
Katedra Informatiky
Fakulta Matematiky, Fyziky a Informatiky
Univerzita Komenského, Bratislava

# Evolučné stratégie v RPG systémoch

(Bakalárka práca)

Stanislav Buštor

**Vedúci:** RNDr. Michal Forišek

Bratislava, 2008

Čestne prehlasujem, že som túto bakalársku prácu vypracoval samostatne s použitím citovaných zdrojov.

.....

# Obsah

# Abstrakt

V tejto práci som riešil problém optimalizácie stratégie skriptovateľného pomocníka do MMORPG Ragnarok Online. Vytvoril som skript – plugin do pomerne rozšírenej umelej inteligencie, ktorý optimalizuje stratégiu na základe genetického algoritmu. Výsledný skript je ľahko použiteľný aj pre neprogramátora, ale ktokoľvek si ho môže v prípade potreby upravovať.

**Kľúčové slová**: genetické algoritmy, MMORPG, hra, umelá inteligencia, Lua

# Kapitola 1

# Úvod

Ako sa výpočtová technika stále zlepšuje, počítačové hry aj umelá inteligencia sú stále rozvinutejšie a čoraz bližšie k reálnej inteligencii. RPG hry sú zaujímavou časťou hier na skúmanie, pretože sa často snažia nejakým spôsobom simulovať realitu.

Cieľom tejto práce bolo vytvoriť plugin do umelej inteligencie komerčnej MMORPG Ragnarok Online, ktorý by zabezpečoval optimalizáciu stratégie skriptovateľného pomocníka na základe genetických algoritmov. Plugin by mal byť ľahko použiteľný pre laika, ktorý nemusí vedieť programovať a zároveň dávať možnosť prisôsobiť si algoritmus či fitness funkciu pre pokročilejších užívateľov či programátorov. Posledným cieľom bolo upozorniť na túto hru ako na experimentálne prostredie pre vývoj a testovanie umelej inteligencie.

# Kapitola 2

# Podobné výskumy

## 2.1 TRON

Webová hra[tro] inšpirovaná známou hrou TRON umožňuje hráčom prostredníctvom java appletu vyskúšať si túto známu hru a zároveň postupne učiť ich genetický algoritmus hrať lepšie túto hru. Hrá sa na štvorcovej ploche reprezentujúcej torus, hráč reprezentuje pohyblivú čiaru, ktorej smer ovláda šípkami, pričom sa nemôže zastaviť.

Tento projekt beží už niekoľko rokov, na stránke sú zverejnené denne aktualizované štatistiky, ako sa postupom času zlepšuje ich hráč. Tieto štatistiky sa generujú už vyše 6 rokov. Samozrejme sa postupom času môžu hráči tiež zlepšovať, alebo aj zhoršovať.

## 2.2 Primordial Life

`http://www.io.com/ spofford/`

Primordial life je experiment v počítačovej evolúcii. Formou šetriča obrazovky sa vyvíjajú bioti – umelé živočíšne formy a bojujú medzi sebou o prežitie. Verzia 3.21 sa automaticky napája na iné ekosystémy zapnuté niekde

na internete, a bioti môžu takto prechádzať na úplne iné počítače.

Bohužiaľ neplatená verzia mala obmedzený počet použití menu, a dokonca mi ani nefungovalo poriadne pripojenie na iné ekosystémy, takže som to poriadne neotestoval. Každopádne to vyzeralo zaujímavo.

## 2.3 Pogamut

`https://artemis.ms.mff.cuni.cz/pogamut/tiki-index.php?page=GeneticBots`

Pogamut je projekt zo susedného Česka, ktorý poskytuje platformu na vytváranie a debugovanie agentov, ako prostredie využíva hru Unreal Tournament 2004.

Jeden z podprojektov aplikuje genetické programovanie na botov.

# Kapitola 3

# Ragnarok Online

Ragnarok online je *MMORPG* vytvorená GRAVITY Co., Ltd. založená hlavne na manhwe Ragnarok od Lee Myung-Jin. Prvý krát bola vydaná v roku 1998 v Južnej Kórei a odvtedy bola vydávaná po celom svete. Väčšina hernej mytológie je založená na nórskej mytológii, ale je ovplyvnená ázijskými kultúrami.

Klient hry je stiahnuteľný zadarmo. Na oficiálnych serveroch sa platí za mesačné pripojenie jedného hráčskeho účtu, pričom sa zvačša pripája na server v blízkej geografickej oblasti. 14. mája 2008 oznámila Gravity spustenie serveru, na ktorom bude hranie zdarma s nejakými nevýhodami oproti normálnym serverom.

Existujú však rôzne emulátory, napríklad eAthena[eat], ktorá je open source a zatiaľ Gravity nepodnikla právne kroky aby ho zrušila.

