Středoškolská odborná činnost

Obor: 18 informatika

Experimenty s přirozeným výběrem

Daniel Čejchan

kraj Královéhradecký
Náchod 2014

Středoškolská odborná činnost

Obor: 18 informatika

Experimenty s přirozeným výběrem

Natural selection experiments

Autor:	Daniel Čejchan
Škola:	Jiráskovo gymnázium Náchod
Kraj:	Královéhradecký
Konzultant:	RNDr. Jan Preclík, Ph.D.

Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval(a) samostatně a použil(a) jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v seznamu vloženém v práci SOČ.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V Náchodě dne ..... podpis: .....

Poděkování

Děkuji panu Preclíkovi za jeho podporu. Děkuji Marušce za její vynikající karbanátky, mamce za její korekturu a tátovi za jeho guláš.

Anotace

Cílem tohoto projektu je prozkoumání, čeho všeho je schopen přirozený výběr, který by měl být logickým důsledkem přírodních zákonů. Uživatel má k dispozici interaktivní prostředí, ve kterém se dá (samozřejmě značně zjednodušeně) simulovat vývoj jakýchsi primitivních organizmů. Ty jsou řízeny neuronovou sítí. Vývoj se snaží být co nejvíce založen na přirozeném výběru.

Klíčová slova: evoluce, buňka, neuron, neuronová síť, simulace

The purpose of this project is to examine the capabilities of the natural selection, as a consequence of the laws of nature. An interactive environment is available, where a user can simulate evolution of some sort of made-up primitive organisms, controlled by a neural network. This evolution is intended to be based on the natural selection as much as possible.

Keywords: evolution, cell, neuron, neural network, simulation

Obsah

1 Úvod

Na Darwinově teorii (resp. darwinismu) se dnes zakládá prakticky veškeré pojetí vývoje života. Základem teorie je tento princip: Jedinci se od sebe mírně odlišují; jejich odlišnosti jim potom buď ztěžují nebo zjednodušují život a reprodukci. Vlivem toho se rozšiřují znaky, které organizmům usnadňují přežití/reprodukci, a organizmy se přizpůsobují prostředí. Toto je samozřejmě velmi zjednodušená verze, a ve výsledku takto jednoduchá pravidla neplatí – to se ostatně ukáže i v mé práci. Názorový směr rozvíjející Darwinovu teorii se nazývá neodarwinismus.

Uvedené téma se mi zdálo zajímavé a připadalo mi, že by stálo za prozkoumání, alespoň v rámci mých možností. Vytvořil jsem pro to prostředí, do kterého jsem umístil jakési ‚buňky'. Každá buňka obsahuje neuronovou síť, která ovlivňuje její chování. Nezaměřuji se na vývoj fyziologie, ale na vývoj ‚mozků' těchto buněk – ten by ale měl fungovat na stejném principu. ‚Program' této sítě se za života jedné buňky nemění, ale mírně mutuje při reprodukci buněk. Uživatel může pozorovat, co se děje s chováním těchto buněk v průběhu generací – program by měl, v závislosti na výkonu počítače, zvládat relativně slušné rychlosti simulace.

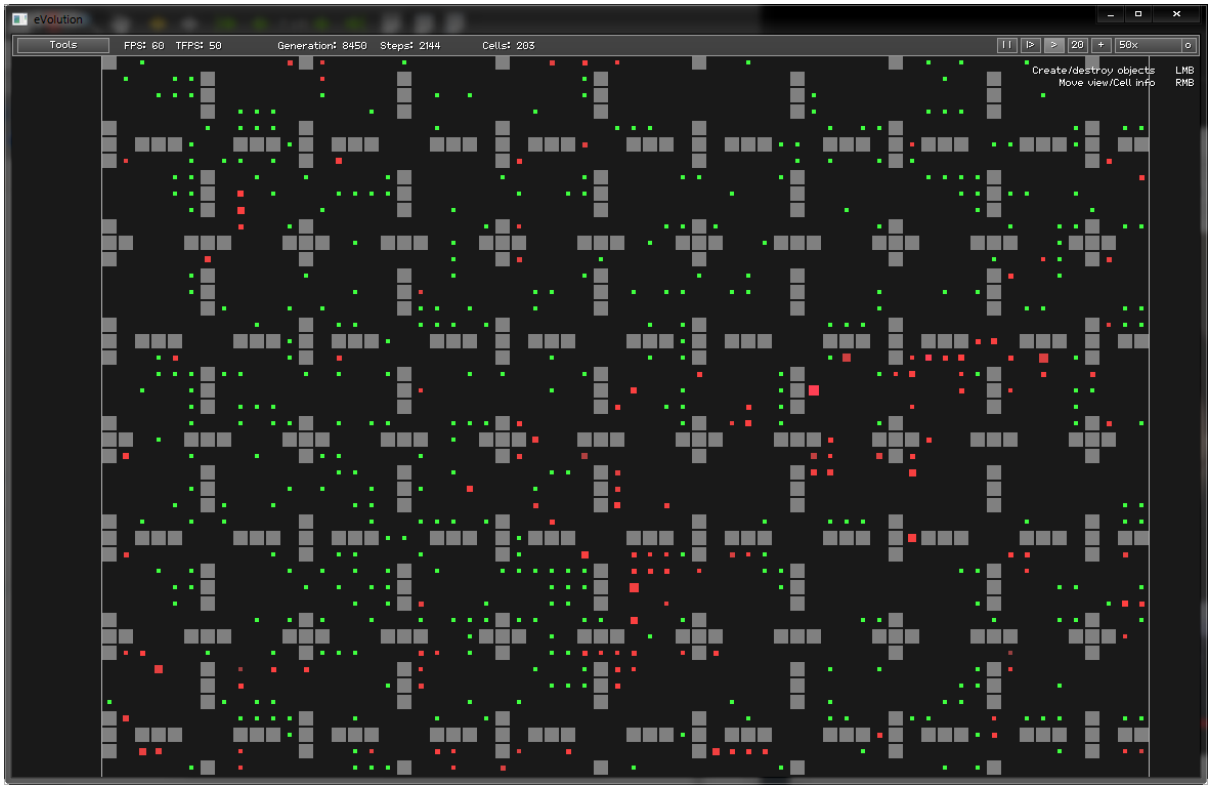
2 Popis prostředí

Prostředí simulace je realizováno dvojrozměrným polem; objekty ve světě jsou zarovnány do mřížky a jejich pozice se dá vyjádřit celými čísly – neexistují žádné mezistavy. Svět je ‚souvislý' *(seamless)* – poslední políčko vpravo sousedí s první políčkem vlevo a nejspodnější políčko sousedí s tím nejvíc nahoře. Jsou implementovány tři typy objektů – buňka, zeď a jídlo.

Buňka je klíčovým objektem tohoto projektu. Ve vizualizaci má červenou barvu. Její chování je řízeno neuronovou sítí – každá buňka má vlastní neuronovou síť se vstupy (podněty, které přijímá od okolí) a výstupy (které určují, co buňka udělá). Základními akcemi buňky jsou pohyb a rozmnožování. Více informací o buňce viz 2.2.

Jídlo je statický objekt. Ve vizualizaci má zelenou barvu. Už z názvu je patrné, že slouží jako potrava pro buňku. Hledání jídla je jakýsi přirozený cíl, který buňka má. Je pozřeno tak, že buňka dá povel k pohnutí se na políčko, ve kterém je ono jídlo. Poté jídlo zaniká; ve světě se ale pořád náhodně vytváří nové.

Zeď slouží jako překážka pro buňky – ztěžuje jim přístup k jídlu, stejně tak (záleží na tom, jak zdi rozmístíte) jim brání v příliš jednoduchém vzorci pro pohyb. Ve vizualizaci má šedou barvu.



Obrázek 1: Vizualizace prostředí

2.1 Mechanismus simulace

Základní jednotkou času simulace je cyklus *(step)*. V každém cyklu program postupně projde všechny buňky, vyhodnotí senzorické vstupy neuronových sítí, zpracuje síť a vykoná příkaz, který buňka vydá. V každém cyklu může buňka vydat maximálně jeden příkaz (viz 2.3). Toto zpracování je iterativní a postupné, tedy další buňka, která je zpracovávaná, už počítá s akcemi, které provedly buňky v tom samém cyklu, které byly zpracované dřív. Pořadí zpracovávání buněk je víceméně náhodné. Nejedná se zde o jednoduché pravidlo (např. zleva doprava), nicméně pořadí je fixní; buňce, jejíž cyklus se provede dříve, než cyklus jiné buňky, se tento cyklus bude provádět dříve po celou dobu její existence.

Proces simulace světa je plně deterministický – tedy když si vícekrát pustíte nějakou simulaci od nějakého bodu, měla by se (za předpokladu, že jste nijak nezměnili podmínky) chovat vždy stejně.

2.1.1 Pravidla simulace

Simulace probíhá v tzv. generacích. Generace je vlastně jeden samostatný experiment; mezi generacemi se nepřenáší žádná data.

