VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

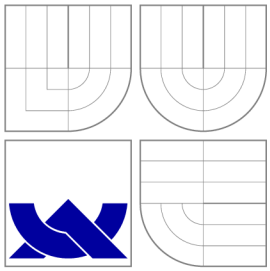
SIMULACE FYZIKÁLNÍCH JEVŮ S VYUŽITÍM CELULÁRNÍCH AUTOMATŮ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

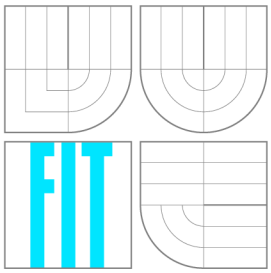
AUTOR PRÁCE
AUTHOR

Bc. DOMINIK MARTINEK

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

SIMULACE FYZIKÁLNÍCH JEVŮ S VYUŽITÍM CELULÁRNÍCH AUTOMATŮ

SIMULATION OF PHYSICAL PHENOMENA USING CELLULAR AUTOMATA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. DOMINIK MARTINEK
AUTHOR

VEDOUCÍ PRÁCE — Dr. Ing. PETR PERINGER
SUPERVISOR

BRNO 2010

Abstrakt

Tato diplomová práce se zabývá modelováním a simulací fyzikálních jevů, u nichž se využívají celulární automaty. Jsou zde vyjmenovány a popsány základní metody, které se k modelování fyzikálních jevů využívají. U každého modelu jsou uvedena přechodová pravidla zároveň s jejich odvozením. Tato pravidla se pak používají v implementovaných modelech. V práci je pak uveden návrh aplikace, která slouží k simulaci jednotlivých modelů. Podle tohoto návrhu byla vytvořena aplikace, s jejíž pomocí byla provedena simulace ukázkových příkladů. Každý příklad se zabývá jednou oblastí fyzikálních jevů. V práci jsou pak zaznamenány výstupy jednotlivých modelů včetně zhodnocení jednotlivých závěrů. Jeden vybraný příklad byl rovněž zparalelizován a byly změřeny časy výpočtu s využitím různého počtu procesorů.

Abstract

This master's thesis deals with modelling and simulation of physical phenomena by cellular automata. The basic methods which model physical phenomena is enumerated and descibed in this thesis. One of the important part of this thesis is a set of demonstration models. Each model is focused on one selected area of physical phenomena. All models are described by transtition rules and the procedure of derivation of these rules is also presented here. There rules were used in implemented models.

Another part of this thesis contains of a simulation application for these models. The real application had been implemented in accord with this design and it has been used to perform the simulation experiments with exemplary models. Results of the simulation experiments are discussed in conclusion of this thesis. One exemplary model had also been adapted for parallel processing. The performances on a computer with different count of working processors were measured and are also discussed in the conclusion of this thesis.

Klíčová slova

modelování, simulace, celulární automat, fyzikální jevy, paralelní výpočet

Keywords

modeling, simulation, cellular automata, physical phenomena, parallel computing

Citace

Dominik Martinek: Simulace fyzikálních jevů
s využitím celulárních automatů, diplomová práce, Brno, FIT VUT v Brně, 2010

Simulace fyzikálních jevů s využitím celulárních automatů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Dr. Ing. Petra Peringera. V práci jsou rovněž uvedeny všechny literární prameny a publikace, ze kterých bylo čerpáno.

.....
Dominik Martinek
25. května 2010

Poděkování

Děkuji svému vedoucímu práce Dr. Ing. Petru Peringerovi za poskytnutí rad a publikací, které pomohly tuto práci vytvořit.

Obsah

Kapitola 1

Úvod

Celulární automaty (v práci též označovány zkratkou CA) představují nástroj, který lze velmi výhodně použít pro modelování a simulaci fyzikálních jevů. Problematikou celulárních automatů jsem se již zabýval ve své bakalářské práci. V této diplomové práci tedy na svou dosavadní činnost navazuji a některé obecné definice jsou z mé bakalářské práce s drobnými úpravami převzaty.

První kapitola je seznámením s tím, co to celulární automaty jsou a jak fungují. Je uvedena formální definice takového automatu včetně zákonitostí, které musí v celulárním automatu platit. Dále jsou uvedeny základní typy celulárních automatů, které se rozlišují jak podle druhů pravidel, tak podle druhů okolí buněk. Součástí kapitoly je část věnující se metodám efektivního výpočtu celulárních automatů. Popisuje výpočet jak v sériovém tak paralelním způsobu řešení. Paralelní výpočet dále dělí na systém distribuovaného výpočtu a výpočtu se sdílenou pamětí. Pro všechny druhy výpočtu jsou vyjmenovány výhody i nevýhody takového přístupu.

Následující kapitoly nás přivedou do oblasti fyzikálních jevů, pro jejichž simulaci lze použít CA. Tyto jevy jsou rozděleny do několika oblastí podle typu daného jevu a podle typu přechodových pravidel, které se při modelování konkrétního jevu používají. Konkrétní odvození pravidel bývá často pouze nastíněno, protože se jedná o kombinaci odborné matematiky a fyziky, která přesahuje zaměření této práce. Tato kapitola je přehledová, jelikož jedním z cílů práce bylo prozkoumání metod, kterých se pro modelování fyzikálních jevů používá.

Pro práci byly vybrány pouze některé metody, jelikož prozkoumání všech metod přesahuje rozsah této práce. V první řadě byly vybrány metody popisující mikroskopické děje v plynech. Jedná se o takzvaná HPP pravidla a FHP pravidla, která pracují pouze se stejnými jednoduchými částicemi pohybujícími se po čtvercové nebo hexagonální mřížce. Rovněž je nastíněn princip Boltzmannovy mřížky, která tvoří myšlenkový základ pro mnoho jiných metod. Dále byla vybrána oblast metod zabývají se sypáním sypkých materiálů, které nejčastěji představuje písek. Další oblastí jsou metody popisující šíření vlnění. Tyto metody jsou právě odvozeny z principu Boltzmannovy mřížky. Těmito modely můžeme popsat jevy, jakými jsou různé odrazy a lomy vlnění, které se pohybuje prostorem. V poslední části je uveden stručný přehled dalších oblastí fyzikálních jevů, kterým se lze v souvislosti s celulárními automaty věnovat, avšak do této práce se již nevešly.

V následující kapitole je představen návrh aplikace pro simulaci celulárních automatů. Návrh aplikace je rozdělen do dvou bloků, které jsou pro simulátor důležité. Nejdříve je prezentován návrh grafického uživatelského prostředí, které slouží uživateli k ovládání simulace. Posléze je představen návrh výpočetního jádra simulátoru, ve kterém se provádí

výpočet během simulace. Dále jsou pak představeny metody, jakými lze zefektivnit tento výpočet celulárního automatu.

Další kapitola popisuje úskalí spojená s implementací jednotlivých částí. Je do podrobna popsána implementace grafického prostředí, ve kterém jsou jednotlivé modely prezentovány. Dále následuje popis jednotlivých modelů a jejich výpočetních mechanismů. V samém závěru je uveden způsob, jakým byla aplikace zparalelizována.

Poslední kapitola této práce je věnována testování. Pro každý implementovaný zásuvný modul, který představuje výpočetní jádro simulátoru, je uveden model, kterým byl tento modul testován. Ke každému modelu je uveden obraz, či sekvence obrázků zachycených přímo z aplikace, na nichž lze vidět grafický výstup. U každého modelu je uvedeno, co daný model představuje a závěry zhodnocující vlastnosti modelu. Součástí této kapitoly je pak analýza výsledků získaných při měření paralelního řešení vybraného modelu. Kapitola je zakončena stručnou zmínkou o metodách validace fyzikálních modelů.

Kapitola 2

Celulární automaty

Tato kapitola se věnuje ve zkratce historii a vzniku CA. Dále je v této kapitole uvedena definice CA a rozdělení do několika skupin, podle kterých je možné CA rozdělovat. Na závěr této části je nastíněna filozofie, proč jsou CA vhodné pro modelování fyzikálních jevů.

2.1 Historie

Historie celulárních automatů se datuje do čtyřicátých let 20. století, kdy přišel John von Neumann s myšlenkou vytvořit stroj, který bude schopen řešit složité komplexní problémy. Jeho inspirací byl mozek, který je schopen mechanizmů jako sebekontrola a sebeopravování. Tehdy přišel s myšlenkou definovat život jako logický proces. Jedním z hlavních Von Neumannových cílů bylo vytvořit systém, který bude dostatečně komplikovaný, aby byl schopen sebereprodukce, čili vytvoření své vlastní kopie.

S možným řešením tohoto problému mu pomohl matematik Stanislaw Ulam, který nalezl inspiraci v přírodě. Každý organismus nebo systém ve vesmíru je totiž složen z určitého počtu diskrétních stavebních prvků, jakými jsou například molekuly, buňky, nebo nižší organizmy. Každý tento prvek je charakterizován svým vnitřním stavem, který bývá většinou vymezen v konečném počtu dosáhnutelných stavů (nabitý, nenabitý, živý, mrtvý,...).

Von Neumann tedy navrhl systém jako soustavu jednotlivých buněk v diskrétních časových krocích, které fungují jako velmi jednoduché automaty, jenž dokážou vypočítat svůj následující stav. Tento výpočet se řídí podle jednoduchých pravidel, je totožný pro všechny buňky v systému a jedná se de facto o funkci, která má za vstup nejen stav dané buňky v diskrétním čase ale i stavy okolních buněk. Výstupem takovéto funkce je hodnota buňky v následujícím kroku, čili její další vývojový stupeň. Takovéto systémy byly nazvány jako celulární či buněčné automaty.

Johnu von Neumannovi se skutečně podařilo tímto mechanismem vytvořit první sebereprodukční celulární automat. Jednalo se o dvoudimenzionální plochu se čtvercovými buňkami, kde buňky mohly dosahovat jednoho z 29 stavů. Na základě hodnot buňky a hodnot buněk jejího nejbližšího čtyřokolí byla vytvořena evoluční pravidla, která zajistila, že stroj byl schopen vytvořit svou vlastní kopii, která měla tytéž vlastnosti. Šlo o převratný objev, protože se do té doby se předpokládalo, že stroj je pouze schopen vytvořit jiný objekt na nižší úrovni komplexity, než je on sám.

Tento úspěch vedl k dalšímu zkoumání CA. Další mezník vytvořil John Conway svou hrou „Life“. Jedná se opět o dvoudimenzionální plochu čtvercových buněk, kde každá buňka nabývá binárních hodnot (1 nebo 0, živá nebo mrtvá). Na základě tří pravidel se pak

buňky rodí, přežívají nebo hynou. Na první pohled se zdá, že jde o primitivní věc, avšak buňky jsou schopny při různých počátečních konfiguracích vytvářet komplikované struktury. Během pokusů s hrou Life bylo zjištěno a formálně dokázáno, že tato hra je výpočetně ekvivalentní Turingovu stroji. Dodnes se hry Life využívá pro modelování některých komplexních problémů.

Celkovou schopnost využití celulárních automatů pro studování a modelování komplexních systémů a fyzikálních jevů shrnuli Tommaso Toffoli a Norman Margolus z MIT[4]:

> „Celulární automaty jsou stylizované syntetické vesmíry, definované jednoduchými pravidly, dosti podobné deskovým hrám. Mají svou vlastní formu hmoty, která se složitě pohybuje ve svém vlastním prostoru a čase. Lze jich vymyslet ohromné množství. Lze je skutečně realizovat a pozorovat, jak se vyvíjejí.“

Toffoli a Margolus se zabývali analogií mezi teorií informatiky a fyzikálními zákony. Přišli na to, že celulární automaty jsou vhodný prostředek pro modelování fyzikálních jevů, které by se jinak modelovaly velmi obtížně.

2.2 Definice celulárního automatu

Celulární automat [2] je dán:

- pravidelnou sítí buněk pokrývající část n–dimenzionálního prostoru, anglicky *lattice*
- sadou $\mathbf{\Phi}(\vec{r},t) = \{\Phi_1(\vec{r},t), \Phi_2(\vec{r},t), \ldots, \Phi_m(\vec{r},t)\}$ stavů, které mohou nabývat hodnot z oboru reálných čísel, popisující pro každé místo $\vec{r}$ v síti jeho lokální stav v čase $t = 0, 1, 2, \ldots$
- pravidly (*rules*) $\mathbf{R} = \{R_1, R_2, \ldots, R_m\}$, která specifikují vývoj stavů $\mathbf{\Phi}(\vec{r},t)$ následujícím způsobem

$$\Phi_j(\vec{r},t) = R_j\left(\mathbf{\Phi}(\vec{r},t), \mathbf{\Phi}(\vec{r}+\vec{\delta}_1,t), \mathbf{\Phi}(\vec{r}+\vec{\delta}_2,t), \ldots, \mathbf{\Phi}(\vec{r}+\vec{\delta}_q,t)\right)$$

  kde $\vec{r}+\vec{\delta}_k$ popisuje okolí buňky $\vec{r}$. Pravidla bývají též nazývána lokálními přechodovými funkcemi.

- samotným okolím buňky (*neighborhood*)

Pro celulární automat jsou typické následující tři vlastnosti:

1. **paralelizmus** – všechny buňky jsou zpracovávány v jeden časový okamžik
2. **lokálnost** – pro každou buňku je důležitý pouze její vlastní stav a stavy buněk v jejím okolí
3. **homogenita** – na všechny buňky jsou aplikována stejná pravidla pro vývoj následujících stavů

2.3 Druhy rozdělení prostoru

Z definice je patrné, že CA se rozkládá v n–dimenzionálním prostoru. Pro simulace fyzikálních jevů se používají zejména 2D a 3D prostory, které lze snadno přenést do reálného světa. Trojrozměrný prostor představuje skutečný obraz fyzického světa a dvourozměrný prostor je pak rovinným řezem skutečného světa. Pro každý prostor a pro každou úlohu je vhodné využívat specifickou strukturu rozdělení buněk do mřížky a rovněž různé pojetí okolí buňky.

U 3D prostorů bývá prostor ve valné většině rozdělen na rovnoměrné krychle, u nichž je nejjednodušší souřadný systém pro lokalizaci buněk. V literatuře se lze také setkat s jinými systémy, např. kdy je prostor složen z pravidelných šestibokých hranolů, tak se tomu děje zejména při modelování růstu krystalů [9].

U 2D prostorů se objevuje více alternativ rozdělení plochy. Beze zbytku lze dvourozměrnou plochu rozdělit pomocí pravidelných trojúhelníků, čtverců a pravidelných šestiúhelníků. Nejčastěji se využívají právě poslední dvě možnosti, z nichž každá má své přednosti, ale i omezení. Výhodou čtvercové plochy je její jednoduchá indexace pomocí souřadnic x a y. Díky této jednoduchosti jsme schopni ji využít v mnohých modelech, kde nepotřebujeme takovou věrnost realitě jakou nabízí šestiúhelníková plocha. Ta poskytuje tu výhodu, že buňky mají mezi sebou stejnou vzdálenost ve všech směrech. Skutečné buňky organizmů nacházejících se v přírodě mají rovněž velmi často šestiúhelníkový tvar, což svědčí o efektivní topologii. Nevýhodou tohoto dělení prostoru je však to, že buňky je nutno indexovat výrazně složitěji a je nutno použít funkce pro přepočet souřadnic.

2.4 Druhy okolí buněk

Podíváme-li se do oblasti druhů okolí buněk, které jsou využívány pro evoluční pravidla, je zde patrná taktéž velká pestrost. U 1D automatů je nejpoužívanějším typem okolí, takzvané radiální okolí. Jedná se okolí, kde se buňka dívá napravo i nalevo na počet buněk udávaných poloměrem r, typicky s $r = 1$.

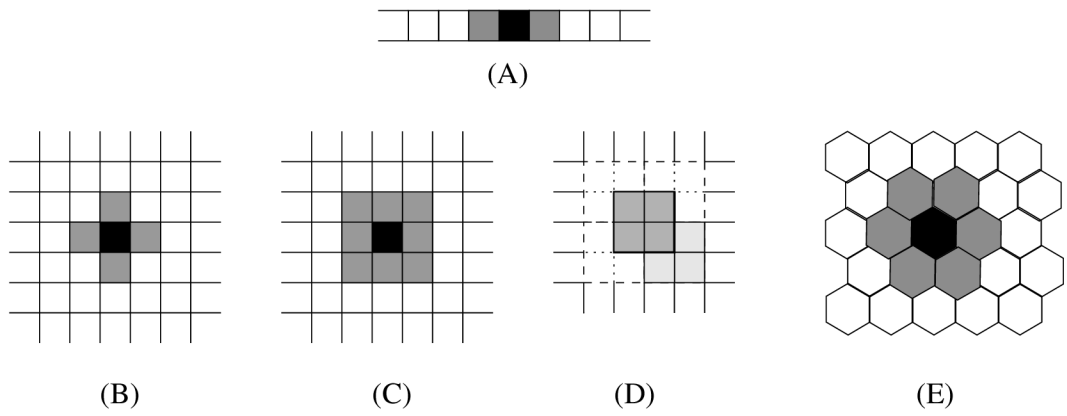
Pro 2D plochy a čtvercové rozložení buněk jsou nejčastějšími druhy okolí von Neumannovo, Moorovo a Margolusovo okolí (je možné vidět na obrázku 2.1). Von Neumannovo okolí je považováno za nejjednodušší a poskytuje analogii s geografickými směry (sever, jih, východ, západ). Toto rozložení bývá také nazýváno čtyřokolím buňky. V tomto okolí je středová buňka od okolních vzdálena vždy stejně. Moorovo okolí bylo odvozeno od von Neumanova okolí tím, že byly přidány buňky po obou diagonálách. Takto jsou popsány všechny buňky, které se zkoumané buňky dotýkají. Toto okolí bývá nazýváno osmiokolím nebo radiálním okolím s $r = 1$. Má-li se buňka dívat do větší vzdálenosti, obvykle se zvětšuje parametr r. Nevýhodou je, že buňky na diagonálách mají své středy více vzdáleny ke středu zkoumané buňky než buňky na horizontální a vertikální ose.

Margolusovo okolí se od dvou výše zmíněných typů významně liší. V předchozích typech jsme měli buňku se kterou se pracovalo a při tom jsme se pouze dívali na její okolí. U Margolusova okolí tomu je však úplně jinak. Plocha s buňkami je zde rozdělena na pravidelné čtverce o velikosti 2 x 2 buňky. Během evolučního pravidla se pracuje pouze s těmito čtverci buněk a hodnota se mění všem buňkám ve čtveřici zároveň. Čtverec je tedy sám sobě zkoumanou buňkou a zároveň i okolím. Další zvláštností je, že rozdělení na čtverce je rozdílné v sudých a lichých krocích časové iterace. Zatímco pro sudé kroky se rozdělování děje od počátku souřadnic, v lichých krocích je rozdělování provedeno s posunem jedné buňky ve směru x i y. Tohoto typu okolí je hojně využíváno při simulacích fyzikálních jevů, jakými

je pohyb plynu nebo sypání písku.

Pro šestiúhelníkové rozdělení plochy se nejčastěji využívá radiální okolí s poloměrem $r \geq 1$. V tomto případě plyne z topologie, že všechny buňky okolí jsou od středové buňky stejně vzdáleny. Takové rozdělení buněk můžeme často vidět u skutečných buněčných systémů a má tedy mnohem větší izotropii, než ostatní druhy okolí.

V mnoha případech lze v modelu použít své vlastní definované okolí. Například u dopravních modelů se zajímáme pouze o okolí, které leží na vozovce a které je ve vzdálenosti úměrné rychlosti vozidla. Záleží vždy na povaze modelu, který modelujeme.