Je to pomerne graficky príťažlivá hra, s 3D prostredím a 2D predrenderovanými obrázkami postáv (*sprite*), NPC (*Non Player Character* - nehráčska postava) a príšer (*creature*).

Svet Ragnaroku je rozdelený na série máp, kde každá má svoj vlastný terén a domáce príšery. Presun medzi mapami je možný na vyhradených miestach – warpoch, cez ktoré nemôžu prechádzať monštrá.

Tak ako vo väčšine MMORPG, primárnym cieľom hráča je dosiahnuť čo

najvyššiu úroveň postavy a zohnať čo najlepšiu výbavu. Zháňanie vývaby je podmienené šťastím, ale hlavne dlhotrvajúcim hraním, ktoré prináša prevádzkujúcej spoločnosti nemalé peniaze.

V hre je dostupné veľké množstvo rôznych povolaní, ako zaujímavé pre programátora mi prišlo povolanie Alchymistu(*Alchemist,Biochemist*), ktorý môže mať skriptovateľné zvieratko(*pet*) – homunkula(*homunculus*)

## 3.1 Prostredie

Prostredie hry je zjednodušene tvorené terénom, príšerami, homunkulom, alchymistom a ostatnými hráčmi. Aby sme sa mohli zaoberať optimalizáciou stratégie homunkula, je potrebné si najprv ujasniť nejaké pojmy, zjednodušené tak aby dostatočne popisovali situáciu.

### 3.1.1 Vlastnosti

Živé objekty majú nasledujúce vlastnosti:

*HP*, *Hit Points*, životy — udáva výdrž objektu voči útokom

*SP*, *Spell Points*, mana — udáva koľko schopností môže použiť v krátkom čase

*XP*, *Experience points*, skúsenosti — udávajú mieru priblíženia sa k ďalšej úrovni, za zabitie príšery je konštantný počet skúseností

*level*, úroveň — udáva približnú silu objektu, pri zmene úrovne sa môžu, ale nemusia zlepšiť iné vlastnosti; úroveň sa pohybuje v intervale 1 až 99

*ATK*, *attack*, útok — udáva počet životov ktoré sa odoberú nepriateľovi po zásahu

*MATK*, *magical attack*, magický útok — udáva počet životov ktoré sa odoberú nepriateľovi po magickom zásahu

*HIT* — šanca na zasiahnutie nepriateľa

*FLEE* — šanca na vyhnutie sa útoku

*DEF*, *defence*, obrana — percentuálne znižuje účinnosť nepriateľovho fyzického útoku *MDEF*, *magical defence*, magická obrana — percentuálne znižuje účinnosť nepriateľovho magického útoku

*ASPD*, *attack speed*, rýchlosť útoku

*Evolution*, evolúcia — po evolúcii získa homunkulus novú schopnosť a zvýšia sa mu vlastnosti

## 3.2 Homunkulus

Homunkulus je skriptovateľné zvieratko ktoré môže mať alchymista. Skript je naprogramovaný v jazyku Lua, ktorého interpreter má klient zabudovaný v sebe.

Homunkulus sa môže ľubovolne pohybovať na dohľad od majiteľa, používať svoje schopnosti(*skill*) a útočiť na príšery. Do cca februára 2008 mohol používať aj skilly majiteľa, toto bolo v novom klientovi zrušené.

Homunkulus dostáva za zabité príšery skúsenosti(*experience points – XP*) a rovnaké množstvo dostane aj majiteľ.

Homunkulov môže majiteľ liečiť schopnosťou *Aid Potion*, čo je takmer jediný hráčsky podporný skill použiteľný na homunkula.

Homunkulom sa určujú ich štatistiky náhodne, každý zo štyroch druhov homunkulov má iný interval prírastkov[1], pričom samozrejme má tendenciu smerovať k priemeru.

### Typy homunkulov

**Amistr** — vysoké HP, ATK, DEF, nízke FLEE, ASPD, HIT

**Filir** — vysoké FLEE, ASPD, HIT, stredný ATK a nízke HP

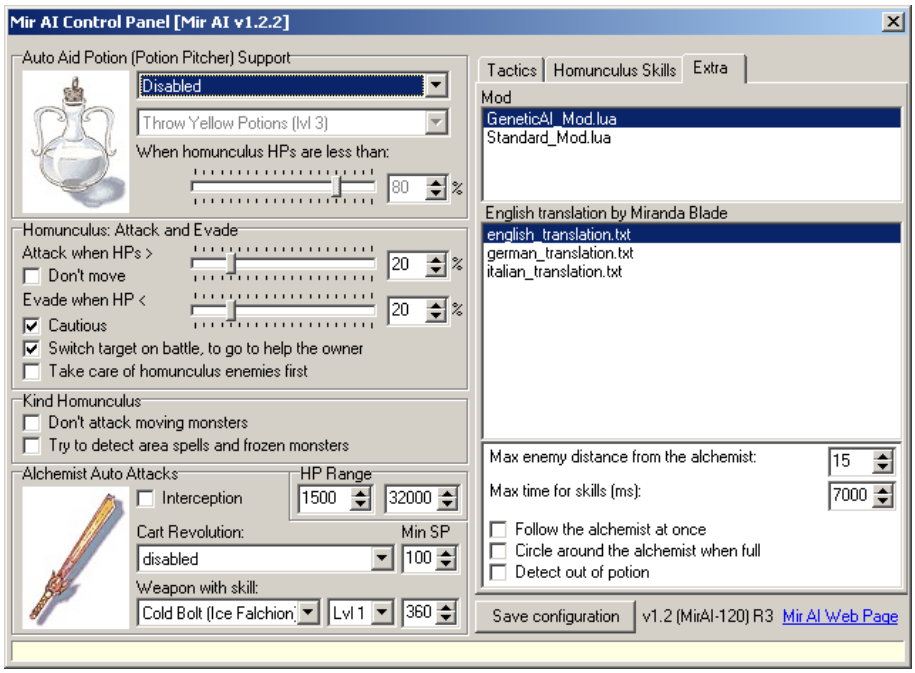
[1]Konkrétne hodnoty napríklad tu: `http://wiki.reborn.cz/doku.php/wiki:alchemist_hom`

**Lif** — po evolúcii najvyšší ATK, vysoké DEF, stredné HIT, FLEE, HP, nízke ASPD
**Vanilmirth** — vysoký ATK, MATK, ASPD, FLEE, HIT, stredné HP

## 3.3 Inštalácia pluginu a použitie

Miranda Blade AI, známa ako MirAI sa presadila hlavne vďaka nie veľkej konkurencii a peknému GUI, takže užívatelia nemuseli meniť nastavenia priamo v textových súboroch so skriptom.

Okrem toho je tu zabudovaný systém pluginov, vďaka ktorému sa stačí extrahovať všetky súbory pluginu do zložky s užívateľskou AI (napríklad c:/program files/gravity/ragnarok/AI/USER_AI/), prekliknúť na záložku Extra a tam už len vybrať *GeneticAI_Mod.lua*




Ako základná populácia je zvolený jedinec bez genotypu. Keď pôvodná AI bude zisťovať potenciálny cieľ, zapíše sa tomuto jedincovi do genotypu

taktika používaná na tohto moba nastavená v konfiguračnom paneli. Ak nie je nastavená, zapíše sa mu tam default hodnota. Takto si jedinec vygeneruje záznamy o tom, aké príšery sa nachádzajú na mape, kde sa pohybuje hráč a nenecháva si v genotype zbytočne všetky príšery, z ktorých väčšinu nikdy nestretne.

# Kapitola 4

# Rozbor problému

V Ragnaroku sa vyskytuje vyše 500 druhov rôznych príšer, a vyše 600 rôznych máp, aj ked nie všetky sú osídlené príšerami. Každý druh homunkula vyžaduje inú stratégiu v závislosti na jeho vlastnostiach, lokácii a konkrétnom cieli hráča.

Niektoré príšery ktoré zabije jeden druh na úrovni 50, nezabije iný druh ani na úrovni 80, napríklad preto, že ho netrafí, alebo naopak dostáva sám príliš veľa zásahov. Preto úloha nie je stacionárna pre rôzne druhy homunkulov.

Na každej mape môže byť iné zloženie príšer, a tie, ktoré sa oplatili zabíjať na jednej mape sa nemusia oplácať na druhej. Úloha teda nie je stacionárna ani pre rôzne mapy.

Zloženie príšer sa našťastie nemení časom, takže ak si raz pre nejaký druh homunkula vyvinieme stratégiu, bude takmer optimálna až kým nebude mať zásadne iné vlastnosti. Homunkulom na leveli 99 sa samozrejme vlastnosti už meniť nebudú, takže tam zostane stratégia na mape stále taká istá.

Keďže množstvo možností je naozaj veľmi veľké, rozhodol som sa aplikovať genetické algoritmy, ktoré majú v takomto prípade šance uspieť tam, kde hráč nevie zvoliť dobrú stratégiu, alebo sa mu len nechce analyzovať, čo je preňho

optimálne.

## 4.1 Genetické algoritmy

Genetické algoritmy používaju princíp, ktorý funguje dobre v prírode. Vo svete žije nejaká populácia, ktorá má určitý cieľ, a zvačša sa ho podarí dosiahnuť iba tým najsilnejším.

Namiesto DNA používa genetický algoritmus pole, v ktorom má uložené zakódované informácie, ktoré určujú jeho správanie. V tomto plugine je to pole pre každé stretnuté monštrum s vlastnosťami popísanými v 4.2.2.

Populácia postupne žije vo svete, a na záver je ohodnotený jedinec *Fitness* funkciou, ktorá označuje jeho vhodnosť.

Je veľa možností, ako vybrať ďalšiu populáciu, ja som si vybral nasledovnú: pár najlepších jedincov presuniem rovno do ďalšej generácie.

Potom ich začnem medzi sebou náhodne krížiť (*Crossover*) – jedinec vznikne tak, že sa zoberie prvá časť z jedného jedinca a zvyšok z druhého jedinca.

Nasledovne sa prejde celá DNA – každý prvok má určitú šancu na to že sa zmutuje, čiže zmení na náhodnú hodnotu.

Postupne vznikne cela nova populácia a cyklus sa môže zase opakovať.

## 4.2 Nastavenie MirAI

### 4.2.1 Globálne

- Pasivita/agresivita homunkula voči príšerám, ktoré nie sú nastavené v taktikách homunkula. Nenastavené hodnoty sa budeme snažiť eliminovať, takže toto nastavenie môžeme vynechať.

- Nastavenia pri akom počte životov homunkulus uteká, respektíve neútočí. Toto takisto necháme na užívateľa.

MirAI obsahuje niekoľko ďalších nastavení, ktoré pre nás z hľadiska optimalizácie stratégie nie sú podstatné, dôležité sú práve tie lokálne.

### 4.2.2 Lokálne

MirAI má niekoľko nastavení, ktoré kódujú taktiku[*Tactics*] homunkula voči konkrétnej príšere. Prvým je Správanie[*Behaviour*], ktoré má 9 nastavení:

`BEHA_avoid` — homunkulus uteká a dokonca ani nepomáha majiteľovi.

`BEHA_coward` — homunkulus uteká ak je zasiahnutý, ale príde pomôcť majiteľovi ak je to potrebné.

`BEHA_react_1st` — homunkulus sa bráni ak je zasiahnuty, alebo spolupracuje; vysoká priorita

`BEHA_react` — homunkulus sa bráni ak je zasiahnuty, alebo spolupracuje; stredná priorita

`BEHA_react_last` — homunkulus sa bráni ak je zasiahnuty, alebo spolupracuje; nízka priorita

`BEHA_attack_1st` — homunkulus útočí; vysoká priorita

`BEHA_attack` — homunkulus útočí; stredná priorita

`BEHA_attack_last` — homunkulus útočí; nízka priorita

`BEHA_attack_weak` — homunkulus útočí na túto príšeru až ako poslednú, ignoruje jej útoky

Ďalšie je nastavenie použitia schopností[*skill*]:

`WITH_no_skill` — nepoužíva schopnosti

`WITH_one_skill` — použije iba jednu schopnosť na začiatku útoku

`WITH_two_skills` — použije dve schopnosti na začiatku útoku

`WITH_max_skills` — používa schopnosti až do nastaveného time out

`WITH_full_power` — používa schopnosti až do vyčerpania *SP*
`WITH_slow_power` — pomaly používa schopnosti až do vyčerpania *SP*

Tretie nastavenie určuje úroveň[*level*] použitej schopnosti, od 1 do 5.

## 4.3 Implementácia

Prečo som implementoval celý kód v Lua? Má to viacero výhod – dá sa to priamo použiť v inej AI bez nejakých modifikácií, hráč nemusí nič kompilovať ani púštať neznámu binárku. A navyše si môže akúkoľvek časť kódu prerobiť tak, ako mu to vyhovuje.

Jeden z implementačných problémov bolo to, že sa AI reštartuje pri prechode warpom alebo použití schopnosti teleport, ktorá presunie majiteľa aj homunkula na náhodné miesto na mape.

Toto som riešil pravidelným zápisom údajov o jedincovi do súboru *GA/settings.lua*, vždy po uplynutí `WriteTimeout` milisekúnd a po zabití nejakej príšery.

Ďalší implementačný háčik bol, že homunkulus nevie zistiť, že zabil príšeru, iba že ju stratil z dosahu, alebo zmizla. Teda vtedy môže predpokladať, že ju zabil, ale nie je to presné. Takisto nemá možnosť zistiť, koľko dostal skúseností za zabitie.

Preto som vytvoril skript *mobdataexport.php*, ktorý exportne niektoré zaujímavé údaje o príšerách do *mobdata.lua*, s použitou databázou eatheny.

Jedinec sa testuje `TestingTime` sekúnd, potom sa presunie do databázy otestovaných jedincov v súbore *GA/tested.lua*, potom ako sa mu spočíta fitness.

Fitness funkcia je implementovaná ako predpokladaný počet skúseností, ktoré získa homunkulus za hodinu:

```
function CalcFitness()
  if (Settings.TotalTime==0) then return 0 end
  return (Settings.TotalExp/(Settings.TotalTime/3600000))
end
```

Toto si samozrejme môže užívateľ zmeniť, tak aby to vyhovovalo jeho cieľom :)

Zvyšok populácie sa skladuje v súbore *GA/population.lua*, keď sa všetci jedinci minú, tak vyberie otestovanú populáciu, `(PopulationSize*Elitism/100)` najlepších presunie do novej populácie. Potom náhodne kríži a nechá mutovať tieto jedince – každý z 3 parametrov pre príšeru s pravdepodobnosťou `(MutationRate/100)`.

Potom novú populáciu presunie do odkladacieho súboru a nahrá ďalšieho jedinca.

Genotyp jedinca je uložený ako pole s názvom DNA, príklad:

```
{
DNA={
  {1152,BEHA_attack_1st,WITH_full_power,3}, -- Orc Skeleton
  {1177,BEHA_attack,WITH_one_skill,4}, -- Zenorc
  {1111,BEHA_react_last,WITH_two_skills,2}, -- Drainliar
  {1042,BEHA_react_last,WITH_no_skill,1}, -- Steel Chonchon
  }
}
```

## 4.4 Nápady do budúcnosti

Aby bol tento plugin ešte ľahšie použiteľný, neuškodilo by keby mal tiež GUI. Ideálne integrované s GUI pre MirAI.

Viac predpripravených nastavení fitness funkcie a kríženia, medzi ktorými by išlo rýchlo meniť. Mutácia by mohla mať váhu pre jednotlivé možnosti – málokedy sa oplácalo dať na príšeru možnosť `BEHA_avoid`

Nuž a posledná a najhoršia časť – keď to má používať veľa užívateľov, prečo by si nemohli navzájom pomáhať a odosielať najlepších na nejaký webový server? Malo by to samozrejme pár háčikov – rozličné nastavenia, nemožnosť kontroly mapy, či overenia hodnoty fitness funkcie, takže by sa všetko muselo počítať u viac ľudí, a musel by u nich bežať nejaký exe súbor, čo by už samozrejme nebol každý ochotný spraviť.

# Kapitola 5

# Záver

Plugin, ktorý som vytvoril, nenaplnil úplne moje očakávania, výsledky jedincov boli približne rovnaké ako tie, ktoré som empiricky, respektíve heuristicky nastavil. Avšak pri testoch, kde som nemal žiadnu predchádzajúcu skúsenosť, alebo málo údajov, dopadli lepšie práve jedince generované genetickým pluginom.

Dúfam však, že som vytvoril základ, na ktorom bude niekto ďalší môcť stavať – experimentovať s genetickými algoritmami, niečo nové sa naučiť alebo aj prekonať iné metódy.

# Dodatok A

# Obsah CD

GA/auth.php
GA/ga_const.lua
GA/mobdata.lua
GA/mobdataexport.php

baka.pdf
control_panel_for_mirai_v1_2_(en_de_it)_R3.zip
GeneticAI_Mod.lua
mirai_v1_2_2.zip

# Dodatok B

# Listing pluginu

```
------------------------------------------------
-- Mir AI -> Genetic AI Config
------------------------------------------------
--[[
--   Webpage: http://ksp.sk/~morgoth/baka/
--]]
------------------------------------------------
-- HOW TO ACTIVATE THIS MOD:
-- 1. open the Control Panel,
-- 2. choose the "Extra" Tab,
-- 3. and select GeneticAI_Mod.lua
-- (if you don't use the control panel: open SelectMod.lua and
-- replace "Standard_Mod" with "GeneticAI_Mod")

------------------------------------------------
-- Extra Globals
------------------------------------------------


-- [### CONFIG START]
WriteTimeout  = 5000 -- kazdych 5000 milisekund
TestingTime  = 600 -- 600 sekund testovaci cas
MutationRate = 50 -- 50/1000
PopulationSize = 20 -- velkost populacie
Elitism = 30  -- 30% elitarov
```

```
-- [### CONFIG END]

require "./AI/USER_AI/GA/mobdata.lua"
require "./AI/USER_AI/GA/ga_const.lua"
require "./AI/USER_AI/Const.lua"

-- Variables

-------------------------------------------------
function Logg(text)
-------------------------------------------------
local CurrTime = GetTick()
local dt= CurrTime - LastLogTime
if LastLogTime == 0 then dt = 0 end
if FirstLogTime==nil then FirstLogTime=CurrTime end
LastLogTime = CurrTime

local f_out=io.open('./AI/USER_AI/GA/logg.txt', "a")
if f_out~=nil then
        f_out:write(CurrTime-FirstLogTime..'ms | '..
                              dt..'ms: '..text .. "\n")
f_out:close()
end
end

-------------------------------------------------
function FileAppend(file,what)
-------------------------------------------------
local f_out = io.open(file, "a")
if f_out~=nil then
f_out:write(what.. "\n")
f_out:close()
end
end

-------------------------------------------------
function file_exists(path)
-------------------------------------------------
  local file = io.open(path, "r")
  if file then file:close() end
```

```
  return file ~= nil
end

-------------------------------------------------
function LoadFile(name)
-------------------------------------------------
Logg('LoadFile<-'..name)
local file=loadfile(name)
if not file then
Logg('LoadFile -> no file found')
return
end
file = file()
if not file then
Logg('LoadFile -> failed parsing')
return
end
Logg('LoadFile -> success')
return file
end

-------------------------------------------------
function SaveSettings()
-------------------------------------------------
Logg('SaveSettings')
local f = io.open('./AI/USER_AI/GA/settings.lua',"w")

Settings.Fitness=CalcFitness()
f:write('return {\n')
f:write(Serialize(Settings))
f:write('}\n')
f:close()
WriteTime=GetTick()
end

-------------------------------------------------
function sizeof(sth)
-------------------------------------------------
local n=0
if (type(sth)~="table") then return 0 end
```

```
for i,v in pairs(sth) do n=n+1 end
return n
end

------------------------------------------------
function Serialize(data)
------------------------------------------------
  Logg('Serialize')
  local str=''

  if (data.TotalTime~=nil) then
    str=str..'TotalTime='..data.TotalTime..',\n' end
  if (data.TotalExp~=nil)  then
    str=str..'TotalExp=' ..data.TotalExp.. ',\n' end
  if (data.TotalKills~=nil)then
    str=str..'TotalKills=' ..data.TotalKills.. ',\n' end
  if (data.Fitness~=nil)then
    str=str..'Fitness=' ..data.Fitness.. ',\n' end

  str=str..'DNA={\n'
  local temp={}
  for i,v in pairs(data.DNA) do
    if temp[v[1]]==nil then
      str=str..'  {'..v[1]..','..beha_const[v[2]]..','..skill_const[v[3]]
      str=str..','..v[4]..'}, -- '..MOBDATA[v[1]][MOB_P_iName]..'\n'
      temp[v[1]]=1
    end

  end
  str=str..'  }\n'

  return str
end

------------------------------------------------
function CalcFitness()
------------------------------------------------
--toto je customizovatelne, ako najrozumnejsie
        --zatial pocet expov za hodinu
if (Settings.TotalTime==0) then return 0 end
```

```
return (Settings.TotalExp/(Settings.TotalTime/3600000))
end

-------------------------------------------------
function Tick()
-------------------------------------------------
  Logg('Tick')
  CurrTime=GetTick()

  if (Settings.TimeStart==nil)then
     --ak nemam udane kedy som zacal, zacal som teraz
    Settings.TimeStart=CurrTime
  else
    Settings.TotalTime=Settings.TotalTime
            +CurrTime-Settings.TimeStart --update celkoveho casu
    Settings.TimeStart=CurrTime
  end

  if (CurrTime-WriteTime>WriteTimeout) then
    --chcem zapisat co jedinec pozabijal?
    SaveSettings()
  end
  if (Settings.TotalTime/1000>TestingTime) then -- koniec jedinca
    Settings.Fitness=CalcFitness()
    local str=Serialize(Settings)

    if (file_exists('./AI/USER_AI/GA/tested.lua')) then
      local tested=LoadFile('./AI/USER_AI/GA/tested.lua')

      file=io.open('./AI/USER_AI/GA/tested.lua',"w")
      file:write('return {\n')
      if (tested~=nil) then
        for i,v in ipairs(tested)
          do file:write('{'..Serialize(v)..'},\n') end
      end
      file:write('{'..str..'}\n')
      file:write('}')
      file:close()
    else
      local f = io.open('./AI/USER_AI/GA/tested.lua',"w")
```

```
      f:write('return {{'..str..'}\n}') --zapisem jedinca do suboru
      f:close()
    end

    FileAppend('./AI/USER_AI/GA/history.lua','{'..str..'},')

    os.remove('./AI/USER_AI/GA/settings.lua') i
    --premazem subor s jedincom, nech ho netestujem 2x
    LoadNextCreature() --nahram dalsieho jedinca z populacie
  end
end

-------------------------------------------------
function InitPopulation()
-------------------------------------------------
Logg('=== InitPopulation ===')
return {
{DNA={}}--prazdna DNA-pobeha, preskuma a pridaju sa mu prisery
}
end


-------------------------------------------------
function sort(x,n,f)
-------------------------------------------------
 local i=1
 while i<=n do
  local m,j=i,i+1
  while j<=n do
   if f(x[j],x[m]) then m=j end
   j=j+1
  end
 x[i],x[m]=x[m],x[i]
 i=i+1
 end
end

-------------------------------------------------
function deepcopy(object)
-------------------------------------------------
```

```
    local lookup_table = {}
    local function _copy(object)
        if type(object) ~= "table" then
            return object
        elseif lookup_table[object] then
            return lookup_table[object]
        end
        local new_table = {}
        lookup_table[object] = new_table
        for index, value in pairs(object) do
            new_table[_copy(index)] = _copy(value)