Na začátku generace se ve světě vytvoří první buňky, tzv. Adamové. Tyto buňky mají náhodně vygenerované rozložení neuronové sítě[1]. Spolu s nimi se vygeneruje i určitý počet jídla (jídlo, které zbylo z předchozí generace je vymazáno a je vygenerováno nové). Zdi se zachovávají.

V určitém (nastavitelném) časovém intervalu se ve světě náhodně vytváří další jídlo, jinak buňky nejsou nijak ovlivňovány. Jejich přežití závisí na jejich mozku (detailní popis viz 2.2). Jakmile všechny buňky v dané generaci vymřou, začne se další ‚experiment' a vytvoří se nová generace. V podstatě tedy generace, které nemají potenciál k přežití, vymřou velice rychle, a ty, které jej mají, se rozmnoží a začnou se vyvíjet. To může taky vyústit v jejich vymření, nicméně to už bývá po mnohem delší časové periodě.

Relevantní informace na toto téma také viz 3.3

2.2 Buňka

Buňka představuje jakýsi autonomní ‚organizmus'. Obsahuje neuronovou síť, která určuje její chování, a disponuje sadou senzorů, které mají sloužit jako podněty k reakcím. Má dvě klíčové vlastnosti – zdraví *(health)* a energii *(energy)*.

Zdraví je faktor ovlivňující buňku z dlouhodobějšího hlediska. Buňka se narodí s určitou hodnotou zdraví (nastavitelné – viz 3.3), které každý cyklus ubývá o jednu jednotku. Jakmile klesne na nulu, buňka zaniká. Jediný způsob, jak si může buňka doplnit zdraví, je sežrat jinou buňku (viz 2.3.4).

[1] Zde záleží na nastavení, uživatel si může zvolit, podle jakých pravidel se síť generuje, popřípadě i nastavit vlastní výchozí síť – viz 3.3. Pokud se generuje víc Adamů, každý Adam má jinou neuronovou síť, generovanou podle nastavených pravidel.

Energie naopak ovlivňuje buňku v rámci malých časových úseků. Buňka se (podobně, jako je to u zdraví) narodí s určitou hodnotou energie (opět nastavitelné – viz 3.3), ale ztrácí ji nejen každý cyklus, ale spotřebovává ji i na jakékoli akce, které chce vykonat[2] – pohyb, rozmnožování, atp. Energii buňka může získat buď tím, že pozře jídlo, nebo tím, že sežere jinou buňku.

Energie a zdraví buňky jsou patrné i z grafické vizualizace – čím víc má buňka energie, tím je větší, a zdraví určuje její ‚červenost' – čím ho má buňka méně, tím je šedší. Pokud se buňka začne chovat jako ‚kanibal' a bude požírat ostatní buňky, zbarví se do fialova (viz 2.3.4).

2.2.1 Neuronová síť

Mozek buňky je realizován čtyřvrstvou perceptronovou neuronovou sítí (jedna vrstva vstupní, dvě skryté, jedna výstupní), v každé z vrstev je 32 neuronů. Pro tento model jsem se rozhodl, protože byl nejjednodušší na implementaci a neviděl jsem důvod pro implementaci složitější sítě. Šířku sítě určil počet senzorických vstupů (zaokrouhlený na mocninu dvou) a hloubku jsem zvolil víceméně náhodnou. Přenosová funkce je skoková – vzhledem k povaze projektu a implementaci prostředí mi to přišlo jako nejvhodnější.

Tyto parametry sítě nelze v aplikaci měnit.

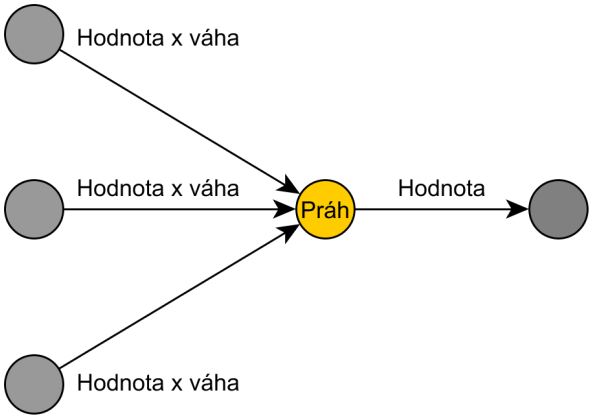
Neuron

Základní jednotkou neuronových sítí jsou neurony, což jsou zjednodušené modely neuronů skutečného mozku. Mají jeden nebo více vstupů a právě jeden výstup. Vstupy a výstupy jsou číselné hodnoty; vstupy mohou být napojeny na ostatní neurony (každý vstup na právě jeden výstup, každý výstup na libovolný počet vstupů), popřípadě na konstantní hodnoty, senzorické vstupy, atp.

Každý neuron má číselný atribut ‚Práh' a pro každý vstup číselný atribut ‚Váha'. Pro systém je také nadefinována tzv. přenosová funkce, což je matematický předpis s jedním vstupním parametrem.

Zpracování neuronu potom probíhá tak, že se projdou vstupy, sečtou se všechny jejich hodnoty vynásobené přiřazenými vahami a od toho celého se odečte práh. Toto výsledné číslo se poskytne jako parametr přenosové funkci a její výsledek je přiřazen jako výstupní hodnota neuronu.

[2] Hodnoty energie spotřebované jednotlivými akcemi se také dají nastavit (viz 3.3)

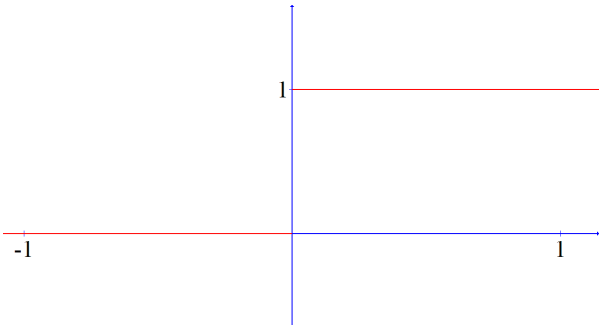


Obrázek 2: Model neuronu

Nejzákladnější přenosovou funkcí je skoková přenosová funkce. Má předpis

$$f(x) = 0 \qquad \text{pro } x < 0$$
$$f(x) = 1 \qquad \text{pro } x >= 0$$

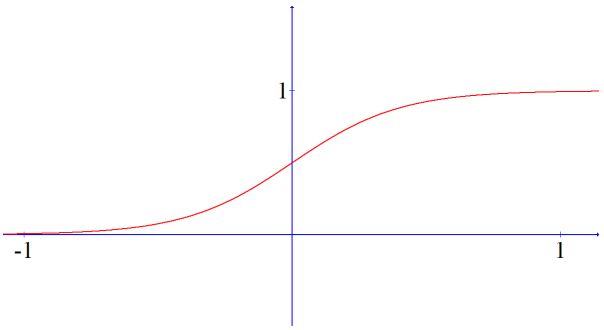
a graf:



Obrázek 3: Graf skokové přenosové funkce

Důsledek je takový, že pokud je součet vstupů vynásobených jejich vahami větší než práh, výstup neuronu bude 1, jinak bude 0. V podstatě tedy pracuji s bitovým výstupem, a tedy i bitovými vstupy. To mi umožňuje používat ordinální datové typy – pro váhy používám 8bitovou ordinální proměnnou se znaménkem (*signed*; nabývá hodnot -127 až 127) a pro práh 32bitovou (cca -2 mld. až 2 mld.). **Neuron, který pracuje s bitovými vstupy a výstupy, se jmenuje perceptron.**

Jen pro zajímavost uvedu ještě graf tzv. sigmoidální přenosové funkce, která už není binární a je nejrozšířenější při práci s neuronovými sítěmi:



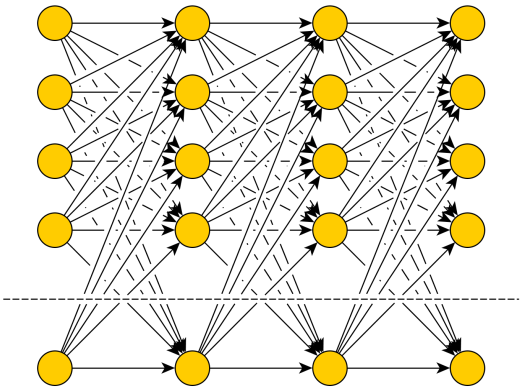
Obrázek 4: Graf sigmoidální přenosové funkce

Předpis této funkce je:

$$f(x) = \frac{1}{1 + exp(-k * x)}$$

Síť perceptronů

Jeden neuron sám o sobě nemá potřebnou výpočetní kapacitu, a tak se jich spojuje víc do struktur – sítí. Existuje více základních druhů sítí, prakticky si lze neurony pospojovat dle libosti. Já jsem se rozhodl použít tzv. vícevrstvou perceptonovou síť. Ta má následující strukturu:



Obrázek 5: Model vícevrstvé perceptonové sítě

Perceptrony jsou v ní rozděleny do vrstev (v modelu vrstvy odpovídají sloupcům), kde každý neuron v jedné vrstvě je provázaný s každým perceptronem ve vrstvě předchozí. Počet perceptronů v každé vrstvě se může lišit, nicméně já jsem nechal všechny vrstvy stejně široké (tj. se stejným počtem perceptronů v každé vrstvě). Šířku jsem určil podle počtu vstupů do sítě (viz 2.3) zaokrouhleného na mocninu dvou.

