ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Návrh a implementace buněčného automatu simulujícího dynamickou rekrystalizaci
Student:	Jakub Tkáč
Vedoucí:	RNDr. Jiří Kroc, Ph.D.
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Cílem práce je navrhnout a implementovat buněčný automat simulující dynamickou rekrystalizaci. Teorie výpočtu je zpracovaná v publikaci [1]. Aplikace bude sestávat z back-end, který již existuje ve speciálním jazyku CellLang a je možné jej snadno portovat do C++ a z front-end, který je třeba navrhnout a implementovat.

1. Seznamte se s koncepcí návrhu a implementace buněčného automatu simulujícího dynamickou rekrystalizaci.
2. Realizujte přepis stávajícího back-end v jazyku CellLang do C++.
3. Navrhněte a implementujte front-end v jazyku C++ s použitím knihovny Qt.
4. Celou aplikaci řádně zdokumentujte a otestujte (Uživatelská a programátorská příručka bude anglicky).

Seznam odborné literatury

[1] Kroc J., Application of cellular automata simulations to modelling of dynamic recrystallization. Proceedings ICCS '02 Proceedings of the International Conference on Computational Science-Part I. ISBN:3-540-43591-3. 2002.

další citace dle dohody

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 7. února 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra Softwarového Inženýrství

Bakalářská práce

Návrh a implementace buněčného automatu simulujícího dynamickou rekrystalizaci

Jakub Tkáč

Vedoucí práce: RNDr. Jiří Kroc, Ph.D.

11. května 2017

Poděkování

Rád bych poděkoval vedoucímu práce RNDr. Jiřímu Krocovi, Ph.D. za uvedení do světa buněčných automatů a za cenné rady nejen v rámci této práce. Mimo jiné bych chtěl poděkovat Ing. Michalu Valentovi, Ph.D. za konzultování výsledné práce. V neposlední řadě děkuji rodině a kamarádům za podporu a trpělivost po celou dobu studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, avšak pouze k nevýdělečným účelům. Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 11. května 2017 .....

Odkaz na tuto práci

Tkáč, Jakub. *Návrh a implementace buněčného automatu simulujícího dynamickou rekrystalizaci*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Cílem této bakalářské práce je navrhnout, implementovat a vizualizovat buněčný automat simulující dynamickou rekrystalizaci. Implementace algoritmu byla sestavena na základě vědecké publikace, doktorské práce a existujícího kódu v jazyku CellLang. Uživatelské rozhraní je implementováno v jazyce C++ s využitím frameworku Qt. Výsledná aplikace zobrazuje mikrostrukturu a vizualizuje spočtené výsledky výpočtů uživateli. Jedná se o křivku mechanického napětí, křivku střední velikosti zrna a hustotu dislokací. Výsledky simulace lze ukládat. Je zde možnost načíst si inicializační mikrostrukturu nebo si vygenerovat vlastní v programu na základě definovaných parametrů pomocí pseudo-statické rekrystalizace. Volitelnost parametrů zaručuje možnost simulace materiálu deformovaného v různých podmínkách.

Klíčová slova Buněčné automaty, Dynamická rekrystalizace, Komplexní systémy, Simulace, Samo-organizace, Emergence, C++, Qt

Abstract

The aim of this bachelor thesis is to design, implement and visualize a cellular automaton simulating dynamic recrystallization. Implementation of the algorithm was based on scientific publications, doctoral thesis and existing code in CellLang. The graphical user interface is implemented in C++ using the Qt framework. The resulting application displays the microstructure and visualizes calculated results to the user: mechanical stress curve, mean grain size curve, dislocation density. The simulation results can be stored. It is possible to retrieve the initialization microstructure or to generate our own in the program based on defined parameters by pseudo-static recrystallization. Optional parameters guarantee simulation of material deformed under different conditions.

Keywords Celullar automata, Dynamic recrystallization, Complex systems, Simulation, Self-organization, Emergence, C++, Qt

Obsah

Seznam obrázků

Úvod

Vědci a materiáloví inženýři často potřebují simulovat chování materiálu v reálných aplikacích za běžného provozu. Matematické a počítačové modely jsou v tