Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

# BAKALÁŘSKÁ PRÁCE

Multiagentní systém modelující dvě soupeřící frakce

2020

Martin Pifka

Vedoucí práce: Mgr. Osička Petr, Ph.D.

Studijní obor: Aplikovaná informatika, prezenční forma

**Bibliografické údaje**

| | |
|---|---|
| Autor: | Martin Pifka |
| Název práce: | Multiagentní systém modelující dvě soupeřící frakce |
| Typ práce: | bakalářská práce |
| Pracoviště: | Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci |
| Rok obhajoby: | 2020 |
| Studijní obor: | Aplikovaná informatika, prezenční forma |
| Vedoucí práce: | Mgr. Osička Petr, Ph.D. |
| Počet stran: | 32 |
| Přílohy: | 1 CD/DVD |
| Jazyk práce: | český |

**Bibliograhic info**

| | |
|---|---|
| Author: | Martin Pifka |
| Title: | Multiagent system modelling two competing groups |
| Thesis type: | bachelor thesis |
| Department: | Department of Computer Science, Faculty of Science, Palacký University Olomouc |
| Year of defense: | 2020 |
| Study field: | Applied Computer Science, full-time form |
| Supervisor: | Mgr. Osička Petr, Ph.D. |
| Page count: | 32 |
| Supplements: | 1 CD/DVD |
| Thesis language: | Czech |

## Anotace

*Práce představuje stručný úvod do problematiky multiagentních systémů, implementaci MAS simulátoru modelující dvě navzájem soupeřící frakce, popis zacházení se simulátorem a technologie použité při vývoji.*

## Synopsis

*The work presents an introduction to the issue of multiagent systems followed by the implementation of MAS simulator which models two independent rival factions and a description of the functionality of the simulator and the technologies used in growth.*



**Klíčová slova:** MAS; multiagentní systém; simulace; genetický algoritmus

**Keywords:** MAS; multiagent system; simulation, genetic algorithm

Děkuji vedoucímu práce Mgr. Petru Osičkovi, Ph.D za pomoc a trpělivost při mém dotazování.

*Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.*

datum odevzdání práce podpis autora

# Obsah

# Seznam obrázků



# Seznam tabulek

# Seznam vět

# Seznam zdrojových kódů

# 1 Úvod

Multiagentní systémy jsou vcelku nevšedními přístupy k řešení informatických problémů. Kromě samotné jejich funkční stránky se po grafickém znázornění a nastavení vhodných podmínek může uživatel dočkat fascinující podívané, stejně tak jako je tomu u celulárních automatů či Lindenmayerových systémů. Jednotliví agenti i když nepříliš sofistikovaní mohou dohromady vytvářet dění, u kterého by pozorovatel očekával nutně mnoho složitých procesů na pozadí. Výsledkem jsou často simulace věrně napodobující jednoduché organismy, jako jsou kolonie mravenců, nebo biologických jevů, jako šíření nemocí v populaci. Mě samotného tento fenomén fascinoval a pokusil jsem se v této práci vytvořit multiagentí systém, jenž nenapodobuje konkrétní prvek reálného světa, avšak rozděluje agenty do dvou soupeřících skupin, jenž se interakcí s okolím dozvídají, které aktivity jsou pro ně užitečné a sami si postupem času vytváří plány, které se snaží optimalizovat.

# 2 Základní pojmy

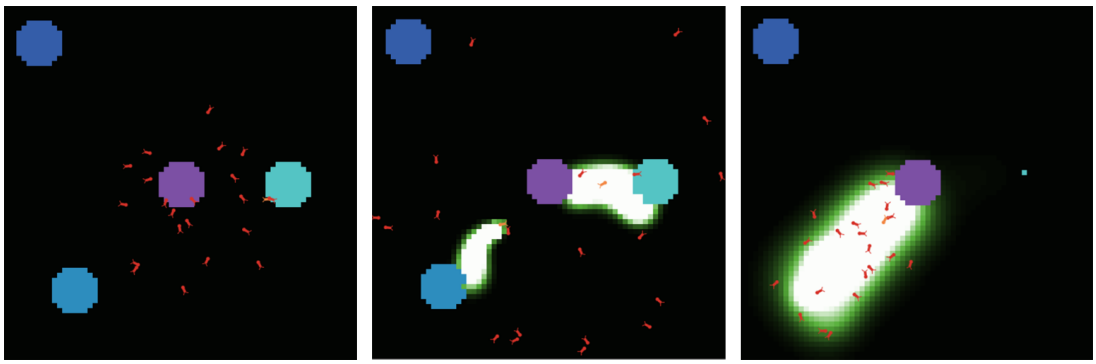
## 2.1 Multiagentní systém (MAS)

*Multiagentní systém je možno popsat jako skupinu diskrétních volně propojených autonomních systémů (agentů), které spolupracují v zájmu dosažení společného cíle.*[2] Jsou využívány k simulování sociálních, nebo přírodních jevů, ale také jsou často zkoumány i v oblasti umělé inteligence. Je vhodné je využít v místech, kde systém už není možné, nebo by nebylo vhodné řešit jako monolitický. Multiagentní systém pak umožňuje dekompozici systému na relativně dílčí úlohy.[3]

## 2.2 Reaktivní agent

Tento typ agentů jedná vždy na základě svého prostředí. Nemají žádný vnitřní model svého prostředí a nemohou ani vytvářet plány. Mohou však uchovávat historii svých stavů. Jednotlivého agenta pak lze reprezentovat jako automat.[4]

Klasickým zástupcem reaktivních agentů je model kolonie mravenců snažících se obstarat si potravu. Po nalezení zdroje potravy za sebou mravenci zanechávají pachovou stopu jenž časem mizí. Ostatní mravenci kteří trpí nedostatkem potravy se pak vydávají po této stopě a po vyčerpání zdroje pachová stopa zanikne. Ačkoliv každý jednotlivý mravenec jedná podle několika jednoduchých pravidel, pak jako celek vykazují vcelku komplexní jednání.