2.3 Senzorické a motorické možnosti buněk

Zde je kompletní přehled vstupů a výstupů neuronové sítě (tento soupis přesně odpovídá rozvržení v editoru neuronových sítí v aplikaci – viz 3.4):

#	**Vstupy**	**Výstupy**
1	Zdraví >= 320	Směr, 1. bit
2	Zdraví >= 160	Směr, 2. bit
3	Zdraví >= 80	Akce, 1. bit
4	Zdraví >= 40	Akce, 2. bit
5	Zdraví >= 20	Nic nedělej
6	Zdraví >= 10	
7	Energie >= 320	
8	Energie >= 160	
9	Energie >= 80	
10	Energie >= 40	
11	Energie >= 20	
12	Energie >= 10	
13	Buňka nalevo	
14	Buňka nahoře	
15	Buňka napravo	
16	Buňka dole	
17	Jídlo nalevo	
18	Jídlo nahoře	
19	Jídlo napravo	
20	Jídlo dole	
21	Zeď nalevo	
22	Zeď nahoře	
23	Zeď napravo	
24	Zeď dole	
25	Něco ob jeden nalevo	
26	Něco ob jeden nahoře	
27	Něco ob jeden napravo	
28	Něco ob jeden dole	
29	Časovač	
30	Vždy aktivní	

Jak můžete vidět, buňka disponuje poměrně velkým množstvím informací jak o svém stavu, tak o stavu okolí. Má přesné detektory na stav své energie a zdraví, dokáže rozlišit typ objektu, který je těsně u ní a dokáže poznat, zda je něco ob jedno políčko od ní (i přes překážky). Vstup ‚Časovač' se každý cyklus přepíná mezi nulou a jedničkou, ‚Vždy aktivní' je konstantní jednička.

Výstupy nejsou na první pohled tak zřejmé. Zatímco vstupy fungují ‚analogově', rozhodl jsem se, že výstup budu řešit ‚digitálně'. První dva bity udávají směr, pro který má buňka danou akci vykonat, a další dva bity určují, která akce se vykoná. Pokud je aktivní výstup #5, buňka neprovede žádnou akci.

Pozorný čtenář si všimne, že ačkoli má neuronová síť šířku 32 perceptronů, jako vstupy je jich definováno jen 30. Zbylé dva fungují jako opakovače – jejich hodnota se nastaví na výstup na odpovídající pozici poslední vrstvy. Využití tohoto nechávám opět na ‚evoluci', na mysl mi přichází například paměťový okruh.

1	**2**	**Směr**	**Akce**
0	0	Vlevo	Pohyb
0	1	Nahoru	Zabití
1	0	Vpravo	Reprodukce
1	1	Dolů	Sex. reprodukce

V této tabulce jsou vypsány všechny kombinace. K digitálnímu pojetí jsem se rozhodl, protože při analogovém byl celkem problém s prioritou příkazů – buňky měly ve zvyku dávat výstup více akcím najednou (buňka smí vykonat pouze jednu akci za cyklus). Musel jsem tedy stanovit přesně prioritu příkazů, díky čemuž se příliš nevyvíjely výstupy s nižší prioritou.

Každá akce spotřebuje určité množství energie (nastavitelné, viz 3.3). Pokud buňka nedisponuje dostatečnou energií, neprovede se žádná akce.

2.3.1 Pohyb

Pohyb je de facto výchozí akcí pro buňku; nevyžaduje žádnou aktivitu. Buňka se pohne o jedno políčko daným směrem. Pokud je na cílovém poli jídlo, pozře ho a k její energii se přičte ‚Výživnost jídla' (viz 3.3); buňka se v tomto případě nepohne. Pokud energie buňky překročí ‚Mez přejedení', buňka zemře. Pokud je na místě jiná buňka nebo zeď, pohyb se nekoná a buňce se odečte ‚Energie za špatnou akci' (viz 3.3).

2.3.2 Reprodukce

Buňka se zreplikuje daným směrem. Nově vytvořená buňka má výchozí hodnoty zdraví a energie a neuronová síť se zkopíruje z mateřské buňky. Nicméně každá váha a každý práh má určitou šanci zmutovat o jednu jednotku (viz 3.3, sekce ‚Brain'). Pokud je na políčku, kam se má buňka zreplikovat, nějaký objekt, akce se nekoná a buňce, která se chtěla replikovat, se odečte ‚Energie za špatnou akci' (viz 3.3).

2.3.3 Sexuální reprodukce

Buňka se ‚spáří' s buňkou v daném směru a vytvoří buňku ve směru opačném. Nově vzniklá buňka bude mít výchozí hodnoty zdraví a energie. Váhy a prahy v její neuronové síti budou mít

hodnotu, která se přesně rovná aritmetickému průměru odpovídajících hodnot mateřských buněk, bez jakýchkoli mutací. Pokud se v zadaném směru nenachází žádná buňka, akce se nekoná a buňce se odečte ‚Energie za špatnou akci' (viz 3.3). Je-li v daném směru buňka, ale v opačném směru není místo, žádná penalizace buňku nestihne, nicméně reprodukce se také nevykoná.

Sexuální rozmnožení stojí energii jenom tu buňku, která ji iniciuje; druhá buňka se podílí pouze pasivně.

2.3.4 Zabití

Buňka svede jakýsi pomyslný boj s buňkou v daném směru. Vyhraje-li útočící buňka, ta druhá je jí ‚pozřena' a její energie a polovina zdraví se převede na agresora (pokud ale energie buňky překročí ‚Mez přejedení', buňka zemře). Prohraje-li, je pozřena sama útočící buňka a té, na kterou útočila, se přičte čtvrtina jejího zdraví. Není-li na políčku v daném směru žádná buňka, akce se nekoná a buňce se odečte ‚Energie za špatnou akci' (viz 3.3).

K určení, zda buňka-agresor vyhrála, slouží tento algoritmus:

```
1 bool won = (
2          random(
3                   ( Energy + Health * 2 ) * 3
4                   + otherCell.Energy + otherCell.Health * 2 )
5          ) < ( Energy + Health*2 ) * 3;
```

Na rozhodnutí se tedy podílí energie obou buněk (s menším účinkem) a zdraví obou buněk (s větším účinkem), přičemž agresor je zvýhodněn tak, že rovnají-li se hodnoty energie i zdraví u obou buněk, agresor má třikrát větší šanci vyhrát.

Zabije-li buňka jinou buňku, její barva ve vizualizaci se zbarví do fialova.

3 Ovládací prvky

Aplikace obsahuje:

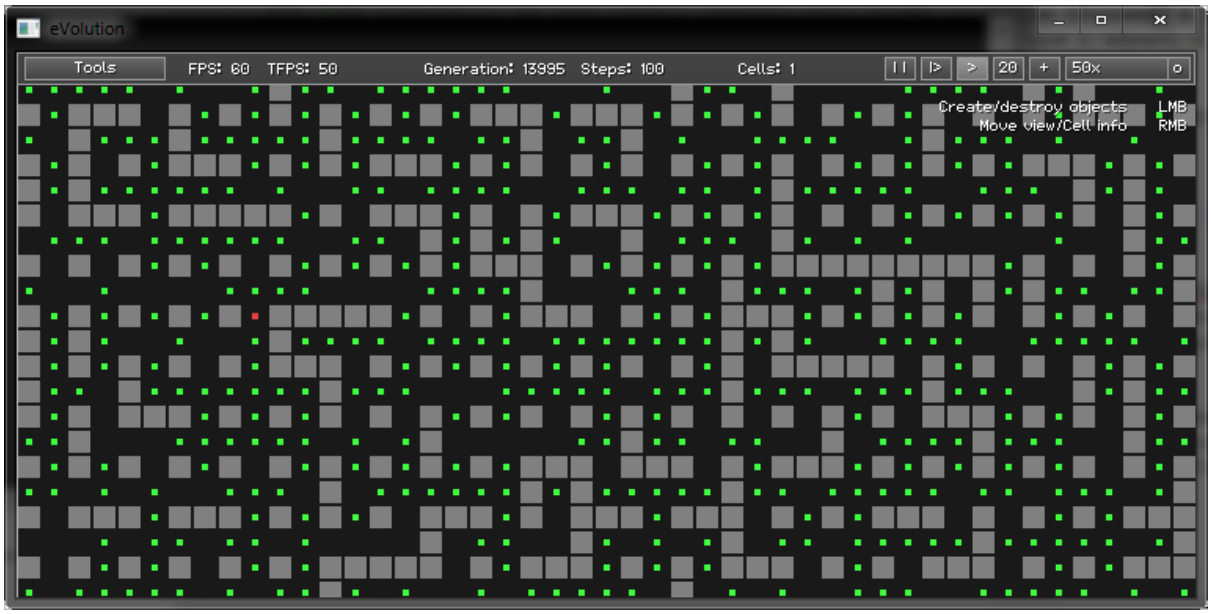
- hlavní panel s nejdůležitějšími ovládacími prvky,
- nástrojový panel,
- okno s nastavením faktorů simulace,
- okno s editorem neuronových sítí a informacemi o buňce.

Použitý jazyk je angličtina, jednak kvůli světovému trendu, jednak kvůli tomu, že verze mého enginu použitá pro vývoj této aplikace ještě nepodporuje UTF-8 (přechod na novější verzi by byl značně problematický).

Samotné simulační okno také disponuje určitými ovládacími prvky. Za držení levého tlačítka myši můžete přidávat/odebírat zdi (popřípadě buňky nebo jídlo – viz 3.2, skupina tlačítek ‚Tool‘) a při držení pravého tlačítka a táhnutí myši můžete hýbat pohledem. Kliknete-li pravým tlačítkem na nějakou buňku, otevře se okno editoru neuronových sítí (viz 3.4), který zobrazí informace o dané buňce.

Při zapnutí aplikace se automaticky načte svět a nastavení takové, jaké jste měli při posledním bezpečném vypnutí aplikace.

3.1 Hlavní panel



Obrázek 6: Nástrojový panel v aplikaci

Komponenty hlavního panelu (zleva):

Tools
Tlačítko zobrazující/skrývající panel nástrojů.

FPS ***(Frames per second)***
Popisek zobrazující vykreslovací frekvenci hlavního okna.

TFPS ***(Thread's frames per second)***
Popisek zobrazující skutečnou frekvenci cyklů simulace.

Generation ***(Generace)***
Čítač zobrazující pořadové číslo generace; s každou novou generací se zvýší o jeden. Můžete ho restartovat tlačítkem ‚Generation = 1' v nástrojové liště.

Steps ***(Cykly)***
Čítač zobrazující počet uplynulých cyklů od začátku probíhající generace.

Cells ***(Buňky)***
Čítač zobrazující počet právě žijících buněk.

Následují komponenty ovládající rychlost simulace:

Tlačítko ‚||'
Tlačítko sloužící k pozastavení simulace.

Tlačítko ‚|>'
Tlačítko sloužící k vyvolání přesně jednoho cyklu simulace; pokud simulace běží, tímto se také pozastaví.

Tlačítko ‚>'
Tlačítko sloužící k obnovení simulace (v nastavené rychlosti).

Tlačítko ‚20'
Tlačítko pro rychlou volbu cílené rychlosti simulace – 20 cyklů za vteřinu (rychlost vhodná k pozorování); pokud byla simulace pozastavena, je její průběh obnoven.

Tlačítko ‚+'
Tlačítko pro rychlou volbu cílené rychlosti simulace – 50 000 cyklů za vteřinu (co nejvyšší rychlost); pokud byla simulace pozastavena, je její průběh obnoven.

Kombinovaný seznam
Kombinovaný seznam slouží k nastavení cílené rychlosti simulace; štítek na něm zobrazuje aktuální cílenou rychlost.

Cílená rychlost simulace je rychlost simulace, které se aplikace snaží dosáhnout. Skutečná rychlost však může být jiná – to záleží na výkonu počítače, na kterém simulace běží. Výpočetní náročnost narůstá s počtem živých buněk (a také velikostí světa, ale ne tolik).

3.2 Panel nástrojů

Panel nástrojů je při startu aplikace skrytý. Zobrazíte ho (popřípadě opět skryjete) tlačítkem ‚Tools‘ na hlavním panelu.

Obrázek 7: Panel nástrojů

Komponenty panelu nástrojů (odshora):

Environment settings ***(Nastavení prostředí)***
: Tlačítko otevírající okno s nastavením prostředí (viz 3.4).

Skupina tlačítek ‚Tool‘ ***(Nástroj)***
: Tato slouží k vybrání typu objektu, který chcete ve světě editovat.
Stisknete-li levé tlačítko myši a pod kurzorem je typ objektu odpovídající tomu, který jste si vybrali k editaci, tahem myši odstraníte všechny objekty tohoto typu, které se ocitnou pod kurzorem.
Stisknete-li levé tlačítko myši a v době stisku není pod kurzorem zvolený typ objektu, tahem myši vytvoříte tento objekt ve všech políčkách, přes která kurzor přejede, a to i v případě, byl-li tam původně jiný typ objektu – ten se odstraní. Budete-li takto vytvářet buňky, budou se generovat se stejným pravidlem jako Adamové.

Skupina tlačítek ‚Clear‘ ***(Vyčisti)***
: Tato skupina slouží k vymazání všech objektů daného typu ze světa.
Pokud odstraníte všechny buňky, aplikace to považuje za konec generace a vytvoří novou – tj. vygeneruje nové jídlo a Adamy. Pokud tomu chcete zabránit, pozastavte simulaci a vytvořte si vlastnoručně Adamy tam, kde je chcete, nebo v nastavení prostředí nastavte počet Adamů na nulu (viz 3.3).

Walls ***(Správa zdí)***

Aplikace umožňuje samostatnou správu zdí a také disponuje některými přednastavenými generátory schémat.

Z kombinovaného seznamu si vyberete, se kterým rozložením chcete pracovat; na výběr máte z několika přednastavených schémat (kompletní přehled viz 3.5), navíc máte možnost uložit/načíst vlastní. Samotný výběr nezpůsobí žádnou akci – k tomu slouží tři tlačítka pod seznamem; tlačítko ‚L' *(load – načíst)* načte vybrané schéma s tím, že zdi, které jsou ve světě před načtením, se nezachovají. Tlačítko ‚L+' také načte dané schéma, ale zachová již vytvořené zdi.

Máte-li vybrané uživatelské schéma (uživatelská schémata jsou označena ‚User #'), můžete do této položky uložit aktuální rozložení zdí. To uděláte tlačítkem ‚S' *(save – uložit)*.

Saves ***(Uložené simulace)***

Program také disponuje možností ukládat a načítat simulace. Pro uživatelské simulace jsou vyhrazeny položky ‚User #'. Popis ostatních naleznete v sekci 5. Ovládání je stejné jako u zdí – z kombinovaného seznamu vyberte položku, tlačítkem ‚L' ji načtete. Do pozic pro uživatelské simulace můžete ukládat pomocí tlačítka ‚S'.

Ukládá se rozložení zdí, buňky, jídlo, nastavení prostředí a statistiky. Neukládají se věci jako rychlost simulace, která okna jsou otevřená, atp.

World size ***(Velikost světa)***

Tento kombinovaný seznam slouží ke změně velikosti světa. Svět změní velikost okamžitě po výběru položky. **Při změně velkosti světa se aktuální svět kompletně smaže.**

Generation start ***(Začátek generace)***

Toto tlačítko vrátí simulaci do doby prvního cyklu aktuální generace. Program v podstatě ukládá simulaci na začátku každé generace (nejde o plnohodnotné uložení – neukládá se nastavení prostředí).

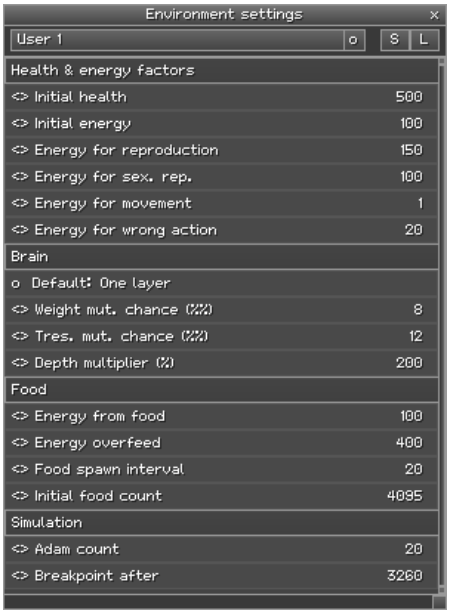
Chování světa je plně deterministické; vrátíte-li tedy simulaci na počátek generace, bude se vyvíjet naprosto stejně (pokud něco nezměníte vy). Toto se může hodit, pokud se například znovu chcete podívat na vývoj generace, o které víte, že se někam dostane.

Generation = 1 ***(Generace = 1)***

Tlačítko sloužící k vynulování (resp. vyjedničkování) čítače generace.

3.3 Nastavení prostředí *(Environment settings)*

Okno slouží k manipulaci s různými faktory, které ovlivňují simulaci. Otevřete ho kliknutím na tlačítko ‚Env. settings' na panelu nástrojů (viz 3.2).



Obrázek 8: Okno ‚Nastavení prostředí'

Oknem můžete pohybovat kliknutím na titulek a táhnutím myši; velikost okna můžete přizpůsobit stiskem tlačítka, které se nachází v pravém dolním rohu okna, a následným pohybem myši.