Obrázek 2.1: (A)–radiální okolí u 1D automatu s $r = 1$; (B)–von Neumanovo okolí; (C)–Moorovo okolí s $r = 1$; (D)–Margolusovo okolí; (E)–šestiokolí u šestiúhelníkové mřížky

2.5 Wolframova pravidla a jejich třídy CA

Hovoříme-li o celulárních automatech a jejich spojení s fyzikálními jevy, musíme zákonitě zmínit jméno Stephena Wolframa. Stephen Wolfram je matematikem a fyzikem, kterému jednorozměrné celulární automaty poskytly prostředí pro studium vzniku komplexity. Zajímal se především o důsledek lokálních pravidel na globální systém. Šlo o to, že na náhodné uspořádání buněk byla aplikována určitá lokální pravidla a zkoumalo se, jak se bude automat vyvíjet v čase. Výsledkem je rozdělení pravidel do čtyř skupin:

1. automaty s těmito pravidly dospěly po několika krocích do stavu, kdy se řádek buď úplně vyprázdnil nebo úplně zaplnil

2. automaty s těmito pravidly se dostaly do stabilního stavu, kdy se jejich výstup po několika krocích periodicky opakoval a již se neměnil

3. tato pravidla způsobila to, že se automat dával za výstupy chaotické stavy, čili nebyla vidět žádná perioda ani systém ve výstupech

4. v této skupině pravidel se na výstupu automatu nepravidelně objevují a mizí některé periodické obrazce

Wolfram toto rozdělení do skupin provedl experimentálně, kdy zkoumal vlastnosti 1D automatů, které pracovaly s radiálním okolím s velikostí poloměru 1. Zkoumáním všech

možných pravidel, kterých je 2^8, čili 256, byla tyto pravidla rozdělena do čtyř výše uvedených skupin.

Z hlediska fyzikálních jevů je velmi důležitá zejména skupina 3. Tento význam spočívá zejména v tom, že příroda a tedy i fyzikální jevy jsou chaotické. I malá změna počátečních podmínek může ovlivnit celý výsledek modelu. Wolframova práce [10] se zabývá podrobným popisem vlastností CA a jejich vztahu k okolnímu světu.

2.6 Výhody CA v modelování fyzikálních jevů

Výhodu užití CA v oblasti modelování fyzikálních jevů si ukážeme na příkladu modelování dynamiky kapalin. Dříve se pro popis kapalin využívalo Navierových–Stokesových rovnic [12]. Ty se však na celý problém dívají příliš obecně a z makroskopického hlediska. Na kapalinu lze rovněž nahlížet z mikroskopického hlediska, kdy se díváme na kapalinu jako soubor molekul. Počet molekul v kapalině na jednotku objemu je však příliš obrovský a tato varianta je tedy výpočetně téměř nekonečná. CA nabízí v této oblasti optimální kompromis.

Kapalinu lze totiž chápat jako soubor jednotlivých částic, které se pohybují po symetrické mřížce a řídí se fyzikálními zákony, kdy se při srážce dvou částic zachovává energie a hybnost částic. Tyto modely se anglicky nazývají lattice gas, čili v českém překladu mřížový plyn. Bylo dokázáno, že takovéto modely s mřížovým plynem jsou skutečně aproximací Navierových–Stokesových rovnic. V mnoha případech tak získáváme skutečně realističtější výsledky. Celulárních automatů se využívá například při modelování směsí ropy a vody, prosakování ropy vápencovými a pískovcovými horninami, modelování chemických difuzí, výpočtech laminárního proudění, turbulencí a mnoha dalších jevů v kapalinách.

2.7 Metody výpočtu CA a jejich zrychlení

V následující části se budu zabývat metodami, kterými lze zefektivnit a zrychlit výpočet modelů tvořených celulárními automaty. Budou zde ukázány principy, které fungují při klasickém sekvenčním výpočtu a následně budou ukázány různé možnosti paralelizace výpočtu.

2.7.1 Sekvenční programování

Pustíme-li se do programování celulárních automatů klasickým sekvenčním způsobem zjistíme hned z definice podstatný problém. Buňky totiž mění svou hodnotu synchronně a v jednom časovém kroku. Všechny buňky musí rovněž pro svůj přechod použít svůj aktuální stav a aktuální stavy svých okolních buněk v čase t. Abychom toto mohli provést v sekvenčním programu, musíme učinit několik opatření.

Prvním opatřením je zavedení takzvaného kvaziparalelního výpočtu. Tímto přístupem jsme schopni simulovat synchronní změnu buněk. Jedná se o to, že na začátku každého kroku je zastaven modelovací čas. Posléze se projdou buňky na ploše sekvenčně jedna za druhou na každou se aplikují přechodová pravidla. Jakmile se projdou všechny je modelovací čas opět spuštěn a je k němu připočtena délka časového kroku τ. Podle modelovacího času to tedy vypadá, že se buňky změnily všechny najednou.

Druhým opatřením musíme zajistit, aby buňky měly časoprostorovou platnost, čili aby pracovaly s buňkami se správným časovým razítkem. Kdybychom tedy vzali první buňku, vypočítali její nový stav a přepsali touto hodnotou její stav, vznikl by nám problém. Její okolní sousedé by již neměli k dispozici hodnotu buňky v čase t ale v čase $t + \tau$. Obvyklým způsobem řešení je zavedení dvou buňkových map. Na první plochu se provede inicializace

buněk. Při prvním kroku simulace se prochází toto pole, ale výsledky se ukládají na stejnou pozici, avšak na druhou plochu. Na konci kroku tedy získáme celou plochu, na které jsou aktuální výsledky. Tuto plochu tedy prohlásíme za aktivní a plochu se starými hodnotami buněk využijeme v následujícím kroku opět na odkládání výsledků.

Máme-li plochu na které jsou řídce rozmístěny buňky, které mění svou hodnotu (typickým příkladem je Conwayova hra Life), můžeme využít následující mechanismus. V tomto případě totiž není nutné provádět množství výpočtů u buněk, které se nemůžou změnit. Stačí nám seznam buněk, které jsou aktivní a mění se spolu s jejich nejbližším okolím, u kterého je rovněž šance, že se během evolučního kroku změní. Průchodem tohoto seznamu změníme jenom nezbytně nutné buňky a zbytek zůstane nedotčen. Výsledky můžeme ukládat do dalšího seznamu, který posléze zkopírujeme do plochy.

U modelů, v nichž se provádí výpočet každé buňky tato metoda pozbývá smyslu, protože veškeré operace se seznamy by způsobily znatelné zpomalení chodu a rovněž by byla více náročnější na paměť. Tento přístup je vhodný tedy zejména pro automaty, kde je aktivní pouze část buněk, typickým příladem buď již zmíněná hra Life.

2.7.2 Paralelní programování

Celulární automaty jsou jasně z principu paralelní. Jak bylo popsáno na začátku, každá buňka představuje svůj vlastní svět, který se vyvíjí podle okolních stavů. Pro fyzikální modely, kde je buněk obrovské množství a v každém kroku je nutno přepočítávat nové stavy všech buněk, je paralelní a distribuované počítání vítanou možností, jak celý proces výpočtu urychlit. Většinou se používají SIMD nebo MIMD architektury [1].

SIMD architektura, čili „single instruction multiple data“, je vhodná zejména pro CA, kde jsou buňky aktivní po celou dobu chodu simulace. Pokud je však větší část buněk neměnná, dochází k tomu, že mnoho výpočetních jednotek zůstane neefektivních. Tento problém však u fyzikálních jevů popisující makroskopické děje odpadá, protože zde se musíme dívat neustále na celou plochu.

MIMD (multiple instruction multiple data) architektura je sice oproti SIMD více náročnější na vzájemnou komunikaci a synchronizaci procesorů, je však více univerzální a dostupná na současných počítačích. MIMD architektura v sobě pojímá jak multiprocesory tak i multipočítače.

Pokud se rozhodneme pro multiprocesory, čili paralelní výpočet, může být náš celulární automat tvořen tak, že na každém procesoru je proces, který se stará o výpočet určité části buněk. Jedná se o systém se sdílenou pamětí, ve které je uložena plocha, se kterou pracuje každý procesor zvlášť. Procesory tak nemusí komunikovat mezi sebou. Systém sdílené paměti však tvoří také omezení ve zrychlení, protože při větším množství procesorů vzniká problém s kapacitou sběrnice. Sběrnice totiž nestihne obsluhovat všechny procesory a ty musejí čekat na příchod dat.

Při distribuovaném výpočtu pomocí multipočítačů, nám problém se zahlcením sběrnice odpadá, protože počítače se vzájemně dorozumívají pomocí posílání zpráv. Paralelizmus je zde dán takzvaným SPMD modelem (single program multiple data). To znamená, že na každé výpočetní jednotce je spuštěn proces, který provádí přechodové funkce na předem dané oblasti buněk. Každá výpočetní jednotka spolupracuje s okolními jednotkami, aby zjistila hodnoty buněk, které jsou mimo její hranice. Systém tedy nemá sdílenou paměť a každá jednotka pracuje autonomně. Pomocí zpráv musíme zajistit společnou synchronizaci, aby měly všechny jednotky společný modelový čas.

2.7.3 Specializovaný hardware

Tato část je uvedena pouze pro úplnost metod, kterými se zrychlují výpočty celulárních automatů. Nebudu se zabývat velkými podrobnostmi, protože tato část je samostatnou odbornou činností hodnou jiných odborníků.

Celulární automat může být implementován i jako samostatný hardware, kde buňky či skupinu buněk představují samostatné výpočetní jednotky, které jsou se svým okolím navzájem fyzicky propojeny. Typickým příkladem takovéto architektury je CAM (cellular automata machine), který v 80. létech navrhnuli T. Toffoli a N.H. Margolus. CAM nabízí možnost implementovat mnoho druhů CA a bylo jej využito k mnoha komplikovaným výpočtům (zejména u 3D modelů) díky jeho velké výpočetní síle, kterou nabízí. Limitním omezením je velikost automatu, který chceme modelovat a počet stavů, kterých mohou buňky dosahovat.

Kapitola 3

Modelování fyzikálních jevů

Následující kapitola popisuje jednotlivé oblasti fyziky, v nichž našly své uplatnění CA. Pro každou oblast jsou vyjmenovány a popsány nejdůležitější druhy přechodových pravidel, které se pro modelování daných fyzikálních jevů používají.

3.1 Pohyb plynu a fluidní dynamika

V této sekci se budeme zajímat o základních metodách, které se věnují pohybu částic v plynu, které se pohybují po předem dané mřížce. Díky tomu bývají v literatuře označeny jako „lattice gas“, čili česky mřížkový plyn.

3.1.1 HPP pravidla

HPP pravidla představují základní metodu, ze které jsou odvozeny mnohé další metody [2]. Hlavním předpokladem této metody jsou bodové částice, které se pohybují ve vymezené čtvercové mřížce. Tento model představuje diskrétní molekulární dynamiku. Smysl takto postaveného modelu je simulace vzájemných kolizí v plynu. K těm dochází ve skutečném světě při uplatnění přírodních zákonů, především zákonu o zachování energie.

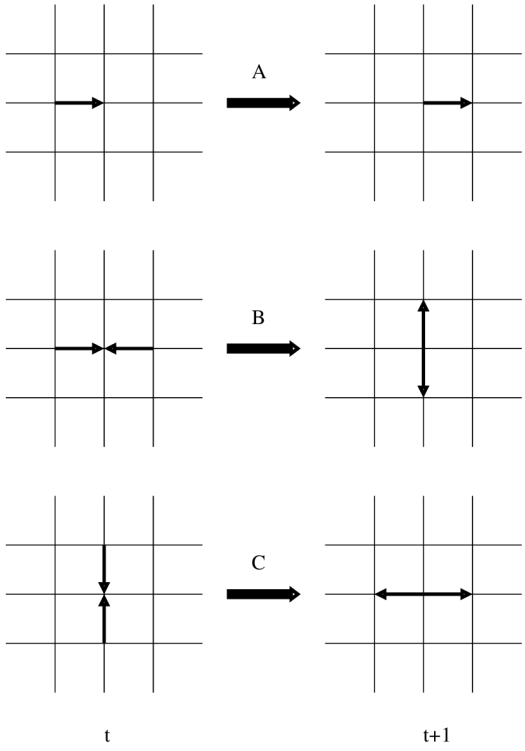
Pro evoluční pravidla je v tomto případě typické, že se skládají ze dvou kroků: kolize a pohybu. Fáze kolize nám udává, jak se částice, které vstupují do bodu kolize zachovají, čili jak změní své směry letu. Během pohybové fáze, často též označované propagací, se částice pohnou jedním krokem ve směru, který získali při kolizi. Na obrázku 3.1 lze vidět, jak mohou HPP pravidla vypadat. Z pravidla B a C je vidět, že je zachováván zákon o zachování energie. Dvě částice, které do sebe čelně narazí, změní směr v kolmé ose, ale hybnost obou částic je zachována.

Z implementačního hlediska označíme jako buňky automatu místa, kde se kříží mřížka a kde dochází ke kolizím. Stav takové buňky $\vec{r}$ v čase t můžeme označit třeba za využití Booleovské reprezentace jako $s(\vec{r}, t) = (0, 1, 0, 1)$. Tento stav znamená, že do buňky míří částice ze směru 2 a ze směru 4 (např. východ a západ). Přechodová pravidla kolizí dvou částic pro tento typ zápisu lze pak napsat jako:

$$(1010) \rightarrow (0101), (0101) \rightarrow (1010)$$

Tato pravidla říkají, že při čelní kolizi změní částice směr v kolmé ose. Všechny ostatní srážky se obejdou bez změny směru částic.

Zajímavou vlastností těchto pravidel je, že jsou reverzibilní. Jsme tedy schopni pro jistý stav systému v čase t analyzovat, jak se systém vyvíjel v čase. V analyzovaném systému se



Obrázek 3.1: A – jednoduchý pohyb částice; B,C – kolize částic ve vertikální a horizontální ose

ale nesmí vyskytovat jediná chyba. Změna orientace byť jedné částice by zapříčinila, že se systém nedostane do stejného stavu, jaký byl v čase t_0.

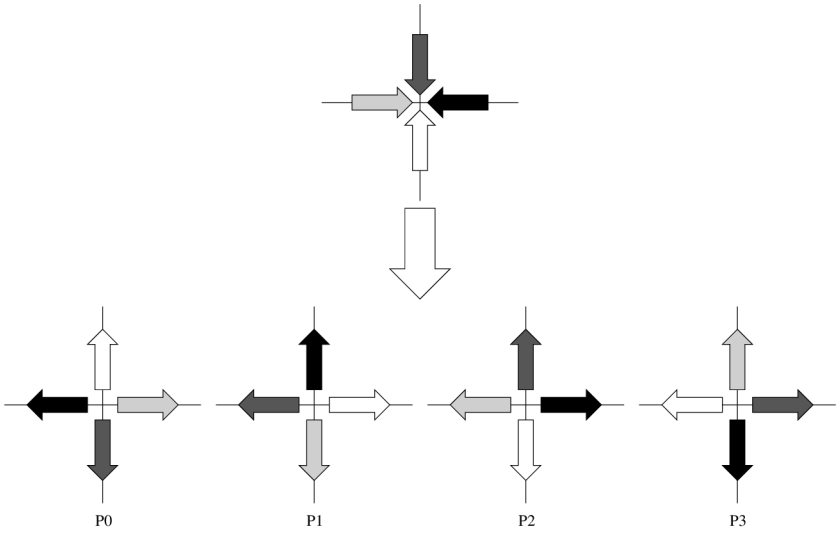
Schopnosti přesně popsat reálný plyn pomocí HPP pravidel jsou však velmi malé díky velké abstrakci. Ta je uplatňována jak při dělení prostoru na čtverce, tak při neakceptování možných rozdílných fyzikálních vlastností částic. Metoda dvoufázového přechodového pravidla však tvoří přínos, který se uplatňuje i v jiných metodách.

Zajímavou obměnou této metody je využití pravděpodobnosti. Tato pravděpodobnost se uplatní při srážce dvou částic, kdy již není striktně dáno, jak změní částice směr, ale změna směru je otázkou náhody. Na obrázku 3.2 je možné vidět, jak může vypadat obecná definice takového kolizního pravidla. Touto obměnou ztrácí metoda svou reverzibilitu, ale více se přibližuje skutečnému světu. Takto upravené metody se často využívá např. při modelování difuze.

3.1.2 FHP pravidla

FHP pravidla jsou modelem dvoudimenzionální kapaliny, který vyvinuli v roce 1986 pánové Frisch, Hasslacher a Pomeau [2]. Model opět popisuje diskrétní částice, které se pohybují po předem dané pevné mřížce. Mezi částicemi znovu dochází ke kolizím na místech křížení mřížky. Hlavním rozdílem oproti HPP pravidlům je to, že mřížka po které se pohybují částice je hexagonální z důvodů větší izotropie.

FHP pravidla jsou abstrakcí skutečné kapaliny na mikroskopické úrovni. Předpokládají tedy platnost všech důležitých zákonů, které jsou uplatňovány v reálné kapalině. Ty jsou zosobněny spojitostí a Navier–Stokesovými rovnicemi hydrodynamiky [12], čili zákonu o za-



Obrázek 3.2: Ukázka pravděpodobnostního pravidla HPP, kdy výběr závisí na pravděpodobnostech P0, P1 ,P2 a P3.

chování hmoty a zákonu o zachování energie. Základem je tedy zachování částic a energie po každé změně stavu.

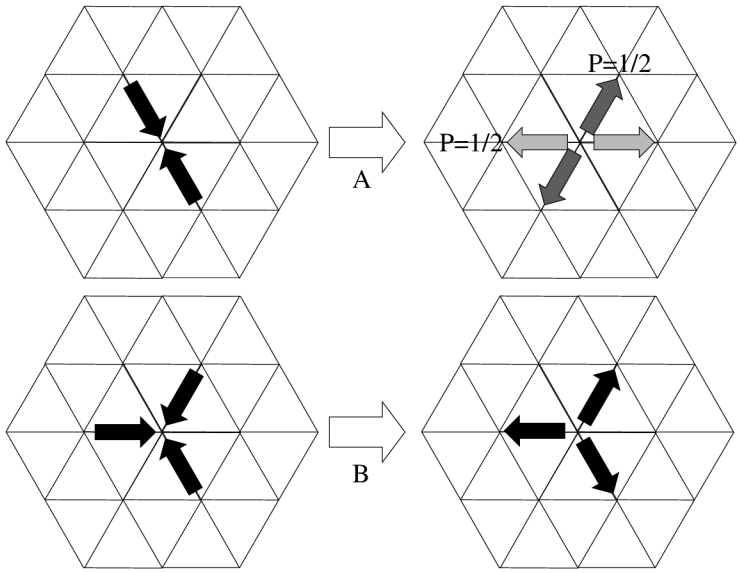
Stejně jako u HPP pravidel je u FHP modelu možné popsat stav každé buňky (kolizního bodu) jako sadu Booleovských proměnných, které představují částice, které míří do tohoto bodu v čase t. Všechny částice se pohybují konstantní rychlostí v šesti možných směrech. Z principu je dáno, že na jednom místě se může vyskytovat pouze jedna částice pohybující se ve stejném čase stejným směrem.

FHP pravidla rovněž přejímají od HPP pravidel způsob dvoufázových přechodových pravidel. Nejdříve jsou tedy vyřešeny kolize bodů a v následující fázi, jsou částice posunuty o jeden krok v novém směru určeném kolizí. Podíváme-li se na problém kolizí, vznikají nám dvě možnosti přímých střetů. První možností je, že se 2 částice setkají ve vzájemném úhlu 180 stupňů. Grafické znázornění tohoto střetu je patrné na obrázku 3.3. Částice jsou odhozeny s posunem o 60 stupňů. Přesný směr je dán symetricky pravděpodobností p. Druhým druhem komplikovaného střetu je, když se střetnou 3 částice se vzájemnými úhly 120 stupňů. V tomto případě se částice odrazí zpět.

U obou případů je patrné, že před i po srážce zůstala celková energie soustavy rovna nule. Při komplikovanějších srážkách je možné situaci řešit tak, že jsou vyjmuty a odděleně vyřešeny částice, které spolu mají vzájemně nulovou energii a pak až následně vyřešíme zbylé částice. Je nezbytně nutné dodržet při tomto pravidlo o jedné možné částici pohybující se na stejném místě stejným směrem.

Chceme-li popsat úplnou mikrodynamiku FHP modelu, můžeme tak učinit pomocí přechodového pravidla. Označme částice vstupující do kolizního místa$\vec{r}$ v čase t pomocí $n_i(\vec{r}, t); i = 1, 2, ..., 6$, kde n_i nabývá hodnot 0 nebo 1 a kde je u každé částice znám vektor směru pohybu částice $\vec{c}_i; i = 1, 2, ..., 6$ viz obrázek 3.4. Dále označíme časový krok jako τ a krok na mřížce jako λ. Z těchto hodnot jsme schopni vypočítat rychlost částice

$$\vec{v}_i = \frac{\lambda}{\tau}\vec{c}_i \tag{3.1}$$

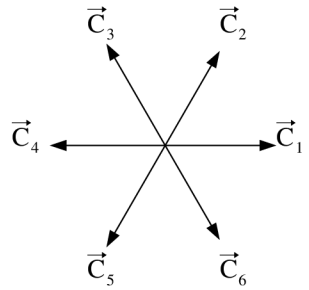


Obrázek 3.3: A– srážka dvou částic, B–srážka tří částic

Přechodové pravidlo bez kolizí mezi částicemi bychom byli schopni napsat jako

$$n_i(\vec{r} + \lambda\vec{c}_i, t + \tau) = n_i(\vec{r}, t) \tag{3.2}$$

což znamená, že každá částice se posune v časovém kroku τ o jeden krok λ ve směru daném $\vec{c}_i$ a bude tedy vstupovat do buňky $\vec{r} + \lambda\vec{c}_i$. Během kolizí však dochází ke změně hodnot $\vec{c}_i$.



Obrázek 3.4: Směry pohybu částic

Mějme případ čelní srážky dvou částic, čili pouze n_i a n_{i+3} mají hodnotu 1. Tehdy částice pohybující se rychlostí $\vec{v}_i$ změní svou rychlost na $\vec{v}_{i-1}$ nebo $\vec{v}_{i+1}$ (index samozřejmě nabývá hodnot od 1 do 6). Potom

$$D_i = n_i n_{i+3}(1 - n_{i+1})(1 - n_{i+2})(1 - n_{i+4})(1 - n_{i+5}) \tag{3.3}$$

udává, že když $D_i = 1$, pak kolize bude mít volné místo a hodnota $n_i - D_i$ udává, kolik částic zmizí ve směru $\vec{c}_i$ po srážce.

Má-li výsledek srážky dvou částic místo, rozhoduje se podle náhodné Booleovské proměnné $q(\vec{r}, t)$, na kterou stranu se odrazí. Jestli doprava ($q = 1$) nebo doleva ($q = 0$). Poté počet částic vygenerovaných ve směru $\vec{c}_i$ je dán

$$qD_{i-1} + (1 - q)D_{i+1} \tag{3.4}$$

Vezměme v úvahu rovněž srážky tří částic, kdy rovněž dochází ke změně směru $\vec{c}_i$. Míra, která detekuje výskyt takovéto kolize je dána vzorcem

$$T_i = n_i n_{i+2} n_{i+4}(1 - n_{i+1})(1 - n_{n+3})(1 - n_{i+5}) \tag{3.5}$$

Počet částic, které změní směr ve směru $\vec{c}_i$ je

$$n_i - T_i + T_{i+3} \tag{3.6}$$

Na základě vzorců 3.2, 3.4 a 3.6 jsme schopni sestavit konečné pravidlo udávající vývoj částic v mikrodynamickém FHP modelu:

$$\begin{aligned} n_i(\vec{r} + \lambda\vec{c}_i, t + \tau) = \quad & n_i(\vec{r}, t) \\ & -D_i + qD_{i-1} + (1 - q)D_{i+1} \\ & -T_i + T_{i+3} \end{aligned} \tag{3.7}$$

Chceme-li postoupit od mikroskopického světa do makroskopického, musíme si uvědomit důležitou skutečnost. V makroskopickém pohledu na plyn nás nezajímá fyzická existence nějaké částice v prostoru, nýbrž jen jistá průměrná hodnota. Tou mohou být např. průměrná hustota částic a vektor průměrné rychlosti, které se nacházejí v každém bodě systému. Tyto hodnoty vycházejí ze základního průměru $N_i(\vec{r}, t) = \langle n_i(\vec{r}, t)\rangle$, kde $N_i(\vec{r}, t)$ de facto představuje pravděpodobnost přítomnosti částice v místě $\vec{r}$, v čase t s rychlostí $\vec{v}_i = (\lambda/\tau)\vec{c}_i$. Hustotu částic můžeme vyjádřit vzorcem

$$\rho(\vec{r}, t) = \sum_{i=1}^{6} N_i(\vec{r}, t) \tag{3.8}$$

Zprůměrováním vzorce 3.7 získáme přechodové pravidlo

$$N_i(\vec{r} + \lambda\vec{c}_i, t + \tau) = N_i(\vec{r}, t) + \langle\Omega_i\rangle \tag{3.9}$$

kde

$$\langle\Omega_i\rangle = -\langle T_i\rangle + \langle T_{i+3}\rangle - \langle D_i\rangle + 0,5\langle D_{i-1}\rangle + 0,5\langle D_{i+1}\rangle \tag{3.10}$$

Hodnota 0,5 vychází z průměrné hodnoty $\langle q\rangle = 1/2$, kdy se částice odrážejí do různých směrů se stejnou pravděpodobností. Rozvedeme-li vzorec 3.10 pomocí Taylorova rozvoje druhého řádu získáme

$$\langle\Omega_i\rangle = \tau\partial_t N_i + \lambda(\vec{c}_i \cdot \nabla)N_i + \frac{\tau^2}{2}{\partial_t}^2 N_i + \frac{\lambda^2}{2}(\vec{c}_i \cdot \nabla)^2 N_i + \lambda\tau(\vec{c}_i \cdot \nabla)\partial_t N_i \tag{3.11}$$

Tento vzorec lze pak následně upravovat za pomocí jiných vzorců, jakými jsou Boltzmannova rovnice, Navier–Stokesova rovnice a jiných rovnic, které zavádějí například vliv hustoty a viskozity kapaliny. Tento postup je velmi matematicky náročný a je uveden v [2] na stranách 81 – 104. Výsledkem je vztah

$$-\langle\Omega_i\rangle = \frac{\rho}{6}\left(1 - \frac{\rho}{6}\right)^3 \tag{3.12}$$

Se systémem FHP pravidel lze dále pracovat například pro pohyb teplého plynu, kdy má každá částice jinou hmotnost a rychlost, čili i jinou kinetickou energii. Od tohoto se odvíjejí další pravidla a způsoby řešení přechodových pravidel. Tyto modely se nazývají multispeed, čili vícerychlostní.

3.1.3 Boltzmannova mřížka

Boltzmannova mřížka (anglicky též Lattice Boltzmann) představuje model, který se výrazně podobá makroskopické části FHP pravidel. FHP pravidla se však specifikovala pouze na hexagonální rozložení plochy. Boltzmannova mřížka je oproti tomu více univerzální a lze je použít pro více způsobů dělení plochy, pro více druhů okolí a pro více dimenzí plochy.

Základem této metody je proměnná $f_i^{in} = \langle N_i \rangle \in [0; 1]$, která představuje pravděpodobnost, že je v kolizním bodě $\vec{r}$, v diskrétním čase t_i fiktivní částice s rychlostí v_i. Tato míra bývá rovněž označována jako hustota částic nebo rozložení hustoty. Základní vzorec evolučního pravidla tedy zní

$$f_i^{in}(\vec{r} + \lambda\vec{v}, t + \tau) = f_i^{in}(\vec{r}, t) + \Omega_i(f_0^{in}, \cdots, f_z^{in}) \tag{3.13}$$

Vzhledem k tomu, že kolizní operátor Ω_i není Booleovská proměnná, nýbrž reálná proměnná, musíme vzít v úvahu, že každý výpočet pro každou buňky zabere velké množství matematických operací s pohyblivou řádovou čárkou, což simulaci neúměrně prodlužuje. Bylo třeba tedy najít jiný způsob efektivního výpočtu kolizí. To se podařilo, když byl kolizní operátor linearizován kolem řešení lokálního ekvilibria, které může nastat v kolizním bodě. Pak můžeme přepsat rovnici 3.13 jako relaxační rovnici

$$f_i^{in}(\vec{r} + \lambda\vec{v}, t + \tau) = f_i^{in}(\vec{r}, t) + \frac{1}{\xi}\left(f_i^{eq}(\vec{r}, t) - f_i^{in}(\vec{r}, t)\right) \tag{3.14}$$

kde ξ představuje dobu relaxace – dosažení ekvilibria ($\xi > 0,5$). f_i^{eq} představuje lokální ekvilibrium a je dáno hustotou

$$\rho(\vec{r}, t) = \sum_{i=0}^{z} m_i f_i^{in}(\vec{r}, t) \tag{3.15}$$

a rychlostí toku kapaliny $\vec{u}(\vec{r}, t)$, která je dána hybností

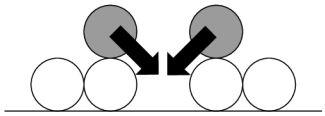
$$\rho(\vec{r}, t)\vec{u}(\vec{r}, t) = \sum_{i=0}^{z} m_i f_i^{in}(\vec{r}, t)\vec{v}_i \tag{3.16}$$

V obou rovnicích se vyskytuje m_i, což představuje váhu mřížky. Tato váha je ve vzorci dána, aby zajistila izotropii zvolené mřížky. Máme-li hexagonální mřížku a za sousedy bereme nejbližší buňky, je váha nastavena na hodnotu 1, protože od středové buňky jsou všichni sousedé stejně vzdáleni. Pro systém, se kterým se nejčastěji počítá na počítačích jsou váhy rozloženy jinak. Tento systém bývá nazýván D2Q9, což znamená dvoudimenzionální prostředí využívající 9 směrů šíření (směr na buňky samotnou, 4 směry diagonální a 4 směry v osách x a y). Zde je v horizontálním a vertikálním směru dána váha 4 a v diagonálních směrech dána váha 1. Toto rozdělení vede z rozdílných velikostí rychlostních vektorů.

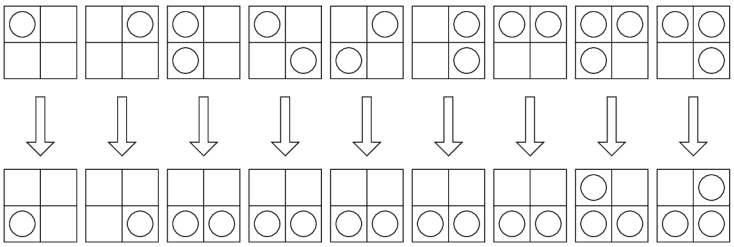
Odvozením byla zjištěna funkce pro výpočet ekvilibria

$$\begin{aligned} f_i^{eq} &= \rho\frac{C_4}{{C_2}^2}\left[1 + \frac{v_{i\alpha}u_\alpha}{c_s^2} + \frac{1}{2}\left(\frac{v_{i\alpha}u_\alpha}{c_s^2}\right)^2 - \frac{\vec{u}^2}{2c_s^2}\right] \\ f_0^{eq} &= \rho\left(1 - \frac{C_0C_4}{C_2^2}\right)\left(1 - \frac{\vec{u}^2}{2c_s^2}\right) \end{aligned} \tag{3.17}$$

kde hodnota c_s představuje rychlost zvuku. Konstanty C_0, C_2 a C_4 jsou odvozeny od druhu mřížky a pro jejich výpočet je použita Kroneckerova funkce, proto se pro náš případ se systémem D2Q9 spokojíme s tím, že $C_0 = 20, C_2 = 12$ a $C_4 = 4$. Index α, který je ve vzorcích užíván označuje dimenzi, ve které hodnotu počítáme. Odvození vzorců a konstant, které byly výše uvedeny je možno nalézt v [5].



Obrázek 3.5: Konflikt dvou zrnek



Obrázek 3.6: Pravidla pro sypání písku pomocí Margolusova okolí

3.2 Pravidla pro sypání písku

Vezmeme-li písek, či jiný materiál, který se skládá z jemných zrnek majících přibližně stejný tvar a vlastnosti, zjistíme, že se jedná o velmi vhodný systém pro modelování pomocí celulárních automatů. Při sypání totiž dochází k interakci pouze mezi nejbližšími buňkami, což je odpovídající mechanismus CA. Našim cílem je tedy vytvořit taková pravidla, která by tento děj jednoduše popsala.

Podíváme-li se na skutečný písek, musíme si uvědomit, že v modelu musíme zanedbat několik skutečností. Definice CA nám totiž říká, že buňky máme rozložené do pravidelné mřížky. Mezi zrnky písku však vzniká mřížka, která je nepravidelná, což má značný vliv na celkovou stabilitu zrnek. Skutečný písek je oproti tomu počítačovému silně nestabilní díky své komplexitě a teorii chaosu, která se na něj uplatňuje. V počítačovém modelu tvořeném na čtvercové mřížce můžeme definovat, že zrnko je stabilní, pokud má pevný základ, čili jeho tři spodní sousedé obsahují také zrnko. Při absenci jakéhokoliv zrnka v základech je již zkoumané zrnko nestabilní a v následujícím kroku by se mělo podle toho patřičně převrhnout.

Při převrhnutí zrnka se nám objevuje další problém a tím jsou případné konflikty. Podíváme-li se na obrázek 3.5, kde je znázorněna situace, kde oba zrnka mají tendenci spadnout na stejné místo. Pokud bychom brali v úvahu Moorovo okolí buňky, kdy se každá buňka podívá na stav svého okolí v současnosti a podle něj se posune, vznikl by nám problém, že by se obě buňky přesunuly na jedno místo a de facto by nám jedno zrnko zmizelo. Toho ale nesmíme dopustit. Nachází se zde možnost, že by buňka viděla, jak se zachovaly okolní buňky a podle toho by změnila či nezměnila svůj stav. Toto však naráží na časovou kontinuitu mezi po sobě následovanými kroky. Je nemožné, aby se buňka dívala do budoucna na výsledky okolních posunů.

Řešení této situace se našlo ve využití Margolusova okolí, kdy je celé prostředí rozděleno na malé oblasti, na něž se aplikují přechodová pravidla, ve kterých nemůže ke kolizím docházet. Vzhledem k tomu, že rozdělení je různé mezi sudými a lichými kroky, nemůže se stát, že by se řešení zarazilo v lokálním extrému. Pravidla, která se využívají při této metodě lze vidět na obrázku 3.6 (převzato z [2]).

Velmi zajímavým prvkem, který nám umožňuje výrazně pozměnit charakter modelu,

je zavedení stochastických pravidel. V tomto případě můžeme pomocí pravděpodobnosti v pravidlech pozměnit některé vlastnosti materiálu. Můžeme tak do jisté míry simulovat například vlhkost materiálu, různou zrnitost, apod.

3.3 Šíření vlnění

Vlnění se nachází všude kolem nás a setkáváme se s ním denně, i když si to mnohdy ani neuvědomujeme. Obklopuje nás světlo, zahřívá nás slunce, přenášíme informace pomocí rádia. Vše jsou různé druhy elektromagnetického vlnění. Na druhou stranu když sledujeme kolečka rozbíhající se od vhozeného kamenu do vody a posloucháme typické šplouchání vody mnohdy si ani neuvědomíme, že se jedná o mechanické vlnění.

Samotné vlnění a jeho šíření je popsáno mnoha rovnicemi a zákony, jakými jsou například Maxwellovy rovnice, Gaussův zákon, Ampérův zákon, Coulombův zákon, aj. Většina těchto zákonů jsou složité rovnice druhého řádu, což dokazuje například obecná matematická rovnice pro síření skalární veličiny ψ rychlostí c

$$\partial_t^2 \psi - c^2 \nabla^2 \psi = 0 \tag{3.18}$$

která však nepopisuje děje, které se dějí na pozadí vlnění jakými jsou interference, odrazy, apod.

Rozdělíme-li prostor na malé části, jsme schopni vypočítat malé odchylky, které se pohybují po mřížce. Ty přicházejí buňce vždy od nejbližších sousedů a tak ji ovlivňují. Pro popis takového modelu se ukázalo jako nejvhodnější využít princip Boltzmannovy mřížky. Stav takového systému lze popsat sadou reálných hodnot f_i, které se pohybují prostorem směrem $\vec{v}_i$ a doplňkovým zbytkovým tokem f_0 (tzn. $\vec{v}_0 = 0$). Můžeme tedy použít vzorec 3.14. Našim cílem je tedy najít ekvilibrium f_i^{eq}, které má pro vlnění jiný charakter než pro fluidní dynamiku [7].

Pro toto si musíme zavést dvě proměnné I a $\vec{J}$

$$I = \sum m_i f_i \qquad \vec{J} = \sum m_i f_i \vec{v}_i \tag{3.19}$$

kde m_i označuje váhu mřížky obdobně jako tomu bylo u fluidní dynamiky. Nyní se ale budeme zabývat případem, kdy se vlny šíří pouze do čtyř stran a tedy váha na všechny strany je rovna 1. Ekvilibrium jsme pak schopni spočítat jako

$$f = a_i I \ + b_i \frac{\vec{v}_i \cdot \vec{J}}{2v^2} \tag{3.20}$$

kde a_i a b_i jsou konstanty, které musí být voleny s ohledem na I a $\vec{J}$ a $v^2 = (\lambda/\tau)^2$ je modulem $\vec{v}_i$.

Zákony o zachování hmoty a energie mohou být dodrženy, když

$$m_i = m \quad a_i = a \quad b_i = b \quad pro \quad i = 1, 2, 3, 4 \tag{3.21}$$

a

$$a_0 m_0 + 4ma = 1 \quad b_0 = 0 \quad b = \frac{1}{m} \tag{3.22}$$

Dosazením těchto hodnot a hodnot z vzorce 3.20 získáme přechodové pravidlo:

$$\begin{aligned} f_0(\vec{r}, t+\tau) = &\ \tfrac{1}{\xi}\left[(a_0 m_0 + \xi - 1) f_0 + a_0 m \textstyle\sum_{i=1}^{4} f_i\right] \\ f_i(\vec{r} + \tau\vec{v}_i, t+\tau) = &\ \tfrac{1}{\xi}\left[\left(am + \xi - \tfrac{1}{2}\right) f_i + a m f_{i+1} + \left(am - \tfrac{1}{2}\right) f_{i+2} + a m f_{i+3} + a m_0 f_0\right] \end{aligned} \tag{3.23}$$

Je obvyklé tato pravidla převádět do maticového tvaru

$$\begin{pmatrix} f_0(r, t\vec{+}\tau) \\ f_1(\vec{r} + \tau\vec{v}_1, t+\tau) \\ f_2(\vec{r} + \tau\vec{v}_2, t+\tau) \\ f_3(\vec{r} + \tau\vec{v}_3, t+\tau) \\ f_4(\vec{r} + \tau\vec{v}_4, t+\tau) \end{pmatrix} = W \begin{pmatrix} f_0(\vec{r}, t) \\ f_1(\vec{r}, t) \\ f_2(\vec{r}, t) \\ f_3(\vec{r}, t) \\ f_4(\vec{r}, t) \end{pmatrix} \tag{3.24}$$

Zvolíme-li ve vzorci 3.23 hodnotu $\xi = \frac{1}{2}$ a určíme-li, že zanedbáme zbytkovou část ($f_0 = 0,\ a_0 = 0$) získáme tím základní matici, kterou je definováno její posouvání

$$W = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ \frac{m_0}{2m} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{m_0}{2m} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{m_0}{2m} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{m_0}{2m} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{pmatrix} \tag{3.25}$$

U vlnění platí několik zásadních zákonů, které musíme vzít v úvahu, chceme-li modelovat skutečné situace. Prvním z nich je fakt, že vlnění se odráží od pevných překážek. Pro tuto situaci platí pravidlo

$$f_i(\vec{r}_i + \tau\vec{v}_i, t+\tau) = \sum_{j=0}^{4} R_{ij} f_i \tag{3.26}$$

kde

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{3.27}$$

Další vlastností vln je jejich lom, kdy vlny prochází z jednoho prostředí do druhého. To je patrné například při průchodu světla čočkou, kde na přechodu mezi vzduchem a sklem probíhá lámání paprsků. Pro toto byla odvozena přechodová matice

$$W = \frac{1}{2n^2} \begin{pmatrix} 2n^2 - 4 & 2\sqrt{n^2-1} & 2\sqrt{n^2-1} & 2\sqrt{n^2-1} & 2\sqrt{n^2-1} \\ 2\sqrt{n^2-1} & 1 & 1 & 1 - 2n^2 & 1 \\ 2\sqrt{n^2-1} & 1 & 1 & 1 & 1 - 2n^2 \\ 2\sqrt{n^2-1} & 1 - 2n^2 & 1 & 1 & 1 \\ 2\sqrt{n^2-1} & 1 & 1 - 2n^2 & 1 & 1 \end{pmatrix} \tag{3.28}$$

kde $n \geq 1$ představuje index lomu.

