Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE

Jiří Michalec

Simulace města

Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce: Mgr. Ondřej Zajíček
Studijní program: Informatika, Obecná informatika

2009

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

V Praze dne Jiří Michalec

Obsah

1 Úvod **7**

2 Problematika **8**
2.1 Stavby, jejich rozdělení, vznik a zánik 9
2.2 Generování mapy 10
2.3 Pracovní trh, nákupní místa a školství 11
2.4 Vnitřní trh města 12
2.5 Vliv staveb na okolí 13
2.6 Příchod obyvatel 13
2.7 Modelování dopravy 14

3 Podobné programy **16**
3.1 SimCity 16
3.2 LinCity 16

4 Uživatelská dokumentace **18**
4.1 Instalace a spuštění 18
4.1.1 Instalace 18
4.1.2 Spuštění 19
4.2 Základní princip 19
4.3 Povrchy 19
4.4 Stavby 19
4.4.1 Zóny 20
4.4.2 Továrna 20
4.4.3 Obchod 20
4.4.4 Dům 20
4.4.5 Škola 21
4.4.6 Dopravní stavby 21
4.4.7 Speciální budovy 21

4.5 Hlavní okno programu 21
4.6 Ovládání hry 21
4.6.1 Start nové hry 21
4.6.2 Uložení hry 22
4.6.3 Načtení hry 22
4.6.4 Ukončení programu 22
4.6.5 Zobrazení informací o městě 23
4.6.6 Akce pravého tlačítka myši 23
4.6.7 Změna akce levého tlačítka myši 23
4.6.8 Simulace nového dne 25
4.6.9 Změna zobrazení města 25
4.7 Informační okna 25
4.8 Konfigurace programu 26
4.8.1 Sdílené elementy 26
4.8.2 Hlavní konfigurace programu 27
4.8.3 Definice povrchů 27
4.8.4 Definice staveb 29
4.8.5 Továrna 31
4.8.6 Obchod 32
4.8.7 Dům 32
4.8.8 Škola 32
4.8.9 Dopravní stavba 32
4.8.10 Křížení silnice a železnice 33
4.8.11 Speciální budova 33

5 Programátorská dokumentace 35
5.1 Platforma a použité technologie 35
5.2 Přehled základních modulů 35
5.2.1 Simulační modul 36
5.3 Struktury staveb 37
5.4 Povrchy 38
5.5 Mapa 39
5.6 Správci simulovaných oblastí 39
5.7 Simulační paměť 40
5.8 Vstup/výstup 40
5.9 Grafické uživatelské prostředí 42
5.9.1 Databáze obrázků 42
5.9.2 Mapa, její vykreslení a akce uživatele 43

5.9.3 Grafy 44
5.9.4 Náhled na vygenerovanou mapu 45
5.9.5 Informační okna 45
5.10 Použité postupy a algoritmy 45
5.10.1 Vznik nových budov z vyhrazených zón 45
5.10.2 Simulace pracovního trhu, nákupních míst a školství . 47
5.10.3 Generování map 48

6 Možná rozšíření **49**
6.1 Ekonomický model hráče 49
6.2 Definice herních událostí 50
6.3 Skriptovaná hra 50

7 Závěr **51**

Literatura **52**

A Obsah přiloženého CD **53**

Název práce: Simulace města
Autor: Jiří Michalec
Katedra (ústav): Katedra teoretické informatiky a matematické logiky
Vedoucí bakalářské práce: Mgr. Ondřej Zajíček
e-mail vedoucího: Ondrej.Zajicek@mff.cuni.cz

Abstrakt: Cílem této práce byl návrh a implementace programu simulujícího chod města a příslušných vnitřních ekonomických a ekologických jevů s možností ovlivnit jeho vývoj skrze grafické uživatelské rozhraní. V práci jsou nejprve rozebrány jednotlivé simulované oblasti, možné způsoby jejich realizace, jejich výhody a nevýhody. V dalších částech se práce zabývá popisem konkrétní implementace tohoto problému. Nejprve je popsáno uživatelské rozhraní a konfigurace programu, dále pak reprezentace nejdůležitějších struktur a popis použitých algoritmů. Poslední část práce rozebírá směry možného dalšího vývoje programu a shrnuje výsledek celé práce.
Klíčová slova: Simulace města, Simulační hra, Modelování dopravy

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

Abstract: The aim of this thesis was to design and implement the program simulating the city and applicable inner economic and ecologic phenomena. The user is supposed to affect the city through graphical user interface. In the first part of the thesis there is analysis of all simulated phenomena, possibility of their implementation and analysis of the advantages and the disadvantages of the possible ways. Other parts of the thesis describes the program created in this thesis. The description starts with user related information and program configuration and continues with the specification of the inner representation of the most important data structures and algorithms used in the program. The last part analyses the ways of the possible future extensions of the program and summarize the result of the whole thesis.
Keywords: City simulation, Simulation game, Traffic modelling

Kapitola 1

Úvod

Cílem práce bylo vytvoření programu simulující chod města a souvisejících jevů. K přesné simulaci by bylo třeba zohlednit veškeré činnosti jednotlivých obyvatel s rozdílnými životními cíly, koníčky apod.

Hlavní myšlenka simulace je vytvoření rovnováhy mezi obyvatelstvem, prací a obchody, která je použitá i v sérii SimCity[1], nejznámějším zástupcem programů simulující město. Základními simulovanými oblastmi jsou tedy pracovní trh, vnitřní trh města se zbožím a růst počtu obyvatel. Těmto oblastem odpovídají i základní typy staveb: domy, továrny a obchody.

Zmíněné oblasti jsou provázané – lidé potřebují práci, domov, nákupní místa a zboží, továrny potřebují zaměstnace a odbyt zboží, obchody potřebují zákazníky a zaměstnance. Ve městěch se nenacházejí jen dospělí lidé, ale také děti, které musí chodit do škol.

Každá stavba nějak působí na své okolí a tímto působením může zapříčinit odchod lidí/společností jinam (např. pokud své okolí příliš znečišťuje).

Tyto všechny vztahy je nutné vzít v úvahu pro vytvoření rozumného simulátoru města.

Kapitola 2 se zabývá podrobným rozborem této problematiky, popisem možných postupů a zjedodušení. V kapitole 3 jsou popsány nejznámější programy zabývající se touto problematikou. Kapitola 4 obsahuje popis jak pracovat s výsledným programem, kapitola 5 popisuje strukturu programu, reprezentaci důležitých prvků a použité algoritmy. Kapitola 6 ukazuje další možné směry vývoje programu a v kapitole 7 je shrnut výsledek celé práce.

[1] Další informace o tomto programu jsou v kapitlole 3

Kapitola 2

Problematika

Simulaci lze provádět z časového hlediska dvěma způsoby – *v reálném čase* (tzv. „*realtimově*“) a po tazích. V prvním případě probíhá simulace nepřetržitě, přičemž se jevy zpravidla simulují po malých časových úsecích a výsledek se více blíží realitě. Uživatel může změny průběžně sledovat a při použitý krátkých časových intervalů lze ovlivnit už rozběhnuté činnosti. V případě způsobu simulování celých tahů (delších uzavřených časových úseků, např. celých dnů) uživatel provede veškeré akce a poté se spočítají a provedou změny za celý časový úsek. Uživatel tak může v klidu prozkoumat, jak se simulovaný svět změnil, a dříve zareagovat na případné problémy. Tento způsob má také výhodu, že se snáze implementuje – není zde žádná automatická akce pro start simulace dalšího kroku a díky uzavřenému tahu zde není nutnost udržování stavu rozběhnutých akcí ale rovnou lze spočítat jejich výsledek – a proto je zvolen pro tuto práci.

Ve městě probíhá mnoho provázaných jevů a modelování všech by bylo velice složité. Proto je třeba celou problematiku zjednodušit a vybrat jen některé simulované oblasti. Nejdůležitější pro tuto práci jsou

- pracovní trh
- nákupní místa
- vývoj populace
- vnitřní trh města
- školství
- doprava
- vliv staveb na prostředí

Tyto prvky jsou určitým způsobem provázané, což je třeba při simulaci zohlednit.

2.1 Stavby, jejich rozdělení, vznik a zánik

V každém městě se vyskytuje mnoho různých staveb s různým účelem. Pro zjednodušení celého problému dále předpokládáme, že každá stavba slouží jednomu účelu, což umožňuje lépe definovat vlastnosti jednotlivých staveb a jejich přímočaré rozdělení do typů – množin budov s podobnými vlastnostmi, např. obytné budovy. Druh stavby je pak konkrétní stavba, např. panelák.

Jak bylo zmíněno výše, ze hry *SimCity* bylo převzato vyhrazování obytných, výrobních a obchodních zón uživatelem a přímé budování všech ostatních staveb. Tedy zatímco u domů, továren a obchodů může uživatel pouze určit místa vyhrazená pro příslušný typ budovy a dále nemá přímý vliv na výběr konkrétní budovy na daném poli, u všech ostatních typů staveb přímo určuje, která bude postavena.

Existují stavby zaměřené na výrobu (továrny), obchodování (obchody), bydlení (domy), na přepravu (cesty, železnice) a na výuku (školy). Další využití nemá pro vybrané simulované oblasti význam a budovy nespadající do žádného z předchozích zaměření lze zařadit do jedné skupiny nazvané speciální budovy.

Každá stavba nějak ovlivňuje své okolí – způsobuje znečištění, má vliv na estetiku místa apod.

Jak již bylo zmíněno, některé budovy vznikají bez přímého rozhodnutí uživatele o jejich postavení. V takovém případě je potřeba rozhodnout, kde a která budova se postaví. Je několik možností, jak daný problém řešit. Jedním způsobem je vybírat herní pole pro danou budovu a druhým výběr budovy pro herní pole.

V prvním případě se nejprve určí, kolik budov daného druhu ve městě vznikne, a poté tyto budovy obsadí postupně nejvhodnější pole své zóny. Zde přichází několik problémů: jak rozhodnout o počtu budov a jak řešit nedostatečný počet vhodných polí. Počet budov by se musel odvíjet od vlastnostní souvisejících s daným typem budov (např. nezaměstnanost a prodejnost zboží u továren), při umísťování budov by bylo potřeba vyřešit pořadí druhů.

V druhém případě se pro všechna volná pole zón spočítá pravděpodobnost, že na něm vznikne nová budova a v případě vzniku budovy se rozhodne

o případném druhu. To lze udělat ohodnocením všech druhů budovy příslušného typu, které na zkoumaném herním poli mohou vzniknout, a přidělí se jim pravděpodobnosti v poměru výsledku ohodnocovací funkce.

Výhodou prvního přístupu je větší realističnost, ale mohou být problémy se situacemi, kdy má vzniknout více budov, než je dostupných míst, a dále není přímočaře viditelné, jak by se určil počet nově vzniklých budov daného druhu.

U druhého způsobu se dá určit pravděpodobnost z „žádanosti“ pole, přičemž „žádanost“ závisí pouze na vlastnostech daného pole. To vychází z předpokladu, že větší zájem je o parcely s malým znečištěním, nízkou kriminalitou a dobrou dostupností zdravotní péče než o znečištěné parcely s centrem zločinnosti. Tuto pravděpodobnost je třeba dále upravit podle podmínek ve městě pro danou budovu (např. snížit jí, pokud není dostatek potenciálních pracovníků). Tento způsob byl zvolen pro realizaci v této práci.

Problém určení ohodnocovací funkce pro daný druh budovy je složitější, avšak je nutné ho vyřešit v obou zmíněných postupech. Lze předpokládat, že továrna nemá velkou motivaci pro příchod do města, kde je trh s jejím druhem výrobku již nasycen a kde je nedostatek potenciálních pracovníků, obchod nepůjde příliš ochotně do města s malým počtem zákazníků a lidé se nenastěhují do obce s nedostatkem práce, zboží, obchodů a škol. Dále je možné předpokládat, že větší podnik je dravější než menší.

Z tohoto také vychází myšlenka možného ohodnocení druhů budov – čím více je budova schopna zaměstnat osob, obsloužit zákazníků, ubytovat lidí a vyrobit zboží, tím pravděpodobnější je její vznik. Ale pokud v některé z oblastí k budově relevantních nemůže uspokojit své minimální požadavky, pravděpodobnost se výrazně snižuje.

2.2 Generování mapy

Budovy nelze postavit všude a proto je třeba při jejich vzniku zohlednit i povrch, na kterém budou stát. Každá budova může mít jiná omezení a proto se musí evidovat povolení povrchy u každého druhu zvlášť.

Před začátkem simulace je nutné určit rozložení povrchů na mapě. Pro program byl zvolen postup používající tzv. metodu přesouvání středního bodu uvedenou v [3], kterou se generují 3D povrchy. Touto metodou se vygeneruje výšková mapa připomínající realistický povrch. Následně jsou určeny rozsahy výšek odpovídající jednotlivým povrchům a výšková mapa je na ně převedena.

Jiným postupem by mohlo být určení náhodných míst pro počáteční herní pole jednotlivých povrchů a od nich by se prohledáváním do šířky rozšiřovaly, dokud by nebyla všechna místa na mapě obsazena nějakým povrchem.

První zmíněná metoda potřebuje oproti druhé zmíněné určit výškové pořadí terénů, což trochu komplikuje konfiguraci. Na druhou stranu ale nevznikají nelogické situace, kdy by sousedila oblast povrchů běžných ve velice malých nadmořských výškách s oblastí povrchů vyskytujících se naopak ve vysokohorských krajinách.

2.3 Pracovní trh, nákupní místa a školství

Dospělí chodí do práce a nakupovat, děti zase do škol. U pracovních míst a škol je situace jasná – každý má přesně určeno, kde pracuje, a dané místo se zpravidla nemění. U obchodů je situace trochu jiná, protože nikdo napevno neříká, kam bude chodit nakupovat, a může jít každý den jinam, pokud se mu to hodí. Typicky ale lidé nakupují v jednom obchodu, do ostatních se vydají méně často.

Z toho také vychází zjednodušení nákupní problematiky – každý si vybere jeden obchod, do kterého bude chodit, a cesty do ostatních se zanedbávají.

Po takovémto zjednodušení nákupní problematiky jsou oblasti pracovních, nákupních a studijních míst principem stejné. Každá osoba si musí najít místo v dané oblasti (dospělí pracovní místo a nákupní místo, děti místo studijní) a každá budova má jen omezenou kapacitu.

Budovy také potřebují nějaké pracovníky/zákazníky pro svou činnost. Proto je nutné vědět, kolik pracovníků musí zaměstnavatel mít, aby mohl existovat a kolik zákazníků potřebuje obchod, aby se uživil[1].

Každá osoba si tedy hledá práci, obchod a dítě školu. Hlavním kritériem pro zvolení daného místa je jeho vzdálenost od bydliště. To vychází z předpokladu, že je příjemnější bydlet blízko práce než daleko dojíždět. V pracovní problematice by bylo také možné nějakým způsobem evidovat platy na daných místech, ale konfigurace zaměstnavatelů by se stala složitou a vznikly by také větší nároky na paměť.

Dlouhodobě nezaměstnaní z města odcházejí s celou svou rodinou, stejně tak lidé, kteří nemají po delší časový úsek kde nakupovat, a rodiny s dětmi

[1] toto by se dalo modelovat přesněji evidencí prodaného zboží v obchodě, důvody tohoto zjednodušení jsou v podkapitole 2.4

bez místa ve škole.

2.4 Vnitřní trh města

Vnitřní trh města je zjednodušen zanedbáním vnějších zdrojů produktů a tedy co se nevyrobí ve městě, to není dostupné. Továrny vyrábí produkty a ty je třeba distribuovat mezi obyvatele.

K tomu je potřeba určit zájem o jednotlivé komodity. Jsou výrobky, které člověk potřebuje několikrát za život (a někteří si je nepořídí vůbec) a výrobky denně nutné k přežití. Zájem o druhou skupinu lze vyřešit jednoduše – nastaví se denní potřeba zboží pro jednoho obyvatele a všichni nakupující si toto množství musí pořídit. První skupina je ale problematická. Muselo by se určit, kdo si kdy dané zboží pořídí. Je zde ale i jiný možný přístup – jednotka daného výrobku nebude výrobek celý, ale jen jeho část, kterou si každý člověk koupí. Tímto se obě skupiny spojí do stejného přístupu a při dobře nastaveném poměru zboží výsledná potřeba může odpovídat skutečnosti.

Po takovémto zjednodušení je třeba rozhodnout o distribuci zboží k jednotlivým nakupujícím. Ve skutečném světě má každá továrna svůj sklad, každý obchod má svůj sklad a zboží vykoná dvě cesty – z továrny do obchodu a z obchodu k zákazníkovi. Modelování tohoto přístupu by znamenalo udržování skladu jak v obchodě tak v továrně. Zároveň by se musela vyřešit strategie plnění skladů obchodů. Celý problém je tedy opět zjednodušen a to tím, že zboží je „skladováno“ pouze v továrnách a je dostupné ve všech obchodech, do kterých z příslušné továrny existuje dopravní spojení.

Evidence prodaného zboží také probíhá v továrnách i v obchodech – oba dva subjekty musí prodat dost zboží pro svou další existenci. V případě obchodů je celý problém o to složitější, že každý druh zboží má jinou hodnotu a prodané zboží by se muselo přepočítat na nějakou městskou měnu. Z tohoto důvodu a z přístupu stejné potřeby zboží pro každého obyvatele je problematika opět zjednodušena na předpoklad, že pokud má obchod dostatek zákazníků, uživí se, a je tedy nutné evidovat množství prodaného zboží jen u továren.

Lidé, na které dlouhodobě nevyzbyde zboží, z města odcházejí s celou svou rodinou.

2.5 Vliv staveb na okolí

Každá stavba nějak mění své okolí a intenzita vlivu je různá pro různé budovy. Za nejdůležitější oblasti bylo vybráno znečištění, kriminalita, estetika, nebezpečí vzniku požáru a dostupnost zdravotní péče. Každá budova má stanoveno, jak ovlivňuje jednotlivé vlastnosti u herních polí podle jejich vzdálenosti od dané budovy.

Hodnota těchto vlastností je důležitá pro stanovení pravděpodobnosti vzniku nových budov na tomto herním poli.

2.6 Příchod obyvatel

Do města přicházejí lidé se vznikem domů a jsou seskupováni do rodin. Tento přístup přináší tu výhodu, že každá osoba ve hře má jasně dané souřadnice domova a tudíž se snáze určují koncové body při přepravě, ale je to vykoupeno nemožností vzniku bezdomovců.

Každá nově vzniklá obytná budova je naplněna rodinami tak, aby v ní bydlel aspoň minimální povolený počet obyvatel pro danou budovu, a zároveň aby stačila obytná kapacita budovy – počet v tomto rozmezí se určí náhodně. Poté jsou vytvářeny jednotlivé rodiny (určení počtu členů jsou věnovány následující odstavce).

Určení počtu dospělých a dětí v jednotlivých rodinách vychází z rozložení jednotlivých typů domácností v České republice, zjištěných při sčítání lidu v roce 2001. Jak je uvedeno v [10], bylo v České republice v roce 2001 30 % domácností jednotlivců, 2 % nerodinných domácností, 34 % rodin bez závislých dětí a 34 % rodin se závislými dětmi, z nichž 76 % bylo úplných a 24 % neúplných. Tabulka 2.1 ukazuje počty dětí v úplných a neúplných rodinách.

Počet dětí	1	2	3	4+
Úplné rodiny	43	47	8	2
Neúplné rodiny	65	30	4,5	0,5

Tabulka 2.1: Počty dětí v rodinách v procentech (zaokrouhleno na půlprocenta)

Pro účely programu jsou zanedbány rodiny s více než čtyřmi dětmi a počet osob v nerodinných domácnostech byl odhadnut bez reálných dat. Celkové statistické rozdělení různých typů rodin ve hře je udáno v tabulce 2.2.

Jednočlenné domácnosti a úplné rodiny					
Počet dospělých	1	2	2	2	2
Počet dětí	0	1	2	3	4
Procentuální zastoupení	30	11,11	12,14	2,07	0,52
Neúplné rodiny					
Počet dospělých	1	1	1	1	2
Počet dětí	1	2	3	4	0
Procentuální zastoupení	5,3	2,45	0,37	0,04	34
Nerodinné domácnosti					
Počet dospělých	2	3	4	5	6
Počet dětí	0	0	0	0	0
Procentuální zastoupení	1	0,5	0,25	0,13	0,12

Tabulka 2.2: Počty členů rodiny v procentech

2.7 Modelování dopravy

K simulaci dopravy je třeba určit počáteční a cílová pole tras jednotlivých obyvatel. Každý dospělý jedinec se přepravuje z domova do práce a z domova do obchodu, další cesty osob po městě se zanedbávají.

Stavby v programu lze pro účely simulace dopravy rozdělit na *dopravní* a *nedopravní*. Dopravní stavby jsou silnice, železnice a jejich křížení, tedy takové stavby, po kterých se mohou přepravovat osoby, nedopravní jsou všechny ostatní.

Lidé se přepravují po „nejkratších“ cestách, kde délka cesty nezávisí jen na počtu polí, ale i na hustotě provozu. K výpočtu těchto cest je použit Floydův algoritmus uvedený v [1], modifikovaný pro neorientované grafy tak, že výpočet hodnot probíhá jen pro indexy $i < j$ a symetrická hodnota je zapsána zároveň, a při aktualizaci hodnoty je pro rekonstrukci nejkratších cest navíc uložen index vrcholu, přes nějž nejlepší nalezená cesta vede.

Hustota provozu přes danou dopravní stavbu za jeden den je určena jako

$$H = \frac{P * 100}{D}$$

kde P je počet osob přepravených za předchozí den a D je maximální počet osob na jejichž přepravu byla budova navržena. Při takto definované hustotě ovšem docházelo při několika paralelních cestách k „přelévání“ osob z jedné na druhou. Důvodem je, že první den se po jedné cestě přepravuje mnoho osob, a hustota je tedy veliká, a po druhé málo. Při výpočtu cest další

den většina obyvatel použije z důvodu menší hustoty provozu druhou cestu místo první a role cest se obrátí. Proto byla místo hustoty cesty za jeden den použita průměrná hustota za několik posledních dní.

Vrcholy grafu pro výpočet nejkratších cest jsou jednotlivá pole cesty, váha hrany pro přepravu na dvě sousední pole je určena

$$V = H + C_p + C_z$$

přičemž C_p je cena za přechod na sousední pole a C_z cena změny dopravního prostředku – ta se bere v úvahu jen při přechodu ze silniční na železniční způsob dopravy. Obě hodnoty jsou nastavitelné v konfiguraci programu (viz kapitola 4.8).

Po změně sítě dopravních staveb (postavení či zbourání dopravní stavby) by mohlo dojít v několika prvních dnech k výrazným oscilacím v rozložení cestujících kvůli datům historie před změnou, která ale nemusí odpovídat současnému stavu cest (nově přidané cesty jsou výrazně zvýhodněné před stávajícími). Proto se první den po změně dopravní sítě vynuluje historie a provádí se několik iterací výpočtu rozložení cestujících, přičemž výsledek po každé iteraci se ukládá do historie přepravovaných osob.

Takto se vypočítají jen nejkratší cesty mezi jednotlivými dopravními stavbami. Lidé se ovšem přepravují ze svých domovů do jiných nedopravních staveb a proto je nutné určit nástupní a výstupní pole. V okolí počáteční a cílové stavby o poloměru určeném jako maximální pěší vzdálenost (viz kapitola 4.8) se vyhledají všechny dopravní stavby a prozkoumají se dvojice, kdy jedna stavba je z okolí nástupního pole a druhá z okolí pole výstupního. Z možných dvojic nalezených staveb se bere v úvahu ta, která má nejmenší součet pěších vzdáleností, při rovnosti pak ta dvojice staveb, mezi nimiž vede nejkratší cesta.

Kapitola 3

Podobné programy

Tato kapitola pojednává o programech s podobným zaměřením. Podrobněji zmíněny jsou dva nejznámější se stejnou tématikou, přestože lze nalézt i další.

3.1 SimCity

Série SimCity je snad nejznámější skupinou programů simulující město. Základním principem je rovnováha mezi třemi skupinami budov – obytnými, obchodními a průmyslovými. Tyto budovy se staví automaticky na místech k tomu určeným (zóna obchodní (commercial), průmyslová (industrial) a obytná (residental)), od druhého dílu série (SimCity 2000) je každý z typů zón rozdělen na „lehký“ (light) a „hustý“ (dense). Třetí díl přidává nové povrchy ([8]), ovšem v celém městě se vyskytuje vždy jen jeden povrch a voda.

Oproti této sérii je program vytvořený v této práci více konfigurovatelný. Celá tato série simuluje oproti této práci navíc energetiku, ekonomickou situaci hráče, přírodní katastrofy, od druhého dílu i vodárenství. V prvních dvou dílech není možné nijak ovlivnit velikost simulované plochy. Dalším rozdílem oproti této práci je používání pouze jednoho druhu povrchu.

3.2 LinCity

Hra LinCity ([6]) vznikala od roku 1995, poslední verze je z roku 2004 a dále se již nevyvíjí. Navazuje na ní projekt LinCity-NG ([7]), který především

přepracoval grafiku a zvuky. Program je vytvořený původně pro operační systém GNU/Linux, později byl předělán do multiplatformní podoby.

Uživatel může postavit obytná sídla tří druhů lišící se počtem obyvatel, porodností a úmrtností. Ve městě se simuluje množství určitých výrobků - jídlo, práce, uhlí, zboží, ruda a železo. Dále je zde ukazatel pro technologickou úroveň, která ovlivňuje, které stavby je schopen uživatel postavit. Pracovní trh je zde simulován pomocí počtu žadatelů o práci. Z povrchů mapy jsou zastoupeny pouze dva – tráva a voda – přičemž na vodě nelze stavět (s výjimkou elektrického vedení přes jedno pole s vodou).

Program vytvořený v této práci má odlišný přístup ke vzniku nových budov – uživatel nemůže u výrobních či obchodních přímo určit, která konkrétní budova vznikne na daném poli, jen určí účel. To více odpovídá situacím ve městech, kdy se v územním plánu nerozhoduje o konkrétních budovách, ale o účelu. Oproti LinCity je v programu také možnost definování více povrchů a podmínek pro vznik budov na daném poli, generování map různých rozměrů a lepší konfigurovatelnost – možné nastavení různého druhu zboží, přidávání nových budov a jejich vlastností a nastavení různých parametrů simulace.

Program LinCity také simuluje jiným způsobem vliv znečištění na obyvatelstvo. Místo konceptu zvoleného v této práci – zmizení celé budovy na základě podmínek na vlastnosti herního pole – je použito zvýšení úmrtnosti. To je dáno i jiným přístupem k budovám, kdy v LinCity všechny domy stojí a neničí se. Také navíc simuluje energetiku stavěním elektráren a elektrického vedení, některé budovy jsou přístupné až po dosažení určité technologické úrovně apod. Jiný postup je také u distribuce zboží, kdy se centrem obchodu stává trh s určitým dosahem a zboží, které má na skladě, je přístupné všem obyvatelům v jeho dosahu.

Kapitola 4

Uživatelská dokumentace

4.1 Instalace a spuštění

4.1.1 Instalace

Program se zkompiluje ze zdrojových kódů příkazem *make* v hlavním adresáři programu. Pro kompilaci je nutná přítomnost knihoven GTK+-2.0 ve verzi 2.6.12, kompilátory jazyků C a C++ a . V případě použití jiných kompilátorů než *g++* a *gcc* je třeba upravit makra *CXX* (kompilátor C++), *CC* (komilátor C), *CXX_FLAGS* (parametry kompilátoru C++) a *CFLAGS* (parametry kompilátoru C) v adresářích *src*, *src/gui* a *src/io*. Jako výchozí linker je použit *g++*, v případě použití jiného linkeru je nutné změnit hodnotu makra *LD* v *src/Makefile*.

Referenční dokumentaci lze vygenerovat příkazem *make doc*. Dokumentace je generována pomocí programu *Doxygen*[1]

Ke smazání souborů vzniklých při kompilaci dojde po zadání příkazu *make clean*, příkaz *make distclean* navíc smaže i výsledný spustitelný soubor a vygenerovanou dokumentaci.

Po kompilace složí pro instalaci příkaz *make install*, který ve výchozím nastavení zkopíruje všechny soubory potřebné pro běh programu do nově vytvořeného adresáře *program* – pro změnu instalačního adresáře je třeba upravit makro *INSTALL_DIR* v *Makefile* hlavního adresáře programu. Odinstalování lze provést smazáním instalačního adresáře, což lze též učinit příkazem *make uninstall*.

[1] *http://www.doxygen.org*

4.1.2 Spuštění

Program se spustí spuštěním souboru *simul_city* v instalačním adresáři.

4.2 Základní princip

Základním principen je vytváření rovnováhy mezi obyvateli, prací, obchody a zbožím. Hráč určí pole pro stavbu továren, obchodů a domů, postaví dopravní infrastrukturu a další budovy, jako např. policejní stanici. Po provedení všech akcí ukončí tah, čímž se provede simulace herního dne. Během něj vzniknou či zaniknou továrny, domy a obchody, ovlivní se kriminalita, znečištění, estetika, nebezpečí vzniku požáru a dostupnost zdravotní péče na jednotlivých polích, někteří lidé si najdou či změní práci a obchod, dospělí půjdou do práce a nakoupit, zatímco děti se vydají do škol.

Vznik a zánik budov závisí na vlastnostech daného políčka (s nižší estetikou a dostupností zdravotní péče a vyšším znečištěním, kriminalitou a nebezpečím vzniku požáru klesá pravěpodobnost, že na daném poli něco vznikne) a faktorech pro daný typ budovy. U továren a obchodů se pravděpodobnost vzniku snižuje s dobrou zaměstnaností ve městě. V případě továrny dále klesá s dobrou dostupností zboží (není dost prostoru na trhu města), u obchodu s dostatkem míst v obchodech. V případě domů se pravděpodobnost snižuje s nízkým počtem pracovních míst, nízkým počtem nákupních míst, malou kapacitou škol a nedostatek zboží ve městě.

4.3 Povrchy

Povrchy jsou uspořádány v hierarchii tříd povrchů. Každá třída buď obsahuje povrchy nebo podtřídy.

Definice povrchů se nachází v souboru *surfaces.data* – viz kapitola 4.8 na straně 26

4.4 Stavby

Včechny stavby, které lze postavit, ovlivňují vlastnosti okolních polí - *znečištění*, *nebezpečí požáru*, *estetiku*, *kriminalitu* a *dostupnost zdravotní péče*. Stavba může mít zadány povolené a zakázané povrchy – lze ji postavit na

poli, jehož povrch nebo třída, jíž je daný povrch členem, je v seznamu povolených povrchů a zároveň jehož povrch nebo třída, jíž je členem, není v seznamu zakázaných povrchů. Stavby tvořící se v příslušných zónách mají dané podmínky pro svou existenci závisející na vlastnostech pole, na kterém stojí a při nesplnění těchto podmínek po několik po sobě jdoucích dnů zanikají.

4.4.1 Zóny

Zóna je pole rezervované pro určité budovy. Ve hře jsou tři druhy – výrobní pro továrny, obchodní pro obchody a obytné pro domy. Továrny, obchody a domy nelze přímo postavit, jediná cesta jejich vzniku je právě postavením příslušné zóny. Budovy se poté staví/zanikají automaticky na základě stavu města.

4.4.2 Továrna

Továrna zaměstnává obyvatele města a vyrábí zboží. Každá továrna má daný minimální počet pracovníků nutný pro její chod, maximální počet pracovníků, který v ní může pracovat, množství vyrobeného zboží za den a množství zboží, které musí za den prodat. Továrna zaniká, pokud po delší dobu neprodá dostatek zboží nebo nemá dostatek pracovníků.

4.4.3 Obchod

Obchod zaměstnává obyvatele a distribuuje vyrobené zboží mezi obyvatele města. Pro každý obchod je stanoven nejmenší a největší počet pracovníků a zákazníků. Obchod zaniká, když nemá delší dobu dostatek zákazníků nebo pracovníků.

4.4.4 Dům

V domech bydlí lidé. Každý dům má dán nejmenší a největší možný počet obyvatel. Pokud v domě po odchodu lidí z města zbyde méně než nejmenší povolený počet obyvatel, dům zaniká.

4.4.5 Škola

Každé dítě studuje v nějaké škole. Škola má danou kapacitu studentů, které je shopna vyučovat. Školy svévolně nezanikají.

4.4.6 Dopravní stavby

Dopravní stavba je taková stavba, přes kterou se mohou lidé přepravovat – např. cesty, železnice či dálnice. Má definován největší počet cestujících, na který je navržena.

Dopravní stavby jsou dle způsobu přepravy rozděleny na *silnice* a *železnice*, přičemž je možné na místě s již postavenou silnicí umístit železnici a obráceně. Na takovém poli vznikne křížení silnice a železnice a je dále v programu považováno za jednu stavbu.

4.4.7 Speciální budovy

Speciální budovy nemají žádné zvláštní vlastnosti, jen ovlivňují vlastnosti okolních polí – policejní stanice snižuje kriminalitu, hasičská stanice snižuje nebezpečí vznilu požáru, fontána zvyšuje estetiku, ...

4.5 Hlavní okno programu

Hlavní okno programu obsahuje ve své horní části menu programu. Zbytek okna je rozdělen na levý ovládací sloupec, mapu města a spodní informační část.

V levém sloupci je umístěno (zhora dolů) pole s popisem akce levého tlačítka myši, výběr akcí levého tlačítka, výběr pohledu na město a tlačítko pro simulaci nové hry.

4.6 Ovládání hry

4.6.1 Start nové hry

Okno pro zahájení nové hry se zobrazí po výběru *Game – New* z hlavního menu programu. V horní části okna je náhled na vygenerovanou mapu a tlačítka pro změnu barev jednotlivých povrchů, ve spodní části ovládací prvky ovlivňující vygenerovanou mapu. Po stisknutí tlačítka *OK* začne nová

hra s mapou zobrazenou v náhledu, tlačítkem *Cancel* se vytváření nové hry zruší a pokračuje se ve hře rozehrané.

Generování nové mapy

V poli *Map properties* je třeba nejprve určit velikost mapy nastavením pole *Map size* na požadovanou hodnotu. Po stisku tlačítka *Generate* bude vytvořena čtvercová mapa o rozměrech *velikost_mapy* x *velikost_mapy* (např. při velikosti mapy 30 se vytvoří mapa o rozměrech 30x30 polí).

Změna rozložení povrchů na mapě

Každé pole na nově vygenerované mapě má svou výšku. V části *Surface levels* okna nové hry jsou definovány hraniční výšky pro všechny povrchy – povrch je na všech polích, jejichž výška je větší než hraniční výška předchozího povrchu a menší nebo rovna své hraniční výšce.

Zvýšením hodnoty v řádce se jménem povrchu (a tedy změnou hraniční výšky daného povrchu) vznikne na mapě více polí s daným povrchem a ubyde polí s povrchem na následující řádce (nebo zbývajícím povrchem, pokud se upravuje poslední řádka). Naopak snížením hodnoty příslušného povrchu ubyde a následujícího přibyde.