Stejně jako u zdí, i zde můžete ukládat/načítat různé konfigurace. K tomu slouží kombinovaný seznam a tlačítka ‚L' a ‚S' v horní části okna.

Samotné okno má potom tyto typy položek:

Položky s tmavým pozadím
: jsou názvy sekcí, které dělí vlastnosti do skupin.

Položky s prefixem <>
: zastupují číselné proměnné a jejich hodnotu změníte tak, že kliknete a podržíte levé tlačítko myši a hýbete s myší doleva nebo doprava. Rychlost změny roste exponenciálně s rychlostí pohybu myši.

Položky s prefixem o
: jsou položky, jejichž hodnotu měníte kliknutím na ně.

3.3.1 Popis položek v nastavení prostředí

Health & energy factors ***(Faktory zdraví a energie)***

Initial health ***(Počáteční zdraví)***
Zdraví, které buňka má, když se narodí. Platí jak pro Adamy, tak pro buňky vzniklé reprodukcí.

Initial energy ***(Počáteční energie)***
Energie, kterou buňka má, když se narodí. Platí jak pro Adamy, tak pro buňky vzniklé reprodukcí.

Energy for reproduction ***(Energie na reprodukci)***
Energie, která je spotřebována při nesexuální reprodukci buněk.

Energy for sexual reproduction ***(Energie na sexuální reprodukci)***
Energie, která je spotřebována při sexuální reprodukci buněk.

Energy for movement ***(Energie na pohyb)***
Energie, která je spotřebována při pohybu buňky o jedno políčko.

Energy for wrong action ***(Energie na pohyb)***
Energie, která je spotřebována, pokusí-li se buňka pohnout na políčko, na kterém je zeď nebo jiná buňka, nebo se reprodukovat na políčko, které není prázdné, nebo se sexuálně reprodukovat s něčím jiným než s buňkou.

Relevantní informace na toto téma také viz 2.3

Brain ***(Mozek)***

Default ***(Výchozí)***
Schéma neuronové sítě Adamů (bližší informace viz 3.4).

Weight mutation chance ***(Pravděpodobnost mutace váhy)***
Udává pravděpodobnost mutace vah při nesexuální reprodukci buněk (udáváno v promilích). Při nesexuální reprodukci tedy zmutuje *n* z tisíce vah v neuronové síti, každá o jednu jednotku nahoru nebo dolů (50/50).

Treshold mutation chance ***(Pravděpodobnost mutace prahu)***
Udává pravděpodobnost mutace prahů při nesexuální reprodukci buněk (udáváno v promilích).

Depth multiplier ***(Činitel pro hloubku)***
Udává, kolikrát klesne pravděpodobnost mutace vah a prahů s každou vrstvou sítě (udáváno v procentech). Vrchní (výstupní, viz 2.2.1) vrstva má pravděpodobnosti definované výše, další vrstva je má ale n-krát menší a další n^2-krát menší.
Díky této vlastnosti (a díky výchozí konfiguraci mozku ‚One layer') se počáteční složitost mozku může značně zjednodušit, což přispívá k lepšímu procesu učení.

Relevantní informace na toto téma také viz 2.3.2

Food ***(Jídlo)***

Energy from food ***(Energie z jídla)***
Hodnota energie, která se přičte buňce, když sní jídlo.

Energy overfeed ***(Mez přejedení)***
Hodnota energie, která když je překročena, buňka umírá.

Food spawn interval ***(Interval generace jídla)***
Udává, za kolik setin cyklu se vygeneruje ve světě jedno jídlo. Hodnota 20 například znamená, že každý cyklus se ve světě vygeneruje 5 jídel. Pokud tuto proměnnou nastavíte na nulu, nebude se generovat žádné jídlo.

Initial food count ***(Počáteční počet jídel)***
Udává, kolik jídla se vygeneruje na začátku každé generace.

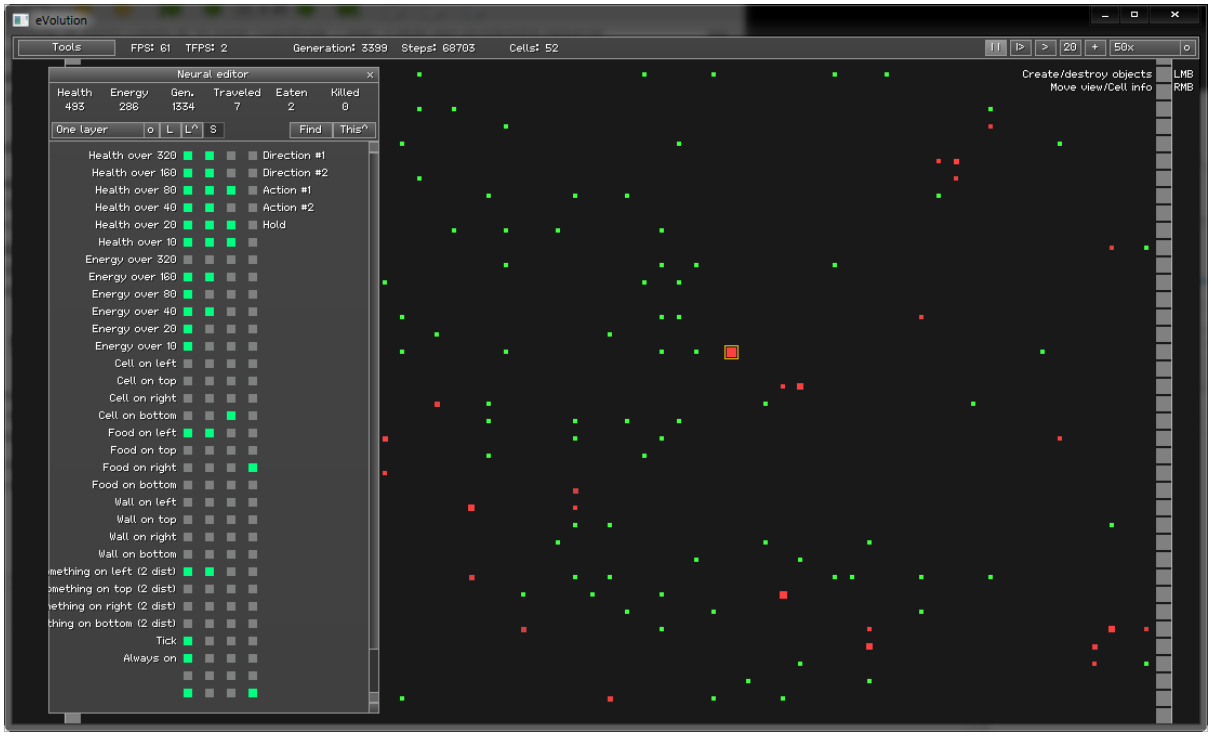
Simulation ***(Simulace)***

Adam count ***(Počet Adamů)***
Udává, kolik Adamů se vytvoří s každou generací. Nastavíte-li tuto proměnnou na nulu, po vymření všech buněk se nevygeneruje nová generace.

Breakpoint after
Pokud se generace dožije n-tého cyklu, je simulace automaticky pozastavena. Nastavíte-li tuto proměnnou na nulu, simulace nebude nikdy automaticky pozastavena.

3.4 Editor neuronových sítí *(Neural editor)*

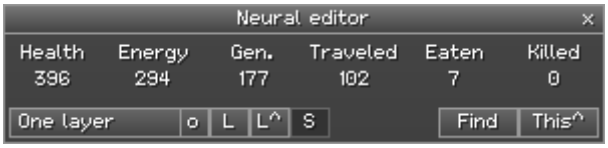
Okno editoru neuronových sítí (dále editor) slouží k sledování (a případné editaci) jedné konkrétní buňky; také se přes něj dá nastavit výchozí konfigurace neuronové sítě pro Adamy. Vyvoláte (popřípadě přepnete na sledování jiné buňky) ho pravým kliknutím na libovolnou buňku ve světě.



Obrázek 9: Aplikace s otevřeným editorem neuronových sítí

Oknem můžete pohybovat kliknutím na titulek a táhnutím myši; velikost okna můžete přizpůsobit stiskem tlačítka, které se nachází v pravém dolním rohu okna, a následným pohybem myši.

Když máte otevřený editor, aktuálně sledovaná buňka je ve vizualizaci prostředí zvýrazněna tak, že má kolem sebe oranžový rámeček (viz obrázek). Také můžete sledovanou buňku zaměřit na prostředek obrazovky pomocí tlačítka ‚Find‘.



Obrázek 10: Hlavička okna editoru neuronových sítí

Horní část hlavičky zobrazuje různé informace o buňce:

Health ***(Zdraví)***
: zobrazuje aktuální zdraví buňky.

Energy ***(Energie)***
: popisuje aktuální energii buňky.

Generation ***(Generace)***
: znamená počet generací, které buňku dělí od jejího Adama; neplést s generací simulace (viz 2.1.1).

Traveled ***(Uražená vzdálenost)***
: popisuje, kolik políček ušla buňka od svého narození.

Eaten ***(Snědeno)***
: popisuje, kolik jídla snědla buňka od svého narození.

Killed ***(Zabito)***
: určuje, kolik jiných buněk buňka úspěšně zabila (viz 2.3.4).

Dále hlavička obsahuje možnost uložení/načtení konfigurace neuronové sítě. K dispozici je několik přednastavených generátorů (popsány níže), navíc můžete ukládat vlastní konfigurace, a to do položek ‚User #'. Tlačítko ‚L' slouží k načtení/vygenerování konfigurace pro aktuálně označenou buňku. Tlačítkem ‚S' můžete síť sledované buňky uložit do některé položky vyhrazené pro uživatelské konfigurace.

Tlačítkem ‚This^ ' nastavíte neuronovou síť sledované buňky jako výchozí mozek Adamů. Tlačítko ‚L^ ' nastaví vámi označenou položku jako výchozí konfiguraci pro Adamy.

Rozdíly jsou dva:

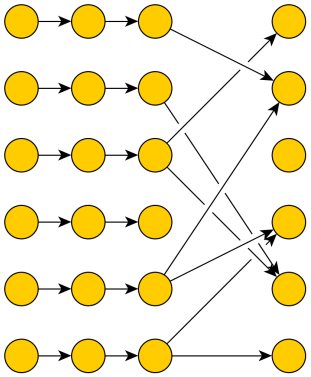
1. ‚L^ ' nenastaví jako výchozí aktuální rozložení, ale rozložení odpovídající vybrané položce v kombinovaném seznamu položce.
2. V případě, že je v kombinovaném seznamu vybrán generátor rozložení, ne konkrétní rozložení, nastaví se jako výchozí generátor, ne jeho produkt. Různým Adamům se potom tedy nenastavuje stejný mozek. Toho by jinak nešlo docílit.
   Např.: Načtete-li pomocí tlačítka ‚L' položku ‚One layer' a poté to nastavíte jako výchozí rozložení pomocí tlačítka ‚This^ ', všichni Adamové se narodí s právě tím rozložením, jaké se vám načetlo do buňky a jaké jste nahráli. Nastavíte-li ale přímo položku ‚One layer' jako výchozí pomocí tlačítka ‚L^ ', každý Adam se narodí s jinou konfigurací sítě, generovanou generátorem ‚One layer'.

3.4.1 Generátory konfigurací sítě

Aplikace disponuje dvěma generátory konfigurací neuronové sítě:

One layer *(Jedna vrstva)*

Toto je generátor, který nejspíš budete používat při vašich pokusech. Dvě skryté vrstvy jsou nastaveny tak, že ‚opakují' hodnotu odpovídajícího perceptronu na předchozí vrstvě (práh 9, váha pro daný perceptron 10, zbytek 0). V poslední vrstvě mají buňky práh náhodně mezi 0 a 20 a 5 % (což vůči šířce sítě odpovídá cca dvěma buňkám) vah má hodnotu 10, zbytek nula.



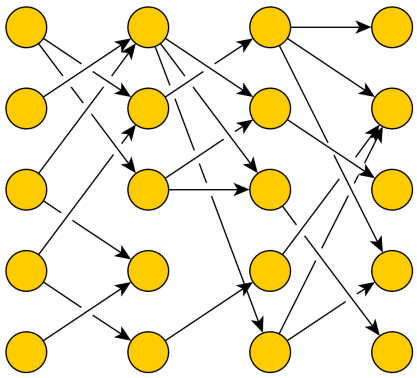
Obrázek 11: Příklad konfigurace generované pomocí ‚One layer' *(Jedna vrstva)* generátoru

Díky tomu má Adam určité, celkem pevně zafixované reflexy, které se poté mohou mutacemi snadno rozvíjet. Spolu s vysokým ‚Činitelem pro hloubku' (viz 3.3.1) je to ideální pro tvorbu zpočátku primitivní konfigurace, která má ale potenciál se vyvinout v něco složitějšího.

Tato metodika už by se dala do jisté míry považovat za formu nátlaku a pokud to tak vidíte, klidně používejte generátor ‚Random'.

Random *(Náhodné)*

Tento generátor nastaví všechny váhy na hodnotu mezi 0 a 50 a všechny prahy na hodnotu mezi 0 a 1600.



Obrázek 12: Ilustrativní příklad konfigurace generované pomocí ‚Random' generátoru

Používání tohoto generátoru se v praxi moc neosvědčilo, protože aby se vytvořil rozdíl ve váze/v prahu natolik velký, aby to ovlivnilo chování buňky, musela by váha/práh mutovat vícekrát po sobě na stejnou ‚stranu' (kladně nebo záporně, viz 2.3.2), což je statisticky velice nepravděpodobné.

3.4.2 Vizualizace neuronové sítě

V samotném okně potom můžete vidět model neuronové sítě; zleva jsou vstupy, vpravo jsou výstupy. Šedé čtverečky označují perceptrony s výstupem 0, zelené znamenají jedničku.

Kliknete-li na nějaký perceptron, místo čtverečku se vám u něj a u celé předchozí vrstvy bude zobrazovat číslo – tyto čísla udávají práh a váhy daného perceptronu (práh je číslo na místě označené buňky, váhy jsou čísla v předchozí vrstvě). Kolem buněk v nižších vrstvách, než je označená buňka, se také vytvoří rámeček – ten orientačně popisuje to, jak perceptron, okolo kterého je rámeček, ovlivňuje hodnotu označeného perceptronu; čím je rámeček tmavší, tím více přispívá k tomu, aby byl výstup 0, čím je světlejší, tím více se podílí na aktivaci perceptronu.

Obrázek 13: Neuronový editor s označeným perceptronem

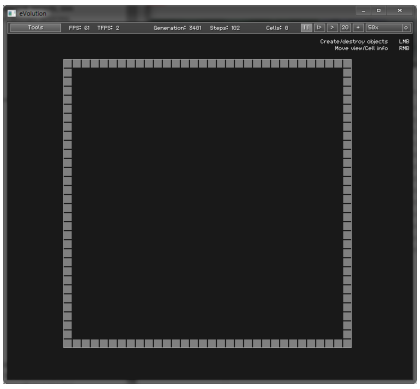
Váhy i práh můžete také buňce měnit – ukažte myší na dané číslo a otáčením kolečka myši přičítejte/odčítejte. Tato manipulace je poněkud nemotorná, nicméně účel editoru (i když se jmenuje editor, paradoxně) není upravovat buňky, spíš má pomáhat při pozorování a analýze.

Relevantní informace na toto téma také viz 2.2.1

3.5 Přednastavená schémata zdí

Jak jsem již napsal při popisu panelu nástrojů (viz 3.2), aplikace disponuje několika přednastavenými konfiguracemi/generátory zdí. Jsou to tyto:

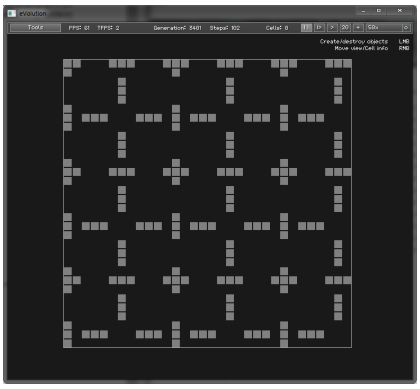
Edges ***(Okraje)***



Obrázek 14: Konfigurace zdí ‚Edges' *(Okraje)*

Základní konfigurace, užitečná pro většinu vašich pokusů. Brání buňkám v udržování jednoho směru pohybu (buňky mají tendenci hýbat se doleva a nahoru - viz 5.1). Také anuluje souvislost světa (viz 2).

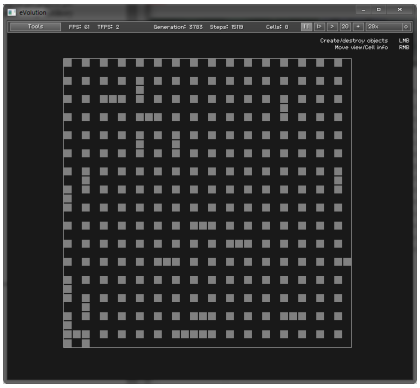
Rooms ***(Místnosti)***



Obrázek 15: Konfigurace zdí ‚Rooms' *(Místnosti)*

Ne příliš uplatitelná konfigurace. Byla vytvořena se záměrem přinutit buňky, aby se nezasekávaly v rozích, ale tento účel moc neplní.

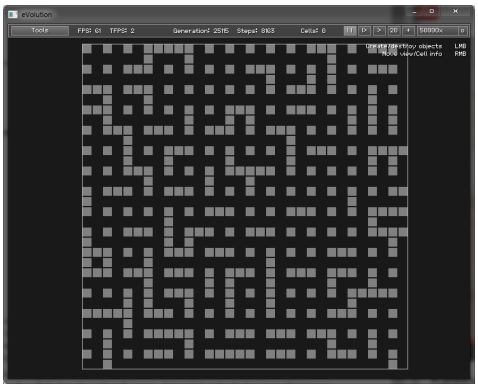
Small maze *(Malé bludiště)*



Obrázek 16: Konfigurace zdí ‚Small maze' *(Malé bludiště)*

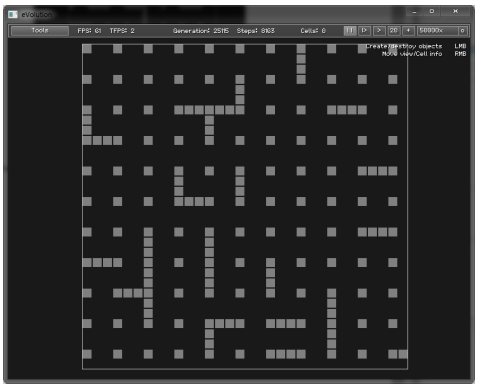
Název mluví sám za sebe. Tato konfigurace je generátor – pokaždé se vygeneruje jiné bludiště. Pokud chcete složitější bludiště, jednoduše ‚aplikujte' více bludišť na sebe pomocí tlačítka ‚L+'. Tato konfigurace bohužel většinou nutí buňky k tomu, aby se pohybovaly jen dvěma směry – jinak by narážely do jiných buněk.

Generátor nekontroluje, jestli chodby nerozdělí bludiště na více samostatných, mezi sebou neprostupných sekcí. Pokud se něco takového stane, doporučuji jednoduchou manuální úpravu.



Obrázek 17: Konfigurace zdí ‚Small maze' *(Malé bludiště)*, aplikováno opakovaně

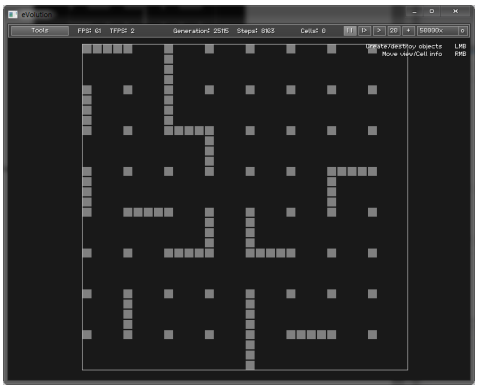
Medium maze ***(Střední bludiště)***



Obrázek 18: Konfigurace zdí ‚Medium maze' *(Střední bludiště)*

Obdoba malého bludiště, ale chodbičky jsou široké dvě políčka.

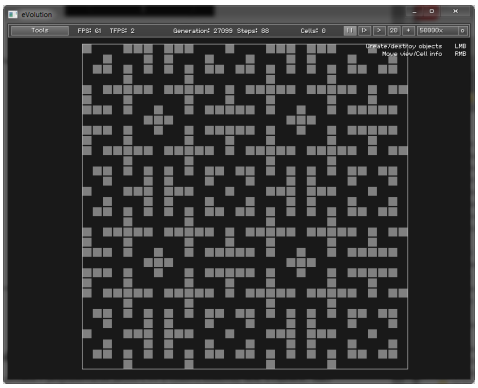
Big maze ***(Velké bludiště)***



Obrázek 19: Konfigurace zdí ‚Big maze' *(Velké bludiště)*

Další verze bludiště, tentokrát s chodbami širokými tři políčka.

China ***(Čína)***



Obrázek 20: Konfigurace zdí ‚China‘ *(Čína)*

Zdobené, ale pro buňky velice těžce stravitelné schéma, úzké plné rohů.

4 Technologie a kód

Program byl vytvořen v programovacím jazyce C++ a využívá knihovny SDL a OpenGL. Jako nadstavbu nad tyto knihovny používá mnou napsaný framework, který jsem použil i v jiných projektech. Kompletní zdrojové kódy můžete najít na přiloženém médiu. GUI komponenty, které jsou použity v uživatelském prostředí, jsou také moje dílo. Nejsou dokonalé, ale pro moje potřeby stačí.

Kód je bohužel kompilovatelný víceméně jen v prostředí Microsoft Visual Studio.

Paralelismus

Aplikace pracuje se dvěma vlákny; první vlákno se stará o celé jádro aplikace, GUI a vizualizace, a na druhém vlákně běží samotná simulace. Asi by bylo vhodnější použít víc vláken na simulaci světa, nicméně výkon je takto celkem postačující a navíc nevím, co bych tam mohl zparalelizovat (teoreticky by šla zparalelizovat senzorika a výpočty neuronových sítí, nicméně protože akce buněk by musely být tak jako tak řešeny sériově, buňky, které by byly vykonávány později, by už nemusely mít aktuální informace).

Vykreslovací informace se každý krok hlavního okna synchronizují s vláknem světa; uzamkne se mutex, aplikace si ze světa zkopíruje již připravená vykreslovací data (která jsou obnovována každých 30 ms) a vykreslí je. Animace buněk se vyvozují ze současné a předchozí pozice buňky a z průměrné rychlosti simulace světa (to je důvod, proč jsou animace chvíli sekavé, když z vysoké frekvence přejdete na pomalou – aplikace sbírá průměrnou rychlost simulace po určitých časových úsecích).

Pseudonáhodný generátor

Pro zaručení deterministického vývoje simulace jsem napsal vlastní pseudonáhodný generátor. Z dosavadních zkušeností s ním (ačkoli jsem ho nijak hlouběji nezkoumal) jsem s ním spokojen a nezaznamenal jsem, že by produkoval nějaké opakující se vzorce. Důležité je, že se dají serializovat a deserializovat jeho vnitřní proměnné.

Vznikl víceméně metodou pokus-omyl.

Kód je následující:

```
typedef UInt32 RandGen_t;

RandGen_t L1, L2, L3, L4, L5, L6, L7, L8
RandGen_t Z, Increment, Seed;
RandGen_t RandGen::roll() {

  L1 = Increment++ + Seed;

  L2 = ( L1 * ( L1 + 2 ) ) / 2 + 3;
  L3 = ( L2 * ( L1 + 3 ) ) / 5 + 7;
  L4 = ( L3 * ( L1 + 5 ) ) / 11 + 13;
  L5 = ( L4 * ( L1 + 7 ) ) / 17 + 19;
  L6 = ( L5 * ( L1 + 11 ) ) / 23 + 29;
  L7 = ( L6 * ( L1 + 13 ) ) / 31 + 37;
  L8 = ( L7 * ( L1 + 17 ) ) / 41 + 43;

  Z = Seed;

  Z = Z ^ ( Z << L1 ) + L1 + Seed + 2;
  Z = Z ^ ( Z << L2 ) + L2 + Seed + 3;
  Z = Z ^ ( Z << L3 ) + L3 + Seed + 5;
  Z = Z ^ ( Z << L4 ) + L4 + Seed + 7;
  Z = Z ^ ( Z << L5 ) + L5 + Seed + 11;
  Z = Z ^ ( Z << L6 ) + L6 + Seed + 13;
  Z = Z ^ ( Z << L7 ) + L7 + Seed + 17;
  Z = Z ^ ( Z << L8 ) + L8 + Seed + 20;

  return Z;
}
```

5 Experimenty v prostředí

Nyní se, s využitím názorných příkladů, pokusím popsat některé jevy, kterých jsem si všiml. P**ro každý z těchto experimentů najdete v aplikaci simulaci pojmenovanou ‚Experiment #‘**, kde křížek zastupuje číslo podkapitoly (načítání viz sekce 3.2).

5.1 Buňky chodí vlevo nahoru

V tomto experimentu používám své standardní podmínky prostředí – uvidíte je ve více experimentech. Buňky nijak neomezuji – dám jim dostatek jídla a neomezený pohyb. Díky těmto povolným podmínkám se vám generace uchytí celkem rychle (v příkladu se uchytila třináctá, ale neváhejte si udělat i vlastní generaci), ale víceméně stagnuje.

Buňky si vystačí s naprosto primitivním předpisem pohybu – v naprosté většině případů se pohybují pouze vlevo a nahoru (v některých ojedinělých případech se může ustálit jiný směr). To se dá odůvodnit následujícím způsobem:

1. Pohyb vlevo je pro buňku nejjednodušší – s generátorem sítě ‚One layer‘ (viz 3.4.1) bývá najednou aktivováno (u Adamovských mozků) cca 20 % perceptronů. Pravděpodobnost, že je alespoň jeden z nich perceptron ovlivňující směr (viz 2.3), není až tak velká.
2. Zmutuje-li nějaká buňka natolik, že se začne hýbat vpravo, většinou narazí na ‚normální‘ buňku, která se hýbe vlevo, a zahyne na nedostatek potravy. Buňce se nevyplatí jít proti proudu.
3. Pohyb nahoru je jednodušší, než pohyb dolů, protože pohyb dolů potřebuje mít aktivní oba dva pohybové bity (viz 2.3).
4. Zmutuje-li nějaká buňka natolik, že se začne hýbat dolů, často narazí na buňky jdoucí v opačném směru a zemře.