Těmito pravidly můžeme zkoumat věci běžného života jakými jsou průchod vlnění čočkou, odraz vlnění na konvexním či konkávním zrcadle až po šíření rádiových vln od antény v místech městské aglomerace. Principem, jak daný model popsat, je neuniformní rozdělení

pravidel. Na buňkách, které představují volný prostor se použije matice, která určuje pohyb částic. Na místech, kde se vyskytují jevy jako odraz či lom se použije jiná přechodová matice. Navíc je zde možnost celou matici vynásobit určitou konstantou $\alpha < 1$, která nám definuje ztrátu energie, která nastává například u odrazu od stěny budovy. Přesné odvození výše zmíněných matic z fyzikálních vzorců lze nalézt v [2, 7].

V publikacích [2, 7] je definován ještě jeden možný způsob modelování šíření vlnění, zejména toho rádiového. Tento model je založen rovněž na čtyřech energetických hodnotách ve čtyřech směrech. Zbytková energie je v tomto modelu zanedbána již na začátku. Přechodová pravidla jsou pak dána vzorcem

$$f_i(\vec{r} + \vec{v}_i, t + 1) = c_1(\vec{r}) \left(c_2(\vec{r}\psi(\vec{r}, t) - f_{i+2}(\vec{r}, t))\right) \tag{3.29}$$

kde

$$\psi(\vec{r}, t) = f_1(\vec{r}, t) + f_2(\vec{r}, t) + f_3(\vec{r}, t) + f_4(\vec{r}, t) \tag{3.30}$$

Hodnota ψ zde představuje integrál energie přes všechny směry. Konstanty c_1 a c_2 představují hodnoty, které popisují danou buňku na pozici $\vec{r}$. Těmito konstantami můžeme definovat, zda se bude buňka chovat jako volný prostor, kterým se šíří vlnění, jako zeď, od kterého se vlnění odráží s jistou ztrátou energie, nebo jako buňka, která záření pohlcuje. Konkrétní hodnoty těchto konstant budou uvedeny v implementační části k danému modelu.

3.4 Ostatní typy fyzikálních jevů

Existuje ještě velké množství kategorií a typů fyziky a fyzikálních jevů, na jejichž modelování lze celulární automaty využít. Podrobné prostudování a pochopení všech těchto kategorií přesahuje rámec této diplomové práce, proto v následující části budou uvedeny alespoň rámcové okruhy, kterými se lze v této oblasti zabývat popřípadě rozšířit jimi aplikaci pro modelování fyzikálních jevů.

První oblastí, kterou je možné se zabývat a která přímo navazuje na výše popsané děje v plynech je fluidní dynamika. Dříve byly představeny pouze děje v plynech a to velmi zjednodušeně. V [5, 2] se tyto základní principy rozvádějí do větších a podrobnějších teorií, kterými lze modelovat takové věci jakými jsou turbulence, vodní proudy, aj. Takováto oblast modelů je velmi důležitá například v oblasti leteckého průmyslu.

Další oblastí, kde se rovněž uplatňují principy Boltzmannovy mřížky jsou takzvané reakčně–difuzní systémy. V [2, 11, 3] jsou uvedeny definice a jejich odvození. V principu se jedná o systémy, které jsou složeny s několika látek, které spolu vzájemně různě reagují. Těchto modelů se používá zejména v oblasti chemie, kde modelují reakce popsané parciálními diferenciálními rovnicemi.

Další kategorií modelů využívající celulární automaty jsou modely růstu krystalů. V těchto modelech se pracuje s různými mřížkami, podle druhu krystalu, který chceme vytvářet. Chceme-li modelovat například růst sněhové vločky, zvolíme hexagonální mřížku. Krystaly se tvoří tak, že na hranách zárodečné buňky se s využitím difuze a pravděpodobnosti generují další buňky a takto začne krystal růst. To jak krystal vyroste ovlivňují různé parametry jako třeba onen difuzní koeficient. Blíže je možno seznámit se např. v [9].

Poslední zmíněnou oblastí je mechanika pevných a pružných těles. Zde se jedná o systémy, kde mezi buňkami jsou vytvořeny pružné vazby. Buňky tedy mohou měnit svou vzdálenost a napětí respektive sílu, která mezi těmito buňkami působí. Takovýmto principem můžeme modelovat například deformace a odrazy pružných těles, jakými jsou gumy nebo

pěny. Rovněž můžeme zkoumat například zlomy a trhliny, které nám vznikají u pevných těles. Bližší informace jsou uvedeny v [8].

Kapitola 4

Návrh aplikace

Jedním z cílů mé práce je vytvoření aplikace, která umožní simulaci jednotlivých modelů fyzikálních jevů. Následující kapitola popisuje návrh jednotlivých částí této aplikace. Vzhledem k odlišnostem jednotlivých modelů nemůže být simulátor univerzální. Požadavkem je ale to, aby aplikace byla pouze jedna a mohli bychom pomocí ní simulovat jednotlivé různé modely. Řešením této situace je vytvoření modelu jako samostatné třídy, kde bude uložena plocha i metody simulátoru. Tuto třídu bude aplikace schopna dynamicky nahrát a komunikovat s ní pomocí několika předdefinovaných metod tvořící rozhraní. Každý model tedy bude zásuvným modulem – pluginem. V následující kapitole budou popsány jednotlivé části modelů i aplikace.

4.1 Plocha

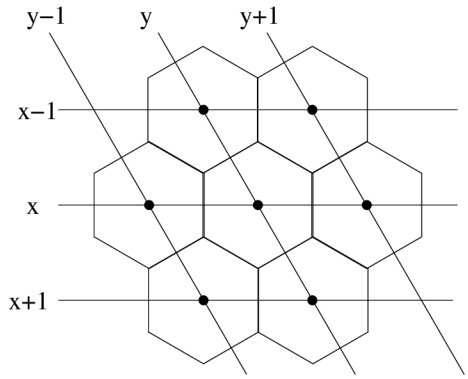
Samotná plocha s buňkami může být reprezentována několika možnostmi. Nejjednodušším typem je uložení buněk do klasické dvourozměrné matice. Tento systém poskytuje jednoduché a rychlé vyhledání buňky podle jejich souřadnic x a y, které definují pozici buňky v ploše. Existují i jiné formy implementace plochy jakými jsou například hashovací pole nebo kvadrantový strom a jiné. Tyto metody jsou vhodné pro plochy, kde jsou buď řídce rozmístěné aktivní buňky (ty které mění svůj stav), nebo kde jsou velké oblasti buněk, které mají stejnou hodnotu. Pro tyto případy mají dané implementace menší paměťové nároky, které jsou ale vykoupeny vyšší časovou náročností na vyhledání hledané buňky. V modelech fyzikálních jevů nenacházejí uplatnění, jelikož zde jsou na ploše aktivní všechny nebo alespoň většinová část buněk a výpočet následujících stavů se snažíme co nejvíce urychlit.

Pracujeme-li s hexagonálními buňkami, můžeme je rovněž uložit do dvourozměrného pole. Princip jejich indexace je naznačen na obrázku 4.1. Jedná se o to, že na buňky je rozmístěna dvousměrná mřížka s vzájemným úhlem 120°. Po transformaci této mřížky na pravoúhlou získáme klasické zobrazení 2D matice.

V aplikaci tedy bude užita matice `plocha [][]`, která bude typu potřebného pro daný model. Její inicializace proběhne po spuštění simulace načtením hodnot ze souboru.

4.2 Výpočetní jádro simulátoru

Každý fyzikální model má odlišný druh a způsob aplikace přechodových pravidel. Rovněž většina modelů zachází jinak s plochou, kde jsou uloženy buňky. Tyto odlišnosti tedy



Obrázek 4.1: Převod souřadnic hexagonálních buněk

jsou v každém modelu řešeny individuálně ve formě vlastních metod, kterými se počítají přechody mezi buňkami.

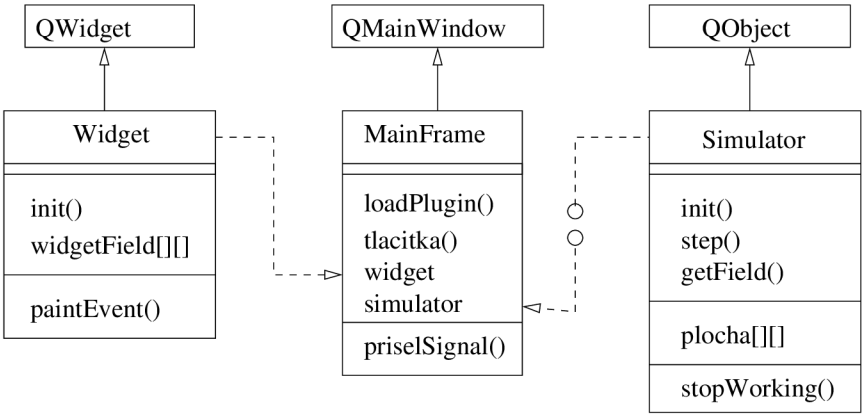
Během výpočtu nových stavů buněk musíme zajistit časoprostorovou jednotu plochy a výsledky je tedy nutné odkládat na pomocné místo. Pro tento případ jsou vytvořeny dvě stejně velké plochy buněk pro časy t a $t+\tau$. První plocha pro čas t se při inicializaci naplní hodnotami buněk ze souboru. Během zpracování buněk se vstupní hodnoty berou z této plochy a výsledky se ukládají na plochu určenou pro $t+\tau$. Po ukončení přechodu všech buněk a posunu času se označení těchto ploch prohodí.

Základním kamenem simulátoru jsou metody `init()` a `steps()`. Metoda `init()` zajistí počáteční inicializace, jakými jsou například načtení hodnot buněk ze souboru a nastavení důležitých proměnných pro simulaci. Metodou `steps()` se spustí simulace na požadovaný počet kroků, který je předán metodě jako parametr. Bez parametru provede metoda standardně pouze jeden krok. Další důležitou metodou je `getPole()`. Tato metoda zajišťuje převedení plochy buněk do podoby, kterou je posléze možno zobrazit. Některé modely totiž umožňují vnitřní reprezentaci stavu ve formě, která není jednoduše zobrazitelná na grafickém výstupu. Takové případy je třeba převést na odpovídající hodnotu. Všechny tyto metody jsou veřejnými a tvoří rozhraní, kterým bude komunikovat aplikace s přídavným modulem, čili s fyzikálním modelem.

4.3 Grafické prostředí

Grafické prostředí aplikace by mělo umožňovat uživateli základní ovládání simulace modelů. Na začátku si uživatel musí jednoduchým způsobem zvolit, jaký model bude chtít zkoumat. Podle výběru bude nahrán příslušný zásuvný modul. Následně by měl mít uživatel možnost změnit hodnoty některých konstant, které ovlivňují model (velikost časového kroku τ, ...). Chceme-li pozorovat výsledek našich pokusů, musí být součástí aplikace grafická plocha, na kterou se výsledky promítnou. Pokud je plocha příliš velká, půjde plochou rolovat. Samozřejmostí jsou ovládací prvky pro spuštění a zastavení simulace a hodnota počtu kroků, po kolika se má obnovit výsledek na grafické ploše.

Pro grafické prostředí aplikace jsem se rozhodl využít knihovny Qt 4.5.3. Jedná se o grafickou knihovnu jazyka C++, která je přenositelná mezi jednotlivými operačními systémy. Tato knihovna rovněž umožňuje vytváření a nahrávání zásuvných modulů, které komunikují s aplikací pomocí předem daného rozhraní.



Obrázek 4.2: Objektový návrh aplikace, zásuvný modul je znázorněn kroužkovým spojením

4.4 Objektový návrh

Na obrázku 4.2 je možné vidět objektový návrh aplikace. Ta se skládá ze tří tříd. Třída `MainFrame` je odvozena od třídy `QMainWindow` a představuje hlavní okno aplikace. Obsahuje metody, které obsluhují tlačítka, což je znázorněno metodou `tlacitka()`. Dále tato třída obsahuje metodu `loadPlugin()`, která slouží k nahrání dynamické knihovny k běžící aplikaci. Kromě všech tlačítek a combo boxů, které se objevují v okně aplikace a slouží k ovládání programu, je ještě součástí třídy objekt `widget`, který je objektem třídy `Widget`. Dále třída obsahuje objekt `simulator`, který představuje simulátor, který je dynamicky nahrán a který komunikuje přes rozhraní. V poslední řadě je třeba zmínit ještě slot `priselSignal()`, který je metodou reagující na příchozí signál o ukončení výpočtu.

Třída `Widget` je odvozena od třídy `QWidget`, která představuje plochu, na kterou se vykreslují hodnoty buněk. Tato třída obsahuje důležitou metodu `init()`, která slouží k základní inicializaci pole `widgetField` a různých vnitřních proměnných. Ono zmíněné pole `widgetField` představuje uložený obraz pole v simulátoru. Vzhledem k tomu, ze pole je veřejně dostupné mohou do něj být hodnoty nahrávány přímo v simulátoru. Pro samotné vykreslování na plochu je potom důležitá metoda `paintEvent()`, ve které jsou řešeny grafické rozdíly jednotlivých modelů. Tato metoda je volána automaticky při jakékoliv grafické události.

Třída `Simulator` představuje zásuvný modul, který je k aplikaci dynamicky připojen skrze předdefinované rozhraní. Jedná se o realizaci jistého modelu, proto každý model implementuje tuto třídu po svém. Vždy je však součástí třídy metoda `init()`, která provádí inicializaci modelu ze souboru. Během inicializace se načtou hodnoty do pole `plocha`. Metoda `steps()` pak představuje výpočetní jádro simulátoru, ve kterém se počítají nové hodnoty. Pomocí metody `getField()` pak můžou být hodnoty plochy zapsány do vykreslovacího pole v odpovídajícím formátu. Posledním důležitým prvkem pro komunikaci s hlavním oknem je signál `stopWorking()`, který oznamuje konec počítání. Tento signál je zachycen příslušným slotem.

4.5 Paralelní řešení

Vzhledem k náročnému a rozsáhlému výpočtu přechodových pravidel některých modelů je vhodné na výpočet aplikovat metody pro jeho zrychlení. Dostupnými metodami jsou paralelní a distribuované výpočty.

Pro paralelní výpočty je nutné mít výpočetní stroj s několika procesory a sdílenou pamětí a mechanismus, který rozdělí jednotlivým procesorům práci. Tímto mechanismem je knihovna OpenMP jazyka C/C++. OpenMP pracuje s vlákny, která vzájemně sdílejí paměť. Součástí OpenMP jsou direktivy překladači, který sestaví vlákna, určí jejich práci a vzájemně je sesynchronizuje. Další součástí jsou specializované funkce, které jednotlivá vlákna mohou využívat. Poslední částí jsou systémové proměnné, které slouží pro kontrolu programu za běhu.

V našem případě představuje onu sdílenou paměť především plocha, na níž jsou umístěny jednotlivé buňky. Tuto plochu je třeba označit za sdílenou proměnnou. Pak tedy budou každému vláknu automaticky přiděleny určité buňky, jejichž výpočet bude mít dané vlákno na starosti. Dále je potřeba se zamyslet nad tím, které další proměnné, které se ve výpočtu využívají mohou být sdíleny, nebo musejí být označeny za privátní proměnné daného vlákna. Privátní proměnné mají tu vlastnost, že jejich instance je nahrána v každém vlákně a všechny změny jsou viditelné pouze v tomto vlákně. Proměnné, které takto musejí být označeny za privátní, jsou především iterační proměnné cyklů pro průchod buňkami, jelikož každé vlákno zpracovává svou část danou těmito proměnnými a nesmí jej měnit ostatním vláknům.

Pro distribuovaný výpočet opět potřebujeme výpočetní prostředky, tentokráte jako samostatné procesory s vlastní pamětí. Tyto procesory (resp. počítače) musí být vzájemně propojeny, aby si mohly navzájem posílat zprávy. Mechanismem, který zajistí přeposílání zpráv a vzájemnou interakci mezi procesory je knihovna MPI. Jedná se opět o knihovnu jazyka C. Pomocí této knihovny se sestaví skupina procesů, které spolu vzájemně mohou komunikovat pomocí speciálních zpráv. Pomocí zpráv se zajišťuje i vzájemná synchronizace.

Chceme-li zparalelizovat celulární automat pomocí distribuovaného výpočtu, musíme sami zajistit rozdělení plochy s buňkami na takový počet částí, kolik strojů chceme zapojit do výpočtu. Každý stroj pak bude podle své značky počítat předepsanou oblast. Jeden počítací stroj musí být určen jako hlavní, který zajistí distribuci dat a následný sběr dat. Pracující procesory však potřebují ke svému výpočtu buňky, které leží mimo jejich území. Tento problém je možné řešit dvěma způsoby. První z nich je ten, že rozdělení polí bude překrývající se, čili dva procesory budou počítat kus stejné oblasti a hlavní procesor poté buď zapíše hodnoty do paměti dvakrát, nebo vezme v úvahu hodnoty pouze od jednoho procesoru. Druhým principem je to, že procesory budou komunikovat pomocí zpráv s koordinačním procesorem o vydání potřebných dat. Kvůli časoprostorové synchronizaci můžeme provádět paralelně pouze jeden krok simulace jak při paralelním tak při distribuovaném výpočtu.

Vhodným cílem pro porovnání efektivnosti jednotlivých metod je vytvoření aplikace se třemi různými implementacemi – sekvenční, paralelní a distribuovanou pro jeden určitý model. Takto vytvořené implementace je vhodné vyzkoušet na několika architekturách s různými parametry a porovnat vzájemnou rychlost, zrychlení a využití procesorů.

Kapitola 5

Implementace

V následující části budou popsány principy, jakým byla implementována aplikace i jednotlivá výpočetní jádra pro jednotlivé modely.

5.1 Grafické uživatelské rozhraní aplikace

Jak již bylo zmíněno v návrhu, byla aplikace implementována s využitím grafické knihovny Qt 4.5.2 a za použití programu Qt Creator 1.2.1, což je vývojové prostředí umožňující poměrně snadnou obsluhu zdrojových souborů a snadné vytváření designu aplikace pomocí grafické plochy.

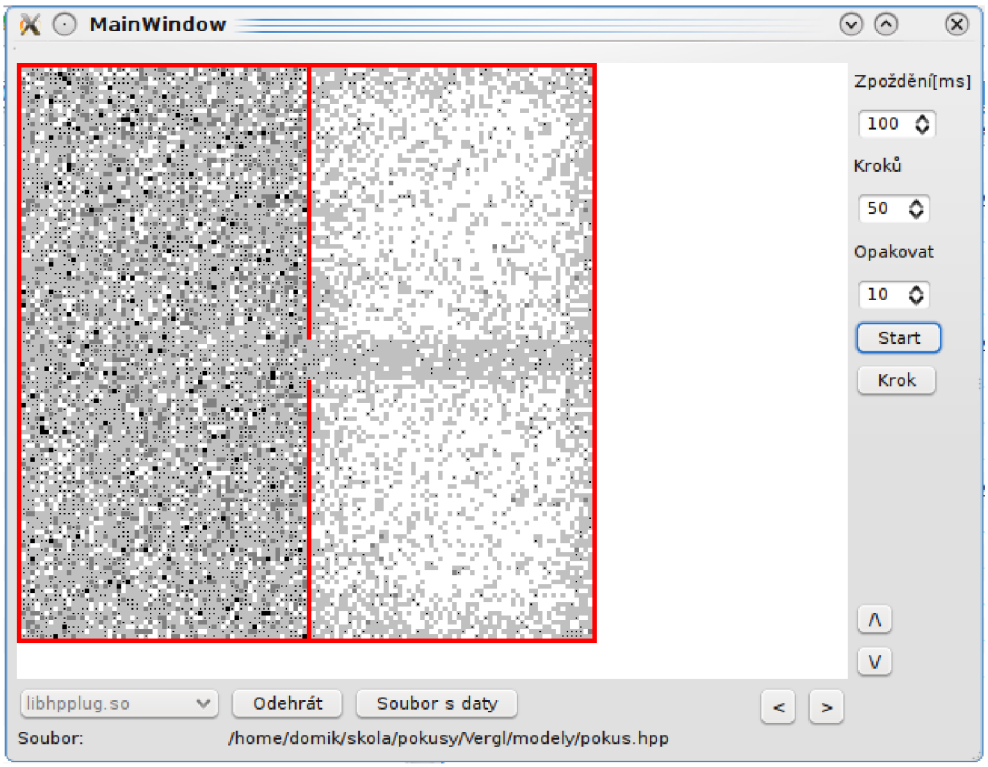
5.1.1 Hlavní okno

Celá aplikace, která je zobrazena na obrázku 5.1 se skládá ze dvou ze dvou hlavních tříd, které jsou do sebe vnořeny. Hlavní třídou, jejíž instance je vyvolána ihned po spuštění je třída `MainWindow`, která dědí vlastnosti třídy `QMainWindow`. Tato třída zajišťuje vykreslení a správu hlavního okna, které uživatel vidí. Součástí této třídy je instance třídy, která je generována automaticky pomocí vývojového prostředí a uživatel do ní nemá právo přímo zasahovat. V této třídě jsou definovány a inicializovány všechny prvky, které se v hlavním okně vyskytují jako tlačítka, combo boxy, popisky aj. K těmto prvkům se posléze přistupuje přes reference.

Umístění těchto prvků v okně lze řešit automaticky pomocí vývojového prostředí, avšak toto ne vždy fungovalo spolehlivě, proto jsem se rozhodl prvky zarovnat ručně pomocí objektu třídy `QGridLayout`, který zajistí zarovnání prvků do mřížky. S oknem je poté možno manipulovat a prvky zůstávají ukotveny v okně a podle potřeby mění svou velikost. Toto zarovnání spolu s nastavením důležitých proměnných se děje v konstruktoru.

Zřejmě nejdůležitější částí je komunikace s uživatelem. Qt využívá systém signálů a slotů. Stane-li se nějaká událost je vyslán signál, který je zachycen příslušným slotem, se kterým je provázán. Slot pak představuje metodu, která se vykoná při zachycení signálu. Pro komunikaci mezi uživatelem a aplikací slouží převážně tlačítka, která emitují signál `released()`. Tyto signály přijímají příslušné sloty, které vytvořeny samostatně pro každé tlačítko.

Tlačítka, která slouží pro posunování ploch ve svých slotech inkrementují nebo dekrementují hodnoty offsetů, které se využijí při následném vykreslování. Tlačítko, které slouží pro vybrání patřičného modelu ve svém slotu vyvolá souborový dialog `QFileDialog`. Výsledné jméno vybraného souboru je poté uloženo do pomocné proměnné a toto jméno je



Obrázek 5.1: Hlavní okno aplikace

vypsáno na místo v hlavním okně. Tlačítko pro nahrávání zásuvných modulů má na starosti ohlídání nahrání a posléze i odehrání modulů, které však bude popsáno dále. Pokud se operace podaří, je nastavena viditelnost tlačítek pro spouštění a krokování simulace. Tato tlačítka mají takřka stejnou funkci a liší se jen s jakým parametrem simulaci pouštějí. V prvé řadě je třeba zkontrolovat, zda již byly inicializovány všechny potřebné proměnné. Pokud ne, je tomu tak učiněno a je nastaven příznak inicializace. Pokud během inicializace vzniknou nějaké komplikace, je vyvoláno výstražné okno, které uživatele o komplikacích informuje. Toto okno je typu `QMessageBox`. Poté je spuštěna simulace. Tlačítko pro start k tomu využije údaje o počtu kroků a počtu opakování, která může uživatel zadat do pole typu `QSpinBox`. Tím se liší od tlačítka pro jeden krok simulace, které spouští pouze jedno opakování se zadaným počtem kroků.

5.1.2 Zásuvné moduly

Knihovna Qt 4.5 poskytuje poměrně elegantní systém, jakým jdou do běžící aplikace zasunout moduly (velmi často označované anglickým slovem plugin), které rozšiřují funkčnost aplikace. Stačí nám k tomu vhodně zvolené rozhraní a umístění několika předdefinovaných maker.

Rozhraní je ve své podstatě třída, která obsahuje virtuální metody. Tyto metody jsou posléze implementovány pomocí zásuvného modulu, který se k aplikaci přihraje. Následující zdrojový kód představuje definici rozhraní, které je použito ve vytvořené aplikaci.

```
#include <QString>

class CellInterface
{
  public:
    virtual ~CellInterface(){}
    virtual int init(QString filename)=0;
    virtual int step(int steps)=0;
    virtual int getType()=0;
    virtual int getSize(int * xradky, int * xsloupce)=0;
    virtual int getPole(int **xpole)=0;
    virtual int uklid()=0;
  signals:
    virtual void stopWorking()=0;
};

Q_DECLARE_INTERFACE(CellInterface,
                    ''cellsim.CellInterface.verze1.1'');
```

Z kódu je patrné, že se jedná o standardní definici třídy, která obsahuje virtuální metody. Klíčovým místem je užití makra `Q_DECLARE_INTERFACE`. Tímto makrem se definuje, že tato třída bude využívána jako interface pro komunikaci se zásuvnými moduly. Tato informace je posléze zanesena v meta-object souboru, který je automaticky vygenerován překladačem při překladu. V makru je uvedeno jméno třídy, která rozhraní tvoří a její unikátní textový popis.

Z definice je vidět, že jsou definovány základní metody, kterých je potřeba pro komunikaci s modulem obsahujícím výpočetní jádro. Metoda `init` zajistí nahrání modelu ze souboru. To znamená alokaci všech potřebných polí podle zjištěných rozměrů a nastavení všech důležitých konstant a polí podle zjištěných údajů. V případě neúspěchu nějaké operace je vrácen chybový kód, podle něhož lze chybu identifikovat.

Metoda `step()` pak představuje ono výpočetní jádro simulátoru, ve kterém se provádí průchody buňkami a jejich výpočet. Parametr steps určuje, kolikrát se má jeden krok provést. Metody `getType()`, `getSize()` a `getPole()` představují metody, kterými si hlavní aplikace zjišťuje parametry modelu, které jsou nutné například pro zobrazení výsledku na plochu. Metoda `uklid()` nahrazuje práci destruktoru, jelikož součástí této metody je dealokace paměťí, které byly zásuvnému modulu přiděleny.

Posledním důležitým prvkem komunikačního rozhraní je signál `stopWorking()`.Tento signál je emitován metodou `step()` po provedení všech požadovaných kroků. Tento signál je zachycen patřičným slotem v hlavním okně. Tento slot pak zajistí případné opakování provedení sekvence kroků. Hlavním úkolem tohoto slotu je však zadání požadavku na překreslení obrazovky.

Doposud byla popisována de facto přípravná část na straně aplikace, která z velké části vyplývá z návrhu systému. Strana samotného zásuvného modulu je mírně obtížnější, protože modul musí mít jistou předepsanou strukturu a zákonitosti, jinak jej není možné do aplikace nahrát. Následující ukázka hlavičkového souboru představuje tyto základy.

```
#include <QObject>
#include <QString>
#include ''cellinterface.h''

class FhpPlugin : public QObject, CellInterface
{
    Q_OBJECT
    Q_INTERFACES(CellInterface)
  public:
    int init(QString filename);
    ....
  signals:
    void stopWorking();
  private:
    char *** pole;
    ....
};
```

První podmínkou je nahrání hlavičkového souboru s rozhraním. Druhou podmínkou je, že třída tvořící modul musí rozšiřovat třídu `QObject` a třídu tvořící rozhraní. Ve třídě `QObject` jsou definována makra, která se pro vytvoření zásuvného modulu používají. Tato makra se poté zapisují hned na začátek definice třídy. Nejdříve je nutné zapsat makro `Q_OBJECT` následované makrem `Q_INTERFACES`, které obsahuje název rozhraní, které budeme v zásuvném modulu implementovat.

Tato makra jsou následována definicí metod tvořící rozhraní. Veřejnými metodami nesmějí být žádné další metody. Poté je definován signál pro komunikaci. Až po něm mohou být definovány privátní metody a proměnné. Experimentálně bylo zjištěno, že závisí na tomto pořadí definic, jinak hrozí, že modul nebude přijat jako platný.

Ve zdrojovém souboru daného hlavičkového souboru jsou pak standardní implementace jednotlivých metod. Na konci souboru však musí být definováno makro `Q_EXPORT_PLUGIN2`, které má dva parametry. Prvním je označení, kterým chceme modul označovat a druhým parametrem je jméno třídy tvořící zásuvný modul.

Všechny tyto kroky však ještě k vytvoření zásuvného modulu nestačí. Je třeba ještě překladači oznámit, že budeme takovýto soubor vytvářet. Pro vytváření `makefile` souborů se v Qt používá program `qmake`. Ten vytváří překládací soubory na základě projektového souboru `*.pro`. V tomto souboru jsou uvedeny všechny soubory, které daný projekt tvoří, dále informace o jménech výstupních souborů a rovněž informace o funkci přeloženého souboru. Pro dynamickou knihovnu je třeba uvést následující parametry, které toto stanoví.

```
TEMPLATE = lib
CONFIG += plugin
 ....
```

Je-li vytvořen soubor představující dynamickou knihovnu, je jej ještě nutno v aplikaci připojit potažmo odpojit. K připojení zásuvného modulu slouží objekt třídy `QPluginLoader`. Celý mechanismus nahrání modulu spočívá ve využití metod této třídy (`load()`, `instance()`, ...). Plugin je pak nahrán jako objekt třídy `QObject` a je jej nutno přetypovat na správnou třídu. K tomu slouží metoda `qobject_cast`.

```
cellinterface=qobject_cast<CellInterface *>(plugin);
```

Takto spojíme metody našeho rozhraní s metodami implementovanými v zásuvném modulu.

Aby se však moduly načetly musí být splněn takzvaný překládací klíč. Ten říká, že jsou načteny pouze ty moduly, které jsou přeloženy na stejné architektuře, stejném operačním systému, za použití stejného překladače a se shodným nastavením Qt knihovny. Rovněž musí být splněny již výše zmíněné strukturální záležitosti.

5.1.3 Grafická plocha

Jedním z nejdůležitějších prostředků pro komunikaci s uživatelem je grafická plocha, na které se zobrazují výsledky simulací. Model by mělo tedy jít zobrazit v rozumném tvaru, aby mu uživatel byl schopen porozumět. V mé aplikaci byl kladen důraz na jednoduchost zobrazení. Qt sice dovoluje využití specializovaných grafických knihoven jako OpenGL, avšak ty jsou pro zvolené modely poněkud předimenzované a byly vhodné zejména pro 3D modely, kde na kvalitě zobrazení závisí srozumitelnost výsledků.

Prvkem, který může fungovat v aplikaci jako kreslící plocha je objekt třídy `QWidget`. Tento objekt se ve vývojovém prostředí vytváří automaticky po vložení widgetu. Pro osobitou práci s plochou je tato třída však nepostačující, protože pro vykreslení modelů potřebuje vlastní metody, jak plochu vykreslit. Jedinou možností je vytvořit novou třídu, která dědí metody třídy `QWidget` a v níž si přidáme rozšiřující metody. Objekt takto vytvořené třídu je ale nutné začlenit do naší aplikace. To způsobuje drobnou komplikaci, protože hlavičkový soubor s definicemi prvků na ploše je automaticky generovaný a nelze jej přepsat ručně. Je nutné objekt třídy `QWidget` pomocí menu povýšit na objekt námi vytvořené třídy. Tím se automaticky generovaný soubor přepíše.

Nyní přistupme k vlastnostem nově vytvořené třídy. Jak již bylo řečeno musí se jednat o potomka třídy `QWidget`, aby byla schopna grafických akcí. Dále je potřeba několik veřejných metod a proměnných, pomocí nichž se bude grafická plocha nastavovat tak, aby odpovídala zobrazovanému modelu. Asi nejdůležitější metodou je metoda `nastavPlochu()`, která přijímá jako parametry rozměry modelu. V této metodě se pak dynamicky alokuje dvourozměrná matice `widgetPole`, které obsahuje hodnoty, které jsou posléze interpretovány na obrazovku. Toto pole je rovněž veřejné, čili do něj může zásuvný modul pomocí své metody zapisovat hodnoty. Dále je součástí třídy metoda, která zjistí vykreslovací typ modelu a proměnná, do níž se zadává informace o tom, je-li nahrán příslušný zásuvný modul. Tato proměnná zajišťuje, aby se program nepracoval s neinicializovanými hodnotami.

O samotné vykreslování se stará privátní metoda `paintEvent()`, která je volána automaticky při vyvolání grafické události. Tu je možné vyvolat například metodou `repaint()`. Samotná metoda vyhodnotí podle typu modelu, která konkrétní metoda pro vykreslení bude vyvolána. Každý model má svou vlastní metodu, která vykresluje jeho buňky.

Ač jsou tyto metody odlišné, jejich princip je shodný. V prvním kroku se zjistí velikost vykreslovací plochy a vypočítají se počty buněk, které se na tuto plochu vejdou v obou rozměrech. Poté probíhá průchod buněk, které se mají vykreslit. Počáteční buňky jsou dány offsety, které jsou taktéž veřejnými proměnnými a lze je nastavovat z hlavní aplikace. Konce jsou dány koncem pole nebo počtem buněk, které se na plochu vejdou. V tomto průchodu je nahlíženo do pole `widgetPole` na odpovídající pozice. Podle hodnoty je vybrána odpovídající barva pera a štětce. Poté je do proměnné typu `QPainter` zakreslen na odpovídající pozici obdélník dané velikosti. V případě modelu se sypáním písku jsou vykreslovány kružnice. Jednotlivé modely se liší tím, jakou barvou reagují na různé hodnoty a velikostí vykreslovaných buněk.

5.2 Navržené modely

V následující části jsou popsány modely, které byly zvoleny, navrženy a implementovány jako ukázkové příklady s ohledem na jejich možné využití ve výuce a při případném výzkumu. Tyto kategorie modelů byly implementovány jako zásuvné moduly, které jsou dynamicky nahrávány aplikací.

5.2.1 Model pohybu částic po čtvercové mřížce - HPP pravidla

Tento model představuje elementární možnost, jak lze popsat chování, ke kterému dochází v plynu. Jedná se o model využívající HPP pravidla, která jsou popsána v části 3.1.1. Model využívá několik abstrakcí. První z nich je ta, že všechny částice mají většinu vlastností jako hmotnost, velikost, energii a rychlost shodnou. Liší se pouze směry, kterými se pohybují. Směr pohybu částice a její pozice na ploše jsou tedy jediné atributy, které nás reálně zajímají.

Pro popis takové plochy se jeví asi jako nejlepší využití dynamicky alokované dvourozměrné matice typu `char **`, kde každý prvek představuje kolizní místo, kde se částice potkávají. Každá buňka nám poskytuje 8 bitů. Spodní 4 bity jsou použity pro definování částice v kolizním bodě. Vyskytuje-li se jedničkový bit na určité pozici, znamená to, že ze směru daného pozicí bitu přichází v časovém okamžiku částice.

Zbývající 4 bity mohou být využity jako další kolizní bod, čímž bychom snížili paměťový nárok tohoto modelu. Během indexace buňky by se však musel index neustále přepočítávat a bity by se musely stále maskovat a bitově posouvat. Tyto operace by výrazně zpomalovaly výpočet, proto jsou tyto bity využity k jinému účelu. V této části je přenášena informace o tom, o jaký typ buňky se jedná. Chceme-li rozlišit buňky kvůli některým pravidlům, je k tomuto vhodné právě toto místo.

V mém případě jsou na ploše pouze dva typy buněk. Buňky představující volný prostor a buňky představující pevnou překážku jakou je zeď. Pro buňky představující zdi platí pravidlo, že narazí-li do této buňky částice, je vrácena v opačném směru. Pro volný prostor se částice šíří dále prostředím.

Pro tento model je typické dvoufázové řešení přechodu buňky do nového stavu. V prvním kroku jsou vyřešeny kolize částic a teprve potom se částice přesune na svou novou polohu. Obě fáze jsou součástí jednoho časového kroku. Během průchodu všemi buňkami na ploše se tedy provádí následující postup. Nejdříve se zkontroluje, zda buňka neleží na okraji plochy, protože by nám mohl nastat problém s indexací buňky mimo hranice pole. Poté se ověří, zda obsahuje buňka pouze dvě částice, které do sebe narážejí v horizontálním či vertikálním směru. V tomto případě se do pomocné proměnné představující buňku po kolizi uloží obraz po kolizi (definováno na obrázku 3.1). Pokud buňka obsahuje jiný počet částic, vymaskuje se jejich obraz a pomocí bitového součtu se přesunou v pomocné proměnné na polohu, která následuje jejich směr (např. pokud do buňky vlétá částice zleva, bude vylétat zprava). Pokud se jedná označena jako speciální, je fáze kolize přeskočena. Tímto jednoduchým efektem zajistíme, že částice, které přijdou do této buňky jsou odraženy ve stejném směru, odkud přišly.

V pomocné proměnné nyní je obraz částic po kolizi a je třeba částice posunout na jejich odpovídající pozice v prostoru. Zde je však problém s časovou sounáležitostí současné a budoucí plochy. Plocha představující aktuální obraz nemůže obsahovat hodnoty, které byly vypočteny jako budoucnost. Je tedy třeba mít dvě plochy a existují dva způsoby jak toho lze dosáhnout. Je možné přidat další časovou dimenzi do naší plochy, nebo mít dvě samostatné

plochy a po učinění jednoho kroku prohodit jejich ukazatele do paměti. S ohledem na jednoduchost tohoto modelu jsem volil první variantu, čili na začátku je vytvořena plocha typu `char ***`, která by se dala přepsat na statickou `char plocha [2][radku][sloupcu]`, kde číslo 2 právě představuje časovou dimenzi – tedy dvě plochy, se kterými se pracuje, plochu současných buněk a plochu kam se odkládají výsledky. Po jednom kroku jsou pouze prohozeny hodnoty indexů na tyto plochy.

Nyní tedy lze částice z pomocné proměnné přesunout na jejich odpovídající hodnoty v následující ploše. Toho se dosahuje opětovným průchodem spodních bitů proměnné. Je-li zjištěna přítomnost částice je pomocí bitového součtu přemístěna na odpovídající buňku podle jejího směru. Zjistí-li se, že buňka byla definována jako speciální (zeď apod.), předá se tato informace do následujícího kroku. Tato buňka zůstává na své pozici. Po umístění všech částic se aktuální buňka vynuluje, aby vytvořila místo pro budoucí částice.

5.2.2 Model pohybu částic po hexagonální mřížce - FHP pravidla

Předchozí model vykazoval velkou odlišnost od reality díky čtvercovému rozdělení prostoru. Model založený na FHP pravidlech však dělí prostor na plochu šestiúhelníku, čímž se blíží více realitě, protože buňka hraničí přímým dotekem s více buňkami a má se všemi sousedy stejnou vzdálenost. Ostatní abstrakce jako stejná rychlost a energie všech částic jsou zachovány.

Buňka na ploše opět představuje kolizní místo, kde se částice střetávají. Díky abstrakci lze částici popsat jako bit, jehož pozice představuje směr letu částice. Může-li nám do kolizního místa vstoupit maximálně šest částic, je tedy vhodné využít opět osmibitový typ `char`. V tom případě ještě zbývají každé buňce dva bity na uchování speciálních informací. V mém modelu se jedná o informace o typu buňky, čili představuje-li volný prostor nebo pevnou překážku na cestě.

Chceme-li vytvořit v paměti plochu tvořenou šestiúhelníkovými buňkami, musíme počítat s komplikacemi při indexaci těchto buněk. Na obrázku 4.1 je naznačen způsob jak se lze vypořádat s indexací hexagonálního prostoru jeho transformací na pravoúhlou mřížku. V praxi je tuto transformaci využít již při alokování pole. Máme-li dánu hexagonální mřížku, kterou je pokryta obdélníková plocha, musíme počítat s tím, že po transformaci nám nebude první sloupec pokrývat všechny první buňky, ale pouze buňky na posledním a předposledním řádku (záleží na sudosti počtu řádků). Celkově tedy naroste počet sloupců transponovaného pole oproti šestiúhelníkovému o hodnotu rovnající se polovině počtu řádků. Zápis statické alokace lze zapsat jako `char pole[2][radku][sloupcu+radku/2]`. Transpozicí hexagonálního pole vznikne kosodélník, který je vepsán do obdélníkového pole. Vzniknou nám tím ale dvě trojúhelníkové oblasti volné paměti, které jsou obětovány jako daň za rychlejší a poměrně jednoduchou indexaci.

