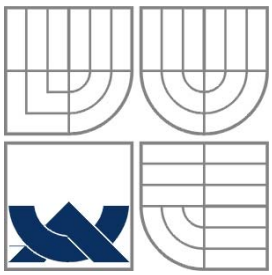


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

INTERAKTIVNÍ SIMULACE S VYUŽITÍM TECHNOLOGIE FLASH

INTERACTIVE SIMULATION BY MEANS OF FLASH TECHNOLOGY

DIPLOMOVÁ PRÁCE
DIPLOMA THESIS

AUTOR PRÁCE **PAVEL LÁTAL**
AUTHOR

VEDOUCÍ PRÁCE **ING. RADOMIL MATOUŠEK, PH.D.**
SUPERVISOR

BRNO 2008

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2007/08

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

student(ka): Látal Pavel

který/která studuje v **bakalářském studijním programu**

obor: **Aplikovaná informatika a řízení (3902R001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Interaktivni simulace s využitím technologie Flash

v anglickém jazyce:

Interactive simulation by means of Flash technology

Stručná charakteristika problematiky úkolu:

Daná bakalářská práce bude realizovat několik interaktivních simulačních modelů z oblasti technických i netechnických věd. K tvorbě bude využita technologie Adobe Flash.

Cíle bakalářské práce:

- Abstrahovat softwarovou technologii Adobe Flash na úrovni potřebné pro tvorbu interaktivních aplikací.
- Vytvoření min. šesti samostatných interaktivních aplikací, realizujících simulaci, školitelem určených, dynamických systémů.
- Stručný teoretický popis modelovaných systémů. Výstup jak písemnou formou (textová část bakalářské práce), tak elektronickou formou (html text).
- Stručný popis vybraných částí zdrojového kódu AS.

Seznam odborné literatury:

[1] Rebenschield, S.: Macromedia Flash 8 Professional, Computer Press, 2007, ISBN: 978-80-251-1696-8
[2] Matoušek, R.: Simulace dynamických systémů, elektronická podpora, http://www.uai.fme.vutbr.cz/~matousek/, 2007

Vedoucí bakalářské práce: Ing. Radomil Matoušek, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2007/08.

V Brně, dne 7. 03. 2008

L.S.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA STROJNÍHO INŽENÝRSTVÍ DĚKANÁT 14

Šeda

doc. RNDr. Ing. Miloš Šeda, Ph.D.
Ředitel ústavu

v.z. R. Doupovec

doc. RNDr. Miroslav Doupovec, CSc.
Děkan fakulty

LICENČNÍ SMLOUVA

(na místo tohoto listu vložte vyplněný a podepsaný list formuláře licenčního ujednání)

ABSTRAKT

Práce se zabývá vytvořením šesti interaktivních simulací pomocí technologie Flash. Vytvořené interaktivní simulace jsou následující: rozšířená varianta Conwayova celulárního automatu realizovaná v ortogonální a hexagonální mřížce, simulace 1D celulárního automatu, demonstrace vybraných selekčních principů evolučních algoritmů, možné grafické zobrazení 2D Turingova stroje a aplikace demonstrující příklad chování mravenčí kolonie.

ABSTRACT

This bachelor thesis is interested in designing of six interactive simulations by means of Flash technology. The presented interactive simulations are follows: an extended version of Conway's cellular automata which was realized on orthogonal and hexagonal grid, simulation of 1D cellular automata, demonstration of selected selection principles of evolutionary algorithms, possible graphic representation of 2D Turing machine, and application for demonstration of ant colony behavior.

KLÍČOVÁ SLOVA

Flash, ActionScript, celulární automat, selekční principy, mravenčí kolonie, Turingův stroj

KEYWORDS

Flash, ActionScript, cellular automaton, selection principles, ant colony, Turing machine

Obsah:

1 ÚVOD

Tato práce se zabývá vytvořením šesti interaktivních aplikací pomocí technologie Flash, které simulují či demonstrují chování vybraných dynamických systémů. Všechny tyto aplikace jsou určeny pro názornější demonstraci zvolených problematik a jsou vhodné i pro výuku.

V textové části je ve druhé kapitole je stručná teorie o technologii Adobe Flash. O jejím vzniku, historii a vývoji. Součástí této kapitoly jsou i základní informace o nástrojích, které jsou jádrem Flash. Zmíněn je i implementovaný programovací jazyk ActionScript a jeho vývoj. Je zde také zmíněna konkurenční technologie. Dále se tato část práce zabývá popisem vytvořených aplikací - interaktivních simulací, kdy v každé kapitole je vždy stručný úvod do dané problematiky, její rozbor v rámci realizované aplikace a nakonec detailní popis sloužící jako manuál k vytvořené aplikaci. Součástí práce je také příloha obsahující vybrané a popsané části zdrojových kódů z jednotlivých aplikací.

Praktickým úkolem je tvorba jednoduchých interaktivních simulací ve vývojovém prostředí Adobe Flash CS3, což je v současné době nejnovější dostupný nástroj pro tvorbu flashových projektů. Flash je současné době je nejvyužívanějším nástrojem pro tvorbu dynamických stránek a jeho možnosti jsou velmi rozsáhlé. Oproti jiným technologiím má z hlediska uživatele mnoho výhod, jako například nezávislost na platformě, cca. 90% rozšíření, či poskytnutí jednotného prostředí a nástroje pro spouštění výsledných aplikací. Toto má za důsledek jednotné zobrazení výsledné grafické podoby ve všech internetových prohlížečích, jenž mají, nebo umožňují zakomponovat tzv. Flash player, nebo-li nástroj pro zobrazování Flash aplikací. Díky implementovanému programovacímu jazyku, tato technologie umožňuje nejširší škálu využití. Od jednoduchých animovaných menu až po složité simulační programy. I při skutečnosti, že Flash nedisponuje silným výpočetním jádrem, tak i přesto je velmi výkonným nástrojem pro tvorbu různých druhů aplikací.

Praktickým výsledkem této bakalářské práce je šest vytvořených Flashových, interaktivních, simulačních aplikací. Byly vytvořeny dvě aplikace na bázi celulárních automatů, které demonstrují chování systému „buněk“ v závislosti na nastavených pravidlech. Toto lze srovnat s chováním reálných, buněčných organizmů. Dále aplikace demonstrující 1D celulární automat, zobrazující rozvoj vstupní posloupnosti buněk na základě zadaných pravidel. Jde o příklad možného získávání pseudonáhodných čísel a to převedením výstupního obrazce na posloupnost bitů. Dále pak aplikace pro demonstraci selekčních principů genetických algoritmů, zobrazující tři základní a jednu speciální metodu výběru jedinců. Vybrané metody znázorňují způsoby jakými dochází k přirozenému výběru, nebo k výběru jedinců při šlechtění dobytka. Následuje varianta zobrazení 2D Turingova stroje, což je aplikace znázorňující zdánlivý paradox, že i jednoduchá pravidla mohou vést ke komplikovaným výsledným obrazcům. Na závěr byla vytvořena zjednodušená simulace chování reálného biologického systému, konkrétně mravenčí kolonie.

2 TECHNOLOGIE FLASH

Adobe Flash, dříve nazývaný Macromedia Flash je soustava multimediálních technologií, jež jsou od prosince roku 2005 vyvíjeny společností Adobe Systems, která získala společnost Macromedia i s touto technologií. Flash je nejčastěji používán k tvorbě animací, reklamních bannerů, interaktivních internetových prezentací, ale také k vytváření běžných aplikací.

Flash využívá přednostně vektorovou grafiku, což do značné míry snižuje velikost výsledné aplikace, ale současně lze využívat i bitmapové grafické prvky. Také podporuje video streaming, který získává stále větší uplatnění. Flash se stále rozšiřuje a je již podporován velkou většinou webových prohlížečů a dokonce mnoha mobilními zařízeními. Mobilní telefony, či PDA již dokáží zobrazovat tyto aplikace.[1]

2.1 Historie technologie Flash

V lednu roku 1993 založili Michelle Welsh, Charlie Jackson a Jonathan Gay softwarovou společnost FutureWave Software. Prvním projektem byla aplikace pro tvorbu grafiky s názvem SmartSketch. Ačkoli byl tento program dosti inovativní, tak si i přes to nedokázal získat pevné místo na trhu. S rozšiřováním internetu přišla na řadu myšlenka vektorové grafiky a vektorových animací, jež by mohly v budoucnu nahradit datově objemnou technologii Shockwave společnosti Macromedia. Roku 1995 společnost FutureWave upravila svůj program SmartSketch přidáním animačních prvků a vydala jej pod názvem FutureSplash Animator. Poté v prosinci 1996 společnost Macromedia získala tento vektorový animační software a později jej po úpravách vydala pod názvem Flash 1.0. [1]

2.2 Vývoj technologie Flash

Prvním v řadě nástrojů pro tvorbu a přehrávání technologie flash byl Macromedia Flash 1 se stejnojmenným přehrávačem z roku 1996. Rok po té následovala verze Macromedia Flash 2 s podporou stereofonního zvuku, rozšířenou integrací bitmap a dalšími vylepšeními. Další verze Macromedia Flash 3 vyšla opět po roce a přinesla vylepšení animací a jednoduché příkazy pro vyšší interaktivitu výsledných aplikací. Verze přehrávače Macromedia Flash 4 z roku 1999 zaznamenala 100 miliónů instalací a to hlavně díky společnosti Microsoft, která jej zakomponovala do svého internetového prohlížeče Internet Explorer 5. Od roku 2000 byl Flash distribuován se všemi Netscape a Internet Explorer prohlížeči a o dva roky později je dodáván se všemi verzemi Windows XP. Program Flash se dostal k 92% uživatelům internetu. Flash 4 mimo jiné již uměl streamovat hudbu ve formátu MP3 a byl rozšířen o prvek Motion Tween, tedy doplnění pohybu do snímků animace. Velkým krokem kupředu byla verze Macromedia Flash 5, jež vyšla opět po roce. Hlavní přínos mělo začlenění programovacího jazyka ActionScript, později pojmenovaného ActionScript 1.0, a možnost úpravy uživatelského prostředí autorského programu.

Následovníkem verze 5 byl Macromedia Flash MX, který vyšel roku 2002 spolu s Flash playerem 6. Této verzi oproti předchozí přibyla podpora videa, sdílených knihoven, a další prvky. Součástí balíku vydaného tohoto roku byl také Macromedia Flash Communication Server MX, jež právě umožňoval streamovat video-soubory do flash playeru 6.

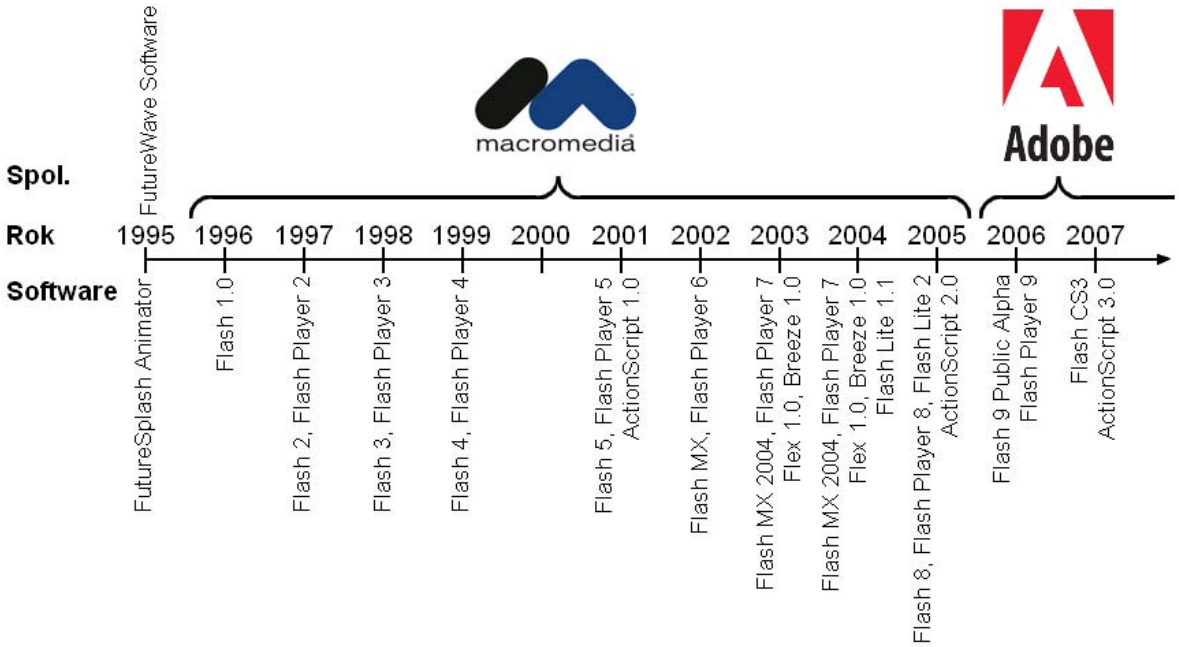
Spolu s dalším Macromedia Flash MX 2004 ze září 2003, vyšel i Flash player 7.