5.2 Zdi, prosím!

Nyní se podívejte, co se stane, když přidáme zdi. Podmínky jsou identické jako v předchozím experimentu, jen jsem načetl schéma zdí ‚Edges‘ *(Okraje)*.

První, co se dá vypozorovat, je to, že trvá mnohem déle najít generaci, která je schopná se uchytit. U ukázkové generace to dalo 1062 pokusů, než se našla taková, která byla schopná dlouhodobě přežít.

Další věc, které byste si mohli všimnout, je to, jakým způsobem řeší buňky problém se zdmi (zde se jedná o analýzu konkrétního případu, věci v jiných experimentech pravděpodobně dopadnou jinak). Zatímco v předchozím pokusu mohly jít donekonečna v libovolném směru, zde jim

v tom zdi brání. Podívejte se v nějakém brzkém cyklu (cca do tisíce) na mozek nějaké buňky. První pohybový bit (0 = vlevo nebo nahoru, 1 = vpravo nebo dolů; viz 2.3) je výrazně ovlivněn vstupem ‚Energy over 80' *(Energie nad 80)*. Tedy buňky, které mají více energie než 80, se hýbou vpravo nebo dolů, a buňky, které mají energie méně, se hýbou vlevo nebo nahoru. Směr je poté upřesněn druhým pohybovým bitem. Ten je závislý:

1. na životech
2. na tom, zda je jídlo nalevo od buňky (v tomto případě poněkud kontraproduktivní – tento reflex způsobuje, že buňka půjde spíše od jídla, než k němu)
3. na časovači (díky tomu chodí buňky ‚cik cak')

Nicméně buňky s takovýmto mozkem už jsou schopné v daném prostředí přežít.

5.3 Pokrok

Nyní se posuneme v té samé simulaci o sto tisíc cyklů dopředu (pro ušetření vašeho času jsem simulaci po 100000 cyklech uložil) a podíváme se, co se změnilo.

Buňky se stále hýbají ‚cik cak', nicméně už celkem reagují na jídlo v jejich okolí. Podívejme se na reflexy buňky:

Pohyb, bit 1 (0 = vlevo/nahoru, 1 = vpravo/dolů)

Zdraví a energie
Po sto tisících cyklech už zmutovaly i hlubší vrstvy sítě. Většina senzorů na hodnotu zdraví a života zmutovala natolik, nejsou aktivní, nebo je jejich chování nejednoznačné.
Buňka reaguje poněkud zvláštně na její zdraví a energii. Díky střídajícím se kladným a záporným vzruchům na jednotlivých hladinách kolísá celkový efekt energie buňky na směr jejího pohybu.
Na vyšších hladinách energie (160+) se kladné a záporné vzruchy do jisté míry vyvažují, buňka je tedy citlivější na ostatní podněty.
Při energii mezi 80 a 160 je ale jednoznačně dominuje vstup ‚Energy over 80' *(Energie nad 80)*, který buňku nutí pohybovat se vpravo nebo dolů.
S energií pod 80 je energie opět relativně v rovnováze.

Buňka nad
Má-li buňka nad sebou jinou buňku, snaží se jít vpravo/dolů, což je dobře.

Jídlo napravo
Má-li buňka napravo od sebe jídlo, má tendenci k němu jít, což je pro ni také dobré.

Zeď nalevo a nahoře
Toto se zdá být pro buňku celkem destruktivní reflex – díky negativním vzruchům, které dostává, je-li nalevo nebo nahoře zeď, má tendenci jít do ní. Nicméně tento reflex je celkem vyvážen tím, že má-li buňka nízkou energii (což jednou bude mít, když bude pořád narážet do zdi), je tlačena jít doprava/dolů.

Něco ob jeden
Tyto reflexy se také mohou hodit – buňka má tendence přiblížit se k neznámému objektu, který je ob jedno políčko (platí opačně pro objekt pod buňkou).

Pohyb, bit 2 (0 = vlevo/vpravo, 1 = nahoru/dolů)

Energie a zdraví
Zde opět nacházíme poněkud kolísavé a nejasné nastavení. Nicméně ono kolísání se nekryje se vzorcem u prvního bitu – tady má váhu spíše zdraví, zatímco u prvního bitu měla váhu energie. Tímto tedy nemusí docházet k tomu, že by buňka chodila pouze dvěma směry (že by oba bity reagovaly stejně na podněty). Váhy na zdraví jsou ale dost silné, takže reakce mladých buněk budou poněkud otupělé.

Okolní objekty
Buňka má vyvinuté dobré reflexy pro buňky dole (ačkoli se zdá, že i pro buňky napravo, tento reflex je ale v hlubších vrstvách narušen). Za zmínku také stojí ‚Food on right' *(Jídlo napravo)* a nevýhodně vyvinutý reflex, když je jídlo nalevo (díky kterému jde potom buňka nahoru nebo dolů). Na okolní zdi není žádná velká reakce.

Časovač
Druhý pohybový bit má silnou reakci na časovač. To způsobuje, že buňka má tendence chodit střídavě nahoru/dolu a vlevo/vpravo – ve výsledku tedy do diagonály.

Akce, bit 1 (0 = pohyb/zabití, 1 = reprodukce/sex.reprodukce)

Jelikož reakce na druhý bit akce jsou dost otupělé (musí se najednou splnit dost podmínek, aby se aktivoval), je na místě brát bit 1 jako bit aktivující reprodukci. V tomto příkazu je zajímavý jenom jeden typ vstupu:

Zdraví a energie
Ačkoli se může zdát, že reprodukce reaguje na zdraví, až tak moc to neplatí – senzorika pro zdraví hodně zmutovala. Celkem dobře se ale ještě drží energie, konkrétně vstup ‚Energy over 160' *(Energie nad 80)*, což se zdá být žádoucí.

5.4 Náhlá změna podmínek

Vzal jsem náhodnou buňku z předchozího experimentu 'Pokrok' (viz 5.3) a nastavil její síť jako výchozí pro Adamy. Poté jsem snížil počet Adamů na jednoho a pozoroval vývoj generací v těchto rozloženích zdí:

Edges
Jelikož se mozek vyvinul v tomto prostředí, není divu, že jen v naprosto minimálním počtu experimentů (experimenty se lišily rozmístěním jídla a počáteční pozicí Adama) se generace neuchytila.

Rooms
Na toto prostředí si buňky celkem dobře zvykají. Vydrží více než desetina experimentů.

Small maze
Zde už je to horší. Na nejzákladnějším bludišti se udrží každý tisící až desetitisící experiment. Na druhou stranu, buňky se pak udrží, když se postupně zvyšuje obtížnost bludiště.

China
Na tomto rozložení se buňky vůbec neuchytily.

6 Závěr

Během pokusů (tedy ve finální fázi projektu) jsem zjistil, že prostředí mnou vytvořené zdaleka nevyužívá plný potenciál daného problému. Míra toho, jak je toto prostředí jednoduché, také degraduje potenciál dalšího vývoje a pozorování.

Udělat plnohodnotné prostředí by ale byl pro mé poměry megalomanský projekt. Myslím, že by bylo nutné nabídnout uživateli mnohem větší možnosti úpravy prostředí - vlastní objekty, možnost přidat další proměnné buňkám, možnost kompletně si přizpůsobit vstupy a výstupy do neuronové sítě, třeba i možnost kompletně navrhnout neuronovou síť. Také by byla vhodná větší podpora paralelismu.

Za cíl práce jsem si dal vytvořit prostředí, což jsem splnil, a poté udělat pár pozorování. To jsem také uskutečnil. Pokusil jsem se popsat základní jevy, které za těchto podmínek vznikaly. S postupem času však výsledek tohoto projektu považuji spíš za hračku, než za nástroj pro výzkum.

7 Obsah disku

Disk obsahuje aplikaci a digitální podobu seminární práce ve formátu PDF. Spustitelný soubor aplikace naleznete ve složce ‚binbin‘. Má dvě verze - normální by měla být spustitelná na všech PC s Windows Vista a novější, verze s postfixem ‚_xp‘ by měla být spustitelná i na Windows XP. Zdrojové kódy jsou potom ve složce ‚src‘.

PDF dokument je přímo v kořenu disku.

8 Seznam obrázků

9 Použité zdroje

- Neodarwinismus. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-03-01]. Dostupné z: http://cs.wikipedia.org/wiki/Neodarwinismus
- Darwinismus. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-03-01]. Dostupné z: http://cs.wikipedia.org/wiki/Darwinismus
- Přirozený výběr. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-03-01]. Dostupné z: http://cs.wikipedia.org/wiki/Přirozený_výběr
- Perceptron. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-03-01]. Dostupné z: http://cs.wikipedia.org/wiki/Perceptron
- Neuronová síť. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-03-01]. Dostupné z: http://cs.wikipedia.org/wiki/Neuronová_síť
- ŠÍMA, Jiří. Teoretické otázky neuronových sítí. Vyd. 1. Praha: MATFYZ press, 1996, 390 s. ISBN 80-858-6318-9.