Nyní chceme-li indexovat jistou buňku v původní hexagonální ploše, musíme si napřed vypočítat offset jejího řádku v transponované ploše. Tento vztah byl empiricky odvozen jako

$$offset\,(radek) = \frac{pocet\_radku}{2} - \frac{radek}{2}$$

Délka řádku je potom dána počtem sloupců netransponované plochy. Přičteme-li tuto hodnotu k offsetu, získáme tím index posledního sloupce, který ještě popisuje buňku netransponované plochy.

Výhoda tohoto dělení plochy je rovněž v dobré indexaci okolních buněk ve všech směrech. Tabulka 5.1 představuje hodnoty základních offsetů pro jednotlivé směrové vektory. Vyná-

sobíme-li tyto vektory konstantou λ určující délku posunu získáme tím hodnoty, které nás při přičtení k současné pozici odkážou na danou buňku v daném směru.

$$\begin{array}{rcrcl} smer = & [& \Delta radek, & \Delta sloupec &] \\ \vec{v}_1 = & [& 0, & +1 &] \\ \vec{v}_2 = & [& +1, & 0 &] \\ \vec{v}_3 = & [& +1, & -1 &] \\ \vec{v}_4 = & [& 0, & -1 &] \\ \vec{v}_5 = & [& -1, & 0 &] \\ \vec{v}_6 = & [& -1, & +1 &] \end{array} \tag{5.1}$$

Jak již bylo popsáno v kapitole 3.1.2 používá se pro evoluci buněk rovněž dvoufázové pravidlo, čili nejdříve se vyřeší kolize v buňce a poté se částice přemístí na odpovídající polohu. U tohoto modelu se řeší speciálně přímé kolize dvou a tří částic podle vzorce 3.7 s využitím pravděpodobnosti q. Pro snadnější a přehlednější výpočet jsou vytvořena tři šestiprvková pole n_i,D_i a T_i. Do pole n_i je na začátku nahrán obraz částic v kolizním bodě. Následně jsou vypočítány kolizní indexy D_i a T_i podle vzorců 3.3 a 3.5. Zde je třeba brát ohled na to, že indexy nesmí překročit hranici pole, což se řeší zavedením operace modulo. Protější částice v kolizním bodě lze tedy vyjádřit jako $n_{(i+3)mod6}$.

S vypočítanými indexy lze postoupit do posunovací fáze. Ta je řešena pomocí vzorce 3.7 a s využitím náhodné proměnné $q \in 0, 1$. Podle vektorů posunu je vypočtena nová poloha buňky, kam se částice přesune. Je-li funkcí pro výpočet polohy vrácena stejná poloha buňky, ze které se chce částice posunout, znamená to, že částice se odrazila od pevné překážky a vrátila se zpět. Částice se poté přesune v pomocné proměnné na svou patřičnou polohu označující její směr. Tato pomocná proměnná je poté pomocí bitového součtu přesunuta na svou novou polohu na ploše v následující časové dimenzi. V té současné se po přesunu všech částic buňka vynuluje, aby uvolnila místo pro následující hodnoty.

5.2.3 Model sypání písku

Chceme-li pochopit alespoň základní vlastnosti, které se vyskytují v sypkých materiálech, je velmi vhodné použití modelu, na kterém můžeme sledovat jednotlivé časové kroky. Dále můžeme odhadnout, jak se může materiál zachovat v reálné situaci a tím pádem se lépe připravit. Jako sypký materiál bývá nejčastěji považován písek také díky tomu, že má nejvíce odhadnutelné vlastnosti.

V kapitole 3.2 je popsán vznik a vlastnosti pravidel, kterých se využívá při sypání písku. Chceme-li popsat plochu představující náš model, musíme si nejdříve uvědomit jaké typy buněk se nám mohou na ploše objevit. V mém případě se jedná o tři typy. První z nich je buňka představující volný vzduch. Druhým typem je pískové zrnko, od kterého se vyžaduje pohyb po ploše a v poslední řadě buňka představující pevnou překážku. Od této buňky se v modelu předpokládá a vyžaduje, že zůstane po celou dobu simulace na své pozici.

Tyto hodnoty tedy jdou zapsat na dva bity. Pokud bychom tedy chtěli maximálně šetřit místem, mohli bychom do osmi bitového typu `char` zapsat vedle sebe čtyři buňky. Tento přístup sice maximálně využije nabízené místo, ale potýká se s problémem složité indexace jednotlivých buněk. Každou buňku bychom museli složitě hledat a maskovat, abychom získali požadovanou hodnotu. Takový přístup jednak zabírá čas simulace, navíc reálně hrozí programátorská chyba. Já proto volil způsob, kdy je každá buňka zapsána do paměťového místa typu `char`. Celé pole je tedy tvořeno jako dynamicky alokovaná matice

`char **`. Tento přístup sice zabírá čtyřikrát více prostoru, avšak při rozměrech polí, která simulujeme, jsme toto schopni obětovat.

Jednoduchá indexace se hodí zejména u Margolusova okolí, které se pro tento druh modelu využívá. Margolusovo okolí totiž bere čtveřici buněk jako jeden prvek, který je třeba změnit. Při průchodu polem pomocí dvou do sebe vnořených cyklů `for` se můžeme pohybovat se skokem o dvě buňky jak na řádku tak na sloupci. Tímto přístupem naindexujeme první buňku dané čtveřice. Další buňky čtveřice tvoří buňka po pravé straně ,buňka na spodní straně a buňka na spodní straně vpravo. Chceme-li pojmout tuto čtveřici, můžeme tak učinit tím, že všechny tyto buňky sloučíme do jedné proměnné typu `char`.

Tuto hodnotu poté můžeme transformovat pomocí pravidel uvedených na obrázku 3.6. Musíme však ještě přidat pravidla, která postihují i výskyt zdí. Transformaci je potom vhodné provádět pomocí konstrukce `switch`, která na danou hodnotu odpoví její transformovanou. Pakliže se vyskytne hodnota, která není v tabulce pravidel, je vrácena původní hodnota a ke změně nedochází. Reálný zápis transformace je popsán následovně:

```
switch(hodnota_bunek)
{
    ...
    case 0x50: return 0x05;  //01010000b -> 00000101b
    default: return hodnota_bunek;
}
```

Tento příklad popisuje pád dvou pískových zrnek, která jsou jsou v horní řadě a spadnou do spodní řady.

Margolusovo okolí je rovněž specifické tím, že rozdělení buněk do čtveřic určuje podle toho, jestli se jedná o sudý nebo lichý krok simulace. Pro tento případ je zavedena pomocná proměnná `krok`, která nabývá pouze dvou hodnot, čímž odlišujeme sudé a liché kroky. V případě sudého kroku je rozdělení na čtveřice zaveden offset -1 pro řádky i sloupce. V reálu to znamená, že pole začíná o jeden sloupec a řádek dříve. S tímto je třeba počítat v základní alokaci paměti pro modelovanou plochu. Skutečná plocha je ohraničena ze všech stran lemem prázdných buněk, které slouží k tomu, abychom se nedostali mimo hranice paměti.

V modelu si můžeme rovněž umístit takzvané sypače. Jedná se o speciální buňky, na jejichž pozicích se pak náhodně generují nová zrnka písku. Pravděpodobnost výskytu nových zrnek je jedním z parametrů modelu.

Tím, že tento model pracuje se systémem, kde pro změnu hodnot čtveřice buněk stačí pouze hodnoty oné čtveřice, můžeme vypustit časovou dimenzi plochy. Buňky, které již byly změněny mohou být zapsány na stejnou plochu, kde jsou ještě nezměněné buňky.

5.2.4 Model fyzikálních vlastností vlnění

Chceme-li modelovat zákonitosti dějící se v prostoru během elektromagnetického vlnění (zejména světla), můžeme k tomu využít principu Botlzmanovy mřížky, který byl popsán v části 3.3. Hlavní ideou je to, že každá částice představuje energetický potenciál, jehož složky se v určitých směrech mohou lišit. Tyto potenciály se posléze podle určitých pravidel přesouvají do daných směrů.

Pravidla pro přesun energie jsou popsány v maticích 3.25, 3.28 a 3.27. Tyto matice byly odvozeny podle parciálních diferenciálních rovnic v [7] s využitím relaxačních koeficientů.

Tyto rovnice počítají často i s energií f_0, čili se zbytkovou energií. Tato energie se však velmi často z modelů vypouští kvůli zjednodušení. Tento postup jsem následoval.

Po odstranění zbytkové energie nám zůstávají pro každou buňku čtyři energie, které směřují do světových stran. Tyto energie nenabývají pouze binárních hodnot, jak tomu bylo v předešlých případech, nýbrž dosahují reálných hodnot. Tyto hodnoty můžeme popsat datovým typem `double`. Variant, jak uchovat energetické hodnoty pro každou buňku je opět několik. Máme možnost vytvořit čtyř prvkovou strukturu, která by obsahovala hodnoty f_1, f_2, f_3 a f_4. Dále existuje možnost využít některý z kontejnerů STL, jako například `vector`. Obě tyto možnosti však v sobě skrývají komplikace s indexováním, který by se obtížně prováděl v cyklu. Rozhodl jsem se pro variantu, kde směr představuje další dimenzi plochy buněk. Vezmeme-li v potaz ještě čas, dostáváme se na hodnotu čtyř dimenzí pole. Staticky by to bylo možné zapsat jako `double plocha[2][4][radku][sloupcu]`, kde číslo 2 představuje časovou dimenzi a číslo 4 dimenzi směrovou.

Jak již bylo zmíněno v kapitole 3.3, pracuje tento model s neuniformním rozdělením plochy, kdy je pro každou buňku třeba rozlišovat, o jaký typ buňky se v prostoru jedná. Toto rozdělení buněk probíhá při základní inicializaci modelu. Typy existují pouze tři – buňky představující prostor, kde se energie volně pohybují prostorem, buňky na kterých probíhá lom a buňky, od kterých se vlnění odráží. Tyto údaje jsou nahrány do dynamicky alokované dvourozměrné matice pokrývající všechny řádky i sloupce a které je typu `char **`.

V samotném simulačním jádře probíhá opět průchod přes všechny buňky na ploše. Pokud má buňka alespoň v jednom směru nenulovou energii je tato buňka brána v potaz. Ty nulové je možno přeskočit. U těch aktivních se nejdříve pro každý směr zjistí buňka kam se bude energie přesouvat. Pokud leží tato buňka mimo hranice pole, nikam se nezapisuje a energie zmizí pryč. V případě, že se energie bude přesouvat do pole, je provedena kontrola typu zdrojové buňky. Podle zjištěného typu se pak vybere, podle které transformační matice se bude nová energie počítat. Nová hodnota energie je pak vyjádřena jako skalární součin určitého řádku matice, který je dán aktuálním směrem a vektorem energií původní buňky. Vypočtená hodnota je nahrána na svou předem vypočítanou pozici. Po průchodu všemi směry jsou hodnoty v původní buňce vynulovány, aby hodnoty neovlivňovaly následné výpočty.

Projdeme-li všechny buňky, které přenáší energii, je nutné ještě nastavit zdroje vlnění. Jejich souřadnice jsou při počáteční inicializaci nahrány do kontejneru typu `vector`. Pro tyto buňky je třeba do simulace zavést i časovou proměnnou. Na začátku se tato proměnná inicializuje na nulovou hodnotu a s každým krokem simulace je k ní přičten časový krok, který tvoří součást modelu, stejně jako amplituda a perioda kmitání.

$$f_i = A \ \sin(\frac{2\pi t}{T}) \tag{5.2}$$

Energie, která se přesune do všech směrů zdroje se vypočítá podle vzorce 5.2. Teprve poté je zvýšen modelovací čas a prohozeny časové dimenze.

Pokud chceme poslat hodnoty grafickému prostředí k vykreslení, musíme si uvědomit jednu komplikaci. GUI přijímá dvourozměrnou matici s prvky typu `int` a my máme každou buňku reprezentovanou čtyřmi hodnotami typu `double`. Udělat z těchto čtyř hodnot jednu se dá jednoduše tak, že vypočítáme aritmetický průměr. Tato hodnota je však pro zobrazení ještě zavádějící, protože jednou může být amplituda velmi malá, podruhé velká. Hodnotu je třeba tedy znormalizovat právě vůči amplitudě a tak, aby se vešla do rozsahu $\langle 0, 255 \rangle$,

kterým můžeme vyjádřit odpovídající odstín šedi. Vzorec pro normalizaci tedy vypadá:

$$normalizovana\_hodnota = \left(\frac{prumer}{amplituda} \cdot 128\right) + 127 \tag{5.3}$$

5.2.5 Model šíření rádiového vlnění

Model šíření rádiového vlnění má zřejmě největší význam pro praktické využití v reálném světě. Se stále rostoucím využívání mobilních telefonů nejen pro telefonáty, ale pro stále větší přenosy dat musí mobilní operátoři ve městech více přemýšlet nad rozmístěním vysílačů. Vysílače totiž musejí pokrývat co největší plochu co nejkvalitnějším signálem. V husté aglomeraci se však nacházejí přirozené překážky v podobě zástavby, které signál různě odrážejí a stíní. Vhodné umístění antény může ušetřit peníze díky nižšímu vysílacímu výkonu antény či díky lepšímu pokrytí města.

Tento model má s předchozím modelem jisté atributy stejné, ale v mnoha se liší. Shodný je v tom, že rovněž využívá pro popis jedné buňky čtyři energetické potenciály, které směřují do čtyř světových stran. Tyto potenciály jsou rovněž popsány datovým typem `double` a je tedy opět využit princip čtyřdimenzionální plochy.

Hlavním rozdílem je systém výpočtu nových energetických hodnot. Pro výpočet se již nepoužívají transformační matice, nýbrž jednoduché koeficienty. Základním vzorcem pro výpočet je rovnice

$$f_i(\vec{r} + \vec{v_i}, t + 1) = c_1(\vec{r})\left(c_2(\vec{r})\psi(\vec{r}, t) - f_{i+2}(\vec{r}, t)\right) \tag{5.4}$$

kde $\vec{r}$ představuje souřadnice zdrojové buňky, ze které se energie přesouvá, $\vec{v_i}$ představuje směr, kterým se energie šíří. Koeficienty c_1 a c_2 představují koeficienty, které budou dále vysvětleny a hodnotu ψ získáme sečtením všech energií v dané buňce, čili

$$\psi(\vec{r}, t) = f_1(\vec{r}, t) + f_2(\vec{r}, t) + f_3(\vec{r}, t) + f_4(\vec{r}, t)$$

Koeficienty c_1 a c_2 slouží k popisu prostředí, kterým se vlnění šíří. Nastavíme-li hodnoty na $c_1 = 1$ a $c_2 = 0,5$ získáme tím hodnoty, které odpovídají matici 3.25. Jedná se tedy o hodnoty, které popisují volný prostor, po kterém se energie pohybuje v zadaném směru.

Chceme-li popsat odraz vlnění od nějaké překážky, můžeme tak učinit tím, že nastavíme koeficient $c_2 = 0$ a koeficientem c_1 určujeme míru, kolik energie se odrazí zpět. Je-li $c_1 = 0$ znamená to, že zeď, od které se vlnění odráží pohlcuje veškeré vlnění. Reálné hodnoty se pohybují kolem hodnot $\langle 0,6; 0,7\rangle$. To znamená, že se vrátí 60% až 70% vlnění.

Pro reálné modely potřebujeme ještě jeden typ buněk. Jedná se o buňky, které absorbují vlnění. Tyto buňky zajišťují „odstínění“ od okolí. V praxi to znamená to, že celá plocha je ohraničená systémem buněk, které postupně absorbují vlnění a nevznikají nežádoucí odrazy v okrajových oblastech. Pro nejvnitřnější ohraničení nastavíme $c_1 = 0,8$ a $c_2 = 0,5$. Druhá vrstva má koeficienty $c_1 = 0,4$ a $c_2 = 0,5$ a vnější hranice má $c_1 = 0$ a $c_2 = 0,5$. Tyto koeficienty jsou součástí inicializačních hodnot modelu a lze je samozřejmě podle potřeby měnit.

V inicializační části je třeba definovat typ každé buňky. To je opět učiněno dynamicky alokovanou dvourozměrnou maticí typu `char **`. Při samotné simulaci se prochází všechny buňky. Nejdříve je k buňce vypočítáno její ψ, se kterým se bude pracovat. Poté se aplikuje cyklus přes všechny směry, kde se vypočítají souřadnice, kam se budou hodnoty přesouvat. Podle typu zdrojové buňky je pak vybrán vzorec, který se aplikuje. Po ukončení průchodu buňkami se poté nastaví hodnoty zdrojů stejně, jako v minulém modelu.

5.3 Paralelizace

Jako prostředek paralelizace výpočtu modelu byla vybrána metoda multiprocesorového výpočtu se sdílenou pamětí. Tento systém byl vybrán jako nejvhodnější pro implementované modely. Implementované modely totiž většinou pracují s plochami, které mají přiměřenou velikost. Rozdělení takové plochy pro jednotlivé počítací stroje by bylo tedy velmi husté, čili úroveň komunikace s ostatními stroji by byla ve špatném poměru vůči samotnému počítání na stroji.

Pro samotnou implementaci byla vybrána knihovna OpenMP, která poskytuje pohodlný nástroj, jak zparalelizovat aplikaci pomocí několika vhodně umístěných příkazů, respektive parametrů pro překladač. Kromě zapsání parametrů pro překladač byla nutná větší úprava zdrojových kódů modelu. Na serveru, na kterém probíhaly testy chyběla instalace patřičné knihovny Qt, proto bylo upuštěno od grafické vizualizace. Zásuvný modul představující vybraný model byl tedy předělán na autonomní aplikaci, která sama provádí načtení modelu ze souboru a výpočet zadaného počtu kroků simulace. Všechny ostatní metody pro komunikaci s uživatelským rozhraním se nevyužívají a jsou tak zbytečné. Je ale nutné zmínit, že v aplikaci s grafickým rozhraním se podílejí také na časové náročnosti, protože se v nich provádí průchod celým polem a samotná vykreslovací funkce je velkým konzumentem času. Když však požadujeme provedení velkého počtu kroků simulace, jedno provedení nahrání a vykreslení pole můžeme klidně zanedbat.

Paralelizace pomocí knihovny OpenMP probíhá pomocí vhodného umístění následující direktivy:

```
int i;
#pragma omp parallel for shared(xxx) private(yyy)
for(i=0;...)
{
   //kód, který bude probíhat paralelně
}
```

Tato direktiva zajistí vytvoření maximálního počtu vláken, pokud není nastaveno jinak funkcí `omp_set_thread_num()`. Tato vlákna si poté rovnoměrně rozdělí práci, která je zapsána v cyklu. Důležitou součástí je nastavení sdílenosti proměnných, v direktivě označované jako `shared()` a `private()`. Proměnné označené jako `xxx` představují ty, které všechny procesy sdílejí. V našem případě se jedná zejména o plochu modelu s buňkami, které se přepočítávají. Tato plocha je rozdělena mezi jednotlivé procesy, které zpracují předepsanou část.

Proměnné označené jako `yyy` představují ty proměnné, jejichž kopie jsou nahrány každému procesu. Tyto proměnné se poté mohou v rámci procesu měnit a jejich změna se neprojeví nikde jinde. Jako tyto proměnné jsou uvažovány zejména iterační proměnné cyklů a jiné pomocné proměnné. V této ukázce se bude jednat zejména o proměnnou `i`.

Kapitola 6

Ukázkové příklady

V následující kapitole jsou popsány ukázkové modely, na nichž byly jednotlivé zásuvné simulátory testovány. Součástí těchto modelů jsou i záznamy grafické plochy, které ukazují vývoj modelu v čase, nebo jeho zachycení v jednom konkrétním časovém momentu. Každý model rovněž obsahuje slovní popis podmínek, se kterými se v modelu pracovalo a rovněž slovní zhodnocení výsledku simulace. Poslední část této kapitoly je věnována analýze paralelního řešení vybraného modelu.

