hra neukončí.

Cesta je poté iterována samotnou herní smyčkou, dokud pozice hlavy hada není shodná s pozicí jablka. V tomto případě je znovu vygenerována náhodná pozice nového cílového bodu a celí proces se opakuje.

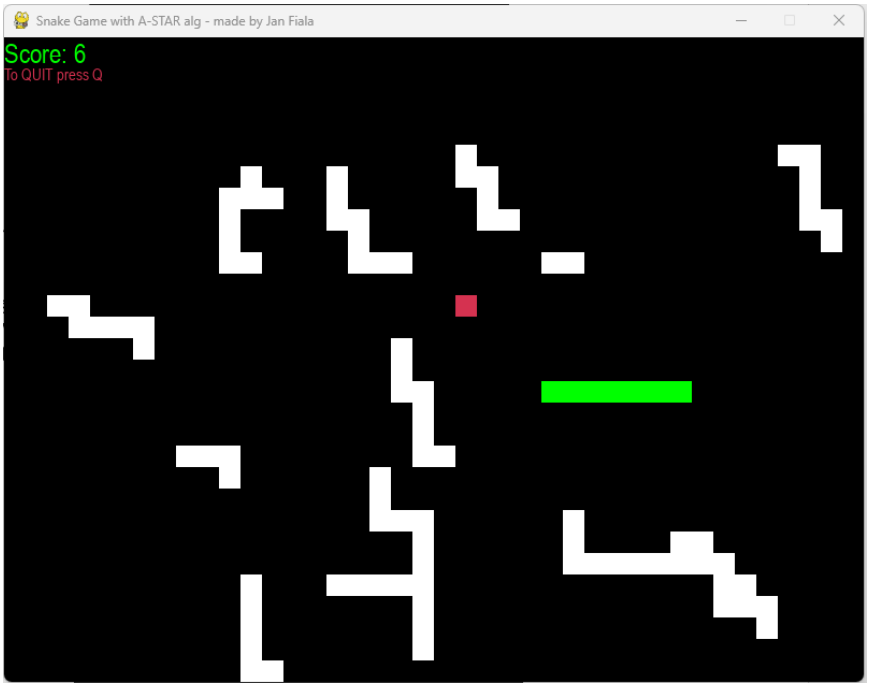
Uživatelský popis programu:

Při spuštění programu je vytvořeno jednoduché menu okno, ve kterém si uživatel zvolí herní mód. V rámci porovnání obtížnosti vygenerované mapy, jsou k dispozici dva herní režimy, kdy první registruje uživatelské vstupy pro manuální ovládání hada dle originální hry. Druhý mód automaticky aktivuje algoritmus A*. ovládání hada hráček je pomocí šipek na klávesnici.



