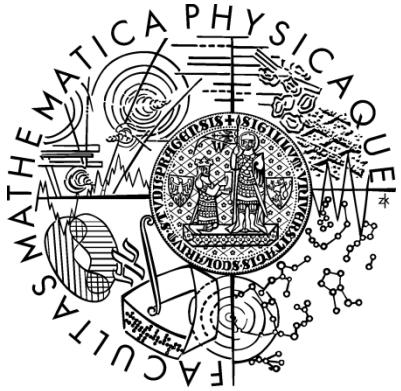
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Jiří Meluzín

Simulace ekonomiky

Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce: Mgr. Ondřej Zajíček

Studijní program: Informatika, programování

2008

Děkuji panu Mgr. Ondřeji Zajíčkovi za odborné vedení mé práce, za rady a za čas, který mi během vypracovávání této práce věnoval.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 29. května 2008 Jiří Meluzín

Obsah

Název práce: Simulace ekonomiky
Autor: Jiří Meluzín
Katedra: Katedra teoretické informatiky a matematické logiky
Vedoucí bakalářské práce: Mgr. Ondřej Zajíček
E-mail vedoucího: Ondrej.Zajicek@mff.cuni.cz
Abstrakt:

Návrh a implementace ekonomické simulace společnosti (na úrovni státu). Cílem je simulovat toky komodit a peněz, vývoj nabídky a poptávky na jednotlivých trzích a rozvoj měst. Důraz není kladen na přesnost simulace, ale na její zábavné a vzdělávací aspekty. Simulace by měla obsahovat grafické interaktivní prostředí, které umožní ovlivňovat model formou hry.

Klíčová slova: diskrétní simulace, ekonomika

Title: Economic Simulation
Author: Jiří Meluzín
Department: Department of Theoretical Computer Science and Mathematical Logic
Supervisor: Mgr. Ondřej Zajíček
Supervisor's e-mail address: Ondrej.Zajicek@mff.cuni.cz
Abstract:

Design and implementation of economy society simulation (at country level). The aim is to simulate comodities and financy flow, supply and demand progress at each market and evolution of the cities. Exactnest of the simulation is not important, but the entertainment and the education aspects. The simulation should contains interactive graphic enviroment, which provides way to influence simulation model.

Keywords: discrete simulation, economy

1 Úvod

Bakalářská práce si klade za cíl vytvořit program simulující toky komodit a peněz, vývoj nabídky a poptávky na jednotlivých trzích a rozvoj měst. Inspirací jsou dnešní strategické hry, zejména budovatelské, ve kterých se hráči předloží určité nástroje pro zásahy do stavu hry a jeho cílem je použít je tak, aby dosáhl požadovaných cílů – spokojenosti obyvatel města, ekonomické prosperity svého města atd. Použití jednotlivých nástrojů bývá omezené zdroji hráče – např. pro stavbu budovy potřebuje několik jednotek stavebního materiálu. Aby bylo poté možné tyto zásahy promítnout do vývoje města, je potřeba nějakým způsobem simulovat jeho chod.

Chodem města se myslí proces přecházení z jednoho stavu do druhého. Takový proces je možné nazvat diskrétní simulací. Průběh výpočtu není většinou příliš složitý, spíše bývá pracné nastavit správně výchozí stav a jeho možné změny.

Avšak tato práce si neklade za cíl reálnost simulace vývoje ekonomiky. Důraz je kladen především na její interaktivitu s uživatelem. Simulační model není v aplikaci napevno, lze jej případně doplnit jiným algoritmem. Současný model se snaží napodobit „neviditelnou ruku trhu“. Tedy zjistí poptávku a nabídku účastníků simulace a pokouší se ji spárovat tak, aby to pro obě strany obchodu bylo výhodné.

Text bakalářská práce je rozdělen do několika kapitol. Úvod je následován seznámením s návrhem samotné simulace, tedy její struktury a právě zmiňované přechody mezi jednotlivými jejími stavy. V následující části se probírá platforma, na které aplikace poběží a její omezení. Čtvrtá kapitola pojednává o struktuře samotné aplikace – hlavně z pohledu programátora. Dále jsou v ní vyjmenovány a popsány implementační problémy a jejich řešení. Následují základní prvku grafického rozhraní. Čtvrtá kapitola obsahuje popisy vybraných podobných aplikací.

Poslední část práce obsahuje shrnutí, použité zdroje a soubory výsledné aplikace.

2 Návrh simulace

K tomu, aby bylo možné ekonomiku simulovat, je potřeba nejdříve definovat, co a jak program bude vykonávat. Nyní budou popsány základní subjekty a operace nad nimi, které může simulace vykonávat.

2.1 Co to ekonomika je

Na prvním místě je potřeba definovat základní pojmy. Jeden z nich je již v samotném názvu práce – Ekonomika.

Denici lze podle WikiPedia.org [1] popsat takto: *Ekonomika (hospodářství) je shrnutí hospodaření určitého subjektu například státu, organizace nebo jednotlivce*.

Aplikace se omezí na hospodaření státu. Státem se v simulaci rozumí množina měst. Ve městě se vyskytují továrny a lidé. Mezi městy mohou vést obchodní cesty. Většina ekonomik je závislá na samotném prostoru. Proto je zaveden do simulace další objekt a to simulační mapa.

Pojmem hospodaření se zde uvažuje posloupnost aplikování některých operací ze souboru operací nad státem. Do tohoto souboru patří například rozšíření plochy města, postavení továrny, jejich produkce a další. Podrobně jsou všechny operace popsány v kapitole Změny stavu světa a její ekonomiky.

Nyní následuje seznam struktur, které definuje uživatel aplikace.

2.1.1 Materiál

Seznam produktů jejichž toky se budou simulovat. Materiálem se rozumí jak vstup tak i výstup továrny. Mezi materiály je zahrnuta i lidská práce.

Příklad: železo, sklo, jídlo...

2.1.2 Kvalita

Kvalitou se rozumí vlastnost pole, pomocí nichž lze upřesnit „úrodnost“ daného pole při výrobě továren.

Příklad: vlhkost, slunečnost, úrodnost...

2.1.3 Pole

Na mapě se rozlišují jednotlivé typy polí. Touto vlastností lze určit, kde všude lze daná

továrna postavit. Dále se u pole nastavuje cena pro stavbu cesty a zda je vůbec možné cestu na daném poli postavit.

Příklad: louka, jezero, hory...

2.1.4 Továrna

Továrnou se rozumí objekt, ve kterém se vyrábí určité produkty. U továrny se nastavuje jméno a typ pole, na kterém se smí postavit. Dále lze nastavit poloměr viditelnosti. Tento údaj říká, jak velký kruh na mapě továrna pro město přivlastní.

U továrny se ještě také nastaví cena postavení. Mezi důležité parametry určitě také patří seznam a množství materiálů, které vyprodukuje a spotřebuje během výrobního procesu.

Výrobu lze dále ovlivnit kvalitativními koeficienty.

2.1.5 Člověk

Aby se simulace přiblížila skutečnosti, byl zaveden další subjekt – člověk. Mezi jeho vlastnosti patří: jméno a potřeby. Pod pojmem potřeby se rozumí seznam potřeb daného typu člověka. Její parametry jsou materiál, jeho množství a uspokojivost. Poslední parametr vlastně říká, jak moc je daná potřeba pro člověka důležitá. Dále je u objektu také možné nastavit cenu za jeho „postavení“.

2.1.6 Mapa

Mapa je čtvercovou sítí polí. Každé pole je určitého typu a má specifické vlastnosti, které určují výtěžnost daného pole během výroby v továrnách. Jedno pole může patřit nejvýše jednomu městu a pokud patří nějakému městu, tak na něm může stát továrna.

2.1.7 Město

Město je objekt na mapě, vlastnící kolekci instancí továren a lidí. Městu dále patří určitá pole na mapě. Na nich lze právě postavit továrny. Když se město poprvé umístí na pole, určí se dané pole jako centrum města. Toto pole ovlivňuje chod a stavbu továren.

2.1.8 Cesty

Cestou se rozumí seznam polí, po kterém se může provádět obchod. Cesty patří státu a nikoli městu. Aby bylo možné uskutečnit obchod mezi dvěma městy, musí existovat posloupnost polí na cestě, jejíž počátek patří prvnímu městu („prodejce“) a konec patří městu druhému („odběratel“).

2.2 Změny stavu světa a její ekonomiky

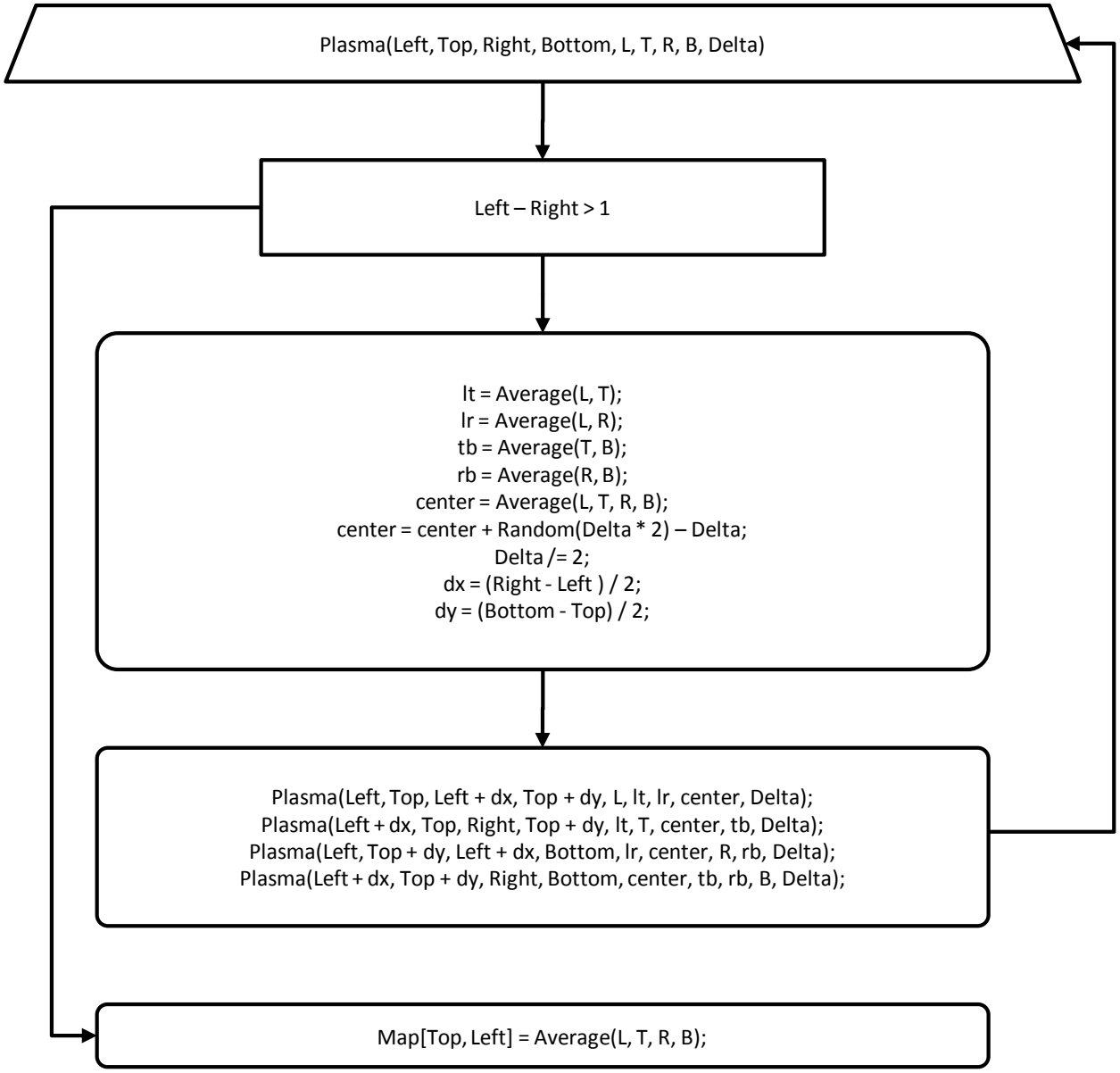
K simulování ekonomiky je potřeba definovat základní operace a vztahy mezi objekty.