Přibyly tabulky, grafy a pokročilé textové efekty. Tato verze Flash byla obohacena o tvorbu aplikací pro mobilní zařízení. Současně byla uvedena nová verze programovacího jazyka ActionScript 2.0, která již poskytuje plně objektově orientované programování. Další změnou bylo, že distribuce této verze byla rozdělena na takzvanou Basic a Profesional verzi. V profesionální verzi byly zahrnuty navíc, oproti základní verzi, další možnosti a rozšíření tvorby Flash aplikací. Současně vyšel i přehrávač pro mobilní zařízení Flash Lite 1.1.

Následující rok přišel na řadu Macromedia Flash 8. Tento představoval veliký skok kupředu v této technologii od doby kdy vyšla verze 5. Obsahoval nové efekty, nový videokodek On2 VP6, vylepšený renderovací engine zvaný Flash Type a několik vylepšení pro ActionScript 2.0, jako například nová třída BitmapData a podobně. V témže roce se mobilní zařízení dočkaly nového přehrávače s označením Macromedia Flash Lite 2.

Dne 3. prosince 2005 společnost Adobe Systems získala firmu Macromedia spolu s technologií Flash a tak následující verze flashe z roku 2006 se již nejmenovala Macromedia, ale Adobe Flash a byl to Adobe Flash Player 9, který vůbec poprvé vyšel aniž by současně byl vydán autorský program. Na ten si zájemci museli ještě chvíli počkat. Ještě týž rok byl uvolněn do prodeje Adobe Flash 9 Public Alpha. Spolu s dalšími vylepšeními a integrací některých Adobe produktů v sobě obsahoval novou verzi známého programovacího jazyka ActionScript 3. Mimo jiného se dočkal i uvedení nový Flex 2.0. Flex je soubor nástrojů rozšiřujících možnosti tvorby inteligentních internetových aplikací na platformě flash.

Brzkým a zatím posledním následovníkem byl Adobe Flash CS3, který právě roku 2007 vzešel z verze Flash 9 Public Alpha, po němž zdědil jazyk ActionScript 3.0 a přidal nový XML engine. Také se dočkala svého uvedení nová verze Flex s pořadovým číslem 3.[1]



Obr. 1 Časová osa vývoje technologie flash.

Budoucí vývoj v oblasti flash technologií se značně zaměřuje na mobilní zařízení, u nichž společnost Adobe Systems usiluje, aby se program Flash Lite stal jakýmsi uživatelským

prostředím. Dalším projektem společnosti Adobe je Adobe AIR Project, který má umožnit vývojářům webových aplikací využít své současné znalosti ke tvorbě takzvaných Rich Internet Aplications (RIAs), tedy inteligentních internetových aplikací. V souvislosti s tímto projektem bude v roce 2009 uveden nový Flex 4.

2.3 Programovací jazyk ActionScript

Od verze Macromedia Flash 5.0 z roku 2001 je součástí této technologie také skriptovací jazyk ActionScript, který dodal Flashovým aplikacím žádanou interaktivitu, tedy možnost ovlivňování běhu aplikace člověkem. Je to programovací jazyk založený na ECMAScriptu. V roce 2000 vznikla první verze tedy ActionScript 1.0. Tato byla do značné míry ovlivněna standardy JavaScriptu a ECMA-262 (třetí edice) skriptu. V této verzi byla již podpora objektového programování. Lokální proměnné mohly být deklarovány příkazem *var* a uživatelé definovali funkce s parametrem *return*, tedy návratovou hodnotou funkce. Hlavní výhodou bylo, že uživatelé namísto toho, aby ActionScript psali v textovém editoru tak výsledný kód se sestavoval pomocí roletových menu a dialogových oken. S následující verzí vývojového prostředí Flash MX zůstal jazyk v základu nezměněn. Došlo jen na drobné úpravy jako například doplnění příkazu *switch.*

Součástí verze Macromedia Flash MX 2004 byl již programovací jazyk ActionScript 2.0, jež byla představena v září roku 2003. Vznikl jako odezva na požadavky uživatelů na tvorbu větších a komplexnějších aplikací. ActionScript 2.0 je nově vybaven kontrolou napsaného kódu během kompilace. Díky tomu mohou programátoři snadněji odhalovat chyby v syntaxi, nebo nesprávné hodnoty v proměnných, neboť v této verzi je možné proměnným určovat daný typ proměnné. Součástí je také dědičnost tříd, takže programátoři mohou využívat třídy stejně jako v jazycích Java, či C++.

V roce 2006 byla zveřejněna dosud poslední verze ActionScriptu s označením 3.0. V této verzi došlo k zásadní změně. ActionScript 3.0 poskytuje nejenom významné zlepšení výkonu, ale i mnohem robustnější programovací model.[2]

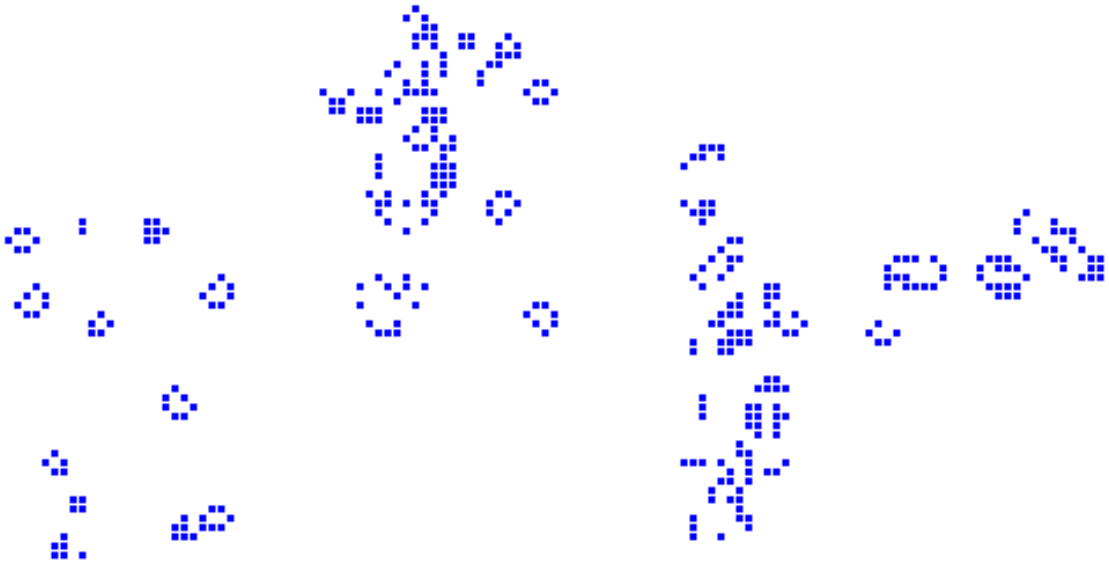
2.4 Konkurence

Konkurencí technologie Flash se v budoucnu může stát společností Microsoft vyvíjená technologie Silverlight, která se má stát součástí balíku Microsoft Expression. Součástí tohoto balíku bude také aplikace Expression Web, jež nahrazuje již zastaralý Microsoft FrontPage a konečně tvoří plně validní kód vyhovující nepsaným standardům XHTML a CSS. Oproti Flash technologie Silverlight pracuje s XML strukturou a sousta věcí se odehrává na serveru. Podobně jako u Flashe je třeba do prohlížeče nainstalovat malý modul pro zobrazení prvků Silverlight. Prozatím je tato technologie dostupná pro operační systémy Windows a MacOS X. Pro Linux se podpora zatím připravuje.[3]

3 CONWAYŮV CELULÁRNÍ AUTOMAT

Celulární automat ve své nejzákladnější formě si můžeme představit jako nekonečnou mřížku, kde každé jednotlivé pole představuje jednu buňku. Každá buňka může mít několik stavů, přičemž dva základní mohou být - živá/mrtvá. Dále je dán soubor pravidel, jakými se bude život na dané mřížce řídit. Tato pravidla určují, za jakých podmínek buňka přežije, zemře, nebo se na jejím místě zrodí nová.

Vyhodnocování pravidel se děje v diskrétních okamžicích, takzvaných generacích, kdy každá následující generace je výsledkem vyhodnocení generace předchozí. V každé generaci je přesně dána generace následující, jedná se tedy o deterministický systém. Příkladem takovéhoto celulárního automatu je populární Game of life, který navrhnul roku 1970 matematik John Conway z Princeton university v USA.



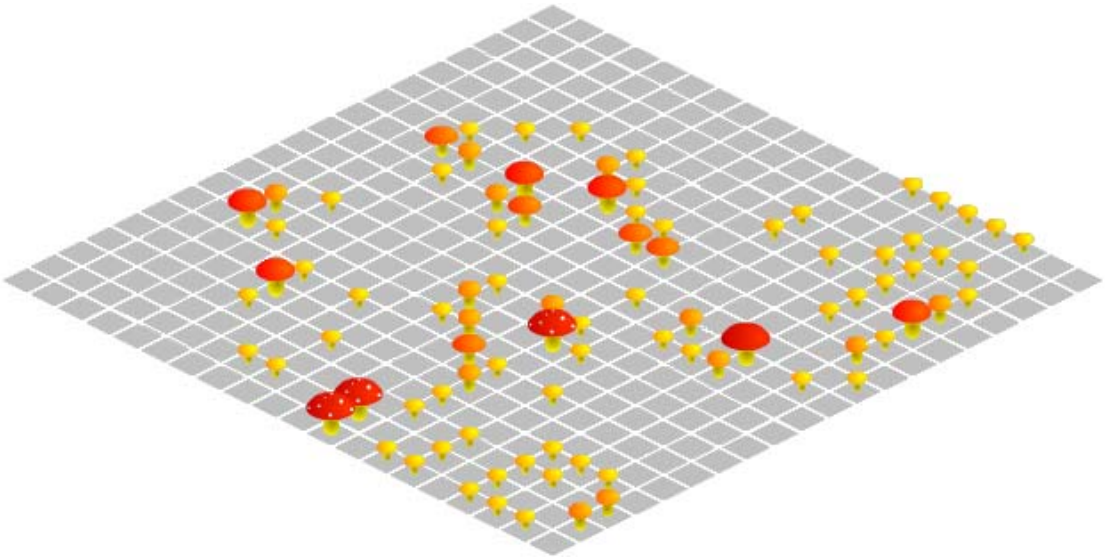
Obr. 2 Ukázka originální varianty celulárního automatu Game of life.

Na myšlenku vytvořit tuto hru jej přivedl roku 1940 renomovaný matematik John Von Neumann, který se zabýval vytvořením hypotetického systému, který by byl schopen replikovat sebe sama. John Von Neumannův systém se skládá z velmi komplikovaných kombinací matematických pravidel a rovnic. Hra Game of life je takovým zjednodušením Von Neumannovy myšlenky.

John Conway poprvé hrál Game of life na desce pro hru GO a jako buňky mu posloužily figurky z této hry. Daná pravidla hry byla zvolena s ohledem na výsledné vlastnosti soustavy, neboť při některých kombinacích pravidel docházelo k příliš rychlému vymírání buněk a při jiných zase k jejich přílišnému přibývání. Pravidla, jimiž se řídí Game of life jsou také označována číselně, například 23/3. První dvě číslice označují při jakém počtu živých sousedů bude buňka dále žít a číslice za lomítkem označuje při jakém počtu živých sousedů se zrodí nová buňka. Existují varianty s označením 16/6, nebo takzvaný HighLife, který má označení 23/36. Existují i jiné varianty, ale ty jsou ve velké většině příliš chaotické, nebo brzy vymírají.