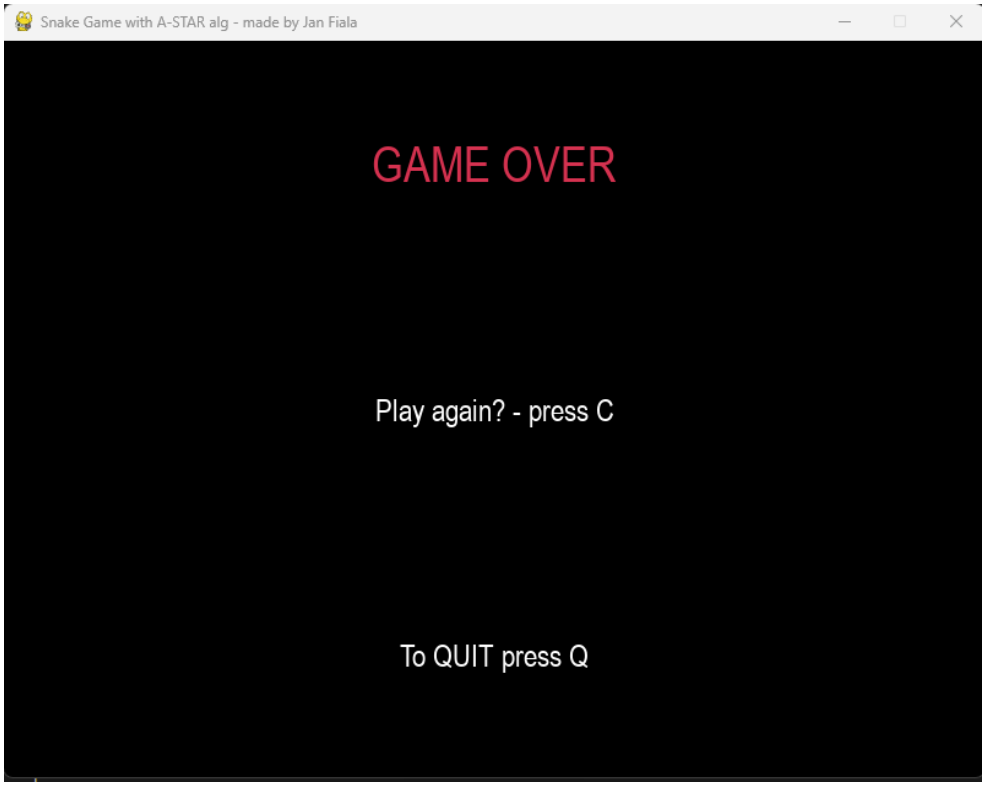
Obrázek 3 Herní menu

Po spuštění vybraného módu stiskem požadované klávesy, jsou vykresleny objekty hry a dojde k jejímu spuštění. Během hraní je možné pomocí klávesy q kdykoliv hru ukončit. V rohu obrazovky je zobrazeno aktuální bodové ohodnocení dané hry.



Obrázek 4 Rozvržení herní plochy

Při neúspěšném vykonání cesty k jablku způsobené nárazem do zvolených překážek je hra ukončena. Vyvolané okno oznamuje konec hry a pomocí klávesy C je možné znovu inicializovat návrat do úvodního menu hry, případně program ukončit.



Obrázek 5 Okno konce hry

Závěr:

Zvolený algoritmus dosahuje požadavků na nalezení nejkratší cesty k jablku a je schopný dosahovat vysokých bodových ohodnocení během hry. V počáteční fázi hry, dokud nemá had příliš dlouhé tělo, velmi rychle generuje požadovanou cestu. V pozdějších fázích ale potřebuje velké množství iterací k prohledání celé mapy, což značně zpomaluje proces vykreslování a plynulost herního režimu. Dále v algoritmu není zahrnuta strategie zabraňující zacyklení hada a náhodnost generovaných překážek v mapě může způsobit situace, kdy k jablko neexistuje žádná cesta. V tomto případě je problém ošetřen prostým ukončením hry. Následujícím možným vylepšením programu je tedy možnost přidání těchto podmínek, vylepšení grafického prostředí uživatelského prostředí. Aktuální řešení demonstruje kladné a záporné vlastnosti zvoleného algoritmu, a potvrzuje, proč je velmi rozšířen při vytváření plánování trasy umělé inteligence v počítačových hrách.

Zdroje:

[1] BAIJAYANTA, Roy. TOWARDS DATA SCIENCE. *A-Star (A*) Search Algorithm* [online]. 2019. [cit. 2023-05-01]. Dostupné z: https://towardsdatascience.com/a-star-a-search-algorithm-eb495fb156bb

[2] LUMIA. Snake charmed! 10 fascinating facts about the world's most popular game. *Microsoft Windows Blogs* [online]. 2012 [cit. 2023-05-01]. Dostupné z: https://blogs.windows.com/devices/2012/02/02/snake-charmed-10-fascinating-facts-about-the-worlds-most-popular-game/

[3] GRIFFIN, Bateson. The History Of Snake The Game. *Coolmathgames.com* [online]. 2023 [cit. 2023-05-01]. Dostupné z: https://www.coolmathgames.com/blog/the-history-of-snake-the-game

[4] RAVIKRIAN, A. S. A* Algorithm Concepts and Implementation. *Www.simplilearn.com/* [online]. 2023 [cit. 2023-05-01]. Dostupné z: https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/a-star-algorithm