BAKALÁŘSKÁ PRÁCE

Jméno Příjmení

Název práce

Název katedry nebo ústavu

Vedoucí bakalářské práce: Vedoucí práce
Studijní program: studijní program
Studijní obor: studijní obor

Praha ROK

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Poděkování.

Název práce: Název práce

Autor: Jméno Příjmení

Katedra: Název katedry nebo ústavu

Vedoucí bakalářské práce: Vedoucí práce, katedra

Abstrakt: Abstrakt.

Klíčová slova: klíčová slova

Title: Name of thesis

Author: Jméno Příjmení

Department: Name of the department

Supervisor: Vedoucí práce, department

Abstract: Abstract.

Keywords: key words

Obsah

Úvod

Počítačové hry mají kromě zábavního prožitku ze samotného hraní další funkci. Herní prostředí často tvoří samostatný, uzavřený svět se svými vlastními zákonitostmi. Agent ve světě dané hry ví, v jakém stavu se nachází, má na výběr konečný počet akcí, které může provést a cíl, kterého chce dosáhnout. A právě takovéto prostředí je pro nás vhodné pro experimentování s umělou inteligencí.

1. Navržená hra - Asteroidy

1.1 Herní logika

Jedná se o hru dvou hráčů. Každý z hráčů ovládá svou vesmírnou loď. Prostředí hry má představovat vesmírný prostor, je to ale prostor zjednodušený, proto zde neplatí gravitační ani odporové síly. To má tedy za následek, že když se vesmírná loď rozletí v nějakém směru, tak v tomto směru letí i nadále i bez dalšího akcelerování.

Vesmírný prostor je v této hře nekonečný a dalo by se říct jistým způsobem cyklický, pokud vesmírná loď proletí dolní hranicí herního prostoru, tak nezmizí, ani nenaboural, ale objeví se na stejné pozici jen na horní hranici a obráceně. Analogicky to platí i s bočními hranicemi. Vesmírné lodě sebou mohou proletět a nedojde ke srážce. Nejsou zde žádné statické překážky, kterým by bylo třeba se vyhnout. Co se ale může srazit s vesmírnou lodí jsou asteroidy.

Asteroidy vznikají v průběhu hry na náhodných místech a letí náhodným směrem. Nové asteroidy se generují častěji, čím déle hra trvá. V boji s asteroidy má hráč v zásadě dvě možnosti. Buď se může pokusit danému asteroidu vyhnout, tím že s lodí pohne mimo trajektorii asteroidu, anebo může asteroid sestřelit. Každý hráč má omezený počet životů a každá srážka lodě s asteroidem ubere hráči část jeho životů.