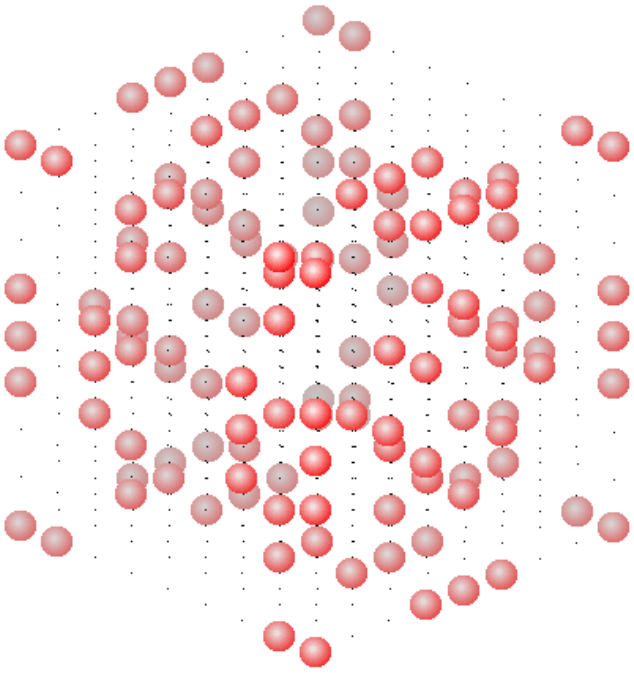
Jistým obohacením této hry může být také vyjádření generací jednotlivých buněk. Tím je myšleno, že čím více generací buňka vydrží živá tím vyšší hodnotu má(vyjádřeno například

změnou barvy). Vyšší hodnota však nemá vliv na daná pravidla hry a tak takováto buňka umírá stejně „snadno“ jako jakákoliv jiná buňka. Variantou takovéhoto modelu je například MushroomLife, kde jako buňky slouží houby, které se s každou další generací od svého zrození zvětšují do doby než dojde k jejich odumření.[6]



Obr. 4 Ukázka výstupu aplikace MushroomLife.

Velmi zajímavou implementací principu Game of life je její převedení do 3D prostoru. Přidaný rozměr znamená, že každá buňka, oproti 2D provedení, má navíc 18 sousedních, což s patřičnou úpravou pravidel vytváří nový náhled na problematiku celulárních automatů, neboť až doposud se takřka výhradně jednalo o plošnou mřížku buněk.[7]

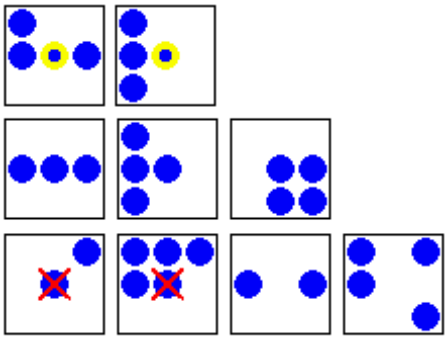


Obr. 5 Ukázka z aplikace 3D Game of life.

3.1 Princip metody

Game of life není vlastně hra v pravém slova smyslu. Nelze ji hrát ani vyhrát, i když existují i varianty, které jistým způsobem hrát lze. Buňky jsou na počátku rozmístěny ve startovních pozicích na čtvercovém poli. Game of life se standardně hraje na čtvercové mřížce, tedy každá buňka má osm sousedních. Podle toho kolik je z nich živých se určuje zda daná buňka v následující generaci přežije, zemře, nebo pokud byla mrtvá, zda se na jejím místě zrodí nová.[5]

Na obrázku č. 3 jsou znázorněna pravidla originální varianty celulárního automatu Game of life. Pokud má mrtvá buňka právě tři živé sousední buňky, pak se na jejím místě v následující generaci zrodí nová. Pokud má živá buňka ve svém okolí dvě nebo tři živé buňky, pak přežije do následující generace. Jestliže má však buňka ve svém okolí více než tři a méně než dvě živé buňky, pak v následující generaci umře.[5]



Obr. 3 Ukázka varianty pravidel celulárního automatu Game of life

3.2 Conwayův celulární automat na ortogonální mřížce

Jedná se o aplikaci Conwayovy Game of life. Buňky jsou umístěny na čtvercové mřížce o konečném počtu polí. Rozměr mřížky je nastavitelný v daných mezích, přičemž maximum bylo stanoveno s ohledem na výpočetní náročnost aplikace. Každá buňka má dva stavy – živá, mrtvá. Pravidla, jakými se bude řídit určování další generace, jsou nastavitelná. Díky tomuto lze rozšířit pohled na tento celulární automat a pozorovat chování systému i za jiných podmínek, než jak jsou stanoveny pro originální variantu Game of life.

3.3 Conwayův celulární automat na hexagonální mřížce

Jde o alternativní verzi předchozího celulárního automatu, avšak se zásadním rozdílem. Oproti klasickým celulárním automatům tento pracuje na hexagonální mřížce. Toto dává nový náhled na tuto problematiku. U drtivé většina konvenčních celulárních automatů je založena na ortogonální, tedy pravoúhlé mřížce. Zde má však každá buňka namísto osmi pouze šest sousedních buněk.

3.4 Popis aplikace

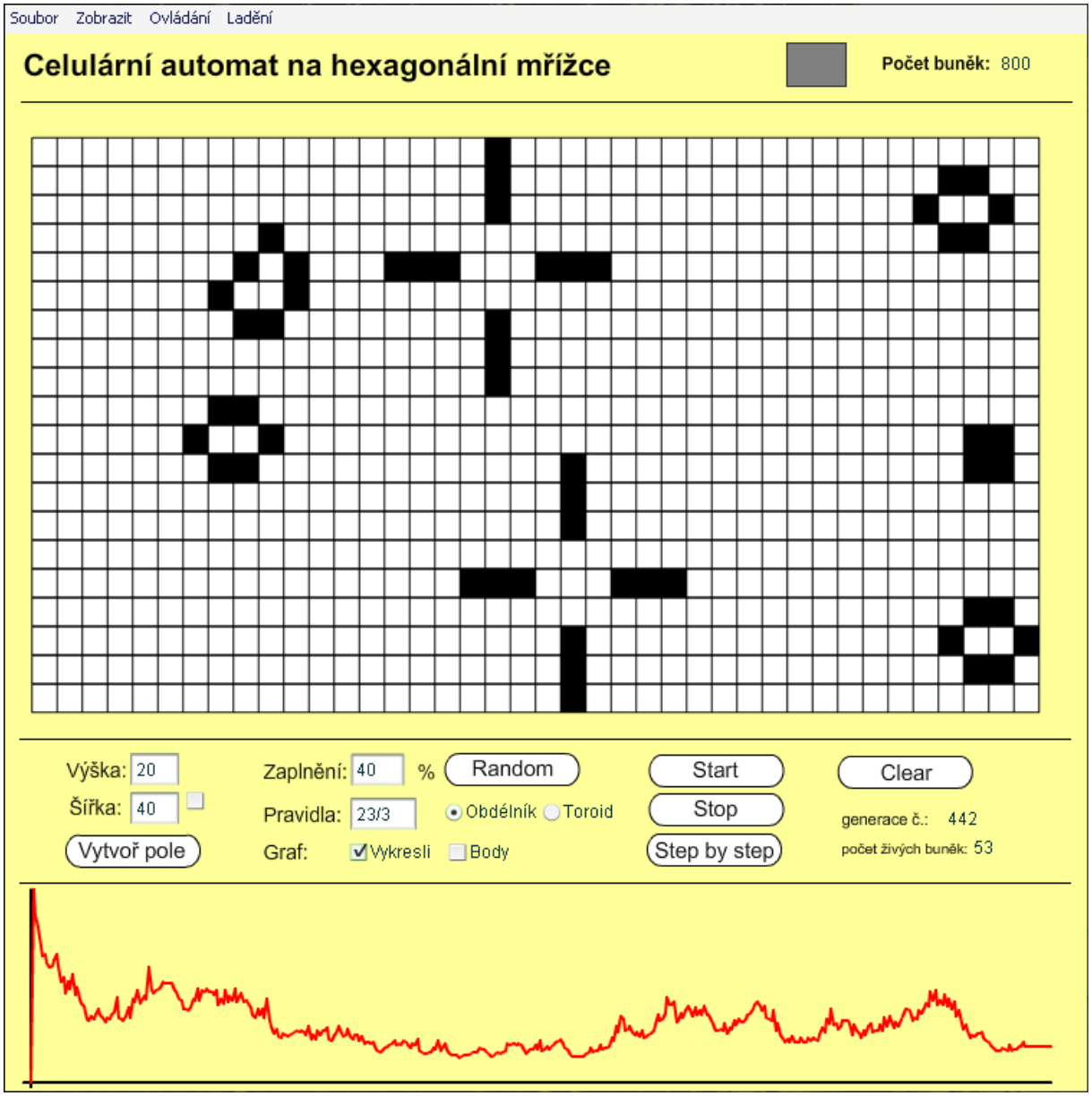
Horní polovinu aplikace tvoří vlastní čtvercové, či hexahgonální pole buněk, kdy kliknutím na jakoukoliv buňku dojde k jejímu překlopení do opačného stavu než v tom, v němž se nacházela. Lze tak nastavit původní konfiguraci buněk pro další vývoj při běhu aplikace.

Pod tímto polem se nachází panel ovládacích prvků. Po spuštění aplikace postupujeme z zleva po sloupcích. V prvním sloupci nejprve nastavíme rozměry pole. Hodnoty udávají počet buněk v horizontálním a ve vertikálním směru. Po zakliknutí zaškrtávacího rámečku

vedle textového pole *Šířka* se bude šířka vypočítávat automaticky při změně výšky pole. Toto má za následek, že výsledné pole bude složeno z rovnoměrných čtverců případně šestiúhelníků. Po kliknutí na tlačítko *Vytvoř Pole* se v horní části aplikace vygeneruje dané pole buněk.

Pro nastavení výchozí konfigurace lze využít buď již zmíněný způsob, tedy ruční zadání živých buněk klikáním na jednotlivá pole, nebo v druhém sloupci ovládacích prvků funkci *random*. Číslo v textovém poli označeném *Zaplnění* udává procento z celkového počtu buněk, které bude nastaveno jako živé. Stiskem tlačítka *Random* se celá akce provede.