4.6.2 Uložení hry

Po výběru *Game – Save* z hlavního menu je zobrazen dialog pro výběr souboru, kam hru uložit. Stiskem tlačítka *Save* dojde k uložení hry do vybraného souboru – pokud již soubor existuje, program se zeptá, má-li být přepsán.

4.6.3 Načtení hry

Po výběru *Game – Load* z hlavního menu je zobrazen dialog pro výběr souboru, ve kterém je hra uložena. Stiskem tlačnítka *Load* dojde k načtení hry.

4.6.4 Ukončení programu

Program lze ukončit zavřením hlavního okna nebo výběrem *Game – Close* z hlavního menu programu.

4.6.5 Zobrazení informací o městě

Okno s informacemi o městě lze zobrazit pomocí *Information – City information* v hlavní nabídce programu.

4.6.6 Akce pravého tlačítka myši

Po uvolnění pravého tlačítka myši se posune mapa města tak, aby bylo pole pod myší v jejím středu, je-li to možné.

4.6.7 Změna akce levého tlačítka myši

Akce vykonávaná po uvolnění levého tlačítka myši nad mapou města se změní stiskem talčítka příslušejícího požadované akci v části *Actions* levého sloupce. V poli *Current action* se objeví popis vybrané akce. Akce se provádějí jen tehdy, mají-li smysl (stavba vznikne, pokud na daném políčku nestojí jiná (s níže popsanou výjimkou u dopravních staveb), . . .)

Akce jsou řazeny do kategorií:

Zones - rezervace zón

Schools - stavění školy

Transport facilities Budování dopravních staveb (viz podkapitola 4.4.6) – každé tlačítko odpovídá jednomu druhu

Special buildings Stavba speciálních budov (viz podkapitola 4.4.7) – každé tlačítko odpovídá jednomu druhu

Surfaces Změna povrchu na daném poli

Další dvě tlačítka mimo zmíněné kategorie jsou *Destroy building* pro zbourání postavené stavby/zóny a *Field informations* pro zobrazení podrobných informací o políčku a stavbě na něm stojící.

Rezervace zón

Jsou zde tři zóny: *industrial* (výrobní), *commercial* (obchodní) a *residental* (obytná). Po výběru zóny se uvolněním levého tlačítka nad mapou města rezervuje políčko pod myší pro příslušný typ budovy (továrny ve výrobní, obchody v obchodní a domy v obytné zóně).

Stavba škol, dopravních staveb a speciálních budov

Každé tlačítko v příslušné kategorii odpovídá jednomu druhu stavby. Budovu lze postavit jen na políčkách s vyhovujícím povrchem (viz kapitola 4.3, strana 19).

Na poli pod myší se po uvolnění levého tlačítka myši postaví příslušná stavba.

V případě dopravních staveb lze na jednom poli postavit dvě stavby – jedna musí být silnice a druhá železnice. Takto vznikne křížení silnice s železnicí. Silnici lze na poli s železnicí postavit právě tehdy, když na daném poli netvoří železnice zatáčku či křižovatku (nesmí mít dva sousedy s železnicí v na sobě kolmých směrech) a zároveň nevznikne trojice sousedních křížení která tvoří pravý úhel. Symetrická podmínka platí pro stavbu železnice na poli se silnicí. Například pokud tedy pole s železnicí sousedí ze severu a západu s železnicemi, nelze na něm postavit silnici. Stejně tak není možné postavit železnici na pole se silnicí, pokud se na jižním a jihovýchodním sousedním poli kříží silnice se železnicí.

Změna povrchu

Každé tlačítko v této kategorii zastupuje jeden povrch. Po výběru akce způsobí uvolnění levého tlačítka myši změnu povrchu políčka pod myší na vybraný. Akce se provede jen tehdy, pokud na daném poli nestojí stavba, která nesmí být na vybraném povrchu postavena.

Bourání staveb

Po uvolnění levého tlačítka myši nad mapou bude zbourána stavba či odstraněna zóna na políčku pod myší, pokud tam nějaká stojí.

Zobrazení informace o políčku

Po uvolnění levého tlačítka myši nad mapou bude zobrazeno okno s informacemi o poli pod myší a o stavbě na daném poli, pokud tam nějaká stojí. Stejné okno se také zobrazí po uvolnění pravého tlačítka myši.

Popis okna je v kapitole 4.7 na straně 25.

4.6.8 Simulace nového dne

Simulace jednoho dne se provede po stisku tlačítka *Simulate next day* v levém sloupci hlavního okna.

4.6.9 Změna zobrazení města

V části *City view* levého sloupce lze změnit způsob zobrazování jednotlivých polí. Výchozí způsob je *Common view*, který zobrazuje povrch a stavby. Další způsoby zobrazují jednotlivé vlastnosti polí: *Pollution* znečištění, *Fire danger* nebezpečí vzniku požáru, *Aesthetics* estetiku, *Health care* dostupnost zdravotní péče a *Crime* kriminalitu. Posledním je *Traffic density*, který zobrazuje počet přepravených osob přes dané pole, stojí-li na něm dopravní stavba.

U všech pohledů kromě výchozího znamená zelená barva kladné hodnoty, modrá barva záporné hodnoty, bílá barva nulovou hodnotu. Čím tmavší je barva pole, tím dále od 0 je hodnota dané vlastnosti na příslušném poli.

4.7 Informační okna

Informační okna jsou rozdělena do dvou částí. Vlevo je textová a vpravo grafová část.

Textová část obsahuje přehled různých základních veličin (např. míra znečištění u políčka, počet obyvatel u domu, ...). Grafová část obsahuje graf a nad ním rozbalovací seznam zobrazitelných vlastností. Z nich je třeba vybrat, co se má v grafu zobrazit – po změně vybraného prvku seznamu se daná vlastnost zobrazí v grafu. Nejaktuálnější hodnoty jsou v pravé části grafu, nejstarší v levé části.

Okno se zavře po stisknutí tlačítka *OK*.

Okno s informacemi o políčku

Toto okno se zobrazí při uvolnění levého tlačítka myši, pokud je vybraná akce *Field informations*. Okno vždy obsahuje část s informacemi o daném políčku a pokud na něm stojí nějaká stavba, pak i část s informacemi o ní. Tyto části se přepínají pomocí záložek v horní části okna – *Field* pro informace o políčku a *Building* pro informace o stavbě.

Struktura obou částí se shoduje s výše popsaným rozvržením okna s výjimkou zobrazení informací o speciálních budovách – ty obsahují jen jméno druhu budovy.

Městské informační okno

Toto okno zobrazuje informace o celém městě, jako je počet obyvatel, počet volných/obsazených pracovních míst, počet nezaměstnaných, ... V grafu je navíc přehled výroby a spotřeby zboží ve městě a informace o školství.

4.8 Konfigurace programu

Formát konfiguračních souborů vychází ze značkovacího jazyka XML [2], je použita podmnožina tohoto jazyka. V souborech je možné používat počáteční a ukončovací tagy, zkrácenou formu zápisu prázdných elementů a komentáře. Počáteční tag a zkrácený zápis prázdných elementů mohou mít nastaveny atributy.

4.8.1 Sdílené elementy

Zde popsané elementy jsou použity ve více souborech.

image

Element *image* obsahuje text se jménem souboru obrázku. V příkladu 4.1 je definován obrázek ze souboru obrazek.png.

```
<image>obrazek.png</image>
```

Příklad 4.1: Element image

goods

Element *goods* popisuje množství jednoho druhu zboží. Obsahuje povinný atribut *name* udávající jméno druhu zboží, obsahem elementu je číslo definující počet kusů daného zboží – nezáporné celé číslo. Příklad 4.2 ukazuje, jak napsat 15 kusů jídla (food).

```
<goods name='food'>15</goods>
```

Příklad 4.2: Element goods

4.8.2 Hlavní konfigurace programu

Hlavní konfigurace programu je uložena v souboru *config.data* v instalačním adresáři programu. Kořenovým elementem tohoto souboru je element *config*. Tabulka 4.1 obsahuje seznam, popis a výchozí hodnotu všech povolených konfiguračních elementů. Vzdálenost pro pěší přepravu je rovna počtu kroků se zakázaným diagonálním pohybem (tj. $|(x_1, y_1), (x_2, y_2)| = |x_2 - x_1| + |y_2 - y_1|$).

Element *personal_request* obsahuje libovolný počet elementů *goods* (viz sekce 4.8.1) popisující množství daného zboží, které potřebuje jedna osoba během jednoho dne – při vynechání elementu nemají lidé žádné potřeby. Při vícenásobném použití jednoho zboží se bere v úvahu poslední údaj. Ostatní přípustné elementy obsahují celé kladné nenulové číslo.

V příkladu 4.3 je konfigurační soubor, který mění periodu hledání práce bližší domovu na 7 dní a nastavuje denní požadavek člověka na 1 kus jídla.

```
<config>
     <change_job_time>7</change_job_time>
     <personal_request>
          <goods name='food'>1</goods>
     </personal_request>
</config>
```

Příklad 4.3: Možná konfigurace programu

4.8.3 Definice povrchů

Povrchy jsou definovány v souboru *surfaces.data* umístěném v instalačním adresáři. Kořenovým elementem souboru je element *surfaces*, který může obsahovat elementy *surface_class* a *surface_hierarchy*. Tyto elementy zastupují třídy povrchů a mají povinný atribut *name* udávající jméno třídy.

Element *surface_hierarchy* zastupuje hierarchii tříd povrchů a smí obsahovat elementy *sufrace_class* a *surface_hierarchy*.

Element *surface_class* zastupuje dále nedělitelnou skupinu povrchů a smí obsahovat jen element *surface*. Ten musí obsahovat obrázek povrchu v elementu *image* (viz sekce 4.8.1) a jméno povrchu v elementu *name*.

Element	Popis	Výchozí hodnota
change_job_time	Počet dnů, po kterých zaměstnaný člověk hledá práci blíže domovu	10
change_shop_time	Počet dnů, po kterých člověk hledá obchod blíže bydlišti	10
cust_find_limit	Počet dnů, ve kterých obchod hledá zákazníky	7
empl_find_limit	Počet dnů, ve kterých obchod či továrna hledají pracovníky	7
job_find_limit	Počet dnů, které člověk vydrží být nezaměstnaným	7
shop_find_limit	Počet dnů, které člověk vydrží bez obchodu	7
max_selling_time	Počet dnů, ve kterých může továrna prodat méně než nutné množství zboží	5
max_distance	Největší vzdálenost pro pěší přepravu	2
field_price	Obtížnost cesty na sousední dopravní pole	5
change_price	Obtížnost změny způsobu dopravy z/na železnici	10
max_cond_waiting_time	Čas, který stavba vydrží stát při porušení podmínek na vlastnosti pole	7
personal_request	Denní požadavek zboží na 1 obyvatele	

Tabulka 4.1: Elementy hlavního konfiguračního souboru

Nejdříve definované povrchy jsou z hlediska generátoru náhodných map brány jako nejnižší, nejpozději definované jako nejvyšší.