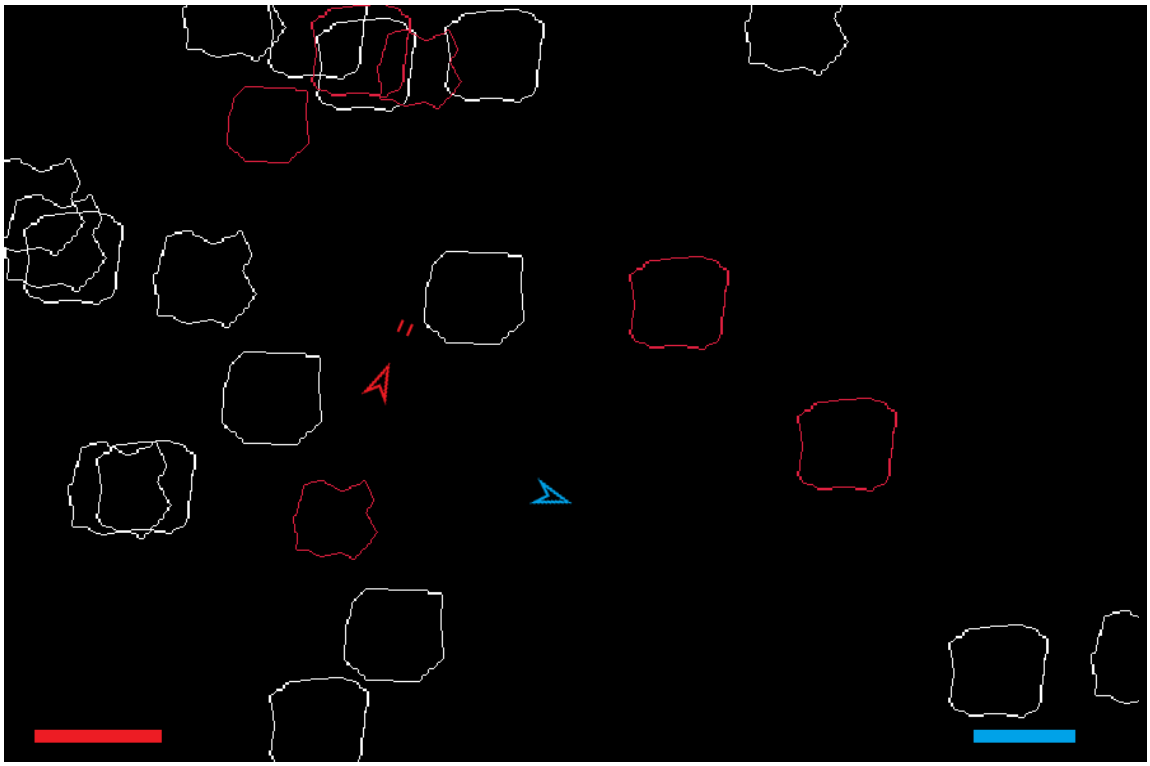
Hráč má k dispozici dva typy střel, obyčejnou a rozdvojovací. Vystřelená střela má značně vyšší rychlost než vesmírné lodě i než kolem letící asteroidy. Střely nejsou určeny k přímému zasažení lodě protihráče, vesmírné lodě jsou k nepřátelské střele imunní. Střely jsou určený k sestřelování letících asteroidů. Asteroidy mohou mít tři velikosti. Náhodně vytvořený asteroid je vždy největší. Každým rozstřelením daného asteroidu vznikají asteroidy o stupeň menší velikosti. Nově vytvořené asteroidy vznikají na místě původního asteroidu. Asteroid se může rozstřelit různými způsoby. Zde záleží na tom, jakou střelou byl asteroid setřelen.

V případě střely obyčejné vznikne namísto původního asteroidu jeden menší, který letí stejným směrem jako střela, která ho zasáhla. V případě střely rozdvojovací se původní asteroid rozstřelí na dva menší, kde každý z nich je oproti směru střely vychýlen o 15°po a proti směru hodinnových ručiček. Pokud je zasažen asteroid nejmenší velikosti, tak již žádné další asteroidy nevznikají. Rychlost asteroidů je nepřímo závislá na jejich velikosti, čím je asteroid menší, tim vyšší rychlost má.

Asteroidy vzniklé rozstřelením se stávají projektily daného hráče. Hráč nemůže být zasažen asteroidem, který sám vytvořil

1.2 Cíl hry

Během hry vzniká postupně více a více asteroidů, čímž je postupně stále obtížnější se všem asteroidům vyhnout, nebo je sestřelit. Hráč nemůže zranit nepřítele střelou přímo, může se ale snažit rozstřelit nějaký z kolem letících asteroidů tak, aby pomocí nově vzniklých asteroidů trefil nepřítele. Cílem hráče je ovládat svou loď takovým způsobem, aby vydržel ve hře déle. Hra končí a hráč vítězí, když nepříteli nezbydou žádné další životy.



Obrázek 1.1: Screenshot ze hry

2. Architektura hry

2.1 Vesmírné objekty

Všechny vesmírné objekty mají některá data společná. Každý vesmírný objekt má souřadnice své současné polohy a také vektor rychlosti.

2.1.1 Asteroidy

Asteroidy nesou navíc informace o tom, jaké jsou velikosti a zda-li byly vytvořené nějakým z hráčů. Na základě těchto dvou informací je asteroidu při vytvoření přiřazen obrázek, pomocí kterého je po dobu své existence vykreslován.