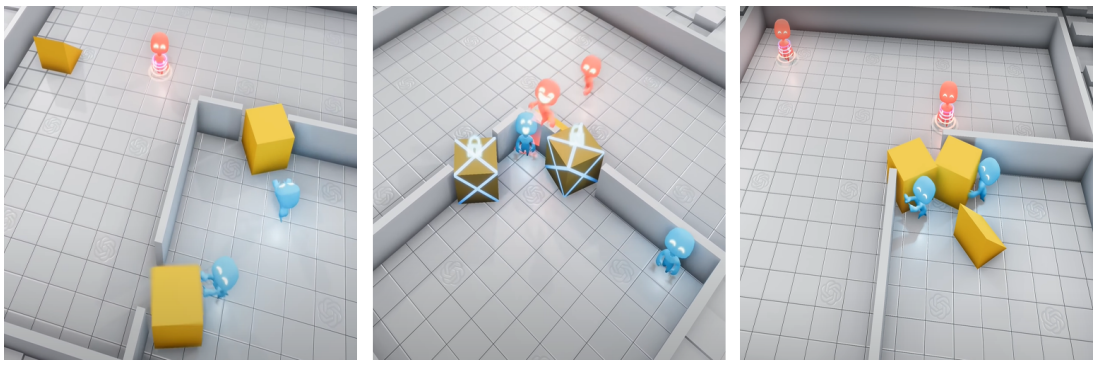
Obrázek 1: Simulace Ant colony v simulátoru NetLogo, hledání, tvorba cesty, spotřebování zdroje

## 2.3 Deliberativní agent

Deliberativní agent, též také uvažující agent. Uchovává vědomosti o svém prostředí a využívá jich pro dosažení svého cíle. Jeho stavy ve kterých se může nacházet nesou ohodnocení, která ovlivňují tvorbu plánů.[4]

Příkladem může být model Hide and Seek kde se nacházejí dvě soupeřící skupinky agentů, přičemž jedna začíná s předstihem a snaží se ukrýt. Druhá skupinka

usiluje o odhalení agentů první skupiny. Agenti mohou k dosažení svých cílů využívat prostředí, které se skládá z pohyblivých i nepohyblivých částí. Agenti se časem učí, jak efektivněji dosáhnout svého cíle, například tím, že prostor kolem sebe uzavřou, avšak na to opačná skupina může zareagovat využití pohyblivého bloku ve tvaru rampy, pomocí něhož se dostanou přes toto opevnění. Postupem času a pokusů se agenti mohou naučit schovávat tento pohyblivý blok do svého opevnění a tím znemožnit opačné straně využití tohoto objektu.



Obrázek 2: Hide and Seek, uzavření prostoru, využití rampy, schování rampy [1]

# 3 Použité technologie

## 3.1 C++

C++ je multiparadigmatický programovací jazyk. Podporuje několik programovacích stylů jako je procedurální programování, objektově orientované programování a generické programování.[5]

Tento programovací jazyk byl vybrán s ohledem k často vyzdvihované rychlosti, jelikož u programu jsou předpokládány vyšší nároky na výkon.

## 3.2 SDL2

SDL2 je multiplatformní knihovna, která se nejčastěji využívá ke tvorbě her. Knihovna poskytuje nízkoúrovňový přístup ke vstupně-výstupním zařízením a to i 2d a 3d grafice. V této práci si vystačím s použitím 2D grafiky. Knihovna samotná obsahuje pouze základní funkcionalitu, pro práci s textem či obrázky je třeba použít přídavných knihoven jako SDL_image, SDL_ttf.[6]

## 3.3 Visual Studio 2019

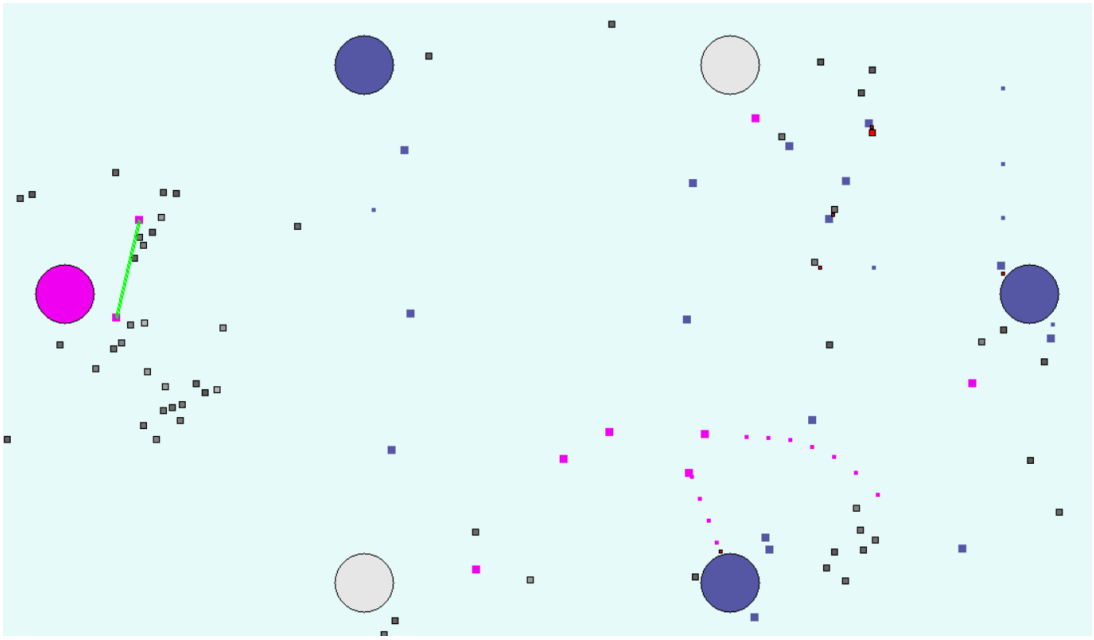
Celý vývoj aplikace probíhal v prostředí Visual Studia 2019. Visual studio je vývojové prostředí (IDE) od společnosti Microsoft. Muže být použito pro vývoj konzolových aplikací a aplikací s grafickým rozhraním. [7]

# 4 Multiagentní systém modelující dvě soupeřící frakce

V této simulaci se agenti dělí do dvou frakcí. Obě frakce začínají souběžně. Na počátku mají jen omezené množství informací. Interagováním s prostředím se učí novým činnostem a jim přiřazené relevantnosti. Agenti mají své plány, zpočátku náhodně sestavené, avšak postupem času dochází k jejich optimalizaci genetickým algoritmem pro maximalizaci získání bodů ze svého okolí. Vítězem se stává frakce vlastnící všechny věže na mapě.

## 4.1 Mapa

Jedná se o ohraničenou plochu na niž se mohou vyskytovat jednotlivé herní objekty. Věže jsou na mapě zasazeny staticky. Ostatní objekty mají možnost se na mapě pohybovat, vznikat a zanikat.



Obrázek 3: Mapa

## 4.2 Věž

Objekt reprezentovaný kruhovým tvarem, nesoucí barvu frakce, jenž ji v aktuální moment vlastní. Věž může být vlastněna jednou z frakcí agentů, nebo být neutrální, náležející frakci C, jenž reprezentuje neutrální prvky. Na mapě se vyskytuje celkem šest věží. Jedna věž náležící od počátku první frakci agentů (frakce A),

jedna druhé frakci (frakce B) a všechny ostatní jsou věžemi neutrálními (frakce C).

Věže slouží jako prostor pro oživování. Zasažený agent se tak znovuzrodí u nejbližší vlastněné věže. Pokud agent dané frakce nevlastní žádnou věž, pak se znovuzrodí uprostřed mapy.

Počet vlastněných věží ovlivňuje jakou délku agent dané frakce bude muset počkat, než se znovu oživí. Za každou vlastněnou věž se délka čekání snižuje o 1/6 z maximální délky smrti (ta je zvolena uživatelem). Vyhraje ta frakce, která v jeden moment vlastní všechny věže na mapě (v potaz není brána neutrální frakce C).

## 4.3 Zeď

Zeď je agenty tvořený nepohyblivý objekt. Vždy náleží frakci C, nehledě na to, kterým agentem byla vytvořena. Zeď má několik životů, snese tak více zásahů, než je zničena. Může sloužit jako překážka v cestě, či jako obrana před střelami.

## 4.4 Střela

Je mobilní objekt pohybující se směrem od agenta, jenž jej vytvořil a pevně určil jeho cíl. Střela je vlastněna frakcí agenta, jenž ji vytvořil. Střela zaniká v případě, že dosáhne okraje mapy, či narazí do jiného objektu, než je agent stejné frakce, jako je ona sama. V případě nárazu do objektu je předána zpráva, na niž objekt následně zareaguje.

## 4.5 Interakce mezi agenty

Interakci mezi agenty je znázorněna barevnou úsečkou mezi aktéry komunikace. Tato úsečka nijak nezasahuje do běhu hry a není překážkou pro jakýkoliv objekt. Jedná se pouze o grafické znázornění probíhající komunikace.

- Zelená – pokus o sdílení informace
- Modrá – pokus o sdílení plánu
- Rudá – rozkaz

## 4.6 Agent

Každý agent má od počátku hry pevně dáno, ke které frakci náleží (A, nebo B). Jeho aktivity jsou podmíněny jeho znalostmi. Agent po zasažení střelou nepřátelské frakce je zabit a znovuzrozen po době v závislosti na počtu vlastněných věží. Během svého života si agent vytváří plány, kterými se snaží o získání co možná nejvíce bodů ze svého okolí.

### 4.6.1 Zorné pole agenta

Jedná se o rádius okolo agenta. Jestliže se v jeho prostoru nachází nějaký objekt, agent je informován o typu objektu, jeho souřadnicích a příslušnosti frakce. Dále je také informován o událostech dějících se v tomto prostoru. Pokud je v jeho zorném poli zasažen agent nepřátelské frakce je mu předána informace s popisem události i bodovým ohodnocením vzhledem k frakci agenta.

### 4.6.2 Odhalené věže

Mimo objekty v zorném poli si agent uchovává také znalosti o jím objevených věžích. Pokud na nějakou narazí, pak si poznačí její souřadnice a frakci. V případě, že by v jeho aktuálním cíli figurovala věž této frakce, pak se nesnaží naleznou novou věž, ale jde k souřadnicím které si dříve zaznamenal, pokud se ukážou jeho informace jako zastaralé (věž byla během doby jeho nepřítomnosti dobyta nepřátelskou frakcí), pak si své údaje o věži aktualizuje.

### 4.6.3 Informace

Jedná se o množinu dat nasbíraných ze svého prostředí, se kterými se buď zrodil, nebo mu byly předány během jeho existence. Agent o své vědomosti nemůže žádným způsobem přijít a to ani jeho smrtí.


Obrázek 4: Informace o aktivitě náhodný výstřel

- Složka dat - určuje o jakou činnost se jedná (v tomto případě o výstřel náhodným směrem)
- Body - množství bodů ovlivňuje za jak užitečnou agent aktivitu považuje a s jakou pravděpodobností bude činnost přidána do jeho plánu, počet bodů jednotlivých činností lze nastavit v prostředí simulace uživatelem, a tak určit ke kterým činnostem bude agent náchylnější
- Souřadnice - struktura Informace může nést informaci o tom, kde se objekt nachází, nebo se něco událo (v informaci o náhodném výstřelu se tento údaj nevyužívá)
- Hodnota - je číselná informace, která nemá jednoznačné určení, může nést informaci o tom kolik kroků musí agent čekat, nebo i kolikrát má na objekt vystřelit

### 4.6.4 Instinkty

Jedná se o informace s nimiž se agent zrodí. Zajišťují agentovo základní chování, díky němuž bude moct interagovat s okolí.
Patří sem:

- Pohyb na náhodné souřadnice
- Výstřel na náhodné souřadnice
- Sdílení informací s jiným agentem téže frakce
- Sdílení plánu s agentem téže frakce
- Stavění zdí na náhodné souřadnice

Pokud by si uživatel přál, aby agent nevyužíval některé znalosti, stačí aby jí jako bodové ohodnocení nastavil nulu. V takovém případě se bude agent domnívat, že aktivita je zbytečná a nebude se ji snažit provádět. Jde tak nastavit aby agenti vůbec nestavěli zdi, nebo spolu nesdíleli informace.

### 4.6.5 Další znalosti

Jak již bylo zmíněno, agent nové informace získává z událostí odehrávajících se v jeho zorném poli, nehledě na tom, zda se na ni nějakým způsobem podílel. Informace, jenž může získat postupem času jsou:

- Stavění opevnění ze zdí poblíž vlastněných věží
- Ničení opevnění poblíž nepřáteli vlastněných věží
- Zásah nepřítele
- Zničení Nepřítele
- Zásah nepřátelské věže
- Zničení nepřátelské věže
- Zásah neutrální věže
- Zničení neutrální věže

Rozdíl mezi zásahem a zničením je v tom, že při zásahu agent vystřelí střelu jednu, či více směrem na souřadnice daného objektu a tímto je cíl splněn. U zničení se agent i přesvědčí, že byl skutečně zničen, pokud tomu tak není, výstřel bude opakovat.

### 4.6.6 Plán

Každý agent vlastní tři plány. Plán se skládá z posloupnosti proměnlivého počtu cílů. Jednotlivým cílem je aktivita ze znalostí agenta. Pseudokód tvorby prvotního plánu:

```
function TvorbaPlanu(delkaPlanu, informace)
    for i ← 0, i ≠ delkaPlanu, i ← i + 1 do
        novyCil ← VratCil(informace)
        plan ← PridejNaKonecPlanu(plan, novyCil)
    end for
    return plan
end function

function VratCil(informace)
    informace ← KladneBodovaneInformace(informace)
    tabulka < informace, body >
    for i ← 0, i ≠ Pocet(informace), i ← i + 1 do
        ntaInformace ← NtaInformace(i, informace)
        bodyNteInformace ← BodyNteInformace(i, informace)
        VlozDoTabulkyRadek(ntaInformace, bodyNteInformace)
        SnizeniBoduPodleVyskytuVjinychPlanech(tabulka)
        SnizeniBoduPodleVyskytuVeStejnemPlanu(tabulka)
    end for
    return VyberCileZTabulky(tabulka)
end function

function VyberCileZTabulky(tabulka)
    sumaBodu ← SumaBodu(tabulka)
    nahodnaHodnota ← Random(0, sumaBodu)
    radek ← PrvniRadek(tabulka)
    acum ← 0
    while radek <= KonecTabulky(tabulka) do
        acum ← acum + BodoveOhodnoceniCile(radek)
        if nahodnaHodnota <= acum then
            return CilVRadku(radek)
        else
            radek ← DalsiRadek(tabulka)
        end if
    end while
end function
```

Funkce SnizeniBoduPodleVyskytuVjinychPlanech a SnizeniBoduPodleVyskytuVeStejnemPlanu snižují bodové ohodnocení cílů v závislosti

na tom, zda jsou již obsaženy v jiných plánech, nebo počtu výskytu v daném plánu.

Každý plán má i své ohodnocení, které je založeno na délce jeho vykonávání a počtu bodů získaných z okolí během jeho vykonávání. Po určité době se vždy odečte několik bodů od plánu, čímž se zajistí, že je přihlíženo i k délce plnění plánu.

Agent na začátku svého tahu přečte zprávy, které obdržel, přičemž tyto informace nesou vždy nějaké bodové ohodnocení. Po přečtení zprávy tyto body přičte k bodům vykonávaného plánu. Pokud nalezne ve zprávách informaci o zásahu nepřátelské věže přátelským agentem (třeba i jím samým) a uživatel ohodnotil tuto aktivitu relevancí deseti bodů, pak přičte k bodovému ohodnocení plánu deset bodů, pokud obdrží informaci o zasažení přátelské věže nepřátelským agentem, pak tato informace nabývá záporné hodnoty, tedy mínus desíti bodů, které „přičte“ k bodům vykonávaného plánu.

Jelikož počáteční plány nemají podklad na jehož by bylo ohodnocení vytvořeno, pak nabývají nulovému ohodnocení.

### 4.6.7 Životní cyklus plánu

Pokud agent momentálně neplní žádný plán, pak si náhodně vybere jeden z jeho tří plánů. Při každém výběru nového plánu dochází ke snížení bodového ohodnocení všech plánů o 3%. Míra pravděpodobnosti výběru plánu je rovnoměrná, nehledě na jejich ohodnocení. Po zvolení plánu se vytvoří jeho kopie a provede se mutace.

Agent se snaží plnit cíle v pořadí stejném jako v plánu (zleva doprava). Může se stát, že se agentovi nebude dlouhodobě dařit dosáhnout nějakého cíle, například nebude moci naleznout nepřátelskou věž, v takovém případě od cíle upustí a tento cíl přeskakuje. Pokud se v agentově zorném poli nachází objekt figurující v některém z jeho informací, pak je možné, že si do plánu přidá cíl obsahující tento objekt. Například agent snažící se postavit opevnění poblíž vlastněné věže, jenž narazí na nepřátelského agenta, přičemž je mu známo, že střelba do nepřátelských agentů je pro něj výhodná, v takovéto chvíli se může před jeho aktuální cíl vložit právě střelba na tohoto agenta a stavění opevnění je tímto odsunuta na později. Pravděpodobnost je závislá na bodovém ohodnocení informace, ve které objekt figuruje.

Ve chvíli kdy jsou všechny cíle plánu splněny (nebo přeskočeny), pak dojde k porovnání bodového ohodnocení plánu před zmutováním a po zmutování, jestliže plán zmutovaný má lepší ohodnocení, pak si plán původní (nezmutovaný) nahradí novým (zmutovaným), v opačném případě se zachovává starý plán a nový je zahozen.

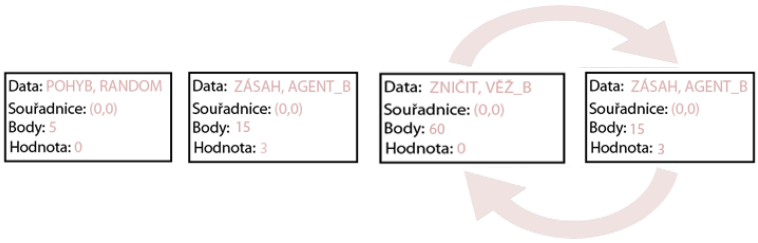
Jako další prostředek pro organizaci agentů je možnost udělení přímého rozkazu. Agent jenž zahyne při aktivitě, která je dle jeho informací hodnocena výrazně lépe, než jeho ostatní činnosti, pak po jeho oživení přikáže agentům v jeho

okolí, aby se společně vydali na souřadnice jeho smrti a ostatní agenti tak sloužili jako podpora při pokusu o znovudosažení tohoto cíle. Nejdříve, kdy agent nemá příliš znalostí, může považovat za takto užitečnou aktivitu zničení nepřátelského agenta, později ale spíše zničení nepřátelské věže. Pokud však budou ohodnocení aktivit uživatelem nastaveny příliš rovnoměrně, agent bude považovat všechny aktivity za podobně relevantní a nebude tedy považovat žádnou aktivitu natolik výjimečnou, aby uděloval rozkaz pro její dosažení.

### 4.6.8 Mutace

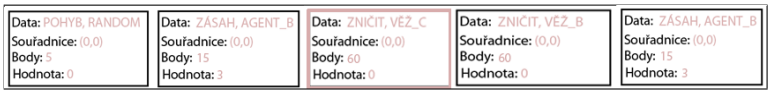
jedná se o úpravu části plánu, mezi než patří:

- `Prohození pozice dvou cílů` - jsou vybrány dva cíle plánu, jejichž pozice se navzájem vymění.
- `Odebrání jednoho z cílů` - jeden z cílů plánu je odebrán
- `Přidání jednoho cíle` - na náhodně vybranou pozici se do plánu přidá cíl vybraný algoritmem VratCil
- `Změna jednoho cíle` - jeden z cílů plánu bude nahrazen cílem který se vybere pomocí funkce VratCil

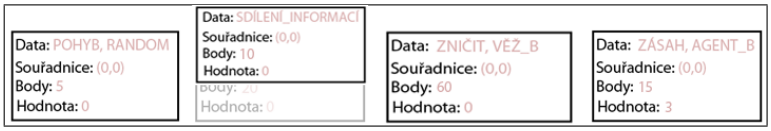



Obrázek 5: Prohození pozice dvou cílů


Obrázek 6: Odebrání jednoho z cílů




Obrázek 7: Přidání jednoho cíle




Obrázek 8: Změna jednoho cíle

# 5 Programátorská příručka

Kvůli velkému počtu tříd i jejich členů, byly vybrány jen ty nejstěžejnější části.

## 5.1 Třída Mapa

Mapa je herní plochou simulace. V tomto prostoru se vyskytují všechny objekty simulace (prvky gui jako třeba tlačítka sem nepatří). Při pohybu jednotlivých objektů, či jejich interakcí s okolím, je třeba neustále zjišťovat, zda tímto počinem budou ovlivněny i jiné objekty. Například samotný pohyb je dlouhý jeden px, pro překonání poloviny mapy nezměněné velikosti, tedy 1100px jedním objektem, je třeba 550krát zjistit, zda došlo ke kolizi s jiným objektem a tak pokaždé zkontrolovat souřadnice i všech ostatních objektů na mapě. Simulace by takto neměla šanci plynule běžet při vyšším počtu objektů na mapě. Z tohoto důvodu se mapa dělí na jednotlivé segmenty. Velikost jednotlivých segmentů je možné zvolit v konstruktoru třídy Mapa. Jednotka je záměrně zvolena v pixelech, pokud by se uváděl počet segmentů pak by jejich velikost byla závislá na velikosti samotné mapy, což není žádoucí vzhledem k tomu, že v efektivitě hraje pouze roli jak velký je samotný segment nehledě na velikost mapy.

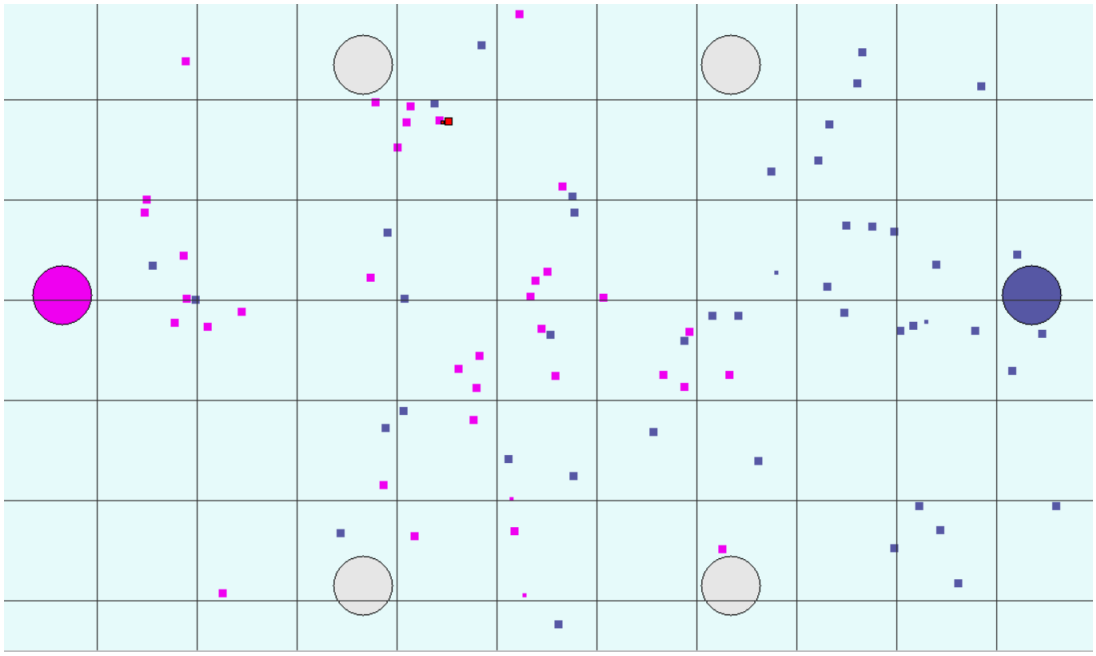
- `UvnitrMapa` – predikát určující zda objekt při daných souřadnicích není mimo mapu, metoda je přetížena tak, aby sloužila více typům objektů, například agenta bude zajímat zda se celý jeho povrch bude vyskytovat na mapě, nikoliv jen jeden jeho bod
- `AgentiVRadiusu` – Metoda vracející list agentů, kteří se na mapě nacházejí v zadaném rádiusu, existují obdobné metody jako ZdiVRadiusu, StrelyVRadiusu, VezeVRadiusu
- `NajdiSegmentRect` – vrací list segmentů na nichž se objekt nachází

## 5.2 Třída Segment

Jedná se o dílek mapy, jenž v sobě nese informaci, které všechny objekty se na něm vyskytují a to byť jen z části. Každý objekt si nese list ukazatelů na segmenty ve kterých se sám nachází a současně je i on sám obsažen v ukazatelích těchto segmentů. Při pohybu objektu se aktualizuje na kterých segmentech se objekt vyskytuje a následně se zkontroluje zda nedošlo ke kolizi s některým z objektů ve stejných segmentech. Tímto není nutné vždy kontrolovat všechny objekty na mapě, ale pouze ty v jeho okolí. Pokud se vchází do nového segmentu, pak je třeba zkontrolovat i objekty tohoto segmentu.