        end
        return setmetatable(new_table, getmetatable(object))
    end
    return _copy(object)
end

------------------------------------------------
function Cross(x,y) --DNA 2 jedincov
------------------------------------------------
--Logg('Cross X:\n'..print_r(x)..'\nWith Y:\n'..print_r(y))
if (math.random(2)==1) then t=x x=y y=t end
local nx=sizeof(x)
local ny=sizeof(y)
local known={}
local backx={}
local backy={}

for i,v in ipairs(x) do known[v[1]]=1 backx[v[1]]=v end
for i,v in ipairs(y) do known[v[1]]=1 backy[v[1]]=v end

--Logg('Known are :'..print_r(known))
nk=sizeof(known)
local point=math.random(nk) --na tomto mieste rozdelim
ret={}
local w=1
for k,v in pairs(known) do
if (w<nk) then ret[w]=deepcopy(backx[k])
                          else ret[w]=deepcopy(backy[k])end
w=w+1
```

```
end
--Logg('Result is: '..print_r(ret))
return ret
end

-------------------------------------------------
function Mutate(x)
-------------------------------------------------
for i,v in pairs(x) do
  for j=2,4 do
    if math.random(1000)<MutationRate
      then v[j]=math.random(const_max[j][1],const_max[j][2]) end
  end
end
end

-------------------------------------------------
function NewPopulation()
-------------------------------------------------
Logg('NewPopulation')

old=LoadFile('./AI/USER_AI/GA/tested.lua')
local n
if (old~=nil) then n=sizeof(old) else n=0 end

if (n==0) then
Logg('=== Pouzivam Default populaciu! ===')
new=InitPopulation()
else
Logg('Stara populacia ma velkost :'..n)
sort(old,n,function (x,y) return x.Fitness>y.Fitness end)
local pocet=math.min(sizeof(old),math.floor(PopulationSize*Elitism/100))
Logg('A do vybralo sa prvych '..pocet..' jedincov do dalsej!')
new={}

for i=1,pocet do new[i]={} new[i].DNA=deepcopy(old[i].DNA) end
for i=pocet+1,PopulationSize do
--nahodny rodic z populacie sa skrizi s dalsim
new[i]={}
new[i].DNA=Cross(new[math.random(pocet)].DNA,new[math.random(pocet)].DNA)
```

```
Mutate(new[i].DNA)
end
end

file=io.open('./AI/USER_AI/GA/population.lua',"w")
file:write('return {\n')
for i,v in ipairs(new) do file:write('{'..Serialize(new[i])..'},\n') end
file:write('}')
file:close()
os.remove('./AI/USER_AI/GA/tested.lua')
end
-------------------------------------------------
function LoadNextCreature()
-------------------------------------------------
Logg('LoadNextCreature')
Population=LoadFile("./AI/USER_AI/GA/population.lua") --nahram populaciu
local size
if (Population~=nil) then size=sizeof(Population) else size=0 end
os.remove("./AI/USER_AI/GA/population.lua")

if (size==0) then --ak dosla populacia, vyrobim si novu
NewPopulation()
Population=LoadFile("./AI/USER_AI/GA/population.lua")
end

Logg('Kopirujem zvysok populacie do population.lua')

first=1
local f
for i,v in ipairs(Population) do
if (first==1)then
Settings=v
first=2
else
if (first==2)then
f=io.open('./AI/USER_AI/GA/population.lua',"w")
f:write('return {')
first=0;
end
f:write('{'..Serialize(v)..'},')
```

```
end
end
if (first==0) then f:write('}\n') f:close() end

--v settings je novy jedinec
CurrTime=GetTick()
ApplyDNA(Settings.DNA)
Settings.TotalTime=0
Settings.TotalExp=0
Settings.TotalKills=0
Settings.TimeStart=CurrTime
Settings.Creatures={}
for i,v in pairs(Settings.DNA)
          do Settings.Creatures[v[1]]=1 end --prisery ktore "poznam"

SaveSettings()
WriteTime=CurrTime
end

-------------------------------------------------
function print_r (t, indent)
-------------------------------------------------
local print_string=''

function printr (t, indent)
  local indent=indent or ''
  for key,value in pairs(t) do
    print_string=print_string..indent..'['..tostring(key)..']'
    if type(value)=="table" then print_string=print_string..':\n'
    printr(value,indent..'\t')
    else print_string=print_string..' = '..tostring(value)..'\n' end
  end
end

printr(t,indent)
return print_string

end

-------------------------------------------------
```

```
function GAOnInit()
-- On Init: loadnut nastavenie, zapisat stary cas
-----------------------------------------------------
Logg('GaOnInit()')
Settings=LoadFile("./AI/USER_AI/GA/settings.lua")--toto sa nepremazava

if (Settings==nil) then LoadNextCreature() end
Settings.Creatures={}

for i,v in pairs(Settings.DNA)
          do Settings.Creatures[v[1]]=1 end --prisery ktore "poznam"

if (Settings.TotalTime==nil) then Settings.TotalTime=0 end
if (Settings.TotalExp==nil)  then Settings.TotalExp=0 end
if (Settings.TotalKills==nil)then Settings.TotalKills=0 end

Logg('Settings: '..print_r(Settings))

ApplyDNA(Settings.DNA)
Tick()
end

-------------------------------------------------
function ApplyDNA(DNA)
-------------------------------------------------
Logg('ApplyDNA')
for i,v in pairs(DNA) do
Logg('Aplikujem DNA na moba '..v[1])
Tact[v[1]]={Tact[v[1]][1],v[2],v[3],v[4],Tact[v[1]][5]}
end
end

-------------------------------------------------
function OnATTACK2_ST()
-------------------------------------------------
Logg('OnAttackST')
if MyEnemy == 0 then return end
local Ex, Ey = GetV(V_POSITION, MyEnemy)
if (Ex == -1) or (MOTION_DEAD == GetV(V_MOTION, MyEnemy)) then
if (MyEnemyTypeID>0) then
```

```
Settings.TotalExp=Settings.TotalExp+MOBDATA[MyEnemyTypeID][MOB_P_EXP]
Settings.TotalKills=Settings.TotalKills+1
Tick()
MyEnemyTypeID=0
end
end
StdOnATTACK_ST()
end

-------------------------------------------
function GetFullTact2(ID)
-- Homun uvidel priseru. Zapisem si jej id a pridam aj do DNA
-------------------------------------------
Logg('GetFullTact : '..ID)
if (Settings.Creatures[ID]==nil) then --nova prisera
Logg('Nova prisera - '..ID)

if (Tact[ID]~=nil) then
Settings.DNA[ID]={ID,Tact[ID][2],Tact[ID][3],Tact[ID][4]}
else
Settings.DNA[ID]={ID,DEFAULT_BEHA,DEFAULT_WITH,0}
end
Settings.Creatures[ID]=1
end
T=StdGetFullTact(ID)
MyEnemyTypeID=ID
EnemyName=T.CName
return T
end

-------------------------------------------------
function AI2(myid)
-------------------------------------------------
Tick()
StdAI(myid)
end


-------------------------------------------------
function ModInit()
```

```
-- plugin initialization
-------------------------------------------------
Settings=nil
WriteTime=0

-- Replace standard Mir AI's functions
--GetFullTact kvoli vyberu nepriatela
StdGetFullTact = GetFullTact
GetFullTact = GetFullTact2
--OnATTACK_ST kvoli zabitemu nepriatelovi
StdOnATTACK_ST=OnATTACK_ST
OnATTACK_ST=OnATTACK2_ST

StdAI=AI
AI=AI2

GAOnInit() --spustim hlavnu proceduru
end
```