2.1.2 Střely

Vystřelené střely neletí věčně, ale mají omezenou životnost kolik kroků hry budou existovat. Tato hodnota se nastavuje z konfiguračního souboru z položky *BULLET_LIFE_COUNT*. V každém kroku hry se střele její živostnost sníží o jedna a pokud se dostane na nulu, tak střela bude zničena. Střele se při vytvoření nastaví úhel, pod kterým poletí. Tento úhel je roven úhlu natočení vesmírné lodi, který měla při vystřelení. Samozřejmě také u střely musíme evidovat, kterému z hráčů patří, toto je řešeno odkazem na objekt vesmírné lodi, která střelu vystřelila. Jak již bylo zmíněno v předchozí kapitole, střely jsou dvojího druhu. Příznakem *split* se určuje zda se jedná o střelu obyčenou nebo rozdvojovací

2.1.3 Vesmírná loď

Vesmírná loď má základní polohové informace rozšířené o úhel. Ten se s každou rotací lodě zvětší nebo zmenší o 12°. Akcelerace funguje vektorovým sčítáním. K současnému vektoru rychlosti se přičte vektor odpovídající současnému úholu lodi. Maximální rychlost vesmírné lodi je omezená, v případě že akcelerací vznikne vektor rychlosti, jehož délka je větší než hodnota maximální rychlosti, se směr vektoru zachová, ale požadovaně se zkrátí.

2.2 Prostředí

Hra běží v cyklu diskrétních kroků, které dohromady simulují plynulý pohyb hry. Herní prostředí je insipirováno projektem open ai gym od google (viz `https://gym.openai.com/`). Jedním rozdílem je však přístup k vykreslování hry. V případě *gym.openai* se prostředí vykresluje zavoláním metody *render()* na instanci prostředí zvenku. Já jsem zvolil přístup jiný. V případě, že chceme hru graficky zobrazovat, předáváme v kontruktoru prostředí grafický modul, který implementuje vykreslování jednotlivých typů vesmírných objektů. A prostředí už poté objekty graficky vyresluje interně samo. Rozhodnutí, že se má grafický modul volitelně injektovat v konstruktoru a nemá být natvrdo svázán s prostředím, jsem učinil pro větší nezávislost modulů. Při práci s různými knihovnami pro evoluční algoritmy se ukázalo být problematické, kdyby bylo herní prostředí svázáno s grafickým modulem.

Herní prostředí se stará o manipulaci všech vesmírných objektů a akcí s nimi spojenými. V každém kroku dostává od hráčů akce, které chtějí provést, a prostředí na to odpovídajícím způsobem reaguje. Akce každého hráče z hráčů jsou pole, které obsahuje elementární možné akce:

- Rotace vlevo
- Rotace vpravo
- Akcelerace
- Obyčejná střela
- Rozdvojovací střela

Hráč může provádět více akcí najednou. Na základě přítomných elementárních akcí se provadí dané reakce. Prostředí se stará o vesmírné objekty přímo. V případě elementárních akcí, které mění rychlost nebo orientaci vesmírné lodi, prostředí zavolá funkce, které požadované změny na vesmírné lodi provede. A v případě elemntárních akcí střel se na základě polohy a orientace dané vesmírné lodi vytvoří nová střela, kterou opět bude mít ve správě právě prostředí.

Hra, jak již bylo řečeno, má být konečná, toho je docíleno narůstajícím počtem asteroidů. O to se také stará herní prostředí. Pamatuje si počet kroků, které uběhly od posledního vytvořeného asteroidu. Pokud tento počet překročil danou mez, tak prostředí vytvoří nový asteroid. Postupného nárůstu nových asteroidů je docíleno incrementálním snižováním této meze.

Další důležitou funkcí herního prostředí je kontrola srážek. Všechny objekty jsou prostorově reprezentovány jako kruhy s danými poloměry. Postupně se prochází všechny objekty, u kterých nás zajímají srážky, a Euklidovskou metrikou se kontroluje, zda od sebe nejsou vzdáleny méně než je součet jejich poloměrů. V případě srážky se prostředí postará o správnou reakci - zničení nebo změnu sražených objektů a případně vytvoření nových objektů vzniklých srážkou.

V rámci srážek se upravují také odměny jednotlivých hráčů. Odměna je hodnota, která vyjadřuje jak úspěšný byl tento krok pro každého z hráčů. V každém kroku, který hráč přežil, je hráč odměnen odměnou hodnoty 1. Jsou ale konkrétní srážky objektů, které hodnoty odměny mohou změnit. V případě, že hráč sestřelil nepřátelský asteroid, nebo svým asteroidem srazil nepřátelovu loď, se výše odměny

zvýší. Naopak, pokud byla jeho vesmírná loď zasažena nepřátelským asteroidem, je hodnota odměny snížena. Koncept odměn nijak neovlivňuje samotný běh hry, ale bude se nám hodit v dalších kapitolách v umělé inteligenci.