## 5.3 Abstraktní třída Zaklad

Obsahuje základní prvky nutné pro správné zasazení do simulace.



Obrázek 9: Segmenty mapy

- `frakce` – ke které frakci objekt náleží
- `zivoty` – počet životů
- `aktualniSegmenty` – list segmentů na kterých se objekt momentálně vyskytuje (v případě úmrtí či pohybu je objekt z těchto segmentů odebrán a vyprázdní se list `aktualniSegmenty`)
- `schranka` – vektor struktur `Informace`, do této schránky jsou agentovi předávány informace o jeho dění v okolí a to včetně informace o jeho zásahu, na což následně zareaguje v závislosti na jeho povaze
- `VlozeniDoSegmentu` – čistě virtuální metoda, kterou si děděný objekt musí přepsat, tak aby se objekt vkládal do správného listu segmentu, agent se tak bude vkládat do listu agentů
- `OdebraniZeSegmentu` – obdobně jako u VlozeniDoSegmentu, avšak s tím rozdílem, že se objekt z listu odebere
- `PridaniZpravy` – předání informace jinému objektu do schránky

## 5.4 Struktura Informace

Prostředek oznamující objektům dění v jejich okolí a také nesení definic jejich cílů. informace má strukturu popsanou v Informace, přičemž složka dat je vektor

výčtového typu (enum) *DataInformace* která zahrnuje: `STRELA, AGENT, AGENT_A, AGENT_B, ZED, VEZ, VEZ_A, VEZ_B, VEZ_C, SOURADNICE, TENTO, HODNOTA, NARAZ, RANDOM, POHYB, SDILENI_INFORMACI, POBLIZ, CEKANI, OZIVENI, ZNICENI, SDILENI_PLANU, ROZKAZ`
Uspořádáním výčtových typů do specifické posloupnosti informace vyjadřuje událost, či přímo akci.

## 5.5 Třída Agent

Je potomek abstraktní třídy Zaklad, jedná se o nejobsáhlejší objekt simulace.

- `velikostZornehoPole` – neznaménkové celé číslo nesoucí informaci o velikosti zorného pole agenta
- `mrtvy` – boolean vyjadřující zda v aktuální chvíli je agent mrtev
- `listAgentu` - statický člen obsahující ukazatele na všechny existující agenty
- `ZpracovaniZprav` – prohlédnutí své poštovní schránky a zpracování každé zprávy, zde se můžou nacházet informace o dění v okolí, pokud agent zaznamenává, že se udála aktivita kterou ještě nezná a je bodově kladně hodnocena, přidá si ji do své trvalé množiny informací a následně může tuto aktivitu zařadit do svého plánu
- `VyberNovehoPlanu` – je spuštěna ve chvíli, kdy agent dokončí, jím dříve vybraný plán, nejdříve se všem existujícím plánům sníží hodnocení o 3%, následně se rozhodne, zda vykonaný plán byl lépe hodnocen před mutací, či po ní, v závislosti na tom dojde k jeho zachování, či zahození a až následně se vybere nový plán, u kterého taktéž nejdříve dojde k mutaci
- `RozhlednutiSe` – agent získá přehled o objektech v jeho okolí, pokud je v tomto okolí věž, dojde k poznamenání její frakce a souřadnicích, tento údaj přetrvá i po ztracení věže ze zorného pole
- `KrokKeSplneniCile` – provedení tahu v závislosti na aktuálním cíli

## 5.6 Třída Planovani

Aby třída agent nebyla příliš objemná, byla část funkcionality separována do samostatné třídy.

- `informace` – list všech aktivit kterých si je agent vědom
- `veze` – list nesoucí informace o všech dosud nalezených věžích
- `plany` – třída obsahující všechny plány agenta a implementující samotné mutace

- `aktualniPlan` – právě vykonávaný plán
- `aktualniCil`- jednotlivý cíl plánu o jehož dosažení se aktuálně usiluje, například aktivita zničení věže zapříčiní vložení do čela listu cíl výstřel na souřadnice věže, případně samotné hledání věže

# 6 Uživatelská příručka

Jedná se o popis instalace a návod použití simulátoru. Pro jeho použití není třeba odborných znalostí a s použitím programu by neměl mít problém ani běžný uživatel počítačového zařízení.

## 6.1 Instalace

Nejdříve je třeba program nainstalovat pomocí Instalace.msi. Po zvolení cílové složky a následné instalaci je na ploše uživatele vytvořena ikona spustitelného programu. Tímto jsou do cílového zařízení přidány jednotlivé komponenty nutné pro jeho běh.

## 6.2 Prostředí aplikace

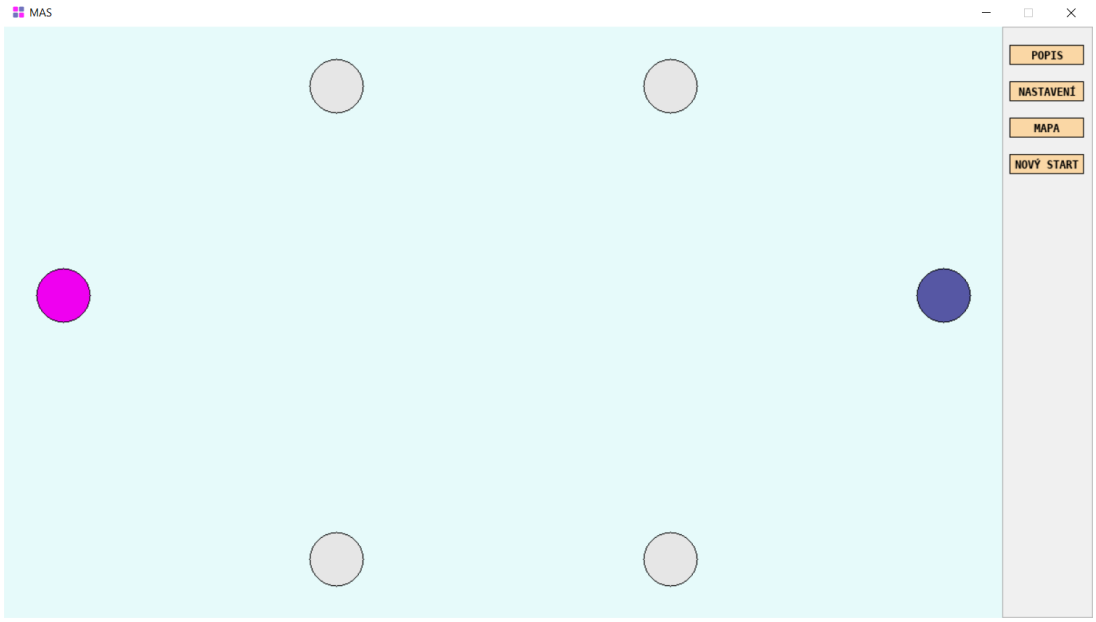
Nativní plochou aplikace je herní mapa. Na ni se odehrává samotná simulace. V sidebaru jsou pro výběr tlačítka:

- `Mapa` – po kliknutí se zobrazí herní plocha
- `Popis` – obsahuje stručný popis MAS, fungování parametru *maximální délka smrti* a významu jednotlivých barev úseček mezi agenty
- `Nastavení` – zde uživatel může pozměnit bodové ohodnocení aktivit, každý vstup má své limitní omezení, obvykle 100 bodů

  Hodnoty se zadávají vždy kladným celým číslem, jiné znaky či hodnoty vyšší, než je limit není umožněno vkládat. Bodové ohodnocení určuje jakou váhu bude agent přisuzovat jednotlivým aktivitám. Je třeba mít na paměti, že agent si dělá úsudek o efektivnosti jednotlivých aktivit z poměru vůči bodovým ohodnocením ostatních aktivit. Pokud budou všechny aktivity hodnoceny maximální hodnotou sta bodů, nebude rozdíl v případě, kde jsou všechny aktivity ohodnoceny jedním bodem, jelikož poměr jednotlivých aktivit zůstává u všech stejně velký.

  *Maximální délka smrti* určuje počet kroků, které agent stráví, než se bude moci opět oživit. Délka této doby pro něj klesá o 1/6 za každou vlastněnou věž.

- `Nový start` – spustí simulaci dle aktuálního nastavení parametrů v části *Nastavení*, po kliknutí se plocha automaticky navrátí do nativní plochy *Mapa*, kde je možné pozorovat průběh simulace. Pokud je stisknuto tlačítko *Popis*, či *Nastavení* během probíhající simulace, je její průběh pozastaven a po návratu na herní plochu se simulace opět uvede v chod. Probíhající simulaci je možné přerušit opětovným spuštěním nového startu.




Obrázek 10: Gui

# 7 Pozorování

Agenti byli vystaveni specifickým podmínkám a na pozorování jejich jednání se opírá následující část textu. Některé tyto podmínky není možné aby prováděl i běžný uživatel, jako je třeba zachování znalostí i do dalšího spuštění simulace, pro takovéto podmínky je třeba pozměnit zdrojový kód programu. Všechny následující simulace měly stejné bodové ohodnocení, jako je výchozí při startu aplikace, pokud není v textu uvedeno jinak.

## 7.1 Zachování informací i plánů

- Frakce A: Agenti si zachovají informace i své plány do dalšího spuštění simulace
- Frakce B: Agenti se jako obvykle zrodí pouze s instinkty a náhodně sestavenými plány
- Odměny za jednotlivé aktivity jsou u obou frakcí stejné

První kolo nebylo nijak důležité, jelikož zde si agenti byli s podmínkami rovni. V následujícím kole již stála frakce A s nabytími informacemi a upravenými plány proti frakci B, která začínala zcela od začátku. Toto kolo a i ta následující měla rychlý průběh, přičemž frakce, která si mohla své znalosti ponechat drtivě dominovala nad frakcí, jenž se ve svých počátcích učila, které aktivity jsou prospěšné, natož aby si utvořili efektivnější plány.

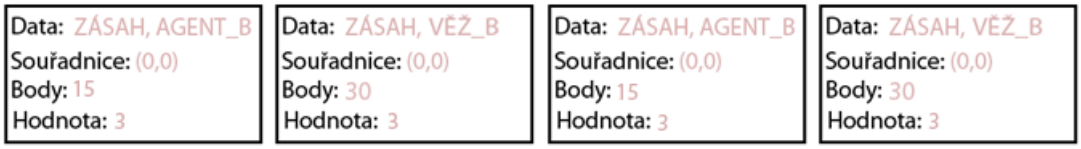
## 7.2 Zachování plánů

- Frakce A: Agenti si zachovají plány do dalšího spuštění simulace
- Frakce B: Agenti se jako obvykle zrodí pouze s instinkty a náhodně sestavenými plány
- Odměny za jednotlivé aktivity jsou u obou frakcí stejné
- Instinkty nyní zahrnují všechny informace jenž agent může znát

Agenti obou frakcí se zrodí se všemi informacemi, díky čemuž již od začátku vkládají do svých plánů aktivity, jako je zničení nepřátelské věže. V prvním kole vyhrála frakce B, avšak to není podstatné, jelikož v prvním kole byly podmínky rovnocenné. Další kolo v němž si frakce A zachovala své plány bylo poměrně zdlouhavé, avšak nakonec vítězí frakce A. Ve všech následujících kolech, až na jedno (není myšleno ono první kolo) vyhrává frakce A, ne však s takovou převahou a rychlostí jako u scénáře kdy si pouze tato frakce zachovávala informace. Zatímco plány frakce B zpočátku obsahují všechny cíle nijak cíleně uspořádané, u frakce A převažovaly cíle jako střílení do nepřátel a věží. Očekávaný „ideální“

plán měl podobu *Obrázek: 11*, avšak ukázalo se, že se pomocí evolučního zlepšování algoritmu více ujala varianta *Obrázek: 12*, při niž agent střídavě střílí do nepřátelské věže a nepřátelských agentů. Jelikož při cíli zničení věže se agent plně soustředí na získání věže a ignoruje nepřátelské agenty, jenž se u vlastněné věže často vyskytují (jelikož je to jejich pozice kde se oživují) stává se tak dosti zranitelným. Tento plán se u agentů vyskytoval vesměs s odlišnostmi, jako je jeho doplnění o sdílení plánu, či zásah neutrální věže, jelikož po zničení nepřátelské věže se stává věž neutrální a plán v první polovině obsahující střílení na nepřátele a věž B, střídá střílení na nepřátele a věž C.