# Literatúra


[CLM04] N. Cole, SJ Louis, and C. Miles. Using a genetic algorithm to tune first-person shooter bots. *Evolutionary Computation, 2004. CEC2004. Congress on*, 1, 2004.

[eat] *eAthena server emulator* `http://eathena.ws`.

[Gra] Gravity Interactive. *Homunculus AI Script User.s Guide*.

[Gre92] J.J. Grefenstette. Genetic algorithms for changing environments. *Parallel Problem Solving from Nature*, 2:137–144, 1992.

[Lua] *The Programming Language Lua*.

[Mic96] Zbigniew Michalewicz. *Genetic algorithms + data structures = evolution programs (3rd ed.)*. Springer-Verlag, London, UK, 1996.

[mir] *Miranda Blade AI* `http://www.mirandablade.altervista.org/index.php?pg=mirai&lng=en`.

[Mit96] M. Mitchell. *An Introduction to Genetic Algorithms*. Bradford Books, 1996.

[rag] *R A G N A R O K online* `http://ragnarokonline.com/`.

[RG02] Franz Rothlauf and David E. Goldberg. *Representations for Genetic and Evolutionary Algorithms*. Physica-Verlag, 2002.

[RK06] Theppatorn Rhujittawiwat and Vishnu Kotrajaras. *Learnable Buddy: Learnable Supportive AI in Commercial MMORPG*. 2006.

[TK06] Worapoj Thunputtarakul and Vishnu Kotrajaras. *AI-TEM: Testing AI in Commercial Game With Emulator*. 2006.

[tro] *TRON* `http://helen.cs-i.brandeis.edu/tron/`.

[wik] *Ragnarok Online - Wikipedia, the free encyclopedia* `http://en.wikipedia.org/wiki/ragnarok_online`.