V příkladu 4.4 je možné nastavení použitých povrchů. Nejnižším povrchem je *water*, nejvyšším *mountain*. Třída *ground_surfaces* obsahuje všechny povrchy kromě *water*, *low_ground_surfaces* povrch *grass*.

```
<surfaces>
     <surface_class name='water_surfaces'>
          <surface>
               <name>water</name>
               <image>water.png</image>
          </surface>
     </surface_class>
     <surface_hierarchy name='ground_surfaces'>
          <surface_class name='low_ground_surfaces'>
               <surface>
                    <name>grass</name>
                    <image>grass.png</image>
               </surface>
          <surface_class name='low_ground_surfaces'>
          <surface_class name='high_ground_surfaces'>
               <surface>
                    <name>mountain</name>
                    <image>mountain.png</image>
               </surface>
          <surface_class name='high_ground_surfaces'>
     </surface_hierarchy>
</surfaces>
```

Příklad 4.4: Definice povrchů

4.8.4 Definice staveb

Nastavení existujících staveb se nachází v souboru *buildings.data* s kořenovým elementem *buildings*, který obsahuje popis všech staveb ve hře. Každá stavba s výjimkou křížení cesty a železnice má povinný atribut *name* udávající jméno druhu budovy a musí obsahovat element *image* s jejím obrázkem (viz podkapitola 4.8.1). Dále může mít zadány vlivy budovy na okolí a přípustné povrchy. Jednotlivé budovy mají další povinné a možné vlastnosti, které budou popsány u popisu konfigurace příslušných budov.

Povolené a zakázané povrchy

Každá budova může mít nastavené povrchy, na kterých smí být postavena, určením povolených a zakázaných povrchů. Budova bude postavena, pokud povrch daného pole nebo jeho předek v povrchové hierarchii je mezi povolenými a zároveň povrch nebo jeho předek není mezi zakázanými. Povolené povrchy se určují elementem *allowed_surfaces*, zakázané *banned_surfaces*. Obsah obou elementů je stejný, obsahují jména jednotlivých povrchů v elementu *surface* nebo jména tříd povrchů v elementu *surface_class*, jejich obsahem je jméno povrchu, resp. třídy povrchů.

Je-li seznam povolených povrchů vynechán či prázdný, jsou povoleny všechny povrchy, při neuvedení zakázaných povrchů není žádný povrch zakázán.

Příklad 4.5 ukazuje, jak při definici povrchové hierarchie podle příkladu 4.4 nastavit, že daná budova smí být postavena všude kromě vody a písku.

```
<allowed_surfaces>
     <surface_class>ground_surfaces</surface_class>
</allowed_surfaces>
<banned_surfaces><surface>sand</surface></banned_surfaces>
```

Příklad 4.5: Omezení povrchů u budovy

Vlivy budovy na okolí

Budova ovlivňuje okolní pole v pěti vlastnostech: znečištění, estetice, kriminalitě, nebezpečí vzniku požáru a dostupnosti zdravotní péče. Vliv na danou vlastnost se určuje elementem příslušející dané vlastnosti (jméno elementu viz tabulka 4.2), který obsahuje celočíselnou hodnotu udávající změnu dané vlastnosti, při neuvedení vlastnosti budova tuto vlastnost nemění.

Vlastnost	Jméno
Znečištění	pollution
Estetika	aesthetics
Kriminalita	crime
Nebezpečí vzniku požáru	fire_danger
Dostupnost zdravotní péče	health_care

Tabulka 4.2: Jména vlastností

Vliv budovy na jedno pole je udán elementem *affect*, který obsahuje vlivy na jednotlivé vlastnosti. Celkový vliv budovy na okolí je v elementu *affects*, jenž smí obsahovat elementy *affect*. První vliv je na pole, na němž budova stojí, následující na pole sousední, ... (i-tý vliv určuje změny na políčku ve vzdálenosti $i - 1$ od budovy). Jako vzdálenost je brán počet kroků se zakázaným diagonálním pohybem. Vlivy na dané pole se sčítají, pole nemá paměť předchozího stavu, vznik/zánik budovy se tedy projeví okamžitě. Příklad 4.6 ukazuje budovu, která znečišťuje prostředí na kterém stojí o 20 jednotek, sousední o 10 jednotek a pole ve vzdálenosti 2 o 5 jednotek.

```
<affects>
     <affect><pollution>20</pollution></affect>
     <affect><pollution>10</pollution></affect>
     <affect><pollution>5</pollution></affect>
</affects>
```

Příklad 4.6: Vlivy na okolí

Podmínky existence

Podmínky existence určují, jaké hodnoty vlastností pole, na kterém budova stojí, jsou pro budovu přípustné. Budova nesplňující tyto podmínky po několik po sobě jdoucích dnů (nastavení viz podkapitola 4.8.2) zaniká. Podmínky jsou uzavřeny v elementu *conditions* a musí být splněny všechny zároveň. Jednotlivé podmínky mají tvar

```
<podminka property='vlastnost'>hodnota</podminka>
```

Hodnota musí být celé číslo, vlastnost je určená jejím jménem (viz tabulka 4.2) a možné podmínky s popisem jsou v tabulce 4.3.

4.8.5 Továrna

Továrna je definována v elementu *factory* a obsahuje povinné elementy *min_workers* a *max_workers* definující nejmenší a největší počet pracovníků. Obsahem těchto elementů je celé kladné číslo.

Dále může obsahovat údaje o produkci v elementu *production* a podmínky pro prodej v elementu *must_sell*. Tvar obou zmíněných vlastností je

Podmínka	Popis
equal	Hodnota vlastnosti daného pole je rovna zadané hodnotě
not_equal	Hodnota vlastnosti daného pole není rovna zadané hodnotě
lesser	Hodnota vlastnosti daného pole je menší než zadaná hodnota
lesser_equal	Hodnota vlastnosti daného pole je menší nebo rovna zadané hodnotě
greater	Hodnota vlastnosti daného pole je větší než zadaná hodnota
greater_equal	Hodnota vlastnosti daného pole je větší nebo rovna zadané hodnotě

Tabulka 4.3: Podmínky

stejný – obsahují libovolný počet elementů *goods* (viz podkapitola 4.8.1) udávající množství denní produkce daného zboží v případě elementu *production* a množství zboží, které musí továrna denně prodat v případě elementu *must_sell*.

4.8.6 Obchod

Obchod je definován pomocí elementu *shop*. Obsahuje, stejně jako továrna, povinné elementy *min_workers* a *max_workers*, a dále povinné elementy *min_customers* a *max_customers* s nejmenším, resp. největším počtem zákazníků daného obchodu.

4.8.7 Dům

Dům je definován v elementu *house* a obsahuje navíc povinné elementy *min_people* a *max_people* definující nejmenší, resp. největší možný počet obyvatel bydlících v budově. Jejich obsahem je kladné celé číslo.

4.8.8 Škola

Škola je zadaná v elementu *school* a má vlastní povinný element *max_students* – kladné celé číslo určující největší počet studentů, kteří se do školy vejdou.

4.8.9 Dopravní stavba

Dopravní stavba, popsaná elementem *transport_facility*, má povinný atribut *type*, který smí nabývat hodnoty *road* pro cestu nebo *railway* pro železnici. Dále musí obsahovat povinný element *max_trans_people*, jehož obsahem je

kladné celé číslo udávající počet přepravených osob za den, na který je stavba projektována.

Pro nastavení obrázků pro budovu slouží 4 elementy – *image*, *t_image*, *l_image* a *x_image*. Všechny tyto elementy mají stejný formát jako element *image* (viz kapitola 4.8.1). Element *image* je určen pro obrázek rovné cesty/železnice jdoucí z levé části pole do pravého. Element *t_image* nastavuje obrázek odbočky (tj. případ tří sousedů), kdy soused chybí na spodní (jižní) straně pole. Element *l_image* obsahuje obrázek zatáčky z levé do horní části pole a *x_image* křižovatky (tj. 4 sousedi). Obrázek bude vybrán podle okolí daného pole, v případě nenastavení některého z nich bude použita rovná cesta zleva doprava.

Příklad 4.7 ukazuje možné nastavení druhu železnice

```
<transport_facility type="railway" name="zeleznice">
     <max_trans_people>30</max_trans_people>
     <image>zeleznice_rovna.png</image>
     <l_image>zeleznice_zatacka.png</l_image>
     <t_image>zeleznice_odbocka.png</t_image>
     <x_image>zeleznice_krizovatka.png</x_image>
</transport_facility>
```

Příklad 4.7: Nastavení dopravní budovy

4.8.10 Křížení silnice a železnice

Křížení silnice a železnice se nastavuje elementem *crossing*. Tento element neobsahuje žádné atributy (přítomné jsou ignorovány). Musí obsahovat element *road*, jehož obsahem je jméno druhu cesty, *railway* se jménem druhu železnice a *image* se souborem obrázku pro křížení. Na obrázku musí vést silnice zleva doprava a železnice shora dolů. Příklad 4.8 ukazuje nastavení křížení silnice s názvem „silnice“ a železnice s názvem „zeleznice“.

Křížení dvojic druhů silnic a železnic, které nejsou takto nastaveny, není možné.

4.8.11 Speciální budova

Speciální budova je definována elementem *spec* a nemá žádné vlastnosti kromě povinných a volitelných pro všechny budovy.

```
<crossing>
     <road>silnice</road>
     <railway>zeleznice</railway>
     <image>zeleznice_silnice_krizeni.png</image>
<crossing>
```

Příklad 4.8: Nastavení křížení cesty s železnicí

Kapitola 5

Programátorská dokumentace

V této kapitole je popsáno rozdělení programu do modulů, použité technologie, popis nejdůležitějších datových struktur a popis nejdůležitějších použitých algoritmů.

Referenční dokumentaci lze vygenerovat ze zdrojových kódů programu příkazem *make doc* v kořenovém adresáři programu. Výsledná dokumentace bude vytvořena v adresáři *doc/hml*.

5.1 Platforma a použité technologie

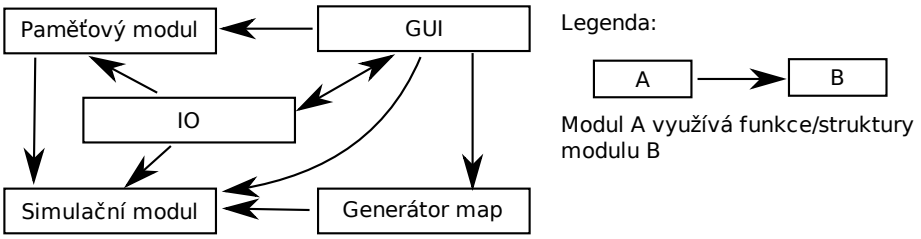
Program je napsán v programovacích jazycích C++ a C a kromě standardních knihoven těchto jazyků využívá knihovnu *GTK+ 2.0* [5] ve verzi alespoň 2.6.12.

Program je určen primárně pro GNU/Linux, ale použití multiplatformní knihovny *GTK+ 2.0* umožňuje jeho kompilaci a používání i v jiných systémech.

5.2 Přehled základních modulů

Program se skládá z modulu simulačního, vstupně–výstupního (IO), paměťového, uživatelského rozhraní (GUI) a generátoru náhodných map. Jejich spolupráci ukazuje obrázek 5.1, přičemž vstupně–výstupní modul využívá z GUI modulu jen funkce pro registrování používaných obrázků. Program dále obsahuje pomocné třídy a funkce v adresáři *libs* využívané všemi částmi.

Další částí je třída *NewDayStepManager* sloužící k řízení spolupráce modulů při simulaci jednoho dne.



Obrázek 5.1: Spolupráce modulů

5.2.1 Simulační modul

Simulační část se stará o základní simulaci města. Její zdrojové kódy se nacházejí v adresářích *map*, *managers* a *buildings* a dále obsahuje soubory *simulace.cpp* a *simulace.h*. Celý modul je rozdělen do čtyř logických celků: definice a implementace budov, definice a implementace mapy, správci řídící jednotlivé jevy a řízení simulace.

Paměťový modul

Paměťový modul slouží k ukládání dat zobrazovaných v grafech hodnot. Jeho zdrojové kódy jsou umístěny v adresáři *memory*.

Vstupně-výstupní modul

Vstupně-výstupní modul se stará o ukládání a načítání dat. Zrojové kódy jsou v adresáři *io* a jeho podadresářích.