Nezmínili jsme zatím pohyb objektů. I o to se samozřejmě stará herní prostředí. Zde se prochází seznamy všech vesmírných objektů a jednoduše se k jejich současné poloze přičte vektor rychlosti. Jediné co se musí kontrolovat je, že se daný objekt nedostal mimo herní prostor. Pokud toto nastane, tak je vrácen zpět do prostoru na své odpovídající místo (viz 1.1)

Pokud byl při vytváření prostředí předán grafický modul, tak se prostředí postará i o vykreslení vesmírných objektů. Pro všechny vesmírné objekty se zavolá příslušná metoda pro jejich vykreslení. Způsob implementace vykreslení jednotlivých objektů není zodpovědností herního prostředí.

Poslední věcí, kterou má prostředí ještě na starosti je kontrola, zda hra neskončila. Na konci kroku se zkontroluje, zda mají oba hráči kladný počet životů.

Jedna instance prostředí odpovídá jedné hře. Herní prostředí má dvě základní metody pro řízení hry.
Metoda *reset()* inicializuje hru do počátečního stavu a tento stav vrátí. Tato metoda se musí zavolat před začátkem hry.
A druhá metoda *next_step(actions_one, actions_two)*, která na na základě akcí hráčů, převede hru do následného stavu. Právě v této metodě je schovaná celá logika manipulace s vesmírnými objekty popsána výše.

```
def next_step(self, actions_one, actions_two):
    self.step_count = self.step_count + 1
    self.reward_one = 0
    self.reward_two = 0

    self.handle_actions(actions_one, actions_two)
    self.generate_asteroid()
    self.check_collisions()
    self.move_objects()
    if self.draw_modul is not None:
        self.render()

    (game_over, player_one_won) = self.check_end()
    if not game_over:
        self.reward_one += 1
        self.reward_two += 1

    current_state = State(self.asteroids_neutral,
                          self.rocket_one,
                          self.asteroids_one,
                          self.bullets_one,
                          self.rocket_two,
                          self.asteroids_two,
                          self.bullets_two)

    return self.step_count, \
           (game_over, player_one_won), \
           current_state, \
           (self.reward_one, self.reward_two)
```

2.2.1 Stav prostředí

Herní prostředí vrací po každém kroku současný stav hry. Stav hry se skládá ze seznamů všech vesmírných objektů včetně kompletních informací o nich. Pravděpodobně by bylo možné vracet i méně obsáhlou informaci o současném stavu hry. Avšak pro mě bylo motivací předat kompletní informaci o všech objektech a nechat následně až na agentech, na základě čeho všeho se budou chtít rozhodnout, jaké akce chtějí provést. Mou snahou bylo, aby herní prostředí poskytnulo všechny informace a neomezovalo tím agenty.

2.3 Agent

Agent je ústřední postavou celé hry a obzvláště v dalších kapitolách pro nás bude nejzajímavějším předmětem zájmu. Je to právě zde, kde budeme později mluvit o umělé inteligenci. V případě agenta, který je ovládán lidským hráčem, se agent, respektive člověk, který jej ovládá, neřídí datovou reprezentací stavu, tak jak jej obdržel od herního prostředí. Ale rozhoduje se na základě toho, jak hráč vizuálně vnímá co se děje ve hře. A příkaz k provedení jednotlivých akcí udává ovládáním kláves na klávesnici. Lidský hráč pro nás ale nebude tolik zajímavý k experimentování, my se budeme soustředit primárně na strojové agenty.

Každý agent musí implementovat jedinou metodu *choose_actions(state)*. Úkolem agenta je, na základě obdrženého stavu, zvolit akci, kterou chce provést. A právě tento rozhodovací problém pro nás bude půdou pro experimentování s různými abstrakcemi a přístupy umělé inteligence.