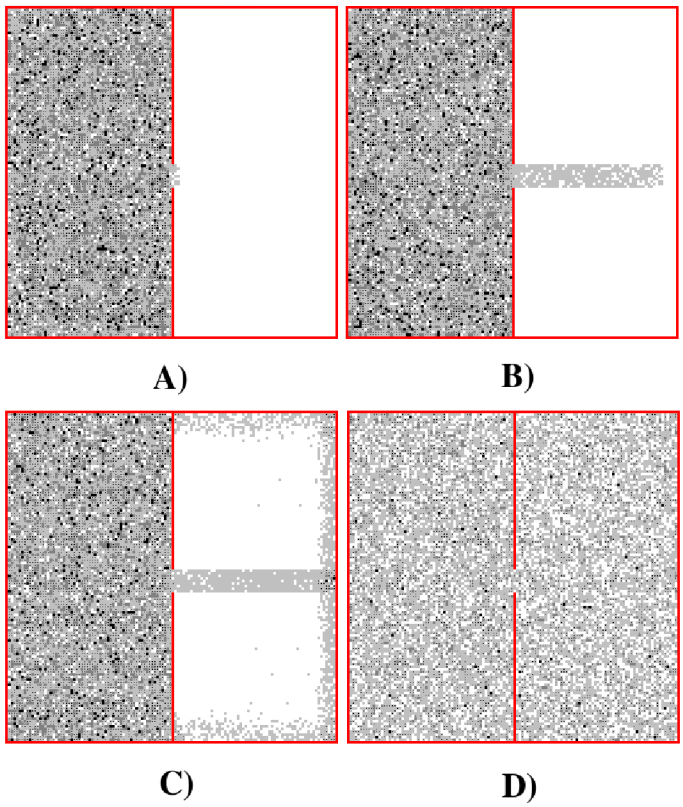
6.1 Simulace expanze částic – HPP pravidla

Na obrázku 6.1 je vidět výstup ukázkového příkladu, představující expanzi částic. Jedná se o příklad, kde na levé straně je v uzavřené oblasti vygenerovaný náhodný rej částic. To může představovat například oblast naplněnou nějakým plynem. Na pravé straně je oproti tomu prostor čistý od tohoto plynu. Viditelné body na ploše představují jednotlivé kolizní místa a jejich barva je určena hustotou částic. Když se v místě nachází méně částic, je buňka světlá. Při obsazení všech čtyř směrů je nejtmavější. Červené buňky představují speciální místa, čili neprostupné zdi.

Na obrázkové sekvenci je vidět, jak mezerou ve zdi oddělující dvě části proniká proud částic. Tento proud je rovný a ohraničený, což je dáno abstrakcí modelu. Při nárazu částic na zeď se začnou částice odrážet zpět a to způsobí rozptýlení i do okolního prostoru. Po uplynutí dostatečného množství kroků je vidět, že plyn je již rovnoměrně rozptýlen v obou komorách.

Tento model se oproti realitě v mnoha ohledech liší díky nízké entropii, kterou získává abstrakcí prostoru a fyzikálních zákonitostí. Avšak i tak představuje myšlenkový základ, který se využívá při modelování fyzikálních jevů. Rovněž je tento model použitelný pro ukázku reverzibilního celulárního automatu, protože pravidla užitá v tomto modelu se touto vlastností vyznačují.

Tento model byl vytvořen na ploše o rozměrech 128 x 128 kolizních buněk. V modelu se dále vyskytuje jedna oblast, kde se mají generovat náhodné buňky a šest oblastí představující zdi. Konkrétní zápis modelu je možno zhlédnout na přiloženém CD v souboru `/modely/expanze.hpp`.



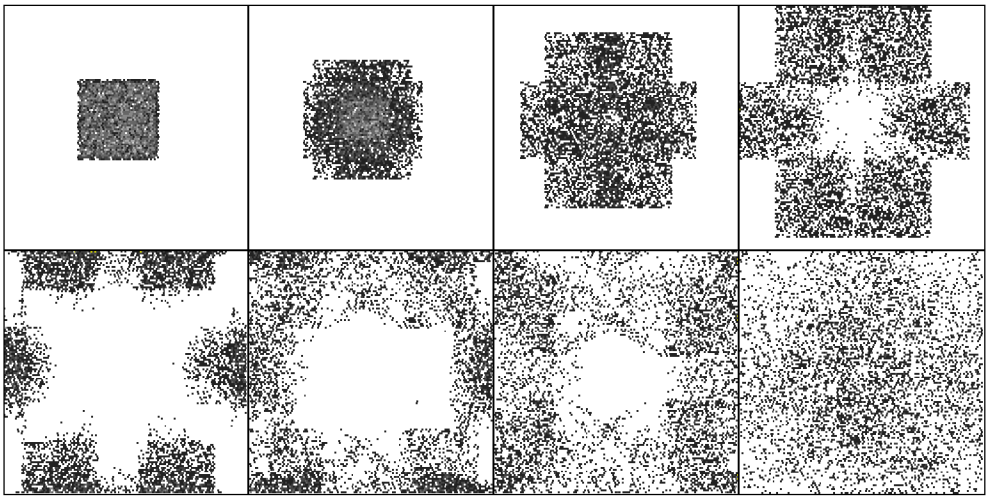
Obrázek 6.1: A) – T=2; B) – T=20; C) – T=50; D) – T=100;

6.2 Simulace vyrovnání energetického potenciálu – FHP pravidla

Na obrázku 6.2 je znázorněn ukázkový model pro FHP pravidla po provedení 1, 10, 25, 40, 55, 70, 90 a 120 krocích. Tento model může být označen jako model exploze. Je z něj vidět, že na začátku jsou částice zhuštěny uprostřed plochy. Poté jsou svými silami vymrštěny do okolí ve shlucích, které odpovídají šesti možným směrům šíření částic. Tyto shluky se pohybují prostorem dokud nenarazí na okraje plochy, které představují zdi. Zdi tyto shluky vrátí zpět do prostoru kde se dojde ke střetům shluků a všechny částice se nakonec rovnoměrně rozptýlí po ploše.

Tímto jednoduchým příkladem lze například dokázat, že energie částic se po jisté době rozprostře po celé ploše a nezůstávají oblasti s extrémní hustotou částic. Po jisté době relaxace tak vznikne na ploše energetické ekvilibrium. Je nutné také zmínit, že tento model má větší entropii oproti předchozímu modelu a to díky využití šestisměrných kolizních bodů a díky využití pravděpodobnosti při výpočtu kolizních operátorů. Tento model se tedy velmi blíží skutečnému světu.

Tento model byl rovněž vytvořen na ploše o rozměrech 128 x 128 buněk. Oblast uprostřed je tvořena čtvercem náhodných buněk o velikosti 40x40 buněk. Soubor s modelem, který byl takto nahrán je na přiloženém CD s názvem `/modely/rovnovaha.fhp`.



Obrázek 6.2: Vývoj modelu představující vyrovnání energetického potenciálu

6.3 Simulace sypání písku

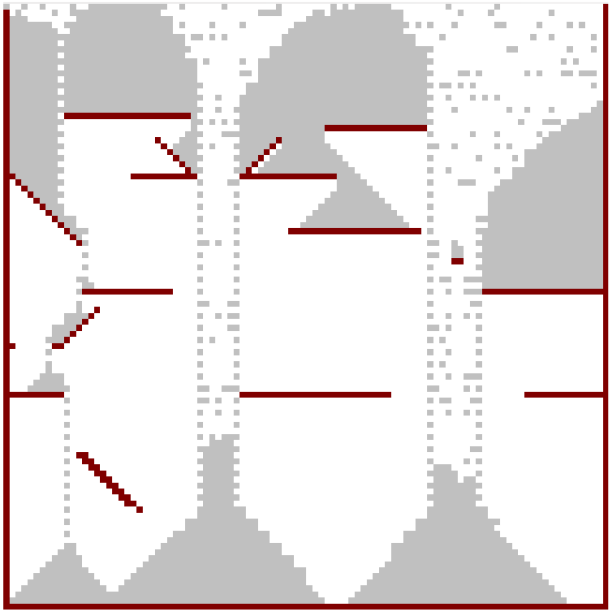
Obrázek 6.3 představuje ukázkový příklad, kde je vidět sypání písku přes soustavu překážek. Písek se na těchto překážkách zachytává a tvoří se hromady, přes které se písek přesypává dále. O sypání písku se starají sypače umístěné po celé horní straně plochy. Nový písek v sypačích se tvoří s pravděpodobností 15%.

Na modelu si lze všimnout, že nevykazuje přesné fyzikální vlastnosti jaké jde spatřit v reálném světě. To lze vidět zejména na stohovatelnosti zrnek. Sype-li se písek v poměrně úzkém proudu avšak s velkým průtokem, vzniká nám na vrcholcích hromad jakýsi komín. Rovněž protéká-li tento proud podél stěny tvořené pískem, má přednost tento proud, místo toho aby se stěna zbortila a sesypala. Tyto vlastnosti jsou však dány abstrakcí modelu a nemůžeme je považovat za chybu.

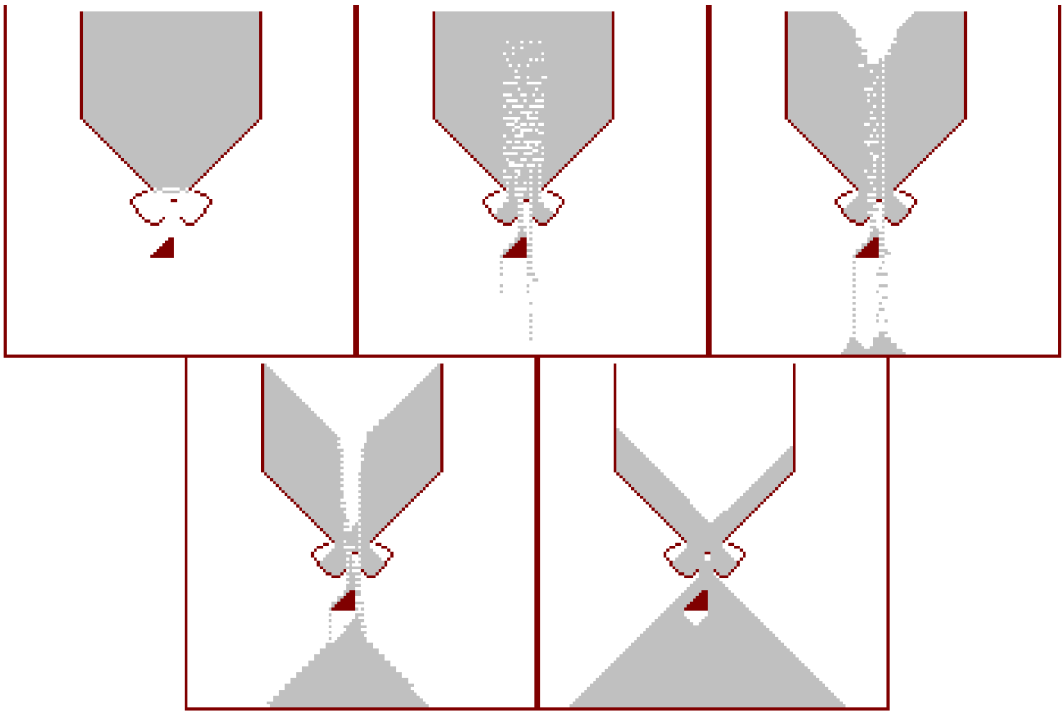
Model je vytvořen na ploše 100 x 100 buněk. Dále je definována oblast sypacích buněk na horním okraji plochy. Překážky a okrajové zdi jsou pak definovány pomocí textového výpisu buněk. Hodnoty tohoto modelu jsou uloženy na přiloženém CD v souboru `/modely/pisek.snd`.

Obrázek 6.4 představuje jiný model, který byl testován. Jedná se o násypku písku s komplikovaným vyústěním. U tohoto modelu bylo využito vylepšení, které umožňuje změnit charakteristiku modelu. Jedná se o využití pravděpodobnosti v transformačních pravidlech. U tohoto modelu bylo nastaveno to, že dvě vedle sebe postavené buňky písku spadnou pouze s pravděpodobností 85%. Model byl vytvořen na ploše 120 x 120 buněk a podrobný popis modelu je na přiloženém CD v souboru `/modely/pisek_nasypka.snd`.

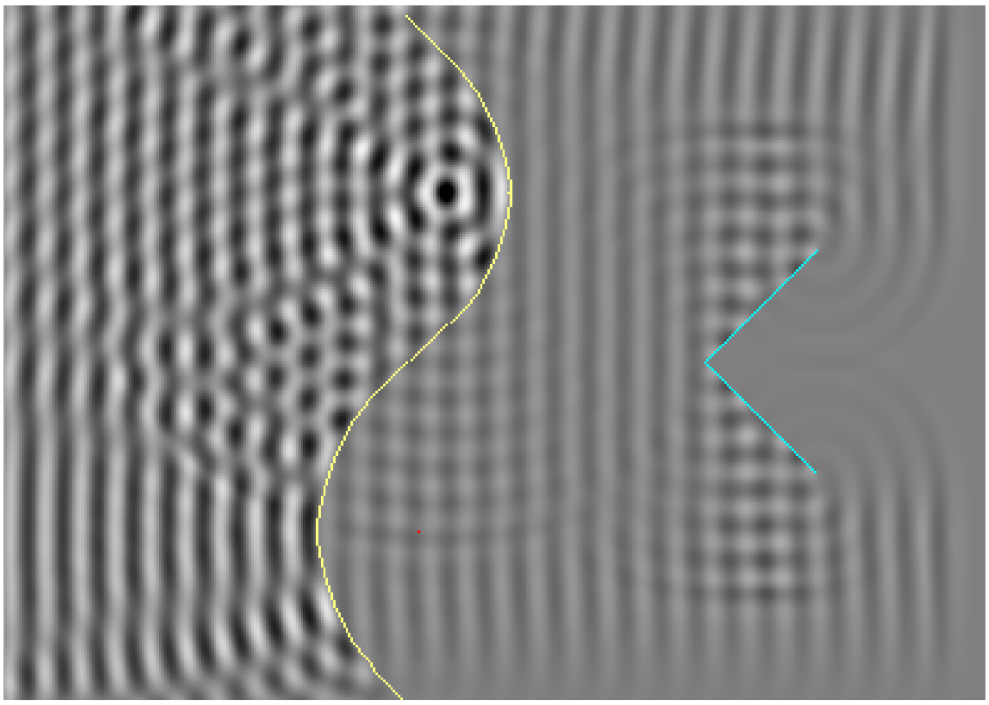
Chceme-li metodu sypání písku lépe využít pro modelování skutečného světa, musíme zavést do modelu komplexitu. Nejjednodušším způsobem je zavedení pravděpodobnosti v přechodových pravidlech. Rovněž můžeme zavést různé druhy pískových zrnek, jako například takové, které druhé zrnko na sobě udrží a jiné, po kterém se zrnko sklouzne. Rovněž by mohlo přispět k větší realitě zavedení šestiúhelníkových buněk. Takto vytvořené modely by měly větší míru nestability, která by se více podobala přírodě.



Obrázek 6.3: Model sypání písku přes soustavu překážek



Obrázek 6.4: Model pískové násypky s komplikovaným vyústěním



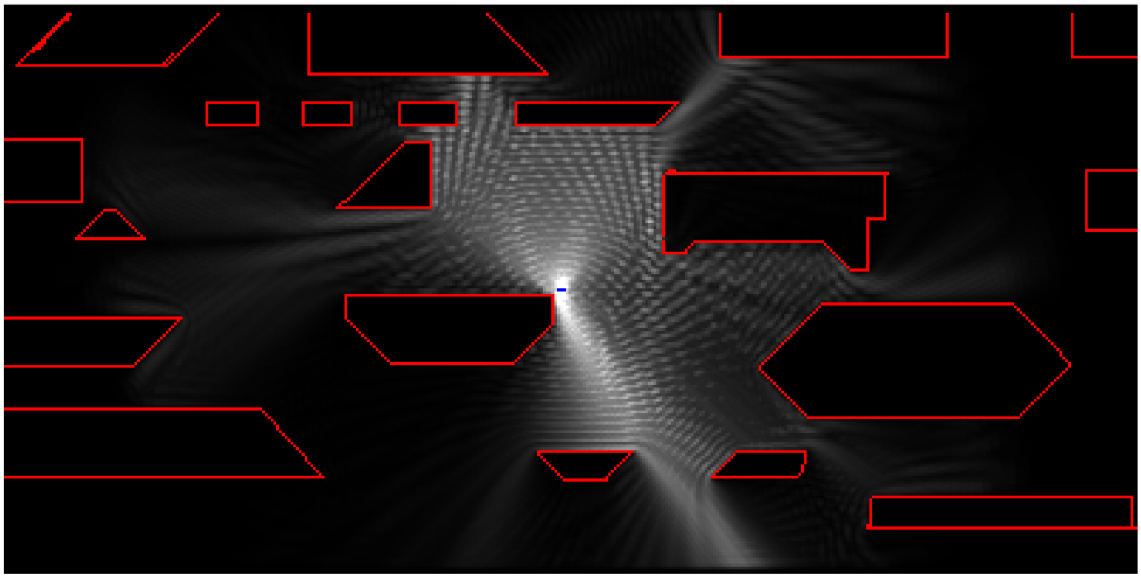
Obrázek 6.5: Model šíření a odrazu vlnění od nepropustných a polopropustných zrcadel

6.4 Simulace odrazu vlnění

Obrázek 6.5 představuje výstup ukázkového příkladu. Jedná se o model, který byl původně zamýšlen jako soustava zdroje vlnění a čoček, které budou soustřeďovat vlnění do jednoho bodu. Přenos, lom a odraz vlnění je řešen pomocí definovaných matic v kapitole 3.3. Ukázalo se však, že pravidla, která měla způsobovat lom světla spíše způsobují polopropustnost vlnění, čili jistá část vlnění je propuštěna ve svém původním směru a část vlnění se odráží. Množství propuštěného a odraženého vlnění určuje koeficient pro lom, který musí být větší roven 1. Pro $index = 1$ je zrcadlo zcela nepropustné a se zvyšujícím indexem vzrůstá i podíl propuštěného vlnění.

Model tedy byl předělán na soustavu dvou polopropustných zrcadel, jednoho vypuklého a druhého dutého. Za tato zrcadla jsou umístěna dva zcela nepropustná zrcadla. Z obrázku 6.5 je vidět, že část vlnění se od zrcadel odráží zpět do prostoru a interferuje s okolním vlněním. U dutého zrcadla je jasně vidět, že vlnění se soustředí v jeho ohnisku. Dále lze vidět, že zbytek vlnění prochází přes zrcadla dále ve svém původním směru. Toto vlnění posléze narazí na nepropustná zrcadla, která jsou vůči vlnění postavena v úhlu 45°. Od těchto zrcadel se vlnění odrazí tak, že pokračuje kolmo na svůj původní směr a rovněž interferuje s ostatním vlněním. Za hranicí zrcadel je vidět, že vlnění se opět láme zpět za zrcadla. Toto je způsobeno lomem paprsku o hranu zrcadla.

Tento konkrétní model byl vytvořen na ploše 300x650 buněk. Perioda vlnění byla nastavena na hodnotu $T = 20ms$, amplituda $A = 25$. Prostorový krok je nastaven na hodnotu $\Delta x = 1$ a jeden časový krok představuje hodnotu $\Delta t = 1ms$. Podrobný popis plochy je možné najít na přiloženém CD v souboru `/modely/zrcadla.wv1`.



Obrázek 6.6: Model šíření rádiového signálu v imaginární zástavbě

6.5 Simuace šíření rádiového signálu v městské aglomeraci

Obrázek 6.6 představuje výstup ukázkového modelu pro šíření rádiového signálu v městské aglomeraci. Tento model používá pravidla uvedena v závěru části 3.3.

Červeně ohraničené oblasti představují domy, čili překážky vyskytující se běžně v městské zástavbě. Zdroj signálu – anténa je vyznačena modrými buňkami. Zbytek buněk představuje otevřený volný prostor, kterým se může signál volně šířit.