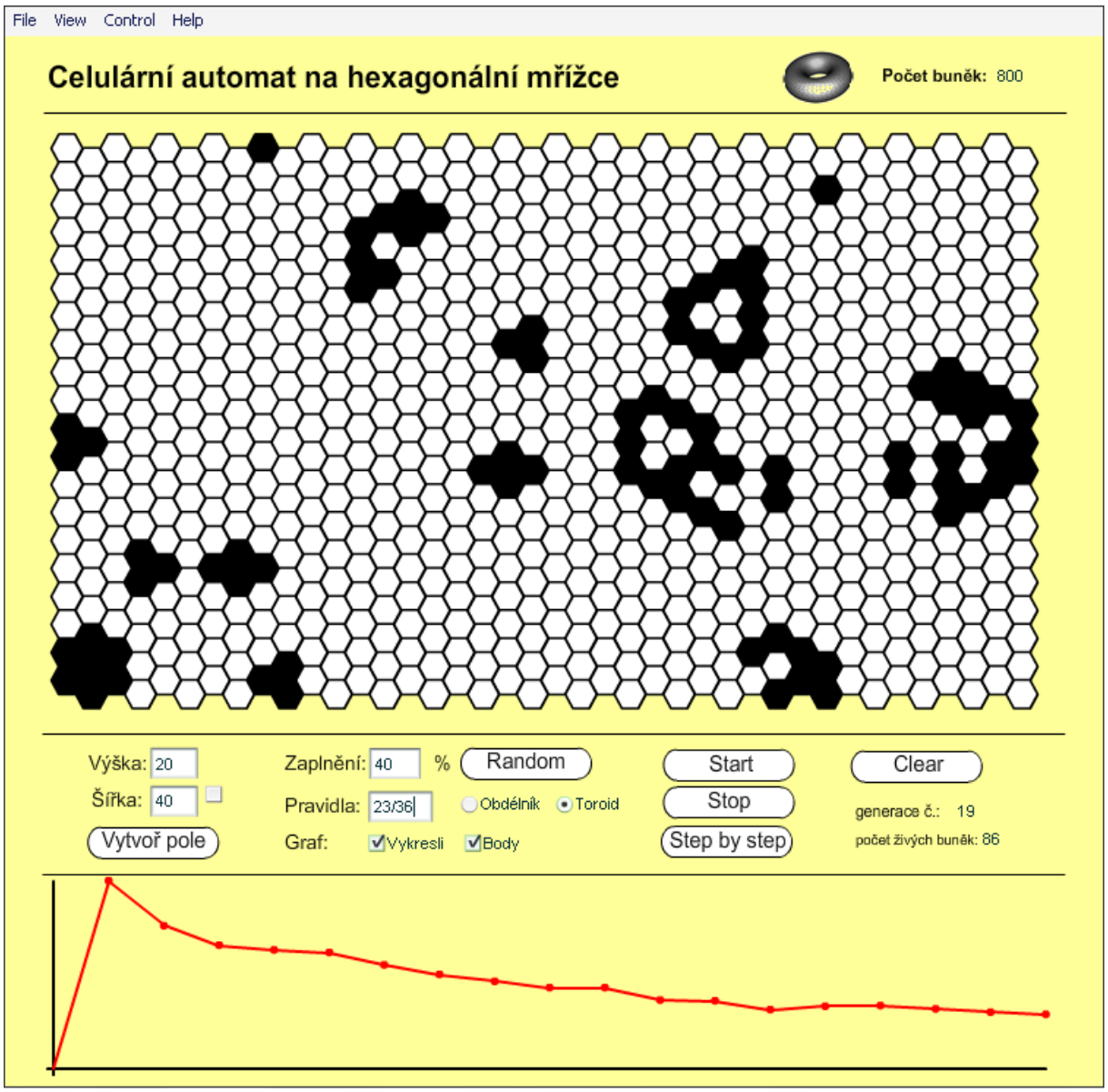
Pod volbou *random* se nachází textové pole pro zadání pravidel ve výše zmíněném tvaru. Následují dva přepínače, které určují jakým způsobem bude toto pole ohraničeno. Pokud bude zakliknuta volba *obdélník*, pak buňky na okrajích pole mají počet sousedních buněk nižší, to znamená že buňky mimo pole jsou brány vždy jako mrtvé. V případně zakliknuté volby *toroid*, pak buňky pravého okraje sousedí s buňkami na levém okraji a buňky na horním okraji sousedí s buňkami na okraji dolním. Stav této volby stejně jako počet buněk v poli je zobrazen v pravém horním rohu aplikace.



Obr. 6 Ukázka aplikace celulárního automatu na ortogonální mřížce.

Poslední sadou prvků ve druhém sloupci ovládacího panelu jsou zaškrtávací pole pro vykreslování grafu. Jedná se o graf znázorňující vývoj počtu živých buněk v závislosti na čísle generace. Pokud je zakliknuta volba *Vykresli*, pak se tento graf vykresluje v oblasti pod ovládacím panelem. V případě zvolené volby *Body* se zvýrazňují vrcholové hodnoty tohoto grafu.

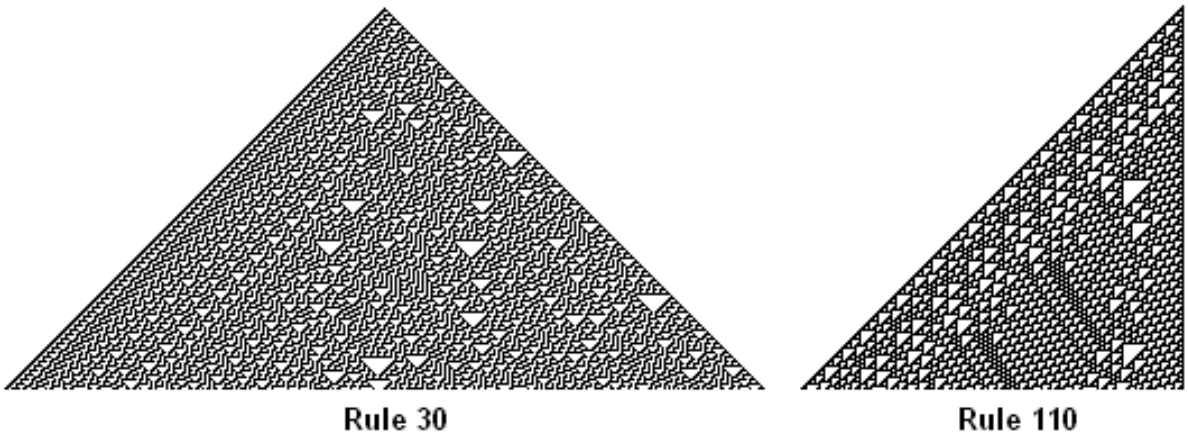
Následují ovládací tlačítka, která se nacházejí ve třetím sloupci ovládacího panelu. Prvním je tlačítko *Start*. Toto spouští automatický vývoj buněk. Tento lze pozastavit tlačítkem *Stop*. Pro krokování generace po generaci slouží tlačítko *Step by step*. Pro vrácení pole buněk do výchozího stavu, tedy kdy jsou všechny buňky mrtvé slouží tlačítko *Clear*. Pod tímto tlačítkem se nachází ještě ukazatel pořadového čísla aktuální generace a také ukazatel počtu živých buněk v této generaci.



Obr. 7 Ukázka aplikace celulárního automatu na hexagonální mřížce.

4 1D CELULÁRNÍ AUTOMAT

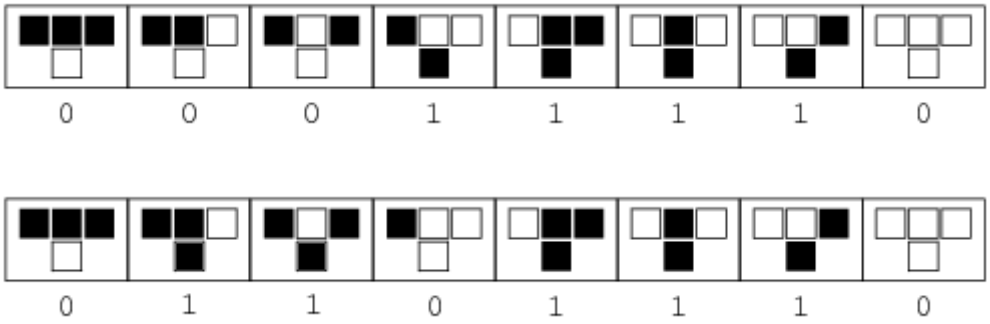
Typickými představiteli jedno-dimenzionálních celulárních automatů jsou automaty R30 (Rule 30) a R110 (Rule 110) jež byly představeny roku 1983 Stephenem Wolframem. Tyto automaty jsou hlavně zajímavé tím, že při převedení jejich grafických výsledků na posloupnosti bitů, získáme takzvaná pseudonáhodná čísla. To jsou taková čísla, která jsou statisticky nerozeznatelná od čísel náhodných, ale bylo jich dosaženo deterministickým postupem. Tedy při zachování pravidel, kterými se řídí celulární automat a stejné počáteční sekvenci bitů, pak dojde po daném počtu generací k vygenerování stejné posloupnosti bitů jako v předcházejícím případě. Díky tomuto se tyto automaty řadí do skupiny generátorů pseudonáhodných čísel. Princip těchto automatů je základem moderní kryptografie, konkrétně metod založených na veřejném klíči a také jsou tímto způsobem kryptovány elektronické podpisy.[8]



Obr. 8 Výsledné grafické obrazce 1D celulárních automatů R30 a R110.

4.1 Princip metody

Je dána čtvercová mřížka, kdy každá buňka představuje jeden bit. Pokud je živá je její hodnota 1, pokud je mrtvá má naopak hodnotu 0. Každé generaci buněk náleží jeden řádek mřížky, kdy 1. generace leží na 1. řádku. Poté se pro každou buňku v následující generaci vyhodnotí její nejbližší sousední buňky z generace předchozí a podle pravidel pro daný typ automatu se určí zda bude žít či nikoliv.



Obr. 9 Pravidla celulárního automatu R110.

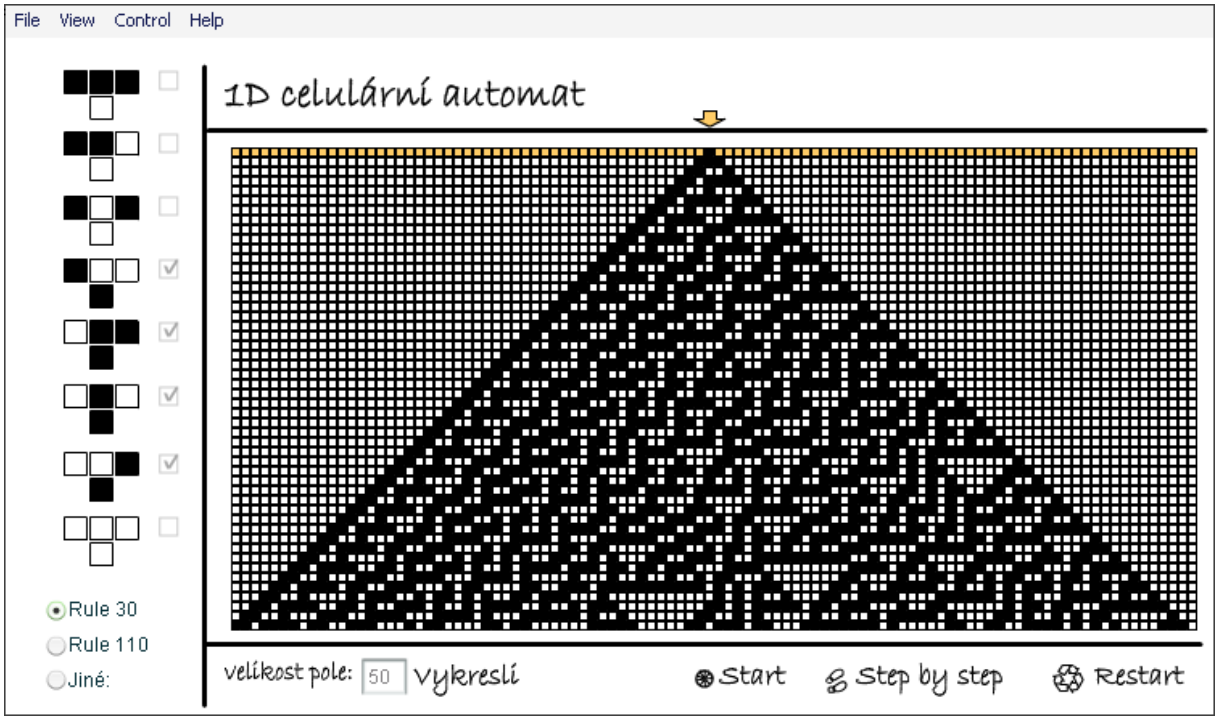
Číselné označení pravidel u jednotlivých typů jedno-dimenzionálních celulárních automatů vychází z decimálního vyjádření binárních hodnot výsledkové tabulky.

4.2 Popis aplikace

V levé části aplikace jsou umístěny ovládací prvky pro nastavování pravidel, jakými se bude celulární automat řídit. Ve spodní části levého panelu jsou umístěny tři přepínací volby, z nichž dvě jsou přednastavené hodnoty pravidel *Rule 30* a *Rule 110* a třetí umožňuje libovolné nastavení. Při označené třetí volbě je umožněno pomocí zaklikávacích polí navolit jakékoliv možné pravidlo, jehož hodnota se zobrazuje dole vedle této volby.

Ve spodním ovládacím panelu je třeba nejprve nastavit velikost pole. Tato hodnota udává počet generací (řádků) mřížky buněk. Z důvodu výpočetní náročnosti a dostatečné vypovídací hodnoty je tato hodnota omezena na rozsah 10 až 50 generací.

Po vygenerování pole, lze v prvním, zvýrazněném řádku mřížky zadat počáteční konfiguraci buněk. Toto lze učinit kliknutím na jednotlivé buňky, které se takto překlopí do opačného stavu, než ve kterém se nacházely. Šipka vyznačuje horizontální střed pole. Poté kliknutím na tlačítko *Start* dojde, dle zadaných pravidel a počáteční sekvence, k vygenerování výsledku. Tohoto lze dosáhnout i klikáním na tlačítko *Step by step*, kdy při každém kliknutí dojde k vygenerování pouze jediné, následující generace. Tlačítko *Restart* vrací aplikaci do původního stavu při zachování nastavených pravidel.



Obr.10 Ukázka aplikace 1D celulárního automatu.

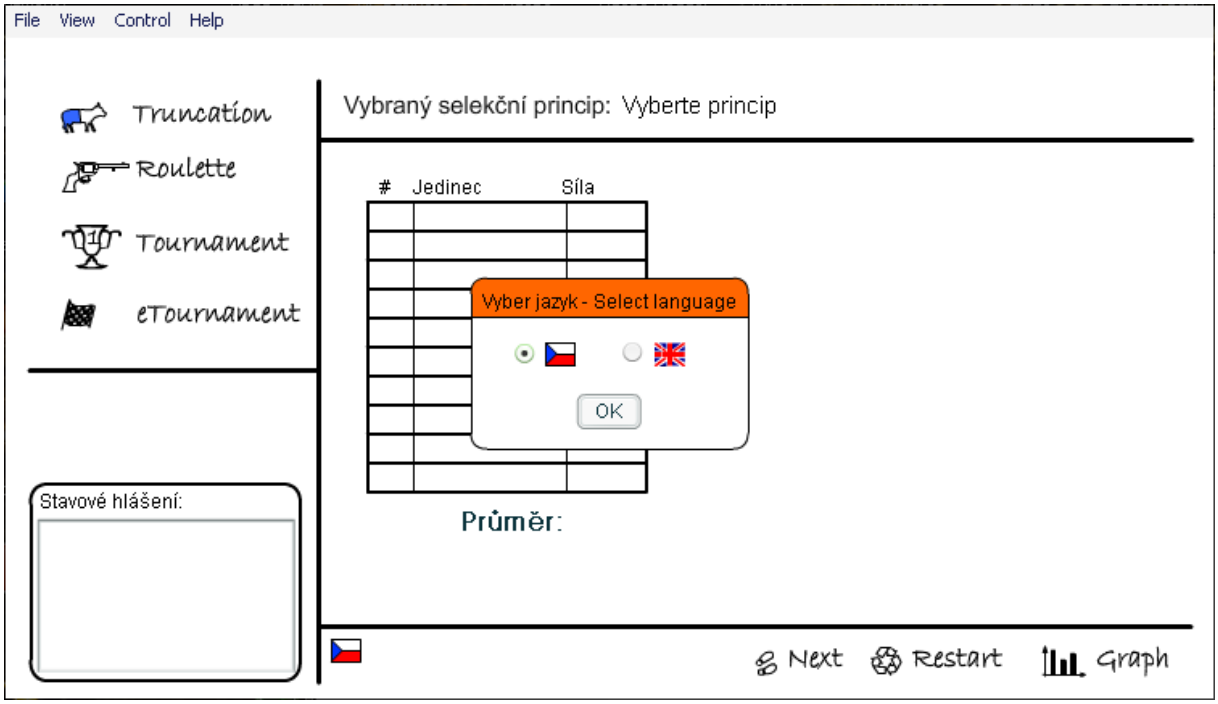
5 SELEKČNÍ PRINCIPY EVOLUČNÍCH ALGORITMŮ

Selekční principy evolučních algoritmů představují vyjádření, jakými metodami dochází ať už k přirozenému přírodnímu výběru živočichů, tak i metody, podle kterých se řídí například chovatelé dobytka a jiného zvířectva při jejich křížení a šlechtění za účelem získání nejsilnějšího a nejproduktivnějšího chovu.

Při těchto demonstracích jsou jako jedinci použity náhodně generované osmibitové posloupnosti, které spolu tvoří celou populaci deseti jedinců. Decimální vyjádření každého jedince určuje jeho sílu, která poté slouží jako měřítko pro srovnávání jedinců. Čím má síla jedince vyšší hodnotu, tím má větší šanci být vybrán do další generace populace.

5.1 Základní popis aplikace

Po spuštění aplikace je nejprve třeba zvolit jazykovou mutaci aplikace. Po zvolení vybrané volby a potvrzení tlačítkem *OK*, je třeba nejprve vybrat z nabídky v levé části obrazovky selekční princip pro následnou demonstraci. Poté se v hodní části aplikace zobrazí celý název zvoleného principu a do tabulky jedinců, uprostřed plochy aplikace, se vygenerují náhodní jedinci. V levém dolním rohu jsou zobrazovány instrukce a bližší popis právě vykonávaných činností aplikace. Pro procházení celým procesem demonstrace je v dolní části tlačítko *Next*. Tlačítko *Restart* slouží k návratu aplikace do výchozího stavu mimo dialogu pro volbu jazyka. Po stisku tlačítka *Graph* se zobrazí graf se znázorněnými četnostmi výskytu jednotlivých hodnot sil jedinců v celé populaci. Původní četnosti zobrazují rozložení sil v původní (výchozí) populaci. Nové četnosti zobrazují aktuální hodnoty po provedeném výběru.



Obr. 11 Úvodní obrazovka aplikace pro demonstraci selekčních principů.

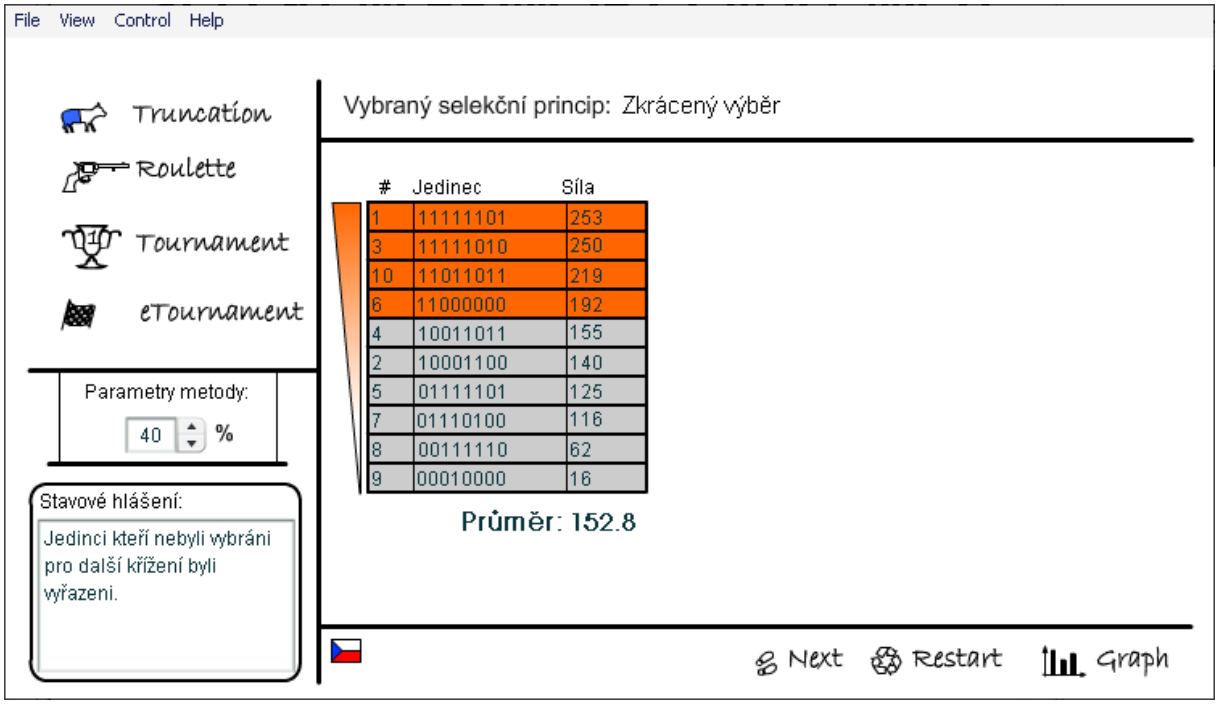
Každý selekční proces lze opakovat stále dokola v libovolném počtu generací, neboť výsledky předchozího výběru jsou zapsány jako vstupní hodnoty populace pro další výběr. Před zvolením dalšího selekčního principu, je třeba kliknout na tlačítko *Restart.*

5.2 Selekční princip – Zkrácený výběr

Tato metoda se zejména využívá u již zmíněného křížení dobytka. Jedná se o jednoduchý mechanismus, kdy je z celé populace vybráno dané procento nejlepších jedinců a toto je určeno pro pokračování v další generaci.

Po vybrání metody *Zkráceného výběru* je třeba ve výsuvném menu, pod volbami metody, zvolit počet procent jedinců, kteří budou použiti pro další generaci. Klikáním na tlačítko *Next* dojde nejprve k seřazení jedinců populace od nejsilnějšího po nejslabšího. Následně jsou vybraní jedinci označeni oranžově a ostatní šedě. Poté jsou jedinci, jež nebyli vybráni, odebráni a nahrazeni vybranými.

Pod tabulkou populace je hodnota průměrné síly populace, která se v nové generaci, oproti předchozí zvýšila.

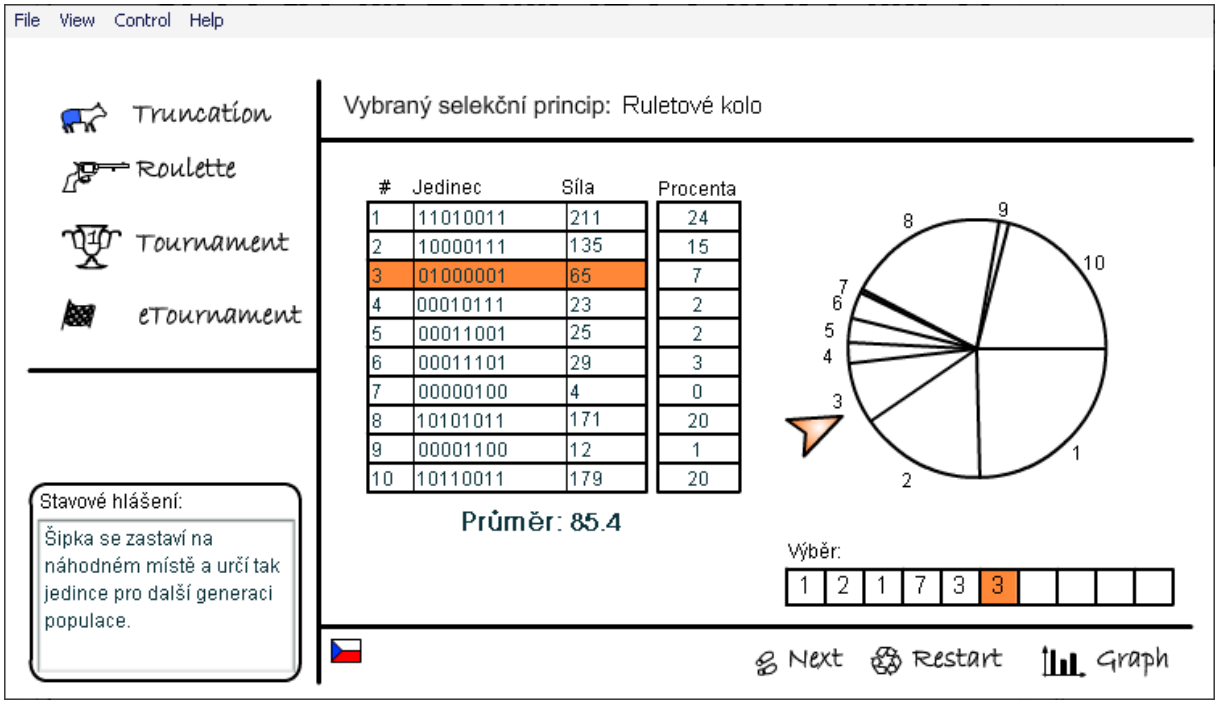


Obr. 12 Demonstrace principu zkráceného výběru.

5.3 Selekční princip – Ruletové kolo

Jádrem této metody je procentuální vyjádření síly každého jedince vzhledem k celkové síle populace, tedy součtu sil všech jednotlivých jedinců. Čím je jedinec silnější, tím má větší podíl na síle celé populace. Tyto procentuální podíly jsou vyneseny do kruhového grafu. Kolem tohoto grafu poté obíhá šipka, která se zastaví na zcela náhodném místě a tím určí vybraného jedince. Čím má jedinec větší procentuální podíl na kruhovém grafu, tím je jeho šance, že bude vybrán, větší. Díky tomu slabší jedinci z velké části odpadnou, ale není vyloučeno, že přesto budou vybráni a dostanou se do další generace.

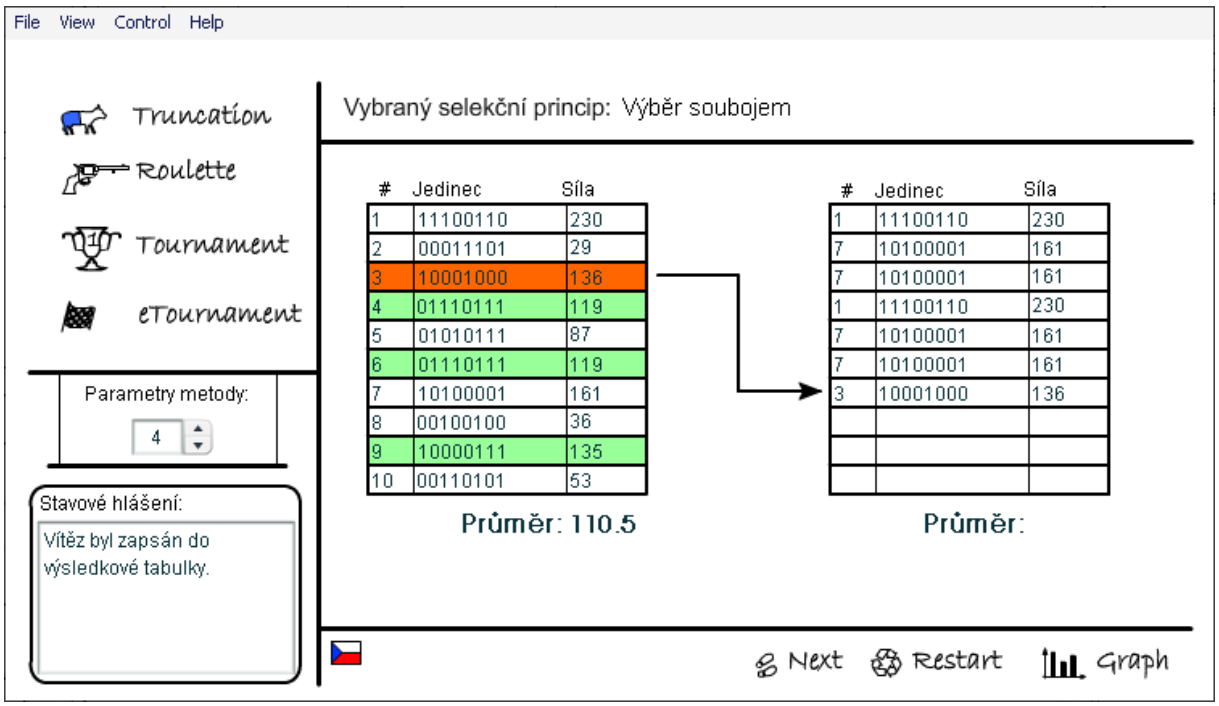
Po zvolení metody ruletového kola a vygenerování populace se zobrazí kruhový graf představující ruletu, rozdělený na výseče označené číslem jedince, jemuž daná výseč náleží. Tlačítko *Next* spouští šipku, přičemž číslo vybraného jedince je zapsáno do tabulky pod ruletou. Po vybrání deseti jedinců dojde k přepsání jedinců z výsledkové tabulky do původní tabulky jedinců v pořadí v jakém byli postupně vybráni. Pro tento postup slouží opět tlačítko *Next*.



Obr. 13 Demonstrace principu ruletového kola.

5.4 Selekční princip – Výběr soubojem

Tato metoda výběru je nejblíže principu přírodního výběru, kdy dochází k souboji mezi několika jedinci téhož druhu, přičemž ten nejsilnější vyhrává a v mnohých případech je jediný který přežívá.

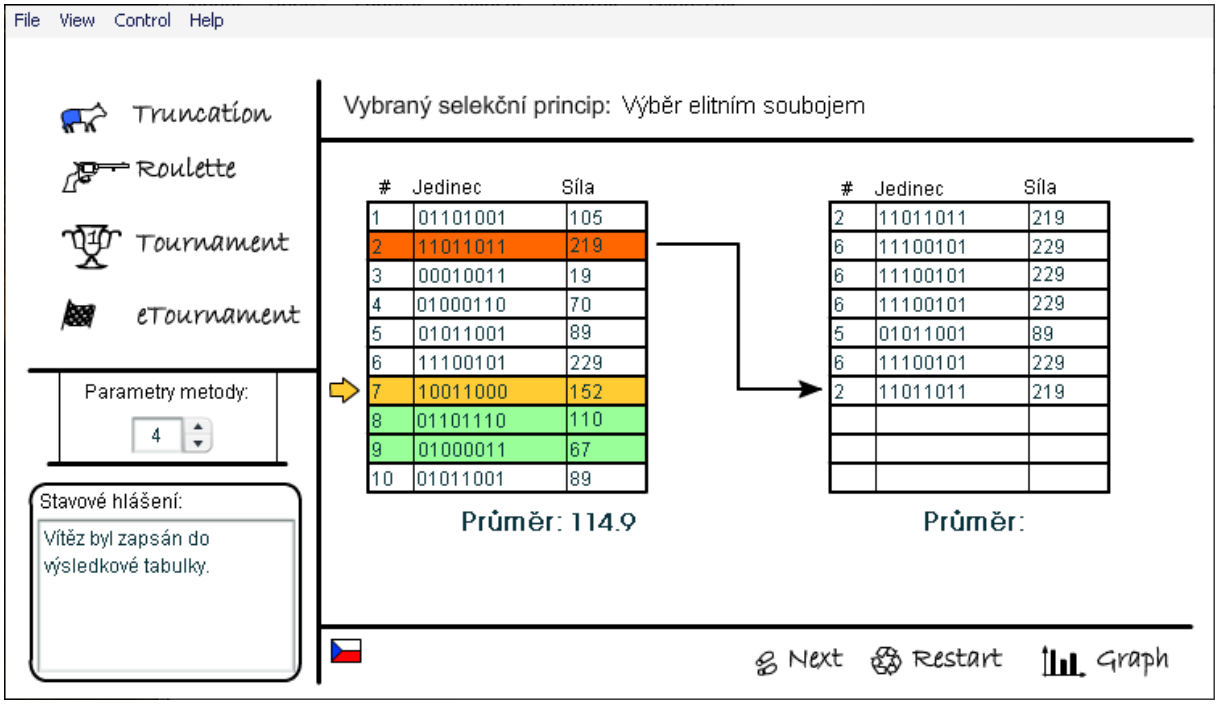


Obr. 14 Demonstrace principu výběru soubojem.

Při volbě tohoto principu je třeba ve výsuvném menu zvolit počet jedinců, kteří budou spolu soupeřit v každém kole souboje. Tito jedinci jsou pak v každém kole náhodně vybráni z celé populace a vyznačeni zeleně. Po srovnání jejich sil je ten nejsilnější, jako vítěz, označen oranžově a zkopírován do nové populace. Takto spolu soupeří náhodně vybraní jedinci tak dlouho, až je nová populace zcela zaplněna. Pro srovnání jsou pod tabulkami staré a nové populace umístěny průměrné hodnoty síly jednotlivých populací, přičemž u nové populace je tato hodnota vyšší, tudíž je nově vzniklá populace silnější než předcházející.

5.5 Selekční princip – Výběr elitním soubojem

Tato metoda je obdobou předchozí. Rovněž zde soupeří, ve výsuvném menu zadaný, počet jedinců. Na rozdíl od předcházející varianty, je zde vždy jeden jedinec vybrán postupně. To znamená, že v prvním kole bude vybrán první jedinec a zbytek jedinců do počtu je vybrán náhodně. V druhém kole bude vybrán druhý, zbytek náhodně. Tak tento systém postupuje celou populací. Tento postup zaručuje, že se do souboje dostanou úplně všichni jedinci. U předchozí varianty mohlo dojít k případu, kdy při náhodných výběrech nebyl vybrán ten nejsilnější jedinec a tak následující populace by byla o tohoto jedince ochuzena.[11]



Obr. 15 Demonstrace principu výběru soubojem.

Systematicky vybíraný jedinec má vždy označení žlutým podbarvením a šipkou na jeho úrovni v tabulce. Ostatní vybraní jedinci jsou podbarveni zeleně a vítězný jedinec je vyznačen oranžově a zkopírován do nové populace. Rovněž jako u předchozí metody slouží pro porovnání hodnoty průměrů obou populací.

6 2D TURINGŮV STROJ – LANGTONOVA DEMONSTRACE

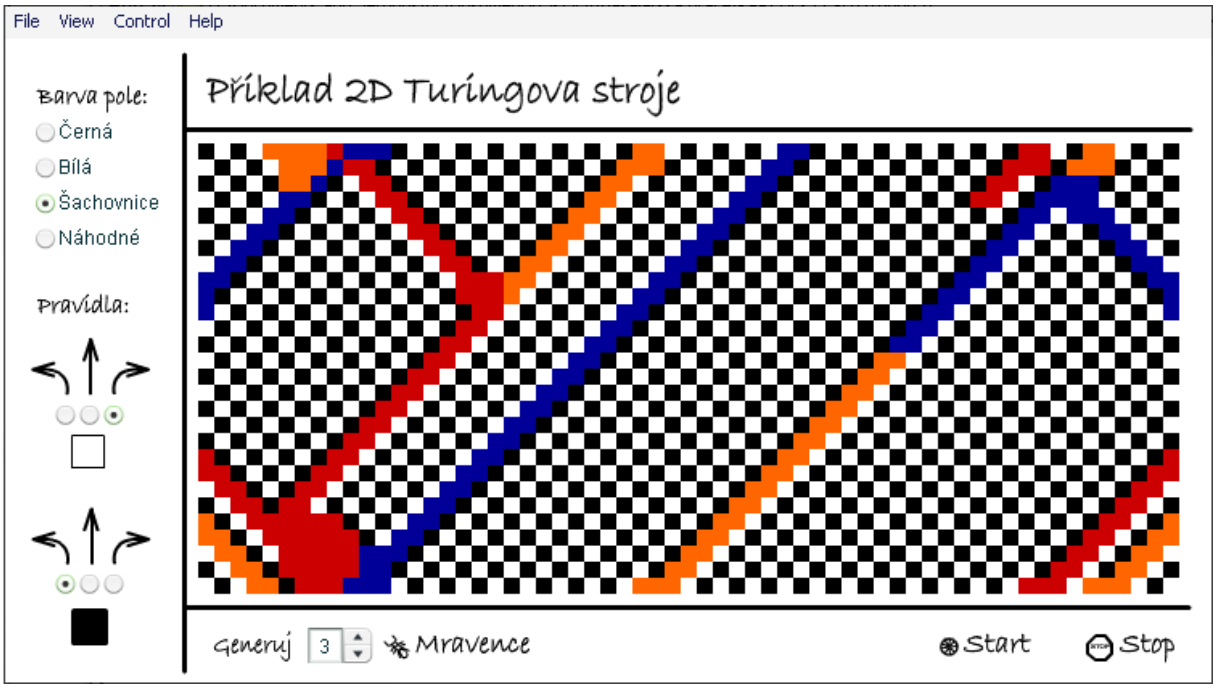
Dvou-dimenzionální Turingův stroj v provedení z roku 1986 podle amerického biologa Christophera Langtona se vyznačuje aplikací jednoduchých pravidel, jež mají za následek komplikované chování.[4] Jedná se o jednu možnou variantu zobrazení 2D Turingova stroje.

6.1 Princip metody

Je dáno čtvercové pole, kdy každé pole může mít bílou, nebo černou barvu. Na náhodně vybrané pole je umístěn mravenec, který se může pohybovat v kterýmkoliv ze čtyř základních směrů. Pohyb mravence se řídí pravidly danými pro barvu pole na němž se právě nachází. Variantou pravidel může být například, pokud stojí na bílém poli, pak se otočí o 90° doleva, posune se o jedno pole vpřed a pole na němž stál změní svou barvu na černou. Obdobně pokud stojí na černém poli, tak se otočí o 90° doprava, posune se v před a pole kde stál změní svou barvu na bílou.[4]

6.2 Popis aplikace

V levé části okna aplikace se nacházejí ovládací prvky pro nastavení výchozí podoby pole se čtyřmi možnostmi volby a pod nimi možnost nastavení pravidel pro pohyb mravenců v tomto poli. Pro nastavení výchozí podoby pole jsou přednastaveny čtyři základní volby, přičemž první z nich je zvolena jako základní po spuštění aplikace. Pokud je zvolena barva pole černá, jsou všechny políčka čtvercové plochy na počátku vybarvena černě. Obdobně u volby – bílá, jsou všechna pole bílá. Další volbou je šachovnice a nakonec náhodné, což znamená, že jednotlivá pole čtvercové plochy budou náhodně vybarvena bíle, nebo černě. V sekci pravidla lze nastavit chování mravence a to tak, že určíme zda na bílém poli půjde mravenec doleva, rovně, nebo doprava. Stejné nastavení je u černého pole.