2.4 Grafické prostředí

Pro grafické zobrazování hry jsem zvolil python knihovnu pygame (`https://www.pygame.org/news`), která jak sám název napovídá slouží k programování jednodušších her v pythnu. Tato knihovna nabízí kromě grafických funkcí, také podporu pro manipulaci s herními objekty. Například je zde zabudovaná podpora pro kontrolu srážek. Mou snahou bylo této funkcionality využít, ale později se to ukázalo být nevhodné. Prvním problémem bylo, že herní objekty jsou v rámci této knihovny reprezentovány jako čtverce a kontrola srážek je tedy realizována jako dotaz, zda se dva čtverce pronikají. Kontrola zda se dva čtverce pronikají je výpočetně náročnější než průnik dvou kruhů. Jako větší problém se ale ukázala být integrace pygame modulu do herního prostředí. Při pokusu o paralelizaci více běhů her se objevily technické problémy, které se mi nepodařilo vyřešit. Proto jsem se rozhodl, že si kontrolu srážek implementuju separátně, nezávisle na modulu pygame a modul budu využívat pouze pro vykreslování hry. A pro toto použití je pro mě knihovna pygame zcela dostačující. Vytvořil jsem si sadu obrázků reprezentující jednotlivé vesmírné objekty. A knihovna pygame mi je umožňuje vykreslovat potřebným způsobem.

2.5 Hlavní herní cyklus

Vysvětlili jsme tedy všechny základní prvky, které v této hře potřebujeme a nyní je propojíme dohromady. K běhu hry potřebujeme mít instanci herního prostředí, tomu můžeme předat grafický modul, pokud chceme graficky vykreslovat, anebo žádnou implementaci nedodáme a pak hra bude běžet bez obrazu, tohoto budeme naším hlavním zájmem později v rámci trénování umělé inteligence. Dále potřebujeme inicializovat dva agenty, kteří budou poředstavovat naše hráče. A po inicializaci herního prostředí budeme v cyklu simulovat hru, dokud prostředí neoznámmí, že daným krokem hra skončila. V každém kroku cyklu agenti na základě stavu, ve kterém se herní prostředí nachází, zvolí své akce a ty předají zpět hernímu prostředí. Kostra herní simulace tedy vypadá následovně:

```
env = Enviroment(draw_module())
agent_one = Some_agent(1)
agent_two = Some_agent(2)


state = env.reset()
game_over = False
while not game_over:
    actions_one = agent_one.choose_actions(state)
    actions_two = agent_two.choose_actions(state)

    _, (game_over, player_one_won), state, _ = \
        env.next_step(actions_one, actions_two)
```

3. Senzory a akční plány

3.1 Motivace

V této kapitole se již přesouváme od hry samotné ke způsobu, jak na daný stav hry nahlížet strojově a také, jak na stav strojově reagovat. Na nejnižší úrovni dostávají agenti od prostředí stav, který obsahuje seznamy všech vesmírných objektů a agenti na to mají reagovat nějakou elementární akcí. Bylo by proto dobré vymyslet princip, jak z obsáhlých a detailních informací nízké úrovně získávat menší objemy potřebnějších informací vyšší úrovně. A podobně by se nám mohlo hodit místo elementárních akcí nízké úrovně, vymyslet princip jak volit akce tak, aby vedly k akcím vyšší úrovně. A právě tyto abstrakce realizujeme pomocí senzorů a akčních plánů.

3.1.1 Senzor

Senzorem nazveme metodu, která nám z kompletního stavu reprezentovaného seznamem vesmírných objektů extrahuje nějakou užitečnou informaci, která není ve stavu explicitně zadána. Informace získané z těchto senzorů, respektive senzorických metod, můžeme využít v rozhodovacím problému vybrání akcí. Většina senzorických metod využívá simulování hry. Na základě současného rozpoložení vesmírných objektů se v rámci simulace pokračuje v jejich pohybu, tak jak by se pohybovaly, pokud by žádný z hráčů neprováděl žádné akce. A na základě toho, co se v simulaci stane v nejbližších krocích hry, můžeme zjistit konkrétní informace, které platí o současném stavu hry. V simulaci žádný z hráčů neprovádí žádné akce, kromě těch, na jejichž dopad se v dané simulaci dotazujeme. Všechny simulace probíhají omezený počet kroků. Tento počet kroků je roven konstantě *IMPACT_RADIUS*. Pro tuto hodnotu se mi ukázal být vhodný počet 25. Je to dostatečně vysoká hodnota, aby senzory včas zaznamenaly potřebné informace a zároveň je dostatečně nízká, aby senzory nebyly zbytečně výpočetně náročné.