Obrázek 11: Předpokládaný plán




Obrázek 12: Nejvíce častý plán

## 7.3 Pouze jedna frakce sdílí informace

- Frakce A: agenti mezi sebou sdílí informace i plány
- Frakce B: Agenti nesdílí informace ani plány

Frakce B má nastaveno hodnocení sdílení plánu a informací na nulu, čímž považují tyto aktivity za neužitečné a tak je nezařazují do svých plánů. Všechny ostatní aktivity jsou hodnoceny u obou frakcí stejně.

Při tomto scénáři bylo očekáváno, že komunikující agenti budou mít po čase značnou převahu oproti svým nepřátelům. Pravdou je však to, že v drtivé většině případů vyhrává skupina B, tedy frakce jenž nekomunikuje. Příčinou je samotný počátek kola. Frakce A zpočátku ve velké míře komunikuje a teprve upravuje své plány s ohledem na to, že komunikace není prozatím efektivní, jelikož neznají informace které jejich blízcí agenti nemají, ani lépe hodnocené plány, jelikož jejich prvotní plány ještě nemají žádné bodové hodnocení. Mezi tím frakce B nekomunikuje vůbec a namísto toho ve větší míře střílí. Komunikující agenti

leckdy zůstanou příliš dlouho na svých počátečních stanovištích a snese se na ně vlna kulek nepřátelské frakce. Dříve než se agenti frakce A opět oživí, ocitají se v tak znevýhodněné situaci, že se jim ji už nedaří překonat.

Tomuto lze alespoň zčásti zamezit tím, že se frakci A sdílení plánů i cílů sníží na velmi nízké bodové ohodnocení oproti ostatním aktivitám. V takovémto případě agenti neprojevují zprvu takovou tendenci komunikovat a tak se vyhnou hromadnému vymření hned na samotném počátku kola. Komunikace je sice méně častá, avšak oproti opačné straně alespoň v nějaké míře probíhá, díky čemuž i ve větší míře vyhrává.

# Závěr

Výsledkem je simulátor v němž může uživatel pozorovat chování dvou skupin agentů, přičemž jednotliví agenti se vždy snaží maximalizovat počet bodů získaných vykonáváním určitých aktivit. Bodové ohodnocení aktivit může měnit samotný uživatel a tím ovlivnit vývoj i podobu simulace. Je možné tyto hodnoty nastavit i tak, že agenti neusilují o vlastnictví věží a tak ani o vítězství, avšak více si takto pohrát s grafickou stránkou simulátoru. Kromě samotné vizuální stránky má simulátor i demonstrovat efektivitu evolučních algoritmů, které čerpají inspiraci v reálném světě. Plány které jsou náhodnými mutacemi během jejich existence pozměňovány a zachovány ty, které se jevili jako prospěšné, měly za následek optimalizaci plánů.

Jako další rozšíření simulátoru se nabízí využití strojového učení, například pomocí frameworku TensorFlow. Agent by si směl přenášet své vědomosti i do dalších kol simulace a nad získanými daty tak provést analýzu, na níž by si sám vytvořil bodové hodnocení jednotlivých aktivit.

# Conclusions

The result is a simulator in which the user can observe the behaviour of two groups of agents, where agents individually try to maximize the number of points earned by carrying out certain activities. The point evaluation of activities can change the user himself, thus influence the development and form of the simulation. These values are possible to set up in such a way that agents do not strive for the ownership of the towers, thus not for victory, however, it possible to experiment with the graphical side of the simulator. In addition to the visual itself, the simulator also can demonstrate the effectiveness of evolutionary algorithms which draw inspiration from the real world. Plans which are generated by random mutations during their existence were altered and preserved, and those that appeared to be beneficial to them resulted in plan optimization.

As the following extension, the simulator would mainly seek to add agents some form of machine learning, for example by TensorFlow framework. The agent would transfer his knowledge to the next rounds of the simulation and perform an analysis of the obtained data on which he would create a point evaluation of individual activities.

.

# A Obsah přiloženého CD/DVD

**bin/**
: Instalátor programu.

**doc/**
: Text práce ve formátu PDF, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu).

**src/**
: Kompletní zdrojové texty programu včetně knihoven, jenž je třeba nalinkovat v IDE. Knihovny jsou umístěny ve složce projektu libraries. Dále je také přiložen stručný popis nalinkování knihoven v IDE Visual Studio 2019.

**readme.txt**
: Instrukce pro instalaci a zdroje použitých materiálů.

# Literatura


[1] Multi-Agent Hide and Seek - YouTube. YouTube [online]. Dostupné z: https://www.youtube.com/watch?v=kopoLzvh5jY

[2] University information system MENDELU [online]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_ cast.pl?cast=32363

[3] MAŘÍK, Vladimír, Olga ŠTĚPÁNKOVÁ a Jiří LAŽANSKÝ. Umělá inteligence. Praha: Academia, 1993-. ISBN 80-200-0472-6.

[4] KUBÍK, Aleš. Inteligentní agenty. Brno: Computer Press, 2004. ISBN 80-251-0323-4.

[5] C++ – Wikipedie. [online]. Dostupné z: https://cs.wikipedia.org/wiki/C%2B%2B

[6] Lekce 3 - SDL - Základy vykreslování. itnetwork.cz - Ajťácká sociální síť a materiálová základna pro C#, Java, PHP, HTML, CSS, JavaScript a další. [online]. Copyright © 2020 itnetwork.cz. Veškerý obsah webu [cit. 27.05.2020]. Dostupné z: https://www.itnetwork.cz/cplusplus/sdl/sdl-zaklady-vykreslovani

[7] Microsoft Visual Studio – Wikipedie. [online]. Dostupné z: https://cs.wikipedia.org/wiki/Microsoft_ Visual_ Studio