Obr. 16 Ukázka aplikace Příklad 2D Turingova stroje.

Ve spodní části aplikace je umístěn ovládací prvek pro umístění jednoho až tří mravenců na náhodná místa na ploše. Toto provedeme navolením požadovaného počtu mravenců a následným kliknutím na tlačítko *Mravence*. Pro spuštění a zastavení procesu slouží tlačítka *Start* a *Stop*. Pro návrat plochy do původního stavu stačí zvolit jiné barevné schéma plochy.

Po spuštění za sebou každý mravenec zanechává stopu své barvy, která zobrazuje jeho pohyb po ploše. Mravenci si navzájem stopy přebarvují, takže pokud jeden mravenec přejde po stopě druhého, pak u druhého již nelze přesně zjistit, kde se všude nacházel.

Pod stopami mravenců však zůstává zachováno bílo – černé schéma základní plochy, tedy byť bylo pole již navštíveno a stopou mravence přebarveno, pokud na něj vstoupí další mravenec, tak se řídí dle bílé, nebo černé barvy, kterou tam zanechal jeho předchůdce.

7 MRAVENČÍ KOLONIE

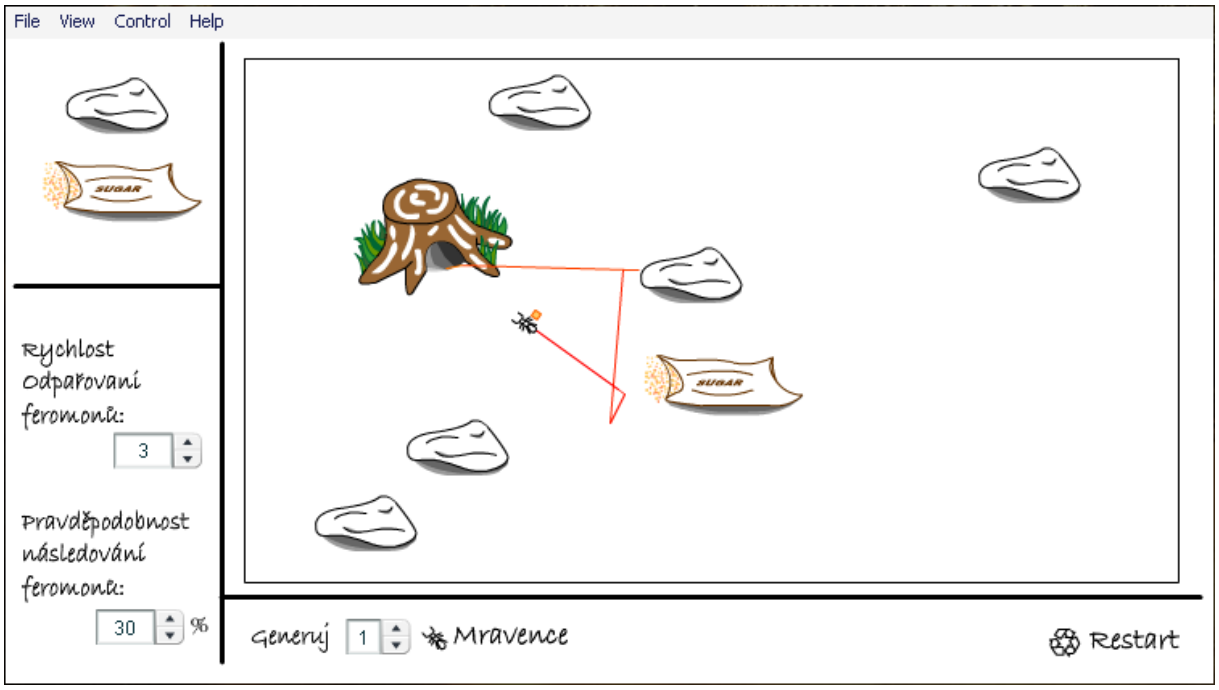
Jedná se o přibližnou simulaci chování reálných biologických organismů, konkrétně se jedná o kolonii mravenců. Tento příklad zobrazuje variantu takzvané rojové inteligence. Systémy rojové inteligence se skládají z jednotlivých agentů, kteří interagují mezi sebou a okolním prostředím. Prostředkem komunikace mezi agenty (mravenci) v mravenčí kolonii je zanechávaná feromonová stopa. I když tyto systémy nemají žádnou centrální kontrolu jednotlivých agentů, tak lokální interakce a jednoduché vzory chování agentů vedou k vytvoření globálního chování.[9] Simulačních modelů mravenčích kolonií se s výhodou využívá pro řešení různě komplikovaných optimalizačních úloh.

7.1 Mravenčí kolonie

Úkolem mravenců je v okruhu kolem mraveniště vyhledat zdroj, či zdroje potravy a tuto potravu přinést zpět do mraveniště. Každý mravenec za sebou zanechává pachovou stopu (feromony) která dává vědět ostatním členům kolonie, že tudy už některý z nich šel. Je dána pravděpodobnost s jakou se mravenci vydají či nevydají po nalezené feromonové stopě, která je dána také intenzitou této stopy. Každá pachová stopa mravence postupně vyprchává a tak pokud na ni mravenec narazí jako na téměř vyprchanou, pak se po ní nemusí vůbec vydat, byť je přednastavena vysoká pravděpodobnost následování stop. Čím více mravenců po dané stopě půjde, tím více se bude udržovat intenzivní. Mravenci se mohou na cestě setkat i s překážkami, kterým se musejí vyhýbat, což komplikuje jejich pohyb v prostředí.

7.2 Popis aplikace

V levé horní části aplikace se nachází dva objekty pro umisťování na plochu pohybu mravenců. Prvním je kámen, který představuje překážku, které se musí mravenci vyhnout a druhý objekt je vlastní zdroj potravy, v tomto případě jde o cukr. Pro jejich umístění na plochu stačí na kterýkoliv z nich kliknout myší a poté kliknout na vybranou polohu pro jejich umístění. Na ploše se již nachází umístěné mraveniště v podobě pařezu.



Obr. 17 Ukázka aplikace Mravenčí kolonie

Pod úvodními objekty v levém panelu se nacházejí dva ovládací prvky. První určuje rychlost odpařování feromonu ve stupnici od 1 do 10, kdy při hodnotě 10 se feromon odpařuje nejrychleji. Druhý ovládací prvek udává pravděpodobnost následování feromonu. To znamená, že pokud mravenec narazí na feromon, tak podle jeho intenzity a této pravděpodobnosti zvolí, zda se po něm vydá, či nikoliv. Čím je hodnota pravděpodobnosti vyšší, tím je větší možnost, že se po této stopě mravenec vydá.

Ve spodním ovládacím panelu lze navolit v rozsahu od 1 do 10, kolik mravenců se bude současně pohybovat po ploše. Kliknutím na tlačítko *Mravence,* dojde ke spuštění procesu. Pro návrat aplikace do výchozího stavu slouží tlačítko *Restart*.

8 ZÁVĚR

Praktickým výsledkem této práce je celkem šest interaktivních simulacích, které byly vytvořeny pro výukové účely. Problematika realizovaných simulací byla volena dle požadavků vedoucího této práce, přičemž se stane součástí širší internetové prezentace. Jednotlivé aplikace se liší svou komplikovaností na vypracování i časovou náročností realizace. Úvodní simulací byly dvě varianty Conwayova celulárního automatu, které demonstrují zjednodušený model chování jednoduchých buněčných organismů. Dalším je jednodimenzionální celulární automat graficky znázorňující generování pseudonáhodných čísel. Následovala aplikace demonstrující vybrané selekční principy evolučních algoritmů, tedy principů jimiž se řídí ať už přírodní výběr, či cílený výběr při šlechtění zvířectva. Předposlední je ukázka možné demonstrace 2D Turingova stroje v podobě Langtonova mravence a nakonec aplikace zobrazující přibližné chování mravenčí kolonie. Časově nejnáročnější byly Conwayovy celulární automaty, neboť byly programovány jako první a tudíž vzhledem k faktu, že až do této práce byla tato technologie promne téměř neznámá, zabraly nejvíce času.

Velmi poutavá a pro mne přínosná byla témata realizovaných simulací. Tato pro mne byla rozšířením studijních znalostí získaných během bakalářského studia. Realizace daných aplikací dále prokázala možnost využití technologie Flash, pro tvorbu jednoduchých interaktivních simulací, či demonstračních příkladů minimálně v ohledu prezentované tématiky. Nutno říci, že i přes zmíněné výhody je technologie Flash určena jiným směrem než profesionální simulační softwary typu Matlab/Simulink apod.

V teoretické části této práce byl pro každou řešenou problematiku vytvořen krátký úvod do dané problematiky obsahující základní fakta a informace o tom co je v aplikacích demonstrováno a také podrobný manuál s popisy funkcí a ovládacích prvků jednotlivých aplikací.

Velkým osobním přínosem bylo pro mne získání zkušeností v tvorbě aplikací novým způsobem, který byť přes svou rozšířenost není součástí výukových osnov mnou studovaného oboru. Naučil jsem se nový programovací jazyk spolu se zvládnutím grafických, animačních a dalších záležitostí úzce souvisejících s vývojem aplikací v prostředí Flash. Přínos shledávám zejména v tom, že tato technologie v současnosti získává stále více na popularitě a nachází stále širší uplatnění a v budoucnu by se mohla stát i mým hlavním oborem zájmu.

9 SEZNAM POUŽITÉ LITERATURY

[1] WIKIPEDIA. The free encyclopedia [online]. Datum poslední revize 16.5.2008 [cit. 2008-04-14]. <http://en.wikipedia.org/wiki/Adobe_flash>

[2] WIKIPEDIA. The free encyclopedia [online]. Datum poslední revize 16.5.2008 [cit. 2008-05-14]. <http://en.wikipedia.org/wiki/ActionScript>

[3] POLZER Jan. Nová konkurence Flashi? *Computer*, 2007, roč. 14, č. 13, ISSN 1214-1887

[4] WIKIPEDIA. The free encyclopedia [online]. Datum poslední revize 6.4.2008 [cit. 2008-04-27]. <http://en.wikipedia.org/wiki/Langton's_ant>

[5] CALLAHAN Paul. Wonders of math [online]. Datum poslední revize 12.1.2006 [cit. 2008-03-18]. <http://www.math.com/students/wonders/life/life.html>

[6] Mushroom Life[online] <http://a.parsons.edu/~joseph/k2/gameoflife/> [cit. 2008-04-05]

[7] 3D Game of Life[online]. Datum poslední revize 8.12.2000 [cit. 2008-03-18] <http://www.people.nnov.ru/fractal/Life/Game.htm>

[8] W. WEISSTEIN Eric . MathWorld[online]. Datum poslední revize 16.5.2008 [cit. 2008-04-05] <http://mathworld.wolfram.com/>

[9] NĚMEC Miloš. Rojová inteligence[online].18.2.2006 [cit 2008-19-05]. <http://www.milosnemec.cz/clanek.php?67>

[10] Adobe[online].[cit. 2008-04-15]<http://www.adobe.com/>

[11] Matoušek, R.: Vybrané metody umělé inteligence – implementace a aplikace. *Disertační práce v oboru Technická kybernetika.* VUT v Brně, 2004.