Nyní budou vyjmenovány operace, které aplikace nabízí.

2.2.1 Generování mapy

Generováním mapy se rozumí nastavení typu a kvalit polí. Pro zopakování – pole musí být právě jednoho typu a obsahovat seznam koeficientů pro jednotlivé kvality.

Pro generování je využit algoritmus přesouvání prostředního bodu inspirován z Root.cz [2]. Jde o rekurzivní postup využívající skutečnosti, že mapa je tvořena čtvercovou sítí polí. Algoritmus je volán (n+1)-krát. Jednou pro typ a n-krát pro každou kvalitu pole. Následuje diagram algoritmu:

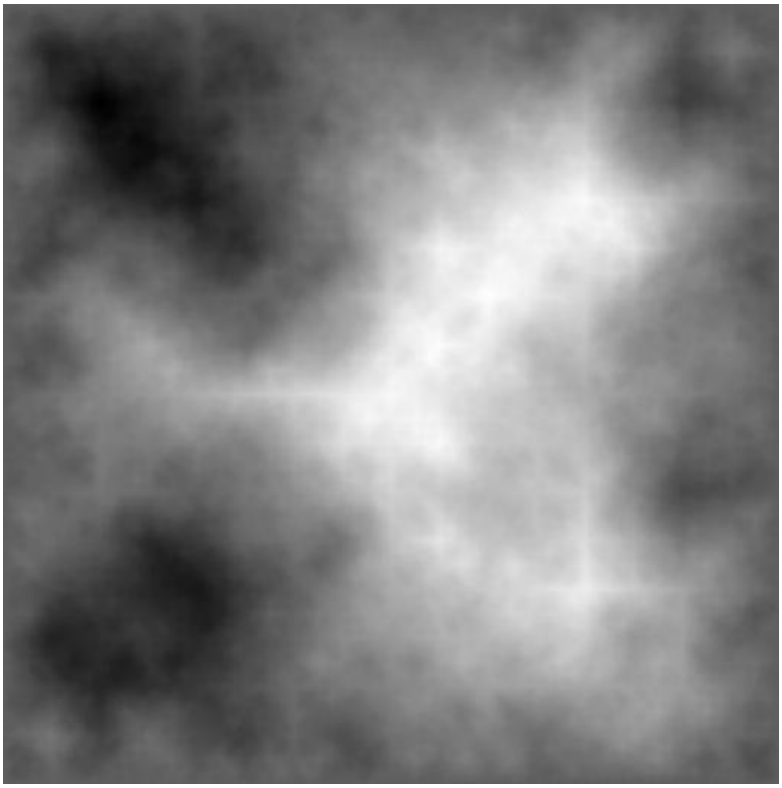


Obrázek 1 Algoritmus generování mapy

Algoritmus tedy postupně rozděluje mapu na poloviční čtverce a průběžně jim počítá barvu hran. Při každé iteraci se posune barva středu čtverce o náhodnou hodnotu. Rekurze pokračuje dokud je hrana čtverce delší než 1 bod – tedy jedno pole na mapě. Poté se nastaví hodnota

daného pole jako průměr barev čtverce.

Výsledkem může být následující mapa:



Obrázek 2 Vygenerovaná plasma

Jak je na obrázku vidět, používaná metoda trpí tzv. „cracked edge“. Jde o „kříže“ či zlomy na hranách čtverců. Vada metody je způsobena tím, že se mezi jednotlivými iteracemi přenáší jen hodnoty hran daného čtverce. Řešením chyby by bylo zohlednění hodnot na okolních čtvercích.

Nicméně zmíněná vada nemá na generování map významný negativní dopad. Navíc je ještě částečně eliminována „blur“ metodou – výsledné hodnoty jsou průměrem z určitého okolí. Konkrétně se jedná o kruh s poloměrem logaritmus z šířky mapy.

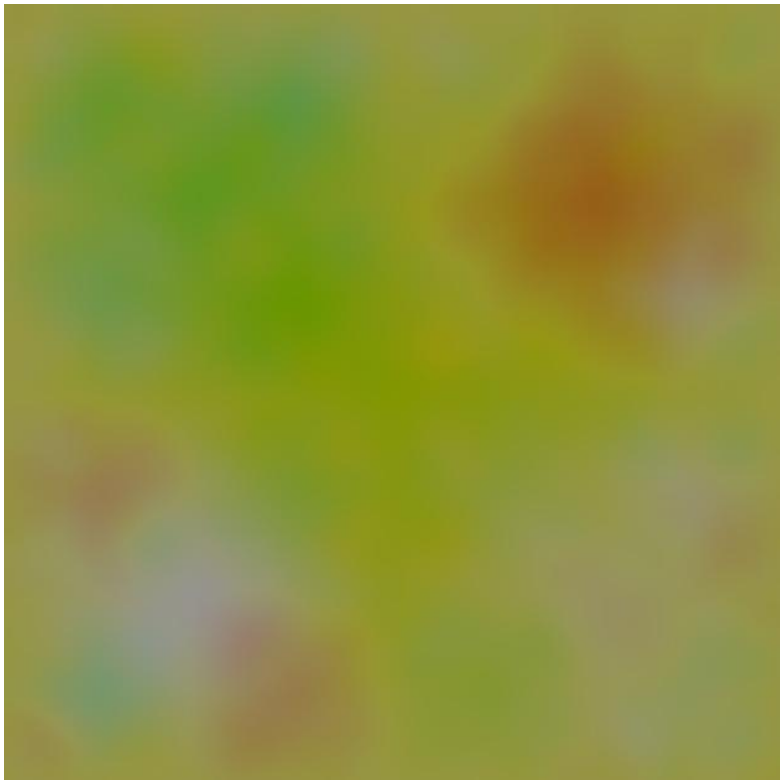
Dalším krokem při generování mapy je normalizace – tedy posun a zvětšení/zmenšení hodnot do požadovaného rozsahu.

Typy polí mapy jsou generovány následujícím algoritmem. Vstupem je velikost mapy, typy polí a jejich relativní množství na mapě. Nejdříve se vygeneruje mapa hodnot metodou Plasma, poté se vypočítá histogram jednotlivých hodnot zaokrouhlených na celá čísla. Z histogramu se určí intervaly hodnot mapy, tak aby se zachoval požadovaný poměr typů polí

na mapě. Výsledná mapa se vytvoří tak, že se pro každé pole mající hodnotu v daném intervalu zvolí daný typ pole. Výsledkem může být následující mapa:

Obrázek 3 Mapy polí světa

Posledním krokem při generování mapy světa je vložení kvalit na jednotlivá pole. Postup je podobný jako u typů polí, nicméně se zde neprovádí část s histogramem. Pouze pro každou kvalitu probíhá normalizace do požadovaného rozsahu.



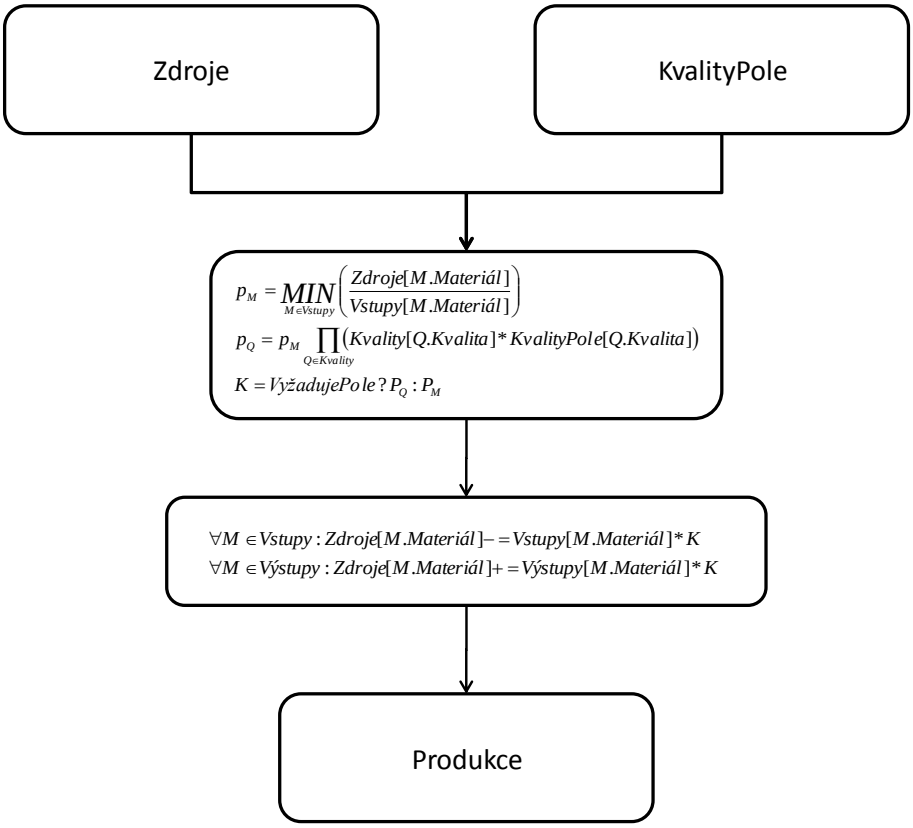
Obrázek 4 Ukázka kvalit polí mapy

2.2.2 Produkce továren

Další důležitou operací simulace je samotná produkce továren. Továrny, jak už bylo zmíněno v jedné z minulých kapitol, jsou objekty města sloužící k přeměně vstupů na výstupy s případným přihlédnutím na kvality pole továrny.