Generátor náhodných map

Úkolem generátoru náhodných map je vytvoření mapy pro novou hru. Skládá se ze dvou částí: *PlasmaFractalGenerator* a *MapGenerator*. Zdrojové kódy jsou v adresáři *map_generator*

Uživatelské rozhraní

Zdrojové soubory uživatelského rozhraní se nacházejí v adresáři *gui* a jeho podadresářích, je implementováno pomocí knihovny *GTK+*.

NewDayStepManager

Třída *NewDayStepManager* řídí spolupráci modulů při simulaci jednoho dne. Spouští simulaci dne modulu simulačního a upozorňuje paměťový modul na nová data. Je *ConcreteSubject* návrhového vzoru *Observer* (viz [4]), čehož využívá uživatelské rozhraní.

5.3 Struktury staveb

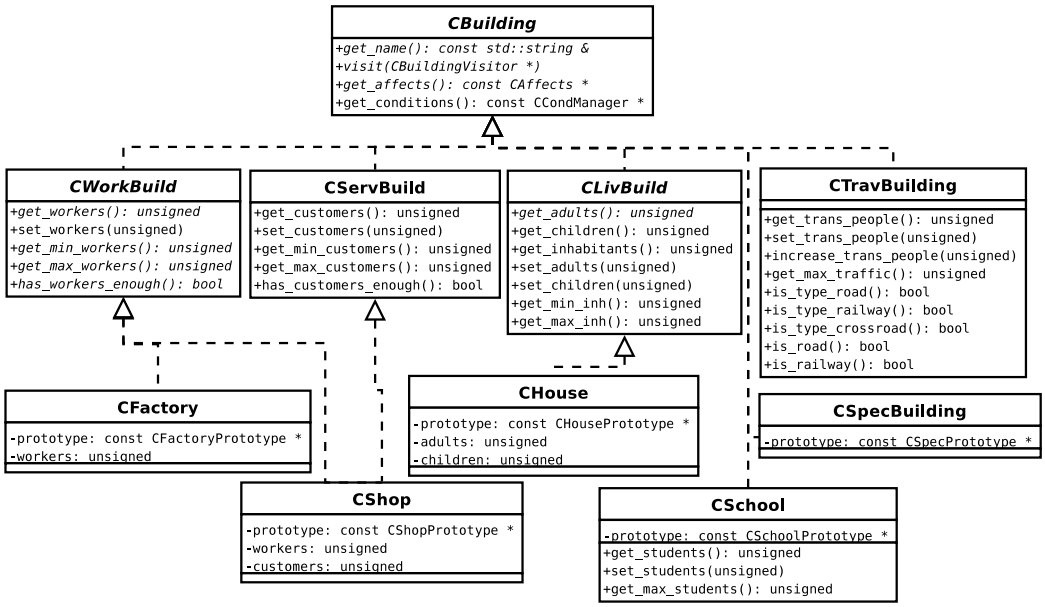
Pro stavby jsou využity dvě hierarchie tříd – hierarchie staveb a hierachie prototypů. Prototypem stavby jsou veškerá data společná pro daný druh stavby nezávislá na konkrétním postaveném exempláři. Příkladem je kapacita školy či jméno druhu budovy.

Prototypy jsou odvozeny od třídy CBuildingPrototype. Instance prototypu se vytváří jednou pro každý nový druh stavby a tento objekt je dále sdílen všemi objekty budov daného druhu. Všechny existující instance prototypů jsou uloženy ve správci prototypů, kterým je singleton *CPrototypeManager*. Třídy prototypů staveb zrcadlí třídy budov s výjimkou dopravních staveb.

Hierarchie staveb je odvozena od abstraktní třídy CBuilding, která je *AbstractElement* návrhového vzoru *Visitor* uvedeného v [4]. Použití tohoto návrhového vzoru umožňuje vykonávání akcí bez nutnosti přetypování a větší přehlednost kódu v místech, kde je potřeba rozhodnout o vykonaných akcích na základě typu budovy (např. odstranění budovy z pole). Konkrétní *visitor*y jsou odvozeny od rozhraní *CBuildingVisitor*. Celá hierarchie tříd staveb bez rozdělení dopravních staveb je na obrázku 5.2.

Dopravní stavby

Dopravní stavby mají základní třídu *CTravBuilding* s dvěma potomky – *ConcreteTravBuilding* a *TravBuildingContainer*. Třída *ConcreteTravBuilding* reprezentuje druh dopravní stavby (silnice, železnice) a má odkaz na



Obrázek 5.2: Základ hierarchie tříd budov

svůj prototyp reprezentovaný třídou *CTravelPrototype*. *TravBuildingContainer* reprezentuje křížení silnice a železnice. Obsahuje pointer na jednu silnici a jednu železnici, jeho kapacita je daná jako součet kapacity příslušné silnice a železnice.

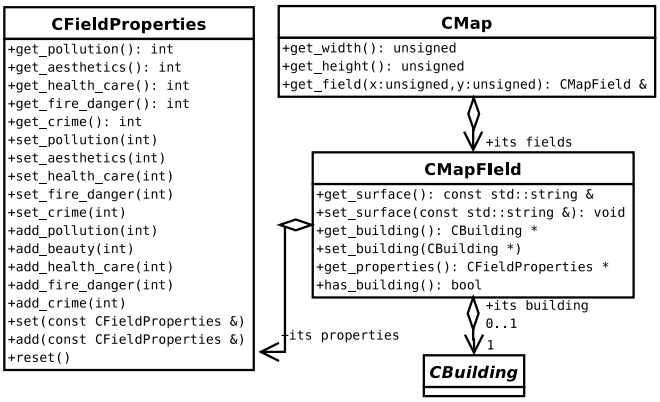
Dokud je na jednom herním poli jen jedna dopravní stavba, je vnitřně reprezentována jako instance třídy *ConcreteTravBuilding*. Při vzniku křížení se vytvoří objekt typu *TravBuildingContainer*, kterému se nastaví pointery na původní stavbu na daném poli a nově budovanou stavbu (přičemž jedna stavba je silnice a jedna železnice).

Továrna

Třída *CFactory* navíc implementuje rozhraní *CGoodsProducer* pro výrobu zboží, dostupné zboží je v soukromém atributu *store* (třída *CStore*).

5.4 Povrchy

Povrchy jsou uloženy v stromové hierarchii tříd, o jejich správu se stará singleton *SurfaceHierarchy*. Povrch je jednoznačně určen svým jménem, které



Obrázek 5.3: Reprezentace mapy

musí být v celé hierarchii jedinečné .

5.5 Mapa

Základním prvkem mapy je herní pole reprezentované třídou *CMapField*. Mapa je reprezentována jako dvourozměrné pole herních polí. Herní pole obsahuje vlastnosti (znečištění, ...) uložené ve třídě *CFieldProperties*. Diagram mapy je na obrázku 5.3

5.6 Správci simulovaných oblastí

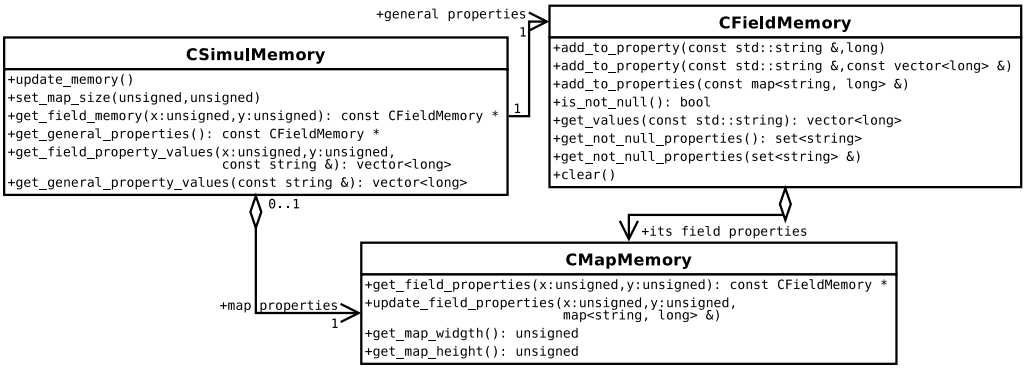
Každou simulovanou oblast má na starost jeden správce, implementovaný jako singleton. Jsou to *CWorkManager* pro pracovní trh, *CStoreManager* pro distribuci zboží k zákazníkům, *SellFactManager* pro kontrolu, jestli továrny prodaly dostatečné množství zboží, *CSchoolManager* pro simulaci školství, *CShopManager* pro nákupní místa, *CBuildingCondManager* pro kontrolu splnění podmínek budovy na vlastnosti herního pole a *CTravelComponents* pro simulaci dopravy.

Celý běh simulace řídí třída *CSimulace*, která dále rozhoduje o vzniku nových budov, příchodu obyvatel a tvoří hlavní rozhraní pro práci se simulací.

Všechny třídy pro správu simulovaných oblastí jsou v adresáři *managers*, třída *CSimulace* je v souborech *simulace.h* a *simulace.cpp* v kořenovém adresáři zdrojových kódů.

5.7 Simulační paměť

Paměťový modul se skládá ze tří tříd – *CFieldMemory* pro držení dat o souboru vlastností, *CMapMemory* pro udržování paměti o polích na herní mapě a singletonu *CSimulMemory*, který je rozhraním celého modulu pro zbytek programu. Diagram struktury modulu zachycuje obrázek 5.4



Obrázek 5.4: Reprezentace paměti

Funkce *CSimulMemory::update_memory* ukládá veškerá zobrazitelná data z modulu simulačního do simulační paměti.

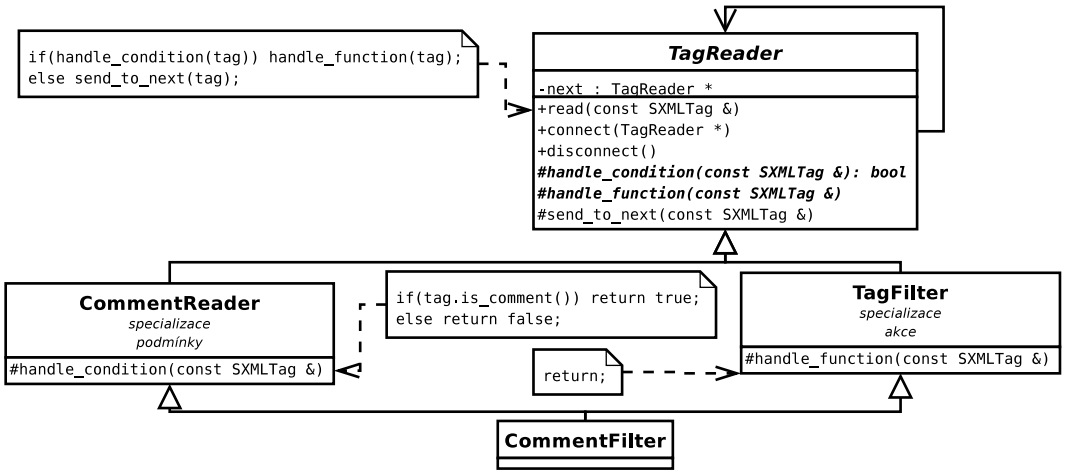
K uložení dat o jedné vlastnosti slouží pomocná třída *const_size_list*, která si pamatuje konstantní počet naposledy vložených hodnot. K ušetření paměti se vlastnost u které jsou uloženy jen nuly odstraní, stejně tak se odstraní paměť celého pole, které má uloženy pouze vlastnosti s nulovými hodnotami.

5.8 Vstup/výstup

Kód pro zpracování vstupních a výstupních souborů je v adresáři *io*.

Zpracování vstupních souboru

Každý vstupní soubor se nejprve převede do DOM modelu, který se dále zpracovává. K převodu je použit konečný automat implementovaný v souborech *xml_like/xml_reader.h* a *xml_like/xml_reader.cpp*. Element je reprezentován třídou *SXmlTag*.