[12] Mařík, V., Lažanský, J.a kol.: Umělá inteligence (3). *Academia*, 2003, ISBN 80-200-0502-1

SEZNAM PŘÍLOH

- Příloha - Implementační detaily

I. CONWAYŮV CELULÁRNÍ AUTOMAT

```
function LiveOrDead (PocetZivychBunek:Number, CisloBunky:Number){
   var Pravidlo:String  = _root.Parametry_tin.text;
   for (i = 0; i <=length(Pravidlo); i++){
      var IndexOfLomitko: Number = Pravidlo.indexOf("/",0);
      if (i!=IndexOfLomitko){
        var TestNum: Number = Number(Pravidlo.substr(i,1));
        if (PocetZivychBunek == TestNum){
          if (i < IndexOfLomitko && StavovaTabulka[CisloBunky] == true){
              return (true);
          }
          if (i > IndexOfLomitko && StavovaTabulka[CisloBunky] == false){
             return (true);
          }
        }else if (i == length(Pravidlo)){
          return (false);
        }
      }
   }
}
```

Cyklus pro vyhodnocení počtu živých okolních buněk vzhledem k pravidlům

Návratovou hodnotu funkce je, zda bude buňka v následující generaci živá či nikoliv

Pro popis byla vybrána funkce vyhodnocující stav dané buňky v následující generaci. Vstupními parametry funkce je počet živých okolních buněk a číslo buňky, kterou právě tato funkce řeší. Do proměnné *Pravidlo* je načtena hodnota textového pole *Pravidlo* na ovládacím panelu. U textového řetězce v této proměnné se zjišťuje pozice lomítka, která se při procházení tohoto textového řetězce přeskakuje. Do proměnné *TestNum* se uloží vždy jedna cifra z celého pravidla a tato se poté porovnává s počtem živých okolních buněk. Pokud dojde ke shodě, tak se určí zda se daná cifra nachází před, nebo za lomítkem a jaký byl předchozí stav dané buňky. Návratovými hodnotami této funkce je *true*, nebo *false*, což znamená zda buňka bude v následující generaci žít či nikoliv.

II. 1D CELULÁRNÍ AUTOMAT

```
function UrciNasledovniky() {
    for (var i:Number = 1; i<=SirkaPole; i++) {
        var TestovaciString:String = "";
        for (var j:Number = -1; j<2; j++) {
            if (PolePredchudcuANasledovniku[i-1+j][0] != Undefined) {
                TestovaciString += PolePredchudcuANasledovniku[i-1+j][0];
            } else {
                TestovaciString += "0";
            }
        }
        for (var j:Number = 0; j<8; j++) {
            if (TestovaciString == TabulkaKlicu[j][0]) {
                PolePredchudcuANasledovniku[i-1][1] = TabulkaKlicu[j][1];
            }
        }
    }
}
```

Podmínka pro úspěšné vyhodnocení okrajvých jedinců.

Cyklus pro porovnání proměnné *TestovaciString* s polem obsahující vzory s výslednými hodnotami 1 nebo 0

Základní datová struktura obsahující informace o předchozí generaci, do níž se zaznamenává generace nová

Tato funkce má za úkol určit následující generaci buněk v závislosti na rozložení buněk v předchozí generaci a daných pravidlech platných pro tuto transformaci. Základem této funkce je cyklus, který prochází v řádku nové generace jednu buňku po druhé. V dalším, vnořeném cyklu, jsou pro každou buňku v nové generaci zjišťovány stavy nejbližších buněk v generaci předchozí a výsledky se ukládají do proměnné *TestovaciString*. Tato proměnná je v dalším cyklu porovnávána s tabulkou klíčů, jež byla vytvořena po spuštění aplikace, která obsahuje údaje o pravidlech pro jednotlivé kombinace buněk. Při shodě je do další generace na pozici zkoumané buňky zapsána hodnota dle tabulky klíčů.

III. SELEKČNÍ PRINCIPY EVOLUČNÍCH ALGORITMŮ

III.I ZKRÁCENÝ VÝBĚR

Počet procent jedinců, kteří budou vybráni pro další generaci

```
function Strihej() {
     var TruncationProcenta:Number =
this.ZasuvkaTruncation.ZasuvkaTruncation_koren.ProcentaTruncation_nst.va
lue/10;

      for (var i:Number =TruncationProcenta; i<10; i++) {
            PoleJedincu[i][0] = "";
            PoleJedincu[i][1] = "";
            PoleJedincu[i][2] = "";
            _root["TabulkaLine" + i].RadekTabulky_vypln.gotoAndStop(4);
      }
      for (i = 0; i<TruncationProcenta; i++){
            _root["TabulkaLine" + i].RadekTabulky_vypln.gotoAndStop(2);
      }
}
```

Cyklus pro vymazání nevybraných jedinců z *PoleJedincu*.

Cyklus pro zvýraznění vybraných jedinců

Funkce *Strihej* slouží k oddělení vybraných jedinců od ostatních, jež nejsou určeni pro další populaci. Nejprve je do proměnné *TrucationProcenta* zapsána hodnota z výsuvného menu určující kolik procent jedinců bude použito do další populace. Neboť je jedinců pouze 10, je tato hodnota vydělena desíti. V následném cyklu dojde k vymazání hodnot jedinců, kteří nebyli vybráni a zároveň se na obrazovce zobrazí šedě podbarvení. V druhém cyklu této funkce dojde k oranžovému zvýraznění vybraných jedinců.

III.II VÝBĚR SOUBOJEM A ELITNÍM SOUBOJEM

```
function Souboj(){
      var TestValue:Number = 0;
      for (var i:Number = 0; i<10; i++) {
            var Cislo:Number = PoleNahodnyVyber[i];
            if (Cislo == -1 | i == 9) {
                  NovePoleJedincu[ii][0] = PoleJedincu[CisloViteze][0];
                  NovePoleJedincu[ii][1] = PoleJedincu[CisloViteze][1];
                  NovePoleJedincu[ii][2] = PoleJedincu[CisloViteze][2];
                  NovePoleJedincu[ii][3] = PoleJedincu[CisloViteze][3];
                  ii++;
                  return (CisloViteze);
            } else {
                  if (TestValue<PoleJedincu[Cislo][1]) {
                        TestValue = PoleJedincu[Cislo][1];
                        CisloViteze = Cislo;
                  }
            }
      }
}
```

Pole náhodně vybraných jedinců

Pole všech jedinců

Pole vítězných jedinců

Tato funkce slouží u aplikace simulující selekční principy pro vyhodnocení nejsilnějšího jedince při výběru soubojem a výběru elitním soubojem. Prochází pole náhodně vybraných jedinců ze kterých porovnáváním určí toho nejsilnějšího. V první podmínce dochází ke zjišťování, zda již bylo dosaženo konce pole náhodně vybraných jedinců. Pokud ne, pak je porovnána proměnná *TestValue* která v sobě uchovává aktuálně nejsilnějšího jedince s jedincem následujícím. Pokud je následující jedinec silnější než předchozí, je původní nejsilnější jedinec nahrazen novým a do proměnné *CisloViteze* je uložena hodnota pořadí vítěze v poli *PoleJedincu.* Při dosažení posledního záznamu je do *NovehoPoleJedincu* na pozici odpovídající pořadí souboje *ii* zapsán vítězný jedinec.

III.III RULETOVÉ KOLO

```
function UrciPolohuSipky(Cx:Number, Cy:Number) {
      var PrubeznySoucet:Number = 0;
      var ProcentaUhlu:Number = (Math.acos(Cx/112)*100)/(2*Math.PI);
      if (Cy<=0) {
            ProcentaUhlu = 100-ProcentaUhlu;
      }
      for (var i:Number = 0; i<10; i++) {
            PrubeznySoucet += _global.StarePoleJedincu[i][3];
            if (PrubeznySoucet>ProcentaUhlu) {
               _root.OblastRulety.TabulkaVysledkuRulety
["BunkaTabukyVysledku"+PoradiVysledku].text = PoleJedincu[i][2];
               VysledkyRulety[PoradiVysledku] = i;
               _root["TabulkaLine" + i].RadekTabulky_vypln.gotoAndPlay(5);
               _root.OblastRulety.TabulkaVysledkuRulety
["TabulkaVysledkuRulety_vypln" + (PoradiVysledku+1)].gotoAndPlay(2);
               PoradiVysledku++;
               return (PoleJedincu[i][2]);
            }
      }
}
```

Původní pole jedinců

Pole čísel vylosovaných jedinců

Návratová hodnta – číslo vybraného jedince

Úkolem této funkce je určit, dle vstupních parametrů *Cx* a *Cy* , který jedinec byl vybrán. Vstupní parametry udávají polohu šipky na ploše vzhledem ke středu rulety, která tvoří střed souřadnicového systému. V úvodu funkce jsou souřadnice šipky přepočteny na procentuální hodnotu úhlu oproti výchozí pozici. Poté jsou v cyklu sčítány procentuální podíly sil po sobě jdoucích jedinců. Pokud hodnota takovéhoto průběžného součtu přesáhne hodnotu uloženou v proměnné *ProcentaUhlu,* pak jedinec, který předcházel tomu jehož přičtením došlo k překročení hodnoty *ProcentaUhlu*, je ten který byl vybrán.

IV. 2D TURINGŮV STROJ

```
_root.onEnterFrame = function() {
    if (Zastav == false) {
        for (var i:Number = 0; i<PocetMravencu; i++) {
            var Xpozice:Number = _global.Mravenci[i].KdeJeX;
            var Ypozice:Number = _global.Mravenci[i].KdeJeY;
            var BarvaAktualnihoPole:Number =
                                    StavovaTabulka[Xpozice][Ypozice][1];

            _global.Mravenci[i].DalsiKrokMravence(BarvaAktualnihoPole);
            var _Xpozice:Number = _global.Mravenci[i].KamJdeX;
            var _Ypozice:Number = _global.Mravenci[i].KamJdeY;
            StavovaTabulka[_Xpozice][_Ypozice][0] =
                                                _global.Mravenci[i].Barva;

            _global.Mravenci[i].PripravaProDalsiKrokMravence();
            var _Xpozice:Number = _global.Mravenci[i].KdeBylX;
            var _Ypozice:Number = _global.Mravenci[i].KdeBylY;
            if (StavovaTabulka[_Xpozice][_Ypozice][1] == 1) {
                StavovaTabulka[_Xpozice][_Ypozice][1] = 2;
            } else {
                StavovaTabulka[_Xpozice][_Ypozice][1] = 1;
            }
        }
        VykresliStavovouTabulku();
    }
};
```

Barva pole na němž se mravenec právě nachází.

Pole instancí objektu *Mravenec*.

Pole stavů jednotlivých polí čtvercové mřížky.

Podmínka pro převrácení barvy pole.

Touto funkcí je realizováno vlastní jádro celé aplikace. Při události *onEnterFrame* na hlavní časové ose dojde, při splnění úvodní podmínky sloužící pro pozastavení běhu aplikace, k vyhodnocení a provedení následujícího kroku. V hlavním cyklu se určuje pro každého mravence barva pole na níž se nachází. Poté zavoláním metody *DalsiKrokMravence* objektu *Mravenec* dojde k vyhodnocení zadaných pravidel a předchozí pozice mravence a výsledkem jsou parametry *KamJdeX* a *KamJdeY*, které udávají na jaké pole bude postupovat. Metodou *PripravaProDalsiKrokMravence* se vlastnosti *KdeJeX/Y* přepíší do *KdeBylX/Y* a vlastnosti *KamJdeX/Y se* přepíší do *KdeJeX/Y*. Na závěr se převrátí hodnoty stavové tabulky z bílé na černou, nebo opačně. Poté je stavová tabulka vykreslena.

V. MRAVENČÍ KOLONIE

DelkaTrasy udává vzdálenost mezi polohou mravence a cílovým bodem

```
public function UrciPosuvy(){
    DelkaTrasy = Math.sqrt((NovaPoziceX-AntX)*(NovaPoziceX-
AntX)+(NovaPoziceY-AntY)*(NovaPoziceY-AntY));

    AntSpeedX = ((Math.abs(NovaPoziceX-AntX))*AntSpeed)/DelkaTrasy;
    AntSpeedY = Math.sqrt(AntSpeed*AntSpeed-AntSpeedX*AntSpeedX);
}
```

Vybraná funkce je součástí třídy *Ant*, jehož objekty jsou vlastní mravenci. Výstupem této funkce jsou proměnné *AntSpeedX* a *AntSpeedY*. Tyto udávají velikost posunu mravence v souřadných směrech v každém kroku aplikace. Nejprve je však určena *DelkaTrasy*, která udává vzdálenost bodu, kde se aktuálně nachází mravenec, tedy *AntX* a *AntY* a nového cílového bodu, tedy *NovaPoziceX* a *NovaPoziceY*. Rychlost posuvu mravence *AntSpeed* je pevně dána, platí tedy předpoklad, že se mravenci pohybují stále stejnou rychlostí.