První otázkou je tedy, jak továrna zdroje (vstupy) získá. V simulaci je to řešeno objednávkovým systémem. Každé město má svého „šéfa“, který dané město řídí. Mimo jiné patří mezi jeho úkoly starání se právě o továrny a jejich objednávky. Továrna může jednak jako objednávku vytvořit nabídku – tedy „prodat“ produkty v ní vyrobené, nebo opačně poptávku, ve které poptává zboží potřebné pro výrobu.

Když tedy továrna získá vstupy (resp. prodá výstupy), může vyrábět produkty. Postup během výroby vypadá takto:



Obrázek 5 Algoritmus výrobního procesu továrny

Výsledné množství produktů omezují hlavně dostupné zdroje továrny. Nicméně výrobu lze ovlivnit u továren vyžadujících pole ještě kvalitou pole, na kterém stojí.

Vyrobené zboží továren se opět pomocí objednávkového systému nabízí ostatním objektům.

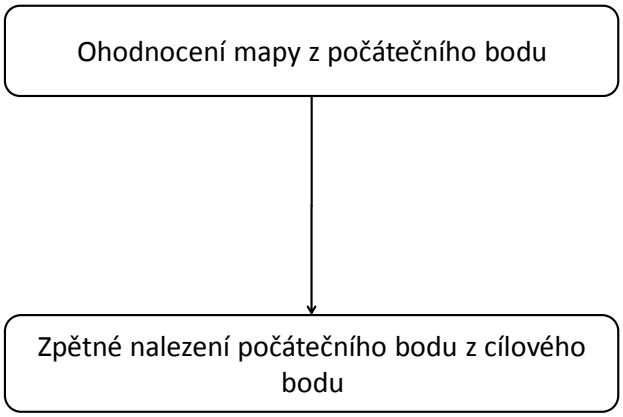
2.2.3 Stavba obchodních cest

K tomu, aby bylo možné mezi městy obchodovat, je potřeba spojit jejich území obchodní cestou. Jak již bylo zmíněno, cesta je seznamem polí, které do ní patří. Jednu cestu může sdílet více měst.

Stavba cest probíhá v několika krocích:

- vybrání počátečního a koncového bodu cesty
- nalezení nejkratší spojnice těchto bodů
- postavení samotné cesty

První krok je z aktuálního pohledu vcelku nezajímavý. Druhý krok však skrývá nejdůležitější bod celého procesu. Nyní bude popsán algoritmus pro hledání spojnice dvou bodů na mapě:

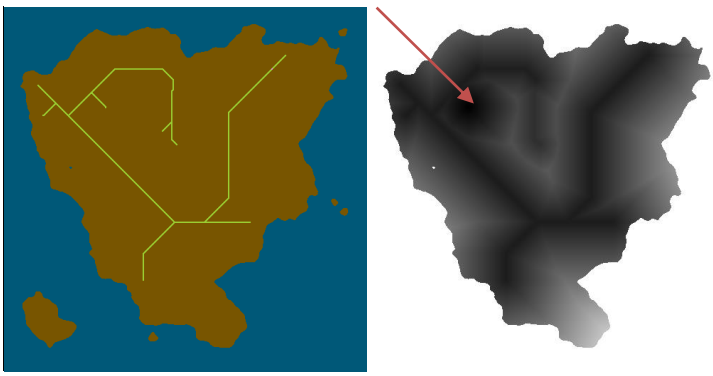


Obrázek 6 Algoritmus hledání spojnice bodů na mapě

První část algoritmu je ohodnocení mapy průchodem mapy do šířky z počátečního bodu. Každý krok do nového bodu je ohodnocen následujícím vzorcem:

$$vzdálenost * (Pole.JeNaNěmCesta\ ?\ 0.001 : Pole.CenaCesty)$$

Cesta je navíc omezena tím, že pole, přes které cesta vede, musí být typu umožňující stavbu cesty, nesmí patřit cizímu městu (tedy městem, které není v počátečním ani koncovém bodě cesty) a není na něm postavena továrna.



Obrázek 7 Mapa obchodních cest a ohodnocení stavěné cesty

Levý obrázek zobrazuje skutečnou mapu, na pravém je její ohodnocení. Červená šipka označuje počátek stavěné cesty. Odstín šedi říká cenu cesty do daného pole. Čím světlejší pole je, tím je dražší se na něj dostat. Na první pohled je zřejmé, že postavit cestu za použití stávajících cest je velmi výhodné.

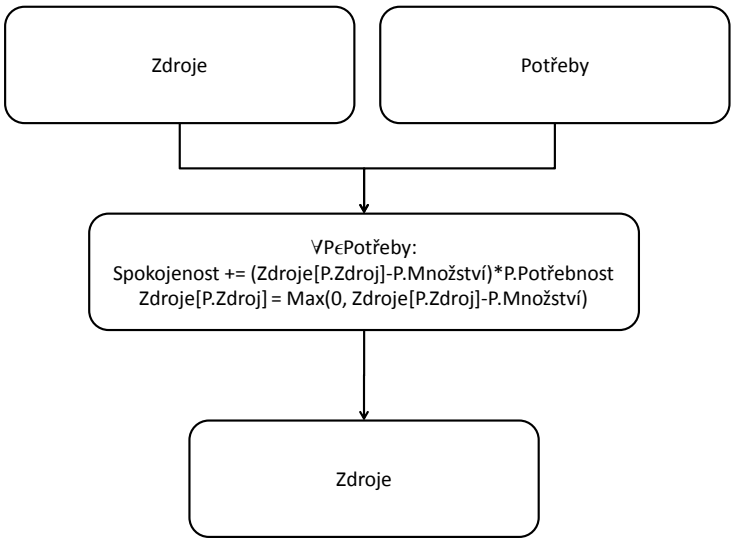
Druhou částí tohoto kroku algoritmu je hledání výsledné cesty. Postupuje se cyklicky od koncového bodu cesty a vždy se hledá sousední pole s nejnižším ohodnocením – toto pole se zvolí jako další pole cesty. Pokud je dané pole zároveň počátečním, byla nalezena cesty. Pokud neexistuje sousední pole s nižším ohodnocením, cestu není možné postavit a algoritmus končí.

Následuje postavení samotné cesty. Vyberou se pole cesty a daná cesta se vloží mezi cesty na mapě.

2.2.4 Lidé

Simulace pracuje také s lidmi. Jedná se objekty nabízející lidskou práci jako svůj „produkt“ a poptávají své materiální potřeby. Podobně jako u továren se nabídky a poptávky uplatňují pomocí vytváření objednávek u „šéfa“ města.

Jeho produkční cyklus vypadá takto:



Obrázek 8 Uspokojení potřeb člověka

2.2.5 Řízení měst

Vývoj a řízení měst má na starost „šéf“ města. Rozhoduje v každém simulačním kroku o následujících činnostech:

- výroba
    - továren

- výstavba
    - továren
    - obchodních cest
    - lidí
- obchod
    - výměna zboží s ostatními městy
    - uspokojení poptávky a nabídky v rámci města
- výběr daní
    - od továren
    - od lidí
- generování statistik
    - produkce továren
    - spokojenosti lidí
    - a mnoho dalších ukazatelů, záleží na implementaci metod GenStats v třídách Human a Factory

2.3 Optimalizační metody

Rozhodování během řízení města může usnadnit mnoho optimalizačních metod. Zkoumají nejrůznější ukazatele ekonomiky a na jejich základě počítají/navrhují další rozvoj města, tak by to bylo pro město výhodné.

Implementována byla „neviditelná ruka trhu“.

2.3.1 Neviditelná ruka trhu

V dnešních ekonomikách (především těch západních) se řízení a rozhodování o budoucnosti z velké části nechává na vlastnících subjektů. Ovšem z důvodu obecnosti simulačního procesu bylo rozhodnuto, že se továrny budou řídit vždy městem. Nicméně to tolik neomezuje vlastní rozhodování města. Může si počítat jakékoli statistiky výroby a simulovat onu „neviditelnou ruku trhu“.

Tato metoda funguje tak, že se na základě poptávky a nabídky počítají ceny zdrojů a určuje se na základě rentability, zda továrny budou vyrábět, zastaví se, zbourají se nebo se naopak postaví nové. Stejně tak se nakládá s lidmi a obchodními cestami.

2.4 Simulační model

Jak bylo zmíněno, každé město je řízeno svým „šéfem“. Jejich chování je do jisté míry společné. Simulace je prováděna diskrétně. V každém kroku simulace se každý z šéfů rozhodne, jak naloží se svým městem. Jejich rozhodování je rozděleno do následujících podkroků.

Každý podkrok je rozdělen do dvou částí – v první je zmíněno chování obecného šéfa a v druhé je implementace „InvisibleHandChief“.

2.4.1 Inicializace simulačního kroku

Base incializují se struktury pro výpočet statistik města

Invisible neprovádí nic

2.4.2 Produkce továren

Base nechá veškeré továrny produkovat své zboží a lidi uspokojit jejich potřeby

Invisible neprovádí nic

2.4.3 Nákup a prodej

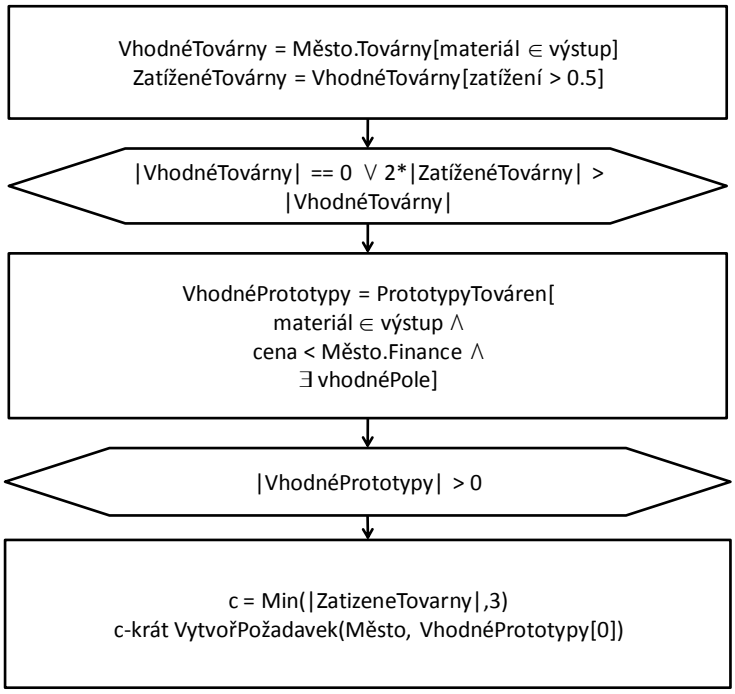
Base vyvolá u továren a lidí metodu, která jim umožní projevit jejich tlak na ceny, tedy zmíněné objekty mohou upravit ceny zboží ve městě. Tlak na ceny se určuje podle obratu materiálů v objektech. Čím vyšší obrat objekt má, tím je jeho tlak na cenu vyšší. Obrat je počítán vždy provedení transakce poptávky/nabídky. Jeho vzorec je:

$$Obrat[materiál] = \frac{Obrat\,[materi\,ál] + \frac{Objedn\;\;ávka.Vym\;ěněnéMno\;žstv\;í}{Objedn\;\;ávka.Celkov\;\;éMno\;žstv\;í}}{2}$$

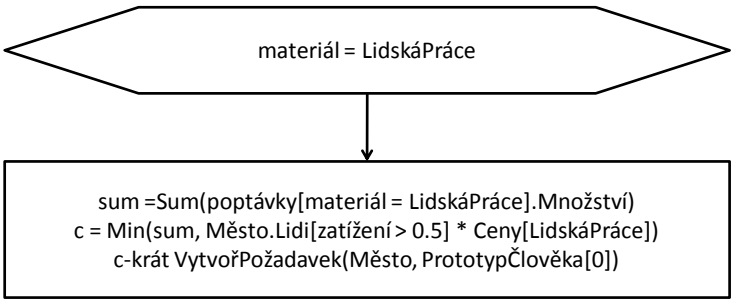
Hodnota Objednávka.VyměněnéMnožství je kladné v případě nabídky, v případě poptávky je záporné.