Příklady senzorů

- **První sražený neutrální asteroid - Zde se simuluje pohyb vesmírné lodi, střel a neutrálních asteroidů. V případě, že v simulaci dojde k srážce vesmírné lodi a neutrálního asteroidu, vrací tento senzor daný asteroid a počet kroků, po kterém došlo ke srážce. Berou se zde v úvahu i vlastní střely. Pokud dojde k sestřelení asteroidu střelou, jsou jak asteroid, tak i střela odstraněny ze simulace.**
- **První sražený nepřátelský asteroid - Jde o téměř identický senzor, jen s rozdílem, že se nesoustředí na asteroidy neutrální, ale na asteroidy nepřátelské.**
- **Asteroid zasáhne nepřátelskou loď - Zde se simuluje pouze pohyb konkrétního asteroidu a nepřátelské lodí. Opět je zde nastaven daný limit na počet simulovaných kroků. Senzor vrací informaci**

zda, a v kolika krocích se střetl s nepřátelskou lodí. V simulaci nepřátelská loď neprovádí žádné reakce.

- Střela zasáhne konkrétní asteroid - Jde o simulaci podobnou předchozí. Simuluje se pohyb konkrétního asteroidu a konkrétní střely.
- Střela zasáhne libovolný asteroid - Simuluje se pohyb konkrétní střely a všech neutrálních a nepřátelských asteroidů. Tato senzorická metoda vrací zda střela zasáhla sestřelila asteroid, daný asteroid a počet kroků, po kterém střela sestřelila asteroid.
- Vzdálenost dvou bodů - Vrátí vzdálenost vyjádřenou v Euklidovské metrice.
- Přepočítání nejbližší polohy asteroidu od vesmírné lodi - Vzhledem k tomu, že prostor je jistým způsobem cyklický, tak v případě, že nás zajímá nejkratší vzdálenost asteroidu od vesmírné lodi, tak přímá vzdálenost těchto objektů, tak jak jsou graciky objekty zobrazeny v herním prostoru, nemusí být nejkratší. Musíme vzít v úvahu všechny čtyři polohy asteroidu, které získáme postupným posunutím asteroidu o šířku a délku prostoru. Jinak řečeno, vzdálenost souřadnice asteroidu posunutého přes hranici prostoru může být kratší než přímá vzdálenot k původní souřadnici asteroidu.
- N nejblizších asteroidů od vesmírné lodi - Tento senzor spočítá nejkratší vzdálenosti vesmírné lodi ke všem asteroidům ve hře a vrátí relativní polohu N nejbližších z nich k vesmírné lodi.