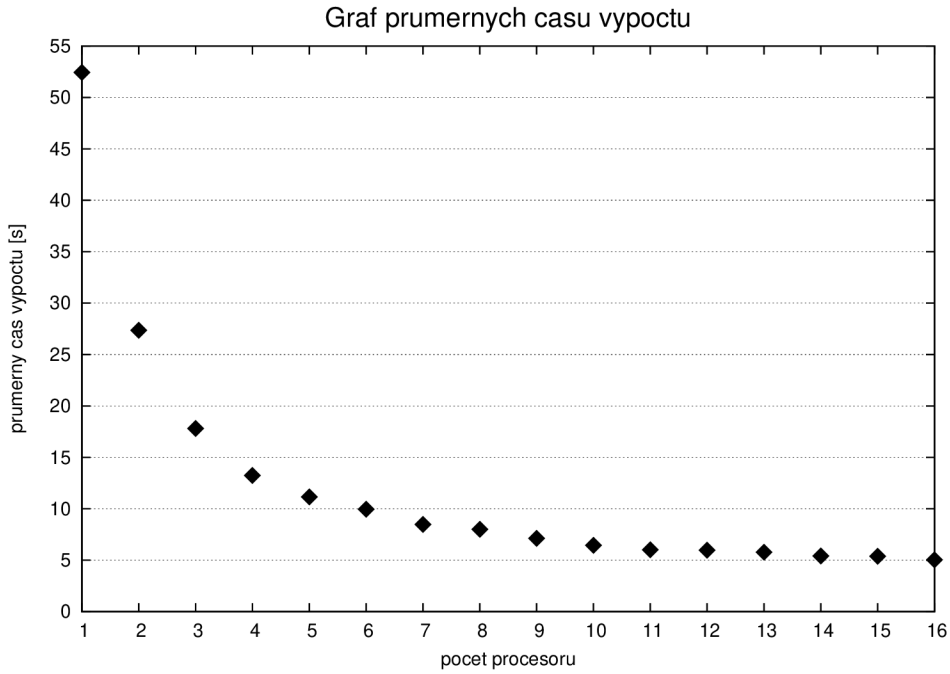
Graficky je znázorněna intenzita signálu, která jde v daném místě naměřit. Černá barva představuje nulovou intenzitu signálu. Se zvětšující se intenzitou se barva více blíží barvě bílé. Intenzita je chápána jako poměr amplitudy signálu naměřené v dané buňce k amplitudě zdroje.

Na obrázku jsou patrné odrazy signálu od zdí budov. Rovněž je možné si všimnout míst, kde dochází k interferencím signálu, tedy místa kde se fáze přijatého a odraženého signálu vzájemně odčítají. Tato místa je však nutné brát s rezervou, přeci jen se stále jedná o model s jistou dávkou abstrakce. Snímek byl pořízen po 150 krocích simulace.

Tento model byl vytvořen na ploše o rozměrech 200 x 650 buněk. Perioda rádiového signálu byla zvolena na $T = 8s$ a amplituda na hodnotu $A = 0,805$. Hodnoty zadávaných konstant c určující odraz a absorbci byly zvoleny na hodnoty: $c_{odraz} = 0,7$, $c_{abs1} = 0,8$, $c_{abs2} = 0,4$ a $c_{abs3} = 0$. Konkrétní zápis modelu je možno nalézt na přiloženém CD v souboru `/modely/zastavba.wv2`.

6.6 Paralelní řešení vybraného modelu

Jako testovací model, který byl zparalelizován a otestován byl vybrán model šíření a odrazu vlnění, který využívá výše zmíněných matic. Tento model byl však zvětšen na plochu o velikosti 500 x 800 buněk, aby byla zajištěna vyšší časová a výpočetní náročnost. Rovněž bylo po modelu vyžadováno, aby bylo provedeno 725 kroků, čímž se náročnost ještě zvětšuje.



Obrázek 6.7: Graf průměrných časů výpočtů pro jednotlivý počet procesů

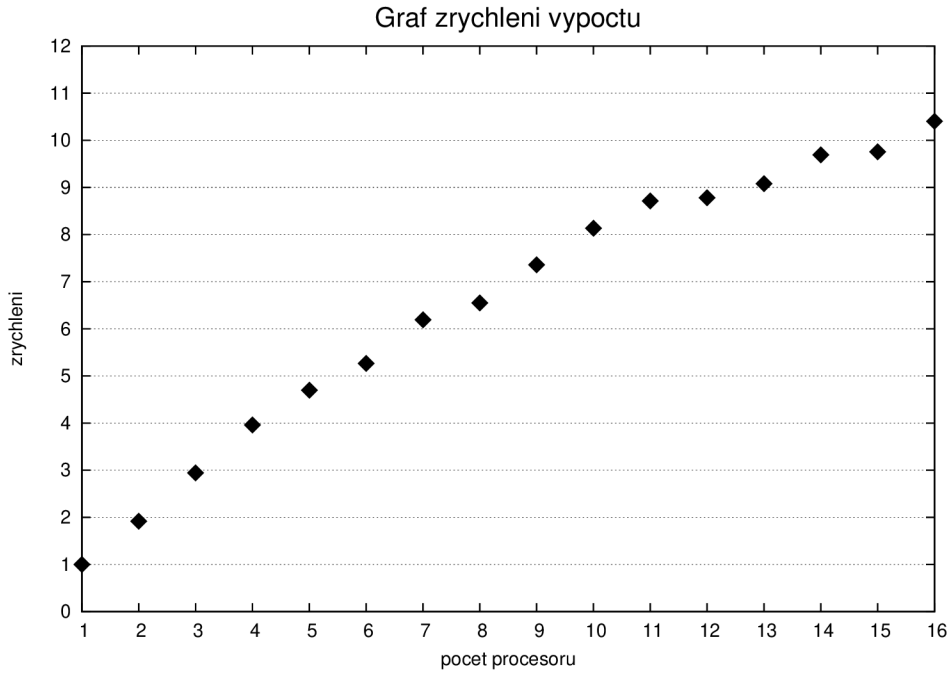
Tento program byl testován na školím serveru *muf.fit.vutbr.cz* tak, že parametrem programu se nastavil maximální počet procesů, které mohou daný výpočet provádět. Tímto se nastavoval počet procesů od 1 do 16. Pak bylo vždy pro každé nastavení počtu procesů provedeno pět testovacích spuštění programu. Během každého chodu programu byl zaznamenán čas, po který se výpočet prováděl. Ze zaznamenaných časů byl poté vypočítán průměrný čas, se kterým se dále pracovalo.

Obrázek 6.7 představuje průměrné časy výpočtu při různých počtech pracujících procesorů. Je vidět, že časy se výrazně měnily zejména v první polovině grafu, čili do rozdělení výpočtu mezi osm procesů. Poté měřené časy klesaly jen velmi pomalu v úrovni sekundových desetinných míst. Čas, ke kterému se doba chodu programu přibližovala, byl z jisté části tvořen sekvenční částí programu, zejména pak načtení modelu ze souboru. Toto načtení totiž nebylo zparalelizováno a tak se tedy výrazně promítalo do celkové doby chodu programu. Zbylý čas pak představovalo ono paralelní počítání a rovněž také systémový čas, který je nutný pro vytvoření a obsloužení jednotlivých vláken. Tento čas se se zvyšujícím počtem pracujících procesů rovněž výrazněji zvětšoval.

Graf na obrázku 6.8 zrychlení, kterého bylo dosaženo zavedením více procesů. Hodnoty jsou vypočítány podle vzorce:

$$zrychleni_i = \frac{cas\_vypoctu_1}{cas\_vypoctu_i}$$

kde i představuje počet procesů, pro které zrychlení počítáme. Zrychlení se váže k času provedení jedním procesem, čili času plně sekvenčního výpočtu.



Obrázek 6.8: Graf zrychlení výpočtů

Z grafu vidíme, že do čtyř procesů odpovídá zrychlení onomu počtu procesů. Dále se však růst zrychlení začíná zpomalovat a na konci nemáme při šestnácti procesech zrychlení 16, ale pouze hodnotu cca 10,4. Za toto snížení může jak zmíněná sekvenční část programu, tak režie spojená s obsluhou procesů.

Snižování zrychlení procesorů se rovněž projevuje na grafu účinnosti využití procesorů, který je vidět na obrázku 6.9. Hodnoty tohoto grafu jsou sestrojeny pomocí vzorce:

$$ucinnost_i = \frac{zrychleni_i}{i} \cdot 100\%$$

Z grafu vidíme, že čím menší je zrychlení výpočtu, tím menší je i využití jednotlivých procesorů během výpočtu.

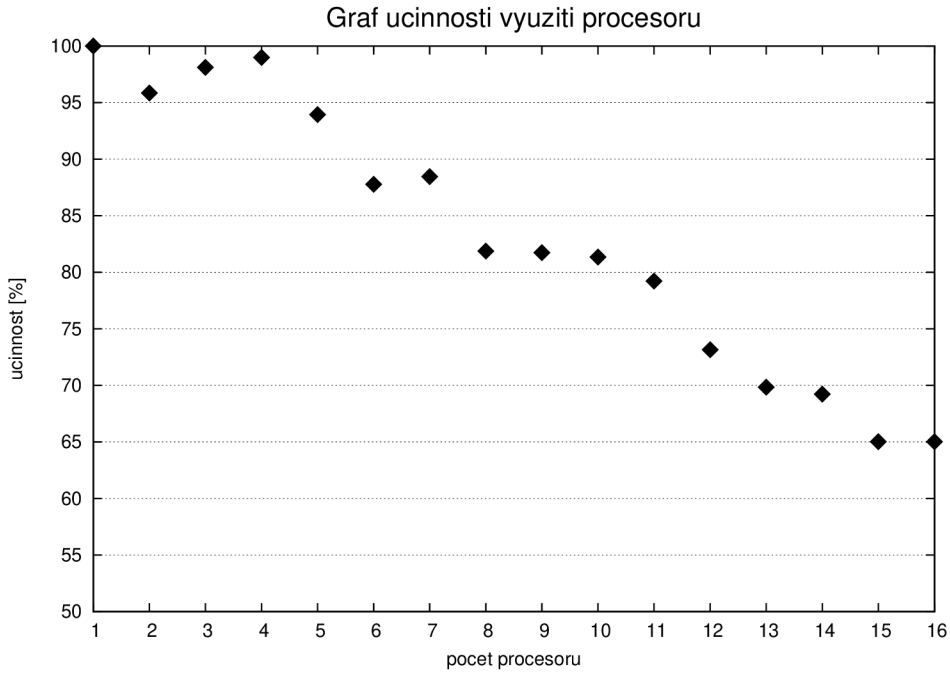
Tabulky naměřených hodnot jsou uvedeny v tabulce v příloze. Zdrojový program i s testovacím modelem se nachází na přiloženém CD v adresáři `/parallel`.

Parametry testovacího serveru:

```
128 GB RAM
4x Quad-Core AMD Opteron(tm) Processor 8389 (16x core)
800 MHz
```

6.7 Validace výsledků

Přestože validace výsledků není obsahem zadání této práce a nebyla prováděna, považuji za vhodné o této fázi v modelování a simulaci zmínit alespoň několik vět. Zejména proto,



Obrázek 6.9: Graf účinnosti využití jednotlivých procesorů

že se jedná o modely fyzikálních jevů a většina fyzikálních jevů a jejich objasňujících tezí je ve svém závěru potvrzována reálnými fyzikálními pokusy. Nejinak tomu je i u modelů vytvořených pomocí celulárních automatů.

Poměrně známým a jednoduchým modelem, jehož výsledky lze snadno ověřit fyzikálním pokusem je model sypajícího se písku. V laboratorních podmínkách lze snadno vyrobit skutečný model s pískem a například pomocí fotoaparátu nebo kamery zkoumat průběh během sypání. Článek [6] například popisuje model, kdy byl zkoumán průběh sypání písku ze sila a porovnáván jeho výsledek s počítačovým modelem. Zde sice nebylo dosaženo zcela stejných výsledků, avšak to je způsobeno onou komplexitou reálného světa. Konečným závěrem článku je ale fakt, že počítačový model se hodí pro rychlé odzkoušení proudů, které vznikají v silech a může tak sloužit během návrhu takových sil.

Dalším modelem, o němž je v literatuře vedena zmínka o reálné validaci výsledků je model šíření rádiového vlnění. V [7] je popisován model šíření signálu pro švýcarského mobilního operátora. Byla vytvořena mapa části města Bern se všemi budovami a ulicemi. V modelu pak byla umístěna anténa a byla spuštěna simulace. Posléze byl vytvořen graf intenzity signálu při imaginárním průjezdu jednou ulicí. Následně byl tento pokus převeden do praxe, kdy byla ve skutečném městě umístěna anténa a onou ulicí projel skutečný měřič intenzity signálu a byl zaznamenán stejný graf. Při porovnání obou grafů jsou vidět jisté rozdíly, avšak základní obrysy intenzit jsou shodné.

Kapitola 7

Závěr

Cílem této práce bylo zejména prozkoumat metody, které se používají pro simulaci fyzikálních jevů a které k tomu využívají prostředků celulárních automatů. Byl tedy proveden průzkum dostupných publikací a vybráno několik metod, které byly ukázkově implementovány. Úvod práce má tedy zejména přehledový charakter, kdy se může čtenář seznámit se základními typy a směry, které se pro modelování fyzikálních jevů využívají. Fyzikálních jevů a metod pro jejich modelování existuje mnohem větší množství, než může být v této práci uvedeno, přesto je v práci obsažen alespoň jejich výčet a odkazy na literaturu, která se těmito problémy zabývá.

Metody, které tvoří hlavní myšlenkový základ modelování fyzikálních jevů byly vybrány a rozvedeny v této práci. K metodám nemohla být rozvedena veškerá matematická báze vedoucí k odvození daných pravidel, byl však ukázán hlavní princip, jak bylo pravidel dosaženo.

Jedním z cílů práce bylo rovněž navrhnout a implementovat vhodný simulátor fyzikálních jevů společně se sadou ukázkových příkladů, které mohou sloužit jak ve výuce, tak pro jejich využití ve výzkumu a v neposlední řadě i při modelování skutečných problémů. Zadání bylo rozšířeno o možnost využití zásuvných modulů v simulátoru, což výrazně zvyšuje univerzálnost použití simulátoru.

Aplikace byla vytvořena tak, aby byla co nejvíce univerzální a umožňovala uživateli velmi jednoduchou modifikaci modelů a metod jejich výpočtu. K aplikaci je pak možno vytvářet nové zásuvné moduly, které budou do aplikace nahrány a bude tak rozšířena její funkčnost bez nutnosti měnit kód celé aplikace. Pokud by pozdější uživatel potřeboval pro model jiné typy vykreslování buněk na obrazovku, je možné pouze modifikovat třídu `Widget`.

K aplikaci byla vytvořena i sada ukázkových příkladů, které mohou posloužit při výuce k ukázce schopností celurárních automatů v oblasti modelování fyzikálních jevů. Modelů pro sypání písku a šíření rádiového vlnění lze využít i pro řešení problémů vyskytujících se v běžném životě, jako například problému pokrytí místností wifi signálem.

Aplikaci a celou práci lze rovněž využít jako základ pro rozšíření a další výzkum této obsáhlé problematiky. Zde je možné provést průzkum dalších typů metod, které se pro modelování fyzikálních jevů používají. Podle mého soudu by však bylo mnohem vhodnější se v budoucnu zaměřit pouze na jednu kategorii modelů a tuto oblast prozkoumat podrobněji. Pro dané modely by se mohla provádět podrobná verifikace přístupů a validace získaných dat. Zároveň by bylo možné zkoušet vlastní přístupy a vylepšení k řešení daného problému.

Literatura

[1] Cannataro, M. – Gregorio, S. D. – Rongo, R. – aj.: A parallel cellular automata environment on multicomputers for computational science. *Parallel Computing*, ročník 21, č. 5, 1995: s. 803–823, ISSN 0167-8191, doi:http://dx.doi.org/10.1016/0167-8191(94)00099-V.

[2] Chopard, B. – Droz, M.: *Cellular automata: modeling of physical systems*. Cambridge University Press, Cambridge, 1998, ISBN 0-521-46168-5, 341 s.

[3] Chopard, B. – Dupuis, A. – Masselot, A. – aj.: Cellular Automata And Lattice Boltzmann Techniques: An Approach To Model And Simulate Complex Systems. *Advances in Complex Systems (ACS)*, ročník 5, č. 02, 2002: s. 103–246.
URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.26.4437&rep=rep1&type=pdf

[4] Conevey, P. – Highfield, R.: *Mezi chaosem a řádem*. Mladá fronta, Praha, 2003, ISBN 80-204-0989-0, 440 s.

[5] Dupuis, A.: *From a lattice Boltzmann model to a parallel and reusable implementation of a virtual river*. Dizertační práce, Faculté des sciences de Université de Genève, 2002.
URL http://cui.unige.ch/~chopard/CA/aDupuisPhD.pdf

[6] Kozicki, J. – Tejchman, J.: Application of a cellular automaton to simulations of granular flow in silos. *Granular Matter*, ročník 7, č. 1, 2005: s. 45–54, ISSN 1434-7636, doi:10.1007/s10035-004-0190-x.

[7] Lüthi, P. O.: *Lattice wave automata - from radio waves to fractures propagation*. Dizertační práce, Faculté des sciences de Université de Genève, 1998.
URL http://cuiwww.unige.ch/~luthi/a.pdf

[8] Marconi, S.: *Mesoscopical modelling of complex systems*. Dizertační práce, Faculté des sciences de Université de Genève, 2003.
URL http://spc.unige.ch/_media/these_stefan_marconi.pdf

[9] Reiter, C. A.: A local cellular model for snow crystal growth. *Chaos, Solitons & Fractals*, ročník 23, č. 4, 2005: s. 1111 – 1119, ISSN 0960-0779, doi:10.1016/j.chaos.2004.06.071.
URL http://www.sciencedirect.com/science/article/B6TJ4-4D5JT6M-5/2/8c0b11e2bb57660a5497fe6e4bb98c9b

[10] S., W.: *New kind of science*. Wolfram Media, USA, 2002, ISBN 1-57955-008-8.

[11] Weimar, J.: *Cellular Automata for reactive systems*. Dizertační práce, Université Libre de Bruxelles, Faculté de Sciences Service de Chimie Physique, 1995.
URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.31.529&rep=rep1&type=pdf

[12] Wikipedia.org: Navier-Stokes equations. [Online], 17.5. 2010.
URL http://en.wikipedia.org/wiki/Navier-Stokes_equations

Dodatek A

Tabulka naměřených hodnot paralelního řešení

A.1 Naměřené hodnoty

počet procesů	t1	t2	t3	t4	t5
1	52,380	52,316	52,509	52,496	52,443
2	26,615	30,113	26,644	26,629	26,753
3	17,671	17,678	17,876	18,018	17,810
4	13,221	13,174	13,167	13,157	13,489
5	10,980	11,023	10,748	10,787	12,271
6	9,937	9,127	9,981	9,662	11,068
7	8,191	8,027	9,503	8,072	8,546
8	8,281	8,522	7,117	8,240	7,866
9	6,528	7,125	7,970	6,943	7,066
10	5,802	6,251	6,375	6,664	7,134
11	5,521	6,090	6,292	6,068	6,110
12	5,641	5,706	5,622	6,747	6,144
13	6,051	5,558	5,937	5,125	6,200
14	5,386	5,311	5,247	5,235	5,869
15	5,754	5,283	5,682	5,081	5,078
16	5,177	4,369	5,100	5,191	5,360

A.2 Vypočítané hodnoty

počet procesů	průměrný čas[s]	zrychlení	účinnost[%]
1	52,429	1,000	100,000
2	27,351	1,917	95,845
3	17,811	2,944	98,123
4	13,242	3,959	98,985
5	11,162	4,697	93,943
6	9,955	5,267	87,776
7	8,468	6,192	88,451
8	8,005	6,549	81,867
9	7,126	7,357	81,744
10	6,445	8,135	81,345
11	6,016	8,715	79,224
12	5,972	8,779	73,159
13	5,774	9,080	69,845
14	5,410	9,692	69,227
15	5,376	9,753	65,021
16	5,039	10,404	65,024