Invisible nechá provést to, co provedl Base. Podle poptávky a nabídky se snaží najít takovou změnu, aby je uspokojil – vytvoří požadavek na vytvoření továrny/člověka/obchodní cesty.

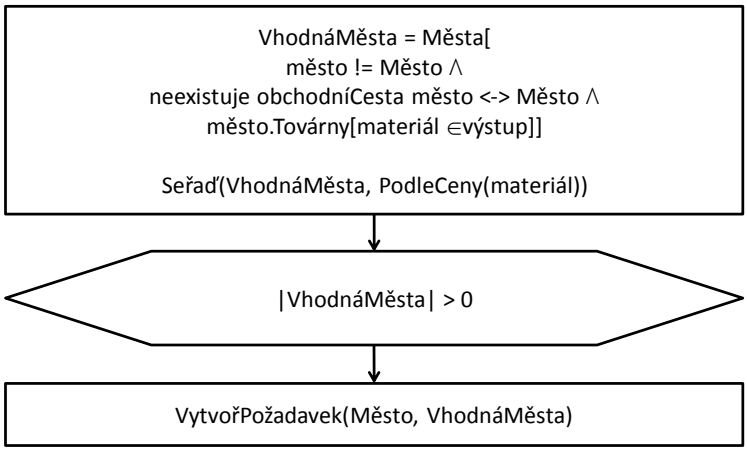
Pro každý typ materiálu mezi poptávkami se provedou následující akce. Pokud některá z akcí uspěje, pokračuje se rovnou na další typ materiálu.



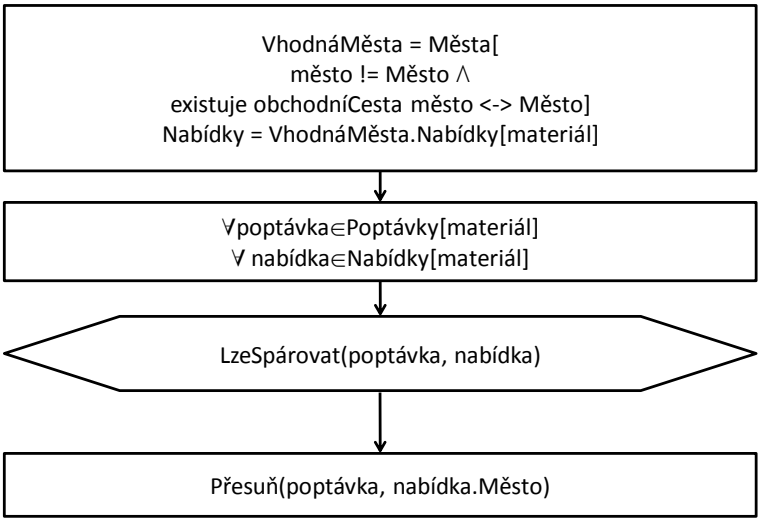
Obrázek 9 Hledání vhodné továrny



Obrázek 10 Vytvoření požadavků na vytvoření lidi



Obrázek 11 Vytvoření požadavku na postavení cesty



Obrázek 12 Přesun objednávek mezi městi – obchodování

Funkce LzeSpárovat zjišťuje, zda jednotková cena nabídky + případná cena za dopravu je nižší nebo rovna ceně poptávky. Cena v poptávce/nabídce je neměnná, ovšem cena zdrojů je v čase proměnná, tedy se tato kontrola musí provádět.

2.4.4 Uspokojení poptávek, nabídek a požadavků

Base vykoná veškeré požadavky města z předchozího kroku. Požadavek má určitou životnost a pokud ho není možné provést, uskuteční se další pokus v příštím cyklu simulace. Všem poptávkám a nabídkám se sníží životnost (počet cyklů simulace, po který je objednávka platná). Všechny uspokojené objednávky nebo objednávky s vypršenou platností se ukončí (továrny a lidi na to mouhou reagovat – připíší si nový stav svých zdrojů)

Invisible snaží se spárovat nabídky a poptávky následujícím způsobem: pro každou poptávku se hledají nabídky s daným zdrojem a cenou vyšší nebo rovnou požadované ceně. U vyhovujících se provede obchod – převede se materiál z nabídky do poptávky a opačným směrem se provede platba.

2.4.5 Stavba a bourání objektů

Base odstraní lidi, kteří jsou neuspokojení – mají spokojenost pod kritickou hodnotou, dále odstraní továrny, které delší (pevně danou) dobu nic neprodukují.

Invisible neprovádí nic

2.4.6 Finalizace kroku

Base odstraňuje požadavky, kterým vypršela platnost. Následuje výběr daní od továren a lidí. Pokud se dostanou finance města do záporných čísel, města se uvede do stavu bankrotu. Město se v tomto stavu přestane vyvíjet.

Invisible neprovádí nic

2.4.7 Generování statistik

Base počítá statistiky přijmů a výdajů města a volá generování statistik lidí a továren

Invisible neprovádí nic

3 Platforma

Program je napsán v jazyce C# ve verzi 3.0. Jeho předností je velmi vysoká abstrakce a řízený kód. Dalším důvodem pro napsaní aplikace ve zmiňovaném jazyku je i propracované vývojové prostředí Visual Studio. Původně byl projekt psán v C# 2.0 a Visual Studiu 2005. V loňském roce (2007) však vyšla nová verze jak C# tak i Visual Studia s množstvím inovací, které byly natolik významné, že došlo k přechodu do nových verzí – tedy C# 3.0 a Visual Studio 2008. Kompletní přehled novinek v C# 3.0 je na webových stránkách MSDN [3]. O Visual Studiu 2008 se lze dočíst na Microsoft.com [4].

3.1 Požadavky pro běh programu

Vzhledem k volbě C# jako implementačního jazyka vyplývají z tohoto rozhodnutí i nějaké požadavky na počítač, na kterém poběží. Tedy zde jsou požadavky na software:

- .Net Framework 2.0 nebo
- Mono >= 1.2.6

Aplikace byla testována převážně pod .Net Frameworkem 2.0, v Monu se občas vyskytly případy odlišného chování visuálních komponent. Pro testování přenositelnosti kódu z .Net Frameworku na Mono existuje velmi užitečný nástroj – MoMA [5].

Jak bylo zmíněno, .Net Framework funguje podobně jako Java, tedy C# zdrojový kód není kompilátorem překládán do nativního strojového kódu, nýbrž do takzvaného Common Intermediate Language [6]. To je na platformě nezávislý jazyk, umožňující ty nejzákladnější operace. Mimo jiné si ukládá metadata o kompilovaných třídách/strukturách, díky čemuž je možné přečíst obsah zkompilovaných assemblies. Nástroj MoMA využívá této vlastnosti a porovnává obsah assembly s implementací Mono. Tím je možné zjistit, zda nebyla v programu použita některá z neimplementovaných funkcí. Velmi usnadňuje migraci existujících .Net projektů pod Mono.

Dále je v aplikaci využit „parser“ a „scanner“ pro načítání skriptů vytvářejících strukturu světa. Oba nástroje pochází z Queensland University of Technology [7]. Aplikace pro běh využívá pouze dynamickou knihovnu *ShiftReduceParser.dll*. Ovšem z důvodu vytvoření gramatiky pro skripty se během překladu používají oba zmíněné nástroje – gplex.exe a gppg.exe. Více jsou popsány v kapitole - Soubory aplikace.

4 Struktura aplikace

Aplikace se skládá hned z několika binárních souborů:

- WorldSIM.exe
- WorldSIM.Simulation.dll
- ShiftReduceParser.dll

V prvním souboru je „start point“ celého programu. Obsahuje grafické a IO rozhraní. Druhým souborem je „Class library“ obsahující důležité pro samotnou simulaci a třídu BaseChief – tedy základní třídu, na které je možné založit vlastní implementaci řízení měst. Posledním souborem aplikace je knihovna, umožňující parsování textových souborů.

Program obsahuje několik jmenných prostorů, zde jsou vyjmenovány ty nejdůležitější:

- WorldSIM.Grammar
- WorldSIM.GUI
- WorldSIM.Lib
- WorldSIM.Simulation
- WorldSIM.World

Mimo zmíněné se v aplikaci používají ještě další prostory – WorldSIM.Examples a WorldSIM.Test, nicméně ty jsou, jak už z názvů může být patrné, jen pro testování a vývoj nových vlastností. Rozebírat je dále nemá smysl.

Každý ze zmíněných jmenných prostorů má podle svého názvu svůj adresář, tedy se zachovává přehlednost zdrojových souborů. Drtivá většina tříd je napsána v samostatných souborech pojmenovaných stejně jako je jméno dané třídy. U některých však došlo ke sloučení z důvodu zachování přehlednosti. Mezi jinými jsou to třídy používané k vyhazováním výjimek. Všichni tyto potomci (i nepřímí) třídy Exception mají implementaci na několik řádků a je jich tolik, že došlo ke zmíněnému sloučení.

Naopak velké třídy jako Parser nebo InvisibleHandChief byly za použití klíčového slova partial rozděleny do několika souborů. V názvech těchto souborů je však zachována následující notace <JménoTřídy>.<ÚčelMetodVSouboru>.cs (např. InvisibleHandChief.Market.cs). Podobně jsou vytvářeny třídy grafických komponent přímo Visual Studiem. V „project“ souboru Visual Studia lze dokonce nastavit sloučení souborů

v „Solution exploreru“, ovšem u ručně slučovaných souborů se změna projevila nekorektně. Zřejmě je potřeba ještě uvést nějaký další parametr.