3.1.2 Akční plán

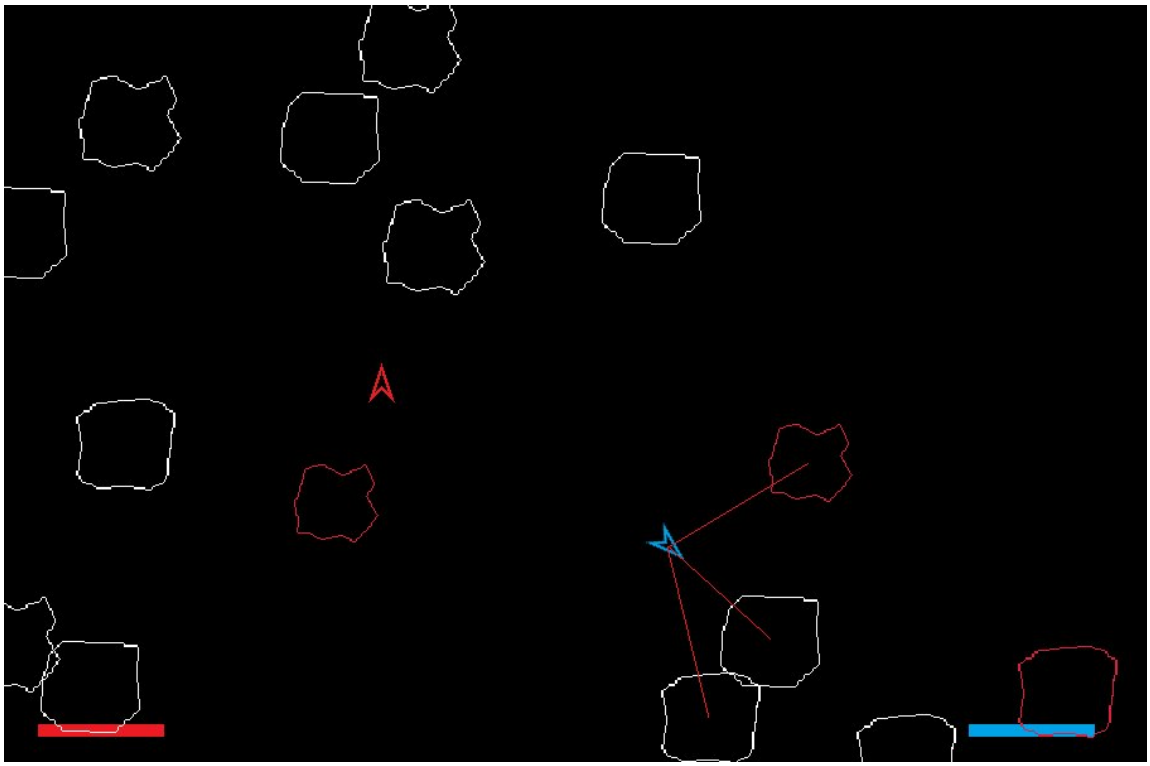
3.2 Jednotliv plány

3.2.1 Útok

3.2.2 Sestřelující obrana

3.2.3 Úhybná obrana

3.2.4 Zastavení letu



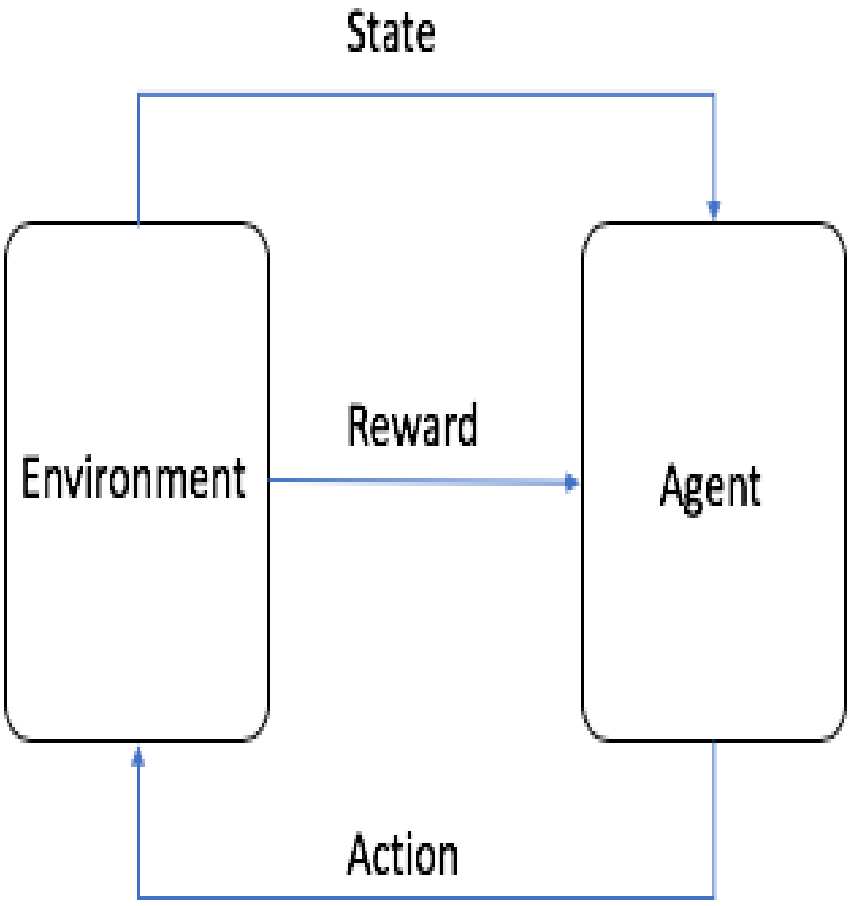
Obrázek 3.1: Ukázka senzoru "N nejbližších asteroidů od vesmírné lodi"pro N=3

4. Genetické programování

4.1 Základní princip

4.2 Využití

4.3 Aplikace



Obrázek 4.1: Herní cyklus

5. Hluboké Q-učení

5.1 Základní princip

5.2 Využití

Kde se používá v praxi.

5.3 Aplikace

Jak jsem to použil já a jakých výsledků jsem dosáhl.

6. NEAT

6.1 Základní princip

6.2 Využití

Kde se používá v praxi.

6.3 Aplikace

Jak jsem to použil já a jakých výsledků jsem dosáhl.

Seznam obrázků