Obrázek 5.5: Příklad odvození filtru komentářů

Pro zpracování elementu se používají potomci třídy *TagReader* – ti se spojují do řetězce, ve kterém se zpracovávaný element posílá, dokud některý objekt posílání nezastaví. Každý objekt přes který je element poslán může na jeho základě vykonat nějakou akci/zapamatovat si některé údaje.

Třída *TagReader* a její potomci zpracovávají element pomocí funkce *read*, což je šablona funkce (Template Method – viz [4]) s implementací

```
if( handle_condition(tag) ) handle_function(tag);
else send_to_next(tag);
```

přičemž funkce send_to_next() předá element následujícímu zpracovávajícímu objektu – ten se připojuje pomocí veřejné metody *connect*. Funkce *handle_condition* a *handle_function* je třeba implementovat v každém potomku.

Pro častěji používané akce a podmínky jsou odvozené třídy implementující pouze jednu z těchto funkcí, např. *CommentReader* pro podmínku odchycení komentáře a *TagFilter* pro akci ukončující další zpracování elementu. Z nich je pak možné odvodit konkrétní třídu pro určitý způsob zpracování, např. *CommentFilter* pro filtrování komentářů, *BlankTextFilter* prázdných řetězců, ... Diagram odvození filtru komentářů je na obrázku 5.5

Pro kontrolu a snažší správu celého řetězce objektů pro zpracování elementů je třída *TagReaderChain*, která umožňuje přidávat na konec řetězce další objekty a přebírá zodpovědnost za uvolnění jejich paměti.

Ukládání hry

Pro uložení hry slouží třídy *MapSaver* pro uložení aktuální mapy, *SimulationDataSaver* pro uložení hlavních simulačních dat, *SimulMemorySaver* pro data paměťového modulu, *SchoolManagerSaver* pro školství, *SellFactManagerSaver* pro přehled prodeje továren, *ShopManagerSaver* pro správce nákupních míst, *TravelComponentsSaver* pro správce dopravy a *WorkManagerSaver* pro správce zaměstnanosti. Ty jsou odvozeny od abstraktní třídy *DataSaver* s čistě virtuální funkcí `save(ostream &)`.

Načítání definice budov

Pro každý typ budov existuje tzv. *„Register“*, což je třída zpracovávající element pro daný typ budovy a zaregistruje typ budovy v programu. Každý „register“ je odvozen od třídy *BuildingRegister*, která zpracovává vlastnosti společné pro všechny typy budov. Přes její chráněné metody přidává odvozený objekt zpracovávání pro něj specifických elementů a spouští zpracování vnitřních elementů.

Načítání uložené hry

Každý objekt, jehož stav se ukládá, má přátelskou třídu pro nastavování načtených dat, tzv. „reader“. Protože se ale data zpravidla čtou pomocí jiného objektu (pro čtení různých elementů jsou implementovány různí potomci třídy *TagReader*) je „reader“ odvozen od rozhraní pro nastavení přečtených dat. Potomku třídy *TagReader* pro čtení a ukládání vnitřních elementů je předán ukazatel na objekt implementující takovéto rozhraní („reader“ pro danou třídu) skrz který se nastavují načítaná data.

5.9 Grafické uživatelské prostředí

5.9.1 Databáze obrázků

Ke správě používaných obrázků slouží třída *ImageBase*. Každý obrázek je identifikován svým jménem (to nemusí být stejné jako jméno souboru).

Udržují se dvě asociace: *Jméno obrázku – Jméno souboru* a *Jméno souboru – obrázek*. Obrázek je udržován ve struktuře *GdkPixbuf* knihovny *GTK+*

a je vytvářen pomocí tzv. líné alokace, tj. až v případě prvního použití. To umožňuje úsporu paměti při nepoužívání všech přípustných obrázků.

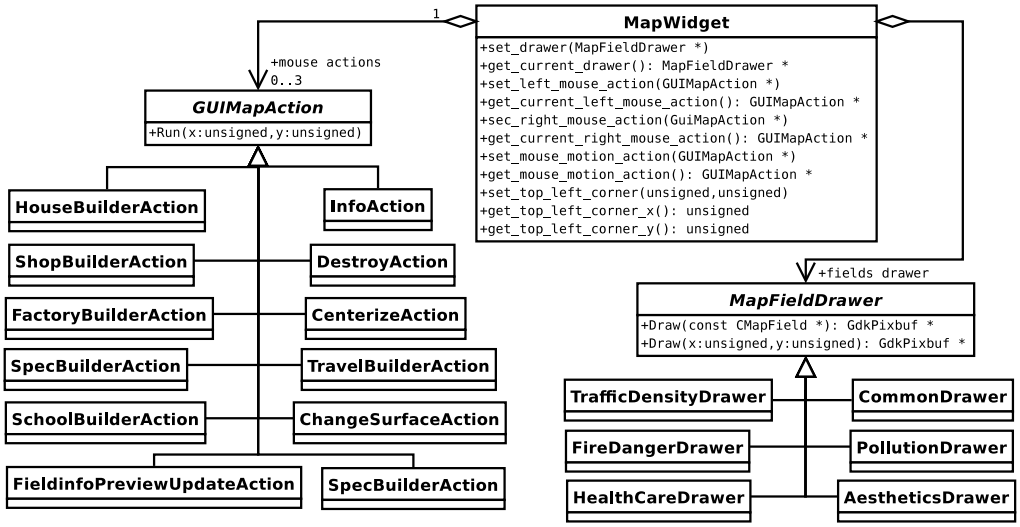
Pro registraci obrázků ze vstupně–výstupního modulu bez rozhraní používající struktury z knihovny *GTK+* jsou určeny funkce třídy *ImageRegistrator*.

5.9.2 Mapa, její vykreslení a akce uživatele

Třída *MapWidget* definuje grafický prvek pro zobrazení mapy. Pro její vykreslení mapy slouží potomci třídy *MapFieldDrawer*. Každý potomek implementuje funkce *Draw* pro vytvoření obrázku určitého políčka. Pokud není nastaven žádný kreslicí objekt, je použit *CommonDrawer*, který vykresluje obrázek povrchu a budovy.

Akce vykonaná po stisku tlačítek myši nebo při přejetí přes pole je určena objektem, který musí být potomkem třídy *GUIMapAction*. Při příslušné události dojde k ověření, jestli je myš nad zobrazeným polem (políčka, která se nevejdou celá, nejsou zobrazena) a poté je spuštěna metoda *Run* pro souřadnice pole pod kurzorem.

Diagram tříd souvisejících s MapWidgetem je na obrázku 5.6.



Obrázek 5.6: MapWidget

Vykreslení dopravních staveb a jejich křížení

Při vykreslování dopravní stavby se vybere příslušný obrázek na základě okolních polí. V případě jedné dopravní stavby na herním poli se nejprve spočítá počet sousedů stejného typu (cesta/železnice) jako je stavba na vykreslovaném poli. S výjimkou dvou sousedů je obrázek jednoznačně určen (při žádném či jednom sousedovi stejného typu se použije rovná cesta), ve zbývajícím případě se zkontroluje, jde-li o zatáčku či rovnou cestu a spočítá se rotace obrázku.

Jde-li o křížení cesty s železnicí, je situace složitější. Sousedí-li několik křížení, měla by být natočena stejným směrem a tudíž nelze brát v úvahu pouze pole sousedící s vykreslovanou budovou, ale celou množinu dotýkajících se křížení. Podmínka pro jejich vznik zaručuje, že budou sousedit pouze v jedné přímce (jinak by nutně existovala trojice sousedících křížení tvořící pravý úhel). U obrázku jsou možné pouze dvě smysluplné rotace – žádná (cesta horizontálně a železnice vertikálně) a o pravý úhel.

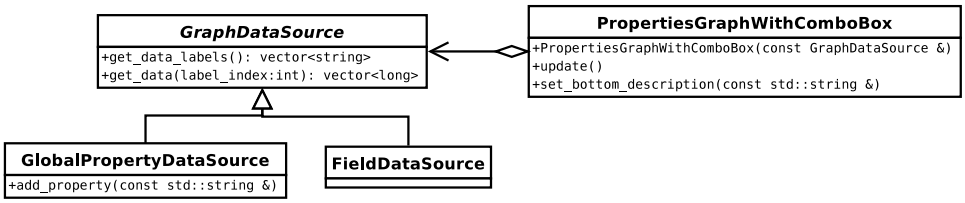
Pro celý sousedící sloupec/řadu křížení se spočítá „špatnost“ (badness) jednotlivých vykreslení, přičemž „špatnost“ je počet sousedních polí s typem dopravní stavby neodpovídajících natočení, křížení se bere jako odpovídající. Například pokud pole s křížením sousedí zleva a zprava se silnicemi, zhora s železnicí a zezdola s dalším křížením, přispěje do „špatnosti“ hodnotou 0 při žádném natočení a hodnotou 3 při rotaci o pravý úhel.

Jako výsledná rotace se použije ta s nižší výslednou „špatností“, při rovnosti je použit obrázek bez rotace.

5.9.3 Grafy

Pro vykreslování grafů byl v programu implementován nový widget GtkLineGraph pro vykreslení čárového grafu. Zdrojové kódy jsou v souborech *gui/gtk_line_graph.c* a *gui/gtk_line_graph.h*.

Ten byl spolu s rozbalovacím seznamem dán do třídy *PropertiesGraphWithComboBox*. Tato třída získává zobrazovaná data z objektů implementujících rozhraní *GraphDataSource* – takové třídy jsou implementovány dvě: *FieldDataSource* pro získání dat vybraných vlastností pole na mapě a *GlobalPropertyDataSource* k získání celkových dat celého města z paměťového modulu. Diagram je na obrázku 5.7. Data vrácená funkcí *GraphDataSource::get_data* jsou zobrazena v pořadí zleva doprava (tj. údaj s indexem 0 je na grafu vlevo).



Obrázek 5.7: Graf hodnot

5.9.4 Náhled na vygenerovanou mapu

Dalším nově vytvořeným widgetem je GtkMapPreview určený pro náhled na vygenerovanou mapu v okně pro start nové hry. Funkcí *gtk_map_preview_set_map* se nastaví aktuální zobrazovaná mapa – dvourozměrné pole indexů povrchů, funkcí *gtk_map_preview_set_colors* se nastaví barvy jednotlivým povrchům. Tento widget je spolu s tlačítky pro výběr barev spojen ve třídě *MapPreviewColorsSet*.

5.9.5 Informační okna

Informační okna jsou odvozena od třídy *InfoWindowInterface* a je *ConcreteObserver* návrhového vzoru *Observer*. *ConcreteSubject* je instace třídy *NewDayStepManager*, která řídí simulaci jednoho dne. Při jejich vzniku se konstruktoru předává pointer na objekt implementující rozhraní *INotificationFunctor* vykonávající všechny potřebné akce po zavření okna (zpravidla vyřazení z aktivních „observerů“ a zničení objektu).

5.10 Použité postupy a algoritmy

V této kapitole jsou rozebrány nejdůležitější postupy použité v programu.

5.10.1 Vznik nových budov z vyhrazených zón

Vznik nových budov ze zón je implementován členskými funkcemi třídy *Csimulace* *build_new_houses()*, *build_new_shops()* a *build_new_factories()* v souboru simulace.cpp.

Pro každý typ budovy se procházejí všechna neobsazená pole příslušné zóny.

Rozhodnutí o postavení nové budovy

Nejprve se zkontroluje, jestli může na zkoumaném políčku s danými hodnotami vlastností a povrchem stát nějaká budova zkoumaného typu. Pokud ano, spočítá se pravděpodobnost vzniku budovy na daném poli, která závisí na žádanosti daného pozemku počítané podle vzorce:

$$dem = aesthetics + health_care - pollution - fire_danger - crime$$

kde dem je žádanost pole, $aesthetics$ estetika, $health_care$ dostupnost zdravotní péče, $pollution$ znečištění, $fire_danger$ nebezpečí vzniku požáru a $crime$ kriminalita na daném poli.