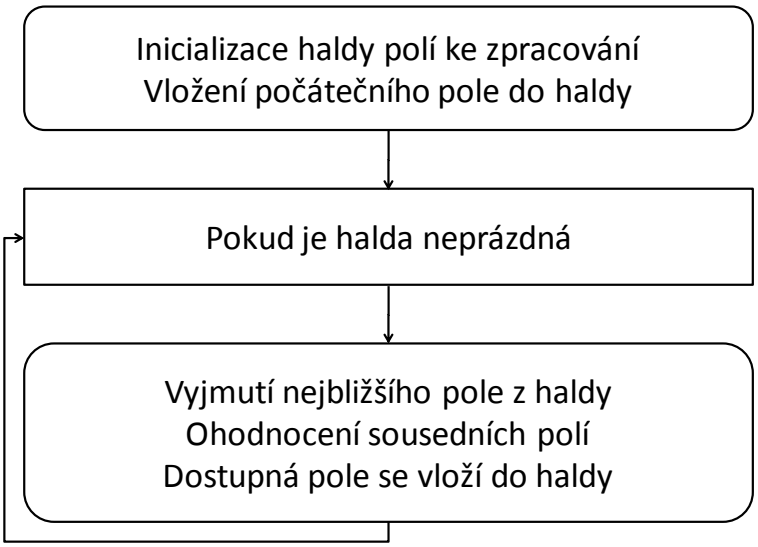
4.1 WorldSIM.World

Tento jmenný prostor je, jak se říká, alfou a omegou simulovaného světa. Jsou zde definovány veškeré jeho struktury a některé algoritmy.

4.1.1 Nejkratší cesta

Hledání nejkratší cesty je implementováno ve třídě Path jako její statická metoda - Shortest. Metoda bere tři argumenty – dvě pole (počátek a konec cesty) a mapu světa. Návratová hodnota je typu Path – tedy zkonstruovaná cesta.

Jak již bylo zmíněno v návrhu, metoda má dvě části – ohodnocení polí mapy a nalezení cesty.



Obrázek 13 Průchod mapou do šířky a její ohodnocení

V ohodnocovací metodě se zvolilo jednoduché „cachování“, aby se výpočet neprováděl zbytečně. Výsledné ohodnocení se tedy uloží do statické proměnné a pokud by se v budoucnu algoritmus volal se stejným počátečním bodem, použije se toto ohodnocení znovu. Cachování se zejména uplatní během grafického návrhu cesty, kdy se vlastně mění jen koncový bod a není potřeba ohodnocení přepočítávat.

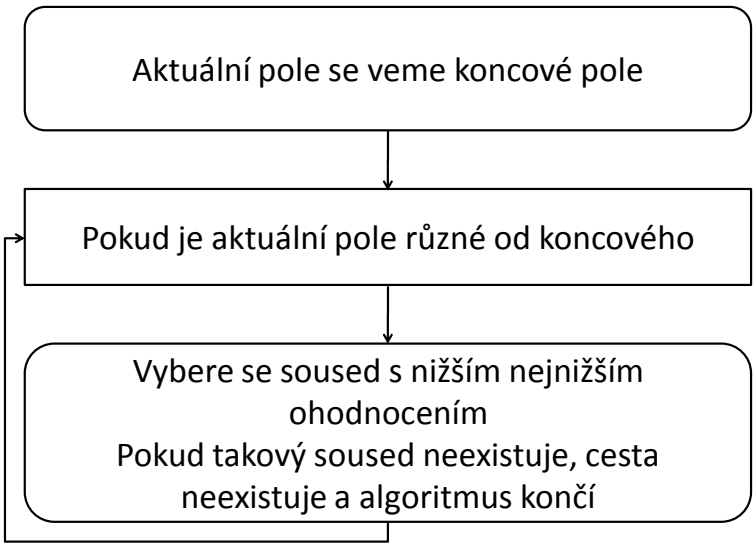
Jako sousední pole se uvažuje všech 8 okolních polí, tedy nejen ty vodorovném a svislém směru, ale i ve směru úhlopříčném. Pole se ohodnotí sečtením délky cesty do aktuálního pole a následujícího vzorce (hodnota *vzdálenost* je rovna vzdálenosti aktuálního a následujícího

pole, tedy buď 1 [pozice polí se liší pouze v jedné ose] nebo $\sqrt{2}$ [následující pole je v jednom ze šikmých směru od aktuálního pole]):

$$vzdálenost * (Pole.JeNaNěmCesta\ ?\ 0.001 : Pole.CenaCesty)$$

Podmínka je ve vzorci proto, aby se využívali již stávající cesty.

Po ohodnocení mapy přichází druhý krok, ve kterém se hledá výsledná cesta. Algoritmus je velmi jednoduchý.



Obrázek 14 Nalezení cesty

Tím algoritmus hledání nejkratší cesty končí. Výsledkem je struktura Path obsahující referenci na seznam polí dané cesty a boolovskou hodnotu říkající, zda cesta existuje. Při přidání cesty do mapy dojde ke sloučení společných polí. Tedy pro všechny cesty obsahující některé společné pole se vloží všechny pole z této cesty do přidávané cesty a původní cesta se zruší.

4.2 WorldSIM.Lib

Jelikož se v simulaci, ve velkém míře využívají kolekce (seznam dat) a někdy je potřeba upravit chování během manipulace s prvky, bylo potřeba potlačit původní chování standardních kolekcí. Ty však nemají označeny své metody jako virtual, tudíž je nelze předefinovat. Proto tedy byly napsány třídy VirtualList<Typ> a VirtualDictonary<Typ>.

4.2.1 Ukládání světa

Pro ukládání stavu světa je zvolena velmi silná technika – serializace. Díky možnosti reflexe

v .Netu je možné tento proces zcela zautomatizovat. Framework nabízí hned několik metod pro serializaci – XML, XML-Soap nebo binární. Byla vybrána poslední, protože první zcela nevyhovuje požadavkům. Jednak generuje XML kód, což má za následek velmi mnoho zbytečných dat okolo. Tato vlastnost by v dnešní době byla uspokojující, nicméně během serializačního procesu touto metodou dojde ke ztracení informace o cyklických vazbách mezi objekty. Zachovávají se pouze vztahy typu rodič ⇔ potomek, ovšem známou vazbu ve tvaru diamantu není možné zachovat.

Další zmíněnou možností by byl XML-Soap – funkčně by již vyhovovala, nicméně jak už bylo řečeno XML sebou nese spoustu dat navíc, v následující verzi .Net (3.5) je „obsolete“ a dokonce neumí serializovat generické typy, proto se v aplikaci serializuje binárně.

Jediný problém během serializace je v serializaci statických položek. Statická položka je např. výchozí prototyp pro zdroj „lidská práce“. Je definován v třídě Human následovně:

```
private static IResource humanResource = new Resource("Lidi");
public static IResource HumanResource
{
  get { return humanResource; }
  set { humanResource = value; }
}
```

Problém nastává ve chvíli, kdy je např. potřeba porovnat nějaký zdroj s lidskou prací. Porovnání se provádí na základě reference a tedy pokud se daný zdroj zaserializoval a deserializoval, je možné (např. kvůli garbage collectoru, restartu aplikace...), že se statický objekt humanResource dostal na jinou adresu a tedy si reference nebudou odpovídat, i když by si při vynechání serializace odpovídali.

Proto je potřeba při serializaci si uložit referenci na humanResource do nějaké nestatické položky, která se serializovat bude vždy a po deserializaci si do humanResource uložit tuto referenci.

Tato skutečnost se vyskytuje v aplikaci na více místech, a proto je ve třídě World několik „zbytečných“ položek navíc právě pro řešení zmíněného problému.

Konečně samotnou serializaci obstarává generická statická metoda, která navíc (de)komprimuje (de)serializované objekty. Její implementace je ve třídě WorldSIM.Lib.LoadSave. Používá se standardní komprimace GZip – součást .Net.

4.3 WorldSIM.Grammar

V jmenném prostoru WorldSIM.Grammar se definuje zpracování skriptu. Pomocí nich lze velmi jednoduše bez zdlouhavého klikání myší naplnit struktury světa. Skripty dokáží struktury pouze rozšiřovat. Pro editaci nebo odstranění stávajících struktur je potřeba použít GUI. Kompletní popis gramatiky je v uživatelské dokumentaci – kapitola 5.3.6.

4.4 WorldSIM.GUI

Velkou část programu zabírá grafické rozhraní. Je rozděleno na dva jmenné podprostory – Forms a Controls. V prvním jsou definovány formuláře (okna) aplikace, v druhém pak používané grafické komponenty.

Následují nejdůležitější komponenty. Formuláře nejsou algoritmicky zajímavé.

4.4.1 Paleta barev

Pro vizualizaci mapy je potřeba rozlišovat jednotlivé objekty na ní zobrazené. Proto se musí nějak odlišit jejich barvy. Aby to bylo pro lidské oko alespoň trochu estetické, jsou použity HSB [8] barvy. Oproti RGB barvám, které jsou jednoduše zobrazitelné počítačem, mají přirozenější význam pro člověka. Výsledná barva závisí na třech složkách – tedy podobně jako u RGB, nicméně význam jednotlivých složek je velmi odlišný. Jedná se o tyto složky:

- Hue - barevný tón, převládající. Neboli odstín - barva odražená nebo procházející objektem. Měří se jako poloha na standardním barevném kole (0° až 360°). Obecně se odstín označuje názvem barvy.
- Saturation - sytost barvy, příměs jiné barvy. Někdy též chroma, síla nebo čistota barvy, představuje množství šedi v poměru k odstínu, měří se v procentech od 0% (šedá) do 100% (plně sytá barva). Na barevném kole vzrůstá sytost od středu k okrajům. Např. červená s 50% sytostí bude růžová.
- Brightness - hodnota jasu, množství bílého světla. Relativní světlost nebo tmavost barvy. Jas vyjadřuje množství světla, které barva odráží.

V aplikaci jsou však rozsahy upraveny pro všechny tři složky na [0; 255].

Během vykreslování mapy je dotazována třída Pallete na barvu nějakého objektu – typicky FieldType, Quality, City a FactoryPrototype. Paleta se podívá do seznamu dosud obarvených objektů, pokud v něm požadovaný objekt je, vrátí jeho barvu, jinak přiřadí objektu novou barvu, jejíž barevný odstín je otočen o určitý úhel od poslední takto přidané barvy.

Jelikož jsou barvy FieldType a Quality vetšinou barvy objektů v pozadí, mají sníženou saturaci. Ostatní objekty jsou pak lépe vidět.

Během běhu programu je možné upravit barevné tóny barev objektů:

Obrázek 15 Použití komponenty PaletteColorsChanger

Po změně barvy některých objektů je potřeba překreslit hrací plán – to se provede stiskem tlačítka „Update“.

4.4.2 MapViewer

Komponenta MapViewer je natolik důležitá, že v této části se popíše její vlastnosti z programátorského hlediska a její použití je popsáno v uživatelské dokumentaci – kapitola 5.3.6.

Jedná se tedy o vizuální komponentu zobrazující simulovaný svět. Mezi její hlavní funkce patří:

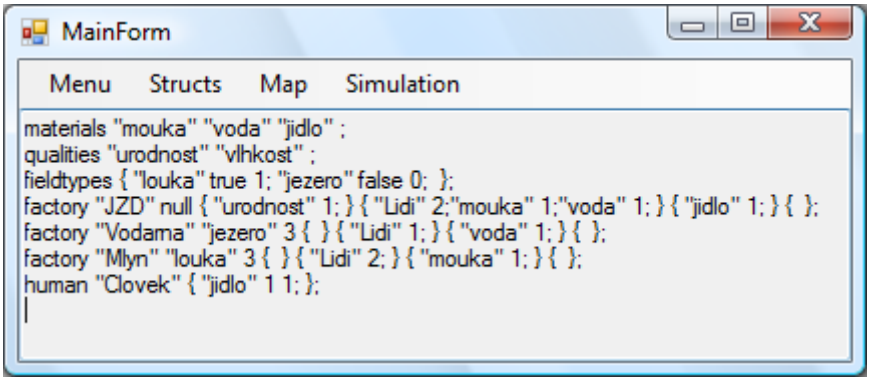
- zobrazování mapy, měst, obchodních cest
- zobrazování dodatečných informací o objektech
- funkce drag&drop pro přidávání objektů

Vykreslování mapy se provádí ve dvou krocích. Nejprve se (během inicializace světa) vykreslují dvě základní vrstvy – typy polí a jejich kvality. Jelikož je to časově náročný proces a stav polí se během simulace nemění, výsledek se uloží do 2 bitových map. Druhým krokem je kreslení „měnících se“ vrstev. Jedná se o plochy měst a jejich továrny a obchodní cesty. Tento krok je prováděn při každé invalidaci komponenty MapViewer.

5 Uživatelská dokumentace

V této kapitole jsou popsány hlavní komponenty a principy aplikace. Kompletní popis se nachází v příloze.

5.1 GUI - MainForm



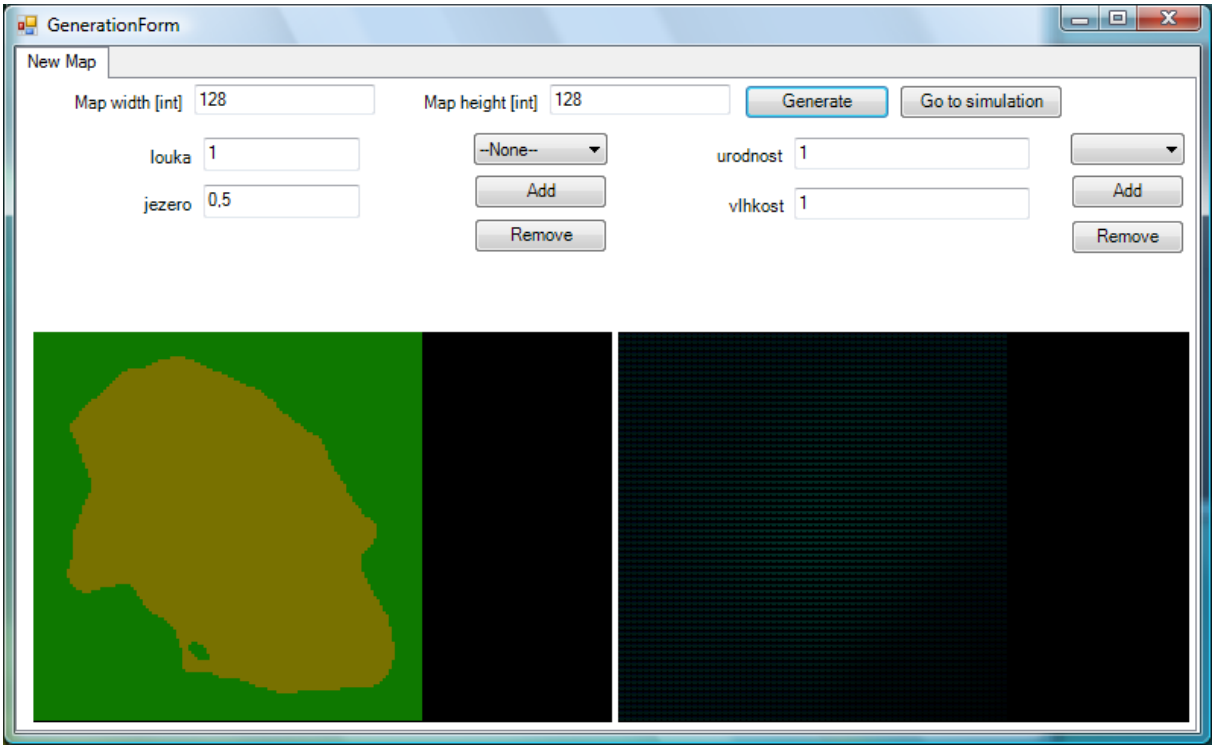
Obrázek 16 Hlavní okno

V hlavním okně aplikace se zejména vytváří struktury světa. K nim se lze dostat přes nabídku „Structs“. V obsahové části okna je vygenerovaný skript struktur. Po každé změně se vygeneruje odpovídající skript.

Struktury nebo celý svět lze též načíst a uložit přes menu „Menu“. Přes stejnou nabídku je také možné struktury vyčistit. Skripty se ukládají do souborů s koncovkou .script a binární obrazy světa do souborů .world.

Přes menu „Map“ se generuje mapa světa a pomocí poslední nabídky se lze dostat k simulaci samotné.

5.2 GUI - GenerationForm



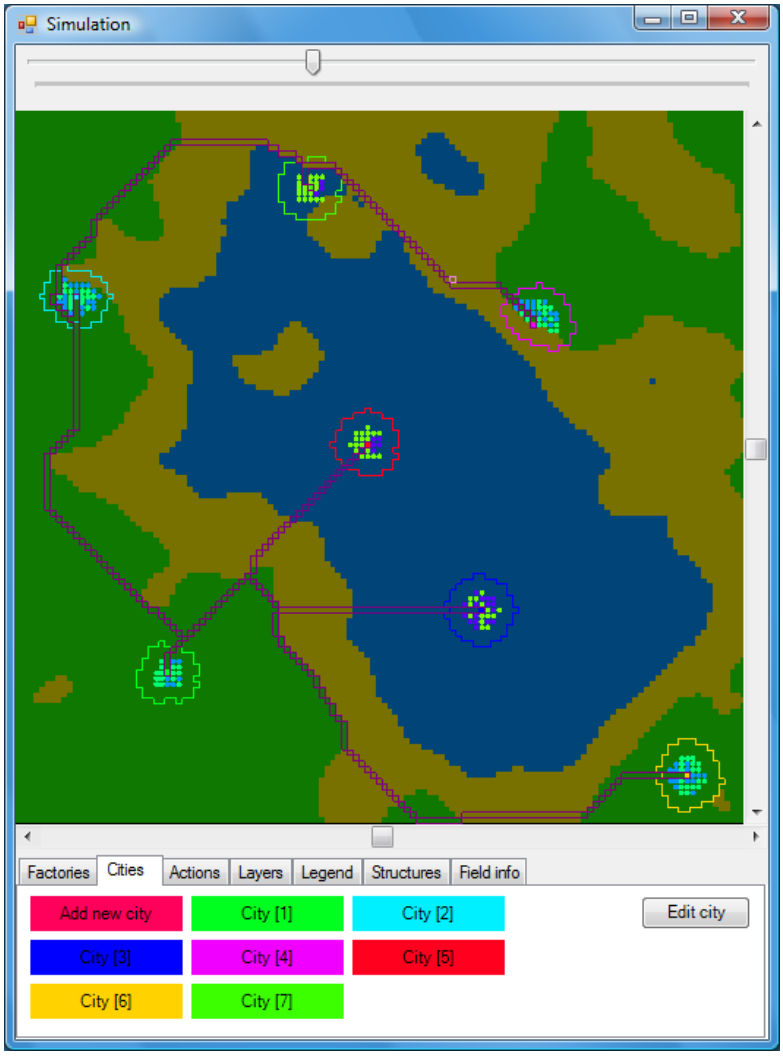
Obrázek 17 Generování mapy

GenerationForm, okno pro generování mapy, obsahuje základní údaje pro vygenerování nové mapy. Nejdříve se zadají rozměry mapy, dále se pokračuje výběrem typů polí a jejich procentuální zastoupení na mapě. Třetím parametrem je seznam kvalit a jejich maximální hodnoty.

Po zadání parametrů se mapa vygeneruje po stisknutí tlačítka „Generate“. Po vygenerování mapy se zobrazí výsledná mapa ve dvou komponentách v dolní části okna.

Pokud je daná mapa vyhovující, je možné přejít rovnou k simulaci – tlačítko „Go to simulation“.

5.3 GUI - Simulation



Obrázek 18 Simulační okno

V simulačním okně je možné sledovat a korigovat běh simulace. V horní části je posuvník pro nastavování přiblížení (zoomu) zobrazované mapy. Uprostřed vykresluje MapViewer mapu samotnou. V dolní části jsou záložky s jednotlivými tlačítky pro různé změny stavu simulace nebo dalšími informacemi o stavu simulace.

5.3.1 Factories

V záložce Factories je seznam dostupných typu továren. Daný typ továrny je možné instanciovat pomocí drag&drop na některém poli mapy. Toto pole musí patřit některému městu, podle toho se nastaví vlastník továrny. Dále na poli nesmí stát obchodní cesta a typ pole musí vyhovovat požadavkům továrny. Pokud daná továrna pole nevyžaduje, lze ji umístit na libovolné pole některého města.

5.3.2 Cities

Podobně jako u továren má i v záložce Cities každé město své tlačítko, pomocí nějž je možné zvětšovat plochu města. Tlačítkem „Edit cities“ je možné přidávat/odebírat města ze světa. „Edit city“ zobrazí informační okno vybraného města. V něm je možné editovat lidské zdroje a továrny města.

5.3.3 Actions

V záložce „Actions“ jsou tři tlačítka – Simulate, Save whole world a Load world. První z nich je to nejzásadnější, které simuluje celý svět. Po kliknutí se zavolá pro každé město jeho „šéf“ a na něm jeho metoda Organize.

Zbylá dvě tlačítka načítají a ukládají stav světa z/do souboru.