Z takto vypočítané žádanosti se spočítá základní pravděpodobnost vzniku nové budovy na tomto políčku jako

$$P_z = \frac{dem - min_dem}{max_dem - min_dem}$$

kde min_dem a max_dem jsou nejmenší, resp. největší žádanosti polí za život města.

Tato pravděpodobnost je dále postihována za

- nulový počet nezaměstnaných u továren a obchodů
- nulový počet lidí bez obchodu u obchodů
- dostatek všech druhů zboží u továren
- nulový počet míst ve školách u domů
- nulový počet volných pracovních míst u domů
- nulový počet volných nákupních míst v obchodech u domů

Výsledná pravděpodobnost je ještě převedena do intervalu $< MIN, MAX >$ vzorcem

$$P = \overline{P}_z * (MAX - MIN) + MIN$$

kde $\overline{P}_z$ je P_z po penalizaci, MIN a MAX jsou nejmenší, resp. největší pravěpodobnost vzniku nové budovy.

S takto spočtenou výslednou pravděpodobností vznikne na zkoumaném poli nová budova.

Výběr budovy

Pokud má být nějaká budova na zkoumaném poli postavena, spočítá se váha všech možných druhů budov daného typu, které na daném místě mohou existovat (tj. jsou splněny nároky na dané políčko a povrch je přípustný). Pravděpodobnost vzniku daného druhu je určena poměrem vah všech přípustných budov (pokud jsou tedy váhy $15, 22, 13$, budou pravděpodobnosti $30\ \%, 44\ \%, 26\ \%$).

Váha továrny se počítá z množství vyráběného zboží a trhu města a dále se upravuje podle zaměstnanosti. Váha pro jeden druh zboží se spočítá jako

$$v_i = \frac{produkce_i - potreba_i}{potreba_i} + produkce_i^t$$

kde $produkce_i$ je celková produkce daného zboží ve městě během předchozího dne, $potreba_i$ je množství daného zboží potřebné ve městě během posledního dne a $produkce_i^t$ je množství tohoto zboží produkované továrnou. Takto vypočtená váha zboží je mírně snížena v případě, že $produkce_i - potreba_i$ (což je množství daného zboží chybějící na trhu) menší než produkce továrny a výrazně snížena, pokud je toto množství menší než množství, které musí továrna za den prodat.

Celková váha zboží je průměr z vah všech druhů zboží vyráběných v továrně. K celkové váze se přičte maximální počet zaměstnanců a sníží se při nedostatku nezaměstnaných ve městě.

Váha druhu obchodu se počítá jako součet maximálního počtu zaměstnanců a maximálního počtu zákazníků. Je výrazně snížena pokud ve městě není aspoň tolik lidí bez obchodu jako je minimální počet zákazníků pro daný druh obchodu nebo je ve městě méně nezaměstnaných než minimální potřebný počet zaměstnanců obchodu.

Váha domu je přímo úměrná maximálnímu počtu obyvatel. Snižuje se, pokud je ve městě volných pracovních či nákupních míst méně než je minimální počet obyvatel domu a pokud není ve městě dostatek zboží a volných studijních míst.

5.10.2 Simulace pracovního trhu, nákupních míst a školství

Obsazování pracovních, obchodních a studijních míst se řeší stejným principem. Řešení vychází z předpokladu, že každý člověk musí pracovat, mít

kde nakupovat a každé dítě musí chodit do nějaké školy. U pracovních a obchodních míst jsou cílovou skupinou obyvatel dospělí, u studijních míst děti. Celý princip bude demostrován na hledání pracovních míst.

Člověk, který se přistěhuje do města, je zadán do správce pracovních míst (viz podkapitola 5.6), stejně tak každá nová továrna či obchod. Správce v každém tahu projde všechny osoby a těm nezaměstnaným se snaží přidělit volné pracovní místa. Kritériem je vzdálenost mezi bydlištěm člověka a umístěním zaměstnavatele – pro každou osobu se vybere místo co nejbližší jeho domovu.

U osob je sledován počet po sobě jdoucích dnů, ve kterých jsou bez práce. Pokud překročí nastavenou hodnotu, celá rodina dané osoby odchází z města.

U zaměstnané osoby se stejný proces opakuje (perioda je nastavitelná v konfiguračním souboru) se snahou nalézt zaměstnání blíže domovu, než je stávající.

5.10.3 Generování map

Generování map probíhá ve dvou fázích:

1. vygenerování výškové mapy
2. určení povrchu z výšky polí v mapě

Pro generování výškové mapy se používá metoda přesunu středního bodu uvedená v [3]:

1. Určení výšky středů obvodových úseček
2. Určení výšky středního bodu obdélníku jako průměr ze středů obvodových úseček
3. Posun výšky středního bodu obdélníku o náhodnou hodnotu, nejvýše o δ
4. Rekurzivní volání na obdélníky se zmenšenou hodnotou δ

Před prvním během algoritmu je náhodně určena výška rohových bodů mapy.

Po získání výškové mapy se rozhodne o površích na daných herních polích podle výškových hranic určených uživatelem (ve výchozím stavu je celý interval možných výšek rozdělen rovnoměrně mezi všechny povrchy).

Kapitola 6

Možná rozšíření

U programu se nabízí další možná budoucí rozšíření

6.1 Ekonomický model hráče

Financování uživatelských akcí a údržby budov, výběr daní, možnost půjček s úrokem závislým na celkové půjčené částce, ...

Jednou možností je začlenit tuto funkčnost do modulu simulačního (vytvořit finančního správce), což by ovšem vedlo k jeho dalšímu zesložitění, ale umožňovalo by to přímé ovlivnění města na základě výše daní či množství peněz na údržbu.

Další ze způsobů realizace tohoto rozšíření spočívá v implementaci nového modulu, který by využíval data simulačního modulu pro výpočet zisků a ztrát hráče. Tento způsob by nevyžadoval žádný zásah do simulačního modulu, ale vývoj města by nereagoval na hráčské ekonomické zásahy.

Třetím způsobem je implementace správce údržby a daní do simulačního modulu a vytvoření nového modulu pro ostatní hráčskou ekonomiku (aktuální stav konta, půjčky, ...), který by neměl přímý vliv na vnitřní vývoj města během tahů. Zde je nevýhodou zásah jak do simulační části programu, tak vytvoření nového modulu – ta je vykoupena menším zásahem do simulační části, kde se budou udržovat pouze data nutná pro výpočet vnitřního stavu města v následujícím tahu, a zároveň možností ovlivnění vývoje města nastavením ekonomických údajů.

Ve všech případech by bylo nutné upravit GUI modul pro zpřístupnění nových dat uživateli a možnosti jejich změny, vytvořit vazbu mezi akcí uživatele a jeho ekonomickým stavem, přidání uložení a načítání ekonomických

dat při ukládání a načítání hry a přidání nových vlastností, jejichž data má paměťový modul zpracovávat.

6.2 Definice herních událostí

Další možným rozšířením je definování akcí pro herní pole a jejich výsledků. Akce by závisela na vlastnostech pole a postavené budově a výsledek by měl přímý vliv na funčnost budovy – ten by mohl záviset i na okolních budovách. Taková událost a následná akce může být např.: pokud nebezpečí vzniku požáru překročí hodnotu 150 na políčku s továrnou, se 40% pravděpodobností se budově nic nestane, s 30% pravděpodobností bude výroba na týden přerušena, s 20% procentní pravděpodobností bude továrna zničena a s 10% pravděpodobností nebude povolen na daném poli v následujícím týdnu vznik další budovy.

Toto rozšíření by si pravděpodobně (podle možných akcí) vyžádalo drobné zásahy do definic budov (možnost pozastavení výroby u továren, ...) a funkcí simulačního modulu (např. znemožnění stavby na určitém poli). Dále by stačilo přidání nového modulu kontrolujícího dané události a vykonávajícího případné akce.

6.3 Skriptovaná hra

Jako další možné rozšíření se nabízí skriptování průběhu hry a tvorba scénářů. Skript by měl definovaná čísla tahů pro vykonání nějaké akce (zničení budovy na určitém herním poli, zničení náhodné budovy/několika budov daného typu, postavení budovy na určitém místě) a výchozí herní plán.

Realizace by byla možná přidáním skriptovacího modulu, který by dané akce vykonával. Dále by se muselo přidat do vstupně–výstupního modulu uložení a načtení skriptu a uložení/načtení čísla tahu a identifikátoru použitého skriptu (např. jméno souboru) při ukládání/načítání hry. Zároveň by bylo nutné implementovat možnost posílání zpráv z tohoto modulu uživateli.

Kapitola 7

Závěr

Cílem této práce byla implementace simulátoru vnitřního vývoje města s možnostní zásahů uživatele.

Výsledkem je program simulující základní prvky města – stavbu budov, pracovní trh, výrobu a distribuci zboží, příchod a odchod obyvatel, školství, dopravu a vývoj základních vlastností herních polí (znečištění, nebezpečí vzniku požáru, estetiku, dostupnost zdravotní péče a kriminalitu). Uživatel má možnost přímo ovlivnit město stavbou nových budov, změnou povrchů herních polí a vyhrazením některých z nich pro obytné, výrobní nebo obchodní využití. Podstatné údaje mohou být zobrazeny číselně, v grafech nebo změnou způsobu zobrazení herního plánu. Částí programu je také generátor náhodných map generující realistické mapy.

Nejdůležitější parametry simulace jsou nastavitelné v konfiguračních souborech a lze tedy sledovat změny chování města při jejich různém nastavení.

Program lze dále rozšiřovat přidáváním nových modulů a úpravou stávajících. Je v něm oddělena simulační logika od ostatních funkcí.

Literatura

[1] DEMEL, Jiří (2002): *Grafy a jejich aplikace*, Academia, Praha.

[2] KOSEK, Jiří (2000): *XML pro každého*. Grada Publishing, Praha. Dostupná online: *http://www.kosek.cz/xml/xmlprokazdeho.pdf*

[3] TIŠINOVSKÝ, Pavel. (2006): *Metoda přesouvání prostředního bodu a obrázky plasmy*[online]. Root.cz, Praha. ISSN 1212-8309. URL: *http://www.root.cz/clanky/metoda-presouvani-prostredniho-bodu-a-obrazky-plasmy/*

[4] ZAVORAL, Filip. *Slajdy k předmětu Návrhové vzory*[online] URL:*http://ulita.ms.mff.cuni.cz/mff/sylaby/prg024.html*

[5] *The GTK+ Project*[online] URL: *www.gtk.org*

[6] *LinCity - A City Simulation Game*[online] URL:*http://lincity.sourceforge.net/*

[7] *LinCity-NG Homepage*[online] URL:*http://lincity-ng.berlios.de/wiki/index.php/Main_Page*

[8] *SimCity 3000*[online], Electronic Arts URL:*http://simcity3000unlimited.ea.com*

[9] *SimCity 4*[online], Electronic Arts URL:*http://simcity.ea.com*

[10] *Rodiny se závislými dětmi*[online], Český statistický úřad, 2004, Úvod URL:*http://www.czso.cz/csu/2004edicniplan.nsf/p/4124-04*

Příloha A

Obsah přiloženého CD

Na přiloženém CD je v souboru *bc_text.pdf* v kořenovém adresáři elektronická verze této bakalářské práce. Adresář *simulace* je hlavním adresářem programu vzniklého v této práci a jeho adresářová struktura je

data Výchozí konfigurační soubory a podadresář *images* s výchozími obrázky

doc Soubory pro generování referenční dokumentace

src Kořenový adresář zdrojových kódů programu

tests Testy pro některé základní části programu

V adresáři *bin* je nainsalovaný program. Pro běh programu musí mít uživatel právo pro spuštění souboru *simul_city* a práva pro čtení všech souborů v tomto adresáři, resp. v adresáři, kam byl obsah tohoto adresáře zkopírován.