5.3.4 Layers

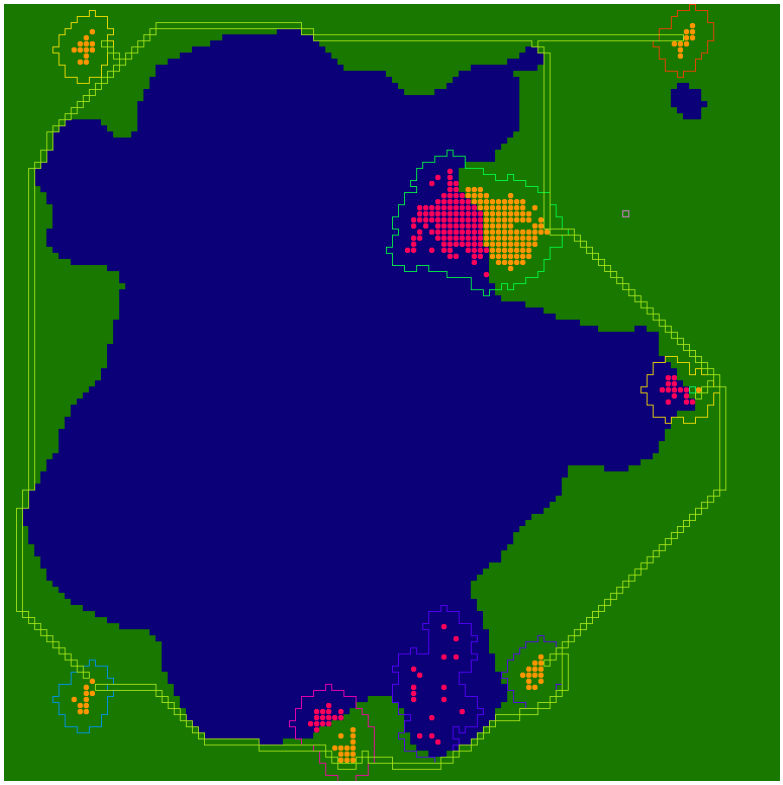
Zde je možné vypínat a zapínat zobrazování jednotlivých vrstev mapy – FieldTypes a Qualities.

5.3.5 Legend

Jak už bylo zmíněno v předcházejících kapitolách, tady se nachází komponenta pro editaci barev objektů na mapě. Po změně barev je potřeba aktualizovat MapViewer stiskem tlačítka Update.

5.3.6 MapViewer

Grafický prvek MapViewer je komponentou pro zobrazování aktuální mapy světa.



Obrázek 19 Ukázka použití MapViewer

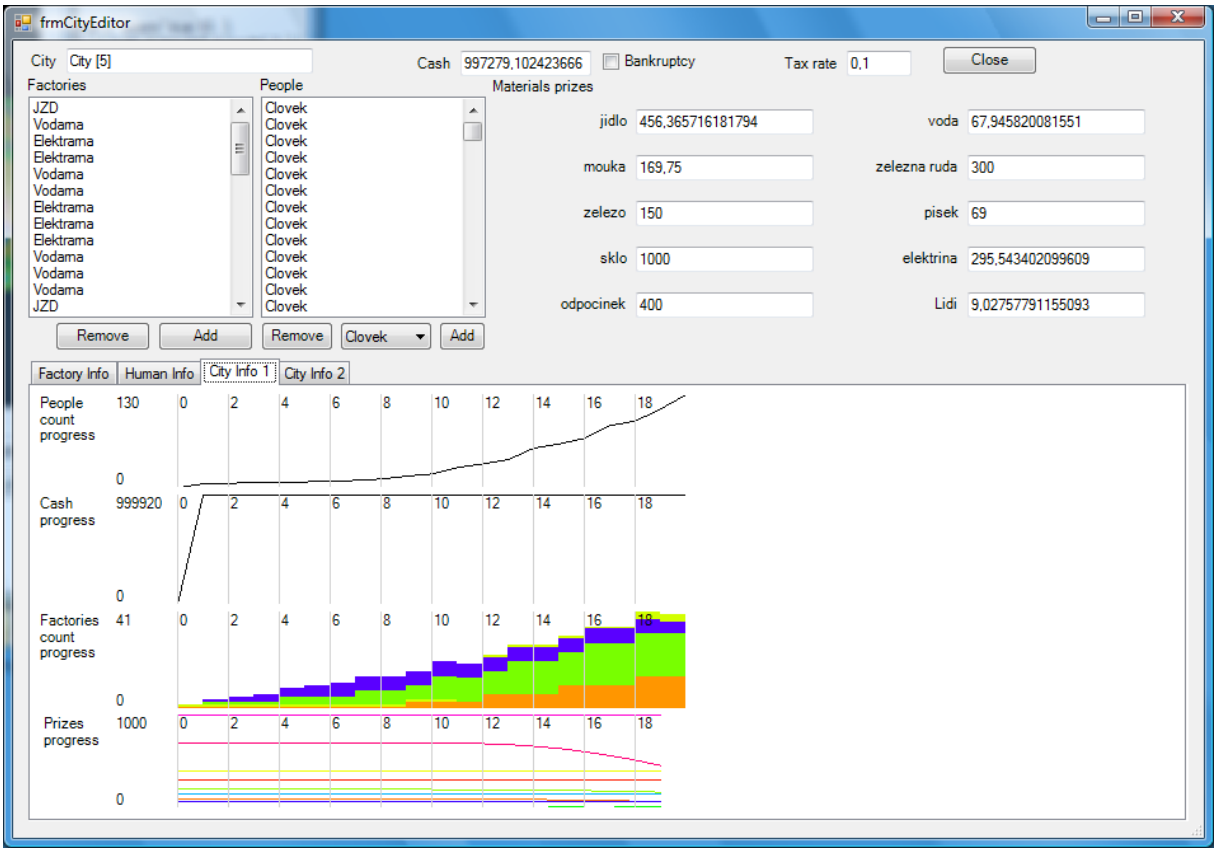
Komponenta zobrazuje mapu světa – typy polí na mapě, města, jejich továrny a obchodní cesty. Na mapě lze pomocí techniky drag&drop vytvářet jednotlivé instance struktur. Dále lze navrhovat obchodní cesty. Nová cesta se začne stiskem levého tlačítka myši na některém poli mapy. Tím se určí počáteční pole cesty, koncové je to, nad kterým je aktuálně myš. Jejich spojnice je definována algoritmem hledání nejkratší cesty popsaným v jedné z předchozích kapitol.

Kliknutím myši lze vybrat objekt na daném poli s následující prioritou objektů:

1. cesta
2. továrna
3. město
4. pole

Dále se u kurzoru myši zobrazuje plovoucí okno s informacemi o poli pod ním. Okno obsahuje typ pole, jeho kvality, jméno města, případně informace o továrně.

5.3.7 CityEditor



Obrázek 20 Statistiky a nastavení města

Velmi užitečným okny simulační aplikace bývají statistiky – to je právě okno CityEditor. Je v něm zobrazeno několik důležitých ukazatelů vývoje města – počet obyvatel, dostupné finanční prostředky, obrat zboží a další. Statistiky jsou ve čtyřech záložkách v dolní části okna.

Dále je zde možné nastavit jméno, finance, daně a ceny zboží. Lze editovat i obyvatelstvo a továrny.

5.4 Gramatika

Skript musí být v jazyce generovaném touto gramatikou:

```
<skript>        ::= <příkazy>
<příkazy>       ::= <příkazy> | <příkaz> ';' | ''
<příkaz>        ::= <kvality> | <materiály> | <pole> | <člověk>
                    | <továrna>
<kvality>       ::= 'qualities' '{' <jména> '}'
<materiály>     ::= 'materials' '{' <jména> '}'
<pole>          ::= 'fieldtypes' '{' <typy polí> '}'
<člověk>        ::= 'human' <jméno> <číslo> '{' <def. potřeb>
                    '}'
<továrna>       ::= 'factory' <jméno> <def. pole> <číslo> '{'
                    <def. kvalit> '}' '{' <def. vstupů> '}' '{'
                    <def. výstupů> '}'
<typy polí>     ::= <typy polí> | <jméno> 'true' <číslo> |
                    <jméno> 'false' <číslo>? | ''
<def. potřeb>   ::= <def. potřeb> | <jméno> <číslo> <číslo> ';'
                    | ''
<def. pole>     ::= <jméno> <číslo> | 'null'
<def. kvalit>   ::= <def. kvalit> | <jméno> <číslo> ';' | ''
<def. vstupů>   ::= <def. vstupů> | <jméno> <číslo> ';' | ''
<def. výstupů>  ::= <def. výstupů> | <jméno> <číslo> ';' | ''
<jména>         ::= <jména> | <jméno> ';' | ''
<jméno>         ::= '[[:IsLetter:]]([[:IsLetter:]|[0-9_])*' |
                    '"[^"]*"'
<číslo>         ::= <plusminus> <cifry> | <plusminus> <cifry>
                    <čárka> <cifry>?
<plusminus>     ::= '+' | '-' | ''
<čárka>         ::= ',' | '.'
<cifry>         ::= <cifry> | [[:IsDigit::]]
```

Význam neterminálů:

Neterminály <kvality>, <materiály>, <pole>, <člověk> a <továrna> se chovají velmi podobně. Pokud je daný prvek (např. jméno kvality nebo jméno továrny...) unikátní vloží se takový prvek do globálního seznamu prvků (globální seznam kvalit, továren...). Jinak se hlásí chyba o duplicitě.

Neterminály začínající „def.“ mají opět podobné chování, pokud se prvek s daným jménem nenalezne v globálních seznamech, je nahlášena chyba použití neznámého prvku, jinak se hledá prvek podle jména.

Zbylé neterminály jsou pro řetězců a čísel. Veškerá čísla jsou reprezentovaná typem double, tedy číselným typem s plovoucí desetinnou čárkou s rozsahem $\pm 5 \times 10^{-324} \sim \pm 1.7 \times 10^{308}$ a

přesností 15 cifer.

Následuje podrobnější popis významu neterminálů:

<kvality>	pro každé jméno vytvoří takto pojmenovaný nový typ kvality pole
<materiály>	pro každé jméno vytvoří takto pojmenovaný nový typ materiálu
<pole>	pro každý typ pole, vytvoří nový typ pole se zadaným jménem, možností cesty na daném poli a případnou cenou cesty po daném poli
<člověk>	nadefinuje nový prototyp člověka – přiřadí mu dané jméno, cenu za vytvoření a jeho potřeby
<továrna>	nadefinuje nový prototyp továren – přiřadí mu dané jméno, požadavek na pole, cenu za vytvoření, vliv kvalit na výrobu, vstupy a výstupy
<def. potřeb>	vytvoří oceněný seznam materiálů, cení se požadované množství a vliv na spokojenost člověka; na materiál je potřeba se odkazovat přes jeho jméno
<def. kvalit>	vytvoří oceněný seznam kvalit, cení se pouze koeficient, kterým se bude násobit vyrobené množství produkce
<def. vstupů> **<def. výstupů>**	vytvoří oceněný seznam materiálů, cení se pouze množství

Tabulka 1 Popis významu neterminálů gramatiky skriptů

6 Srovnání s existujícími programy

6.1 LinCity-NG

LinCity-NG [9] je budovatelská hra, ve které je cílem vybudovat takové město, které bude mít trvale udržitelný rozvoj, nebo vynalézt „vesmírnou loď“ a přesunout obyvatelé města na jinou planetu.

Hráč může rozvíjet město kupováním a stavbou budov, služeb a infrastruktury.

Simulace uvažuje tyto indikátory:

- populace
- (ne)zaměstnanost
- přístup obyvatel k vodě a ekologie
- jídlo
- zboží (jeho dostupnost a produkce)
- zdroje (rudy, železo, uhlí)
- služby (vzdělání, zdraví a rekreace populace, ochrana proti požárům)
- energie (elektřina a dřevěné uhlí, uhlí s omezenými zdroji, sluneční a větrná energie)
- a další omezení jako finance, znečistění a doprava

Mnoho indikátorů je zobrazováno na mini mapce nebo ve statistikách. Hráč se musí starat o růst populace a další ukazatele, aby zajistil socio-ekonomickou rovnováhu.

Program se od bakalářské práce liší hlavně tím, že se soustředí pouze na jedno město.

Obrázek 21 Ukázka hry LinCity-NG

6.2 Age of Empires II

Hra Age of Empires II [10] je real-time strategií. Cíle hráče záleží na konkrétní „kampani“. Mohou být následující:

- Získání určitého území
- Vynalezení nějaké technologie
- Dosáhnutí dané velikosti populace

Hráč má na starosti své město, ovšem na herní mapě není sám. Jsou tam ještě další města, která mohou být řízena buď počítačem nebo jiným hráčem po síti. Hráči mohou povolit obchodování svých měst mezi sebou. K tomu, aby mohli obchodovat, jim stačí postavit budovu „Market“, nepotřebují se spolu nijak jinak spojovat. Ve hře se získávají čtyři základní suroviny – jídlo, zlato, dřevo a kámen. Suroviny těží lidé v dolech nebo farmách. Takto vytěžené zdroje je možné použít na stavbu objektů.

Např. lidé se „vytvářejí“ za spotřebování 50 jednotek jídla v hlavní budově města a pro každých 5 lidí je potřeba mít 1 dům. Na tento dům je potřeba 30 jednotek dřeva. Dále je možné stavět různé typy budov pro tvorbu vojenské síly, která slouží k útočení a ničení ostatních měst na mapě.

Obrázek 22 Ukázka hry Age of Empires II

6.3 Transport Tycoon

V počítačové hře Transport Tycoon [11] kontroluje hráč dopravní společnost. Ta může přepravovat cestující i zboží po silnici, železnici, vodě i letecky. Ve hře jsou konkurující společnosti řízené počítačem.

Hráč má za úkol stavět zastávky v blízkosti průmyslových objektů či měst a v případě jiné než letecké či vodní dopravy i dopravní cestu. Následně zde pak bude přepravovat výrobky, poštu či cestující v dopravních prostředcích, jež pro tento účel koupí. K dispozici jsou čtyři druhy dopravy: železniční, silniční, letecká a lodní. Množství komodit k přepravě ve stanici je ovlivněno hodnocením jeho dopravy, jež je určeno pro každou stanici, z které už bylo dopravováno. To samozřejmě závisí na kvalitě, spolehlivosti a i bezpečnosti přepravy. Dopravní prostředky lze koupit v depu, kde jsou také pravidelně podrobovány servisním prohlídkám. Silniční vozidla se na rozdíl od železničních mohou pohybovat i po silnicích vlastněných jednotlivými městy či konkurencí.

Aby byl hráč odměněn za přepravu osob či zboží, musí je doručit do zastávky v oblasti, kde je po nich poptávka. V takovém případě je hráč odměněn částkou úměrnou vzdálenosti, rychlosti a množství přepraveného zboží či osob. Jakákoli činnost ve městě má vliv na hráčovo hodnocení městskou správou. Pokud hráč na území města provádí činnost zahrnující bourání budov a hlavně likvidaci lesů, jeho hodnocení se snižuje. Pokud nemá dostatečné hodnocení, městská rada mu nepovolí zbourání obecních domů či silnic nebo stavbu zastávky. Doprava v daném městě nebo vysazování stromů v okolí města může

hodnocení vylepšit.

Během hry dochází k vývoji měst i ekonomiky. Objevují se nové budovy, města se rozrůstají a vznikají tak nové nároky na existující dopravní sítě. Průmysl a doly vznikají, zanikají, mění se objem produkce. Vyvíjí se i nové druhy dopravních prostředků.

Obrázek 23 Ukázka ze hry Transport Tycoon

7 Závěr

Cílem této bakalářské práce bylo vytvořit prostředí pro simulaci ekonomiky na úrovni státu a navrhnout a implementovat základní simulační procesy. Naprogramovaná aplikace nabízí plně interaktivní uživatelské rozhraní, umožňující uživateli kontrolovat a měnit stav simulace.

Simulační proces je navržen tak, aby jej bylo snadné rozšiřovat a aplikace mohla umožnit simulovat a využívat i jiné optimalizační metody než ty, které aplikace nyní nabízí. Důraz byl kladen hlavně na co možná nejobecnější provázání jednotlivých struktur v simulačním světě jako je město, továrna, lidská práce a další.

Určitě je zde však prostor pro mnohá rozšíření:

- herní mód, ve kterém by měl hráč omezené možnosti zásahů do jednotlivých struktur, ať už fixní (mohl by provádět operace jen nad svým městem) nebo dynamické – na základě financí, zdrojů, rozlohy města by se mu zpřístupňovaly další operace nebo by opačně při poklesu ukazatelů možnost změn ztrácel
- síťová komunikace, která by umožnila zapojit více hráčů do hry nebo spojit výpočetní sílu více počítačů
- optimalizační metody založené na ekonomických teoriích a možnost jejich ověřování nebo výzkum nových
- lepší grafické rozhraní, umožňující vykreslovat potexturované objekty – továrny, cesty, pole...

8 Soubory aplikace

8.1 Přeložené nebo použité knihovny a aplikace

8.1.1 ShiftReduceParser.dll

Pomocná knihovna pro parsování skriptu – převádí znaky z textového souboru na symboly požadovaného jazyka. Požaduje, aby byl vstupní soubor resp. text vstupního souboru ve formátu UTF8.

Dodávána je společně s gplex.exe a gppg.exe.

8.1.2 WorldSIM.exe

V tomto binárním souboru je grafické, uživatelské a IO rozhraní aplikace.

8.1.3 WordlSIM.Simulation.dll

Knihovna definující struktury světa a operace nad nimi. Dále je v ní velmi důležitá třída BaseChief, která je předkem všech „šéfů“ řídících města.

8.2 Použité nástroje

Následující použité nástroje jsou z webových stránek Queensland University of Technology [7].

8.2.1 gplex.exe

Aplikace pro vygenerování scanneru regulárních výrazů v WorldSIM\Grammar\Scanner.lex. Výstupem je třída WorldSIM\Grammar\Scanner.cs. Je to vlastně konečný automat přijímající regulární jazyk. Každé slovo jazyka je popsáno regulárním jazykem. Příkladem budiž toto:

```
null        yylval.Token = Tokens.Null;
            return (int)Tokens.Null;
\"[^\"]*\" yylval.Token = Tokens.String; yylval.Str =
            yylval.Str; yylval.Data = yytext.Substring(1,
            yytext.Length - 2); return (int)Tokens.String;
```

8.2.2 gplexx.frame

Šablona pro vygenerovanou třídu scanneru aplikací *gplex.exe* přijímající daný regulární jazyk.

8.2.3 gppg.exe

Aplikace generující třídu parseru přijímající danou gramatiku. Gramatika musí být

bezkontextová. Vstupem je soubor WorldSIM\Grammar\Parser.y a výstupem třída WorldSIM\Grammar\Parser.cs. Tato třída je označena jako *partial*, protože v jejích dalších částech jsou definovány prováděné akce pro jednotlivé příkazy – viz. soubory WorldSIM\Grammar\Parser.*.cs.

9 Literatura

[1] **Wikipedia.org.** Ekonomika. *Wikipedie, otevřená encyklopedie.* [Online] http://cs.wikipedia.org/w/index.php?title=Ekonomika&oldid=2521768.

[2] **Tišnovský, Pavel.** Metoda přesouvání prostředního bodu a obrázky plasmy. [Online] http://www.root.cz/clanky/metoda-presouvani-prostredniho-bodu-a-obrazky-plasmy/.

[3] **Anders Hejlsberg, Mads Torgersen.** Overview of C# 3.0. *MSDN.* [Online] Microsoft. http://msdn.microsoft.com/en-us/library/bb308966.aspx.

[4] **Microsoft.** Microsoft Visual Studio: Přehled. *Microsoft.* [Online] Microsoft. http://www.microsoft.com/cze/msdn/produkty/vstudio/default.mspx.

[5] **Mono.** MoMA. *Mono.* [Online] http://mono-project.com/MoMA.

[6] **Wikipedia.org.** Common Itermediate Language. *Wikipedia, the free encyclopedia.* [Online] http://en.wikipedia.org/w/index.php?title=Common_Intermediate_Language&oldid=199559921.

[7] **Technology, Queensland University.** Programming Languages and Systems. [Online] http://plas.fit.qut.edu.au/projects/LanguageProcessingTools.aspx.

[8] **Wikipedia.org.** HSV. *Wikipedia, otevřená encyklopedie.* [Online] http://cs.wikipedia.org/w/index.php?title=HSV&oldid=2345837.

[9] **Wikipedia.org**. LinCity. *Wikipedia, the free encyclopedia.* [Online] http://en.wikipedia.org/w/index.php?title=Lincity&oldid=214976348.

[10] **Wikipedia.org**. Age of Empires. *Wikipedia, the free encyclopedia.* [Online] http://en.wikipedia.org/w/index.php?title=Age_of_Empires_%28video_game%29&oldid=210004642.

[11] **Wikipedia.org**. Transport Tycoon. *Wikipedia, the free encyclopedia.* [Online] http://en.wikipedia.org/w/index.php?title=Transport_Tycoon&oldid=214466088.

[12] **Roeder, Lutz.** [Online] http://www.aisto.com/roeder/dotnet/.

[13] **Wikipedia.org.** Simplex algorithm. *Wikipedia, the free encyclopedia.* [Online] http://en.wikipedia.org/w/index.php?title=Simplex_algorithm&oldid=212069746.

10 Obrázky a tabulky

11 Přílohy na CD

- Aplikace WorldSIM – adresář bin
- Zdrojové kódy aplikace – adresář WorldSIM
- Uživatelská dokumentace – adresář Docs
- Ukázkové skripty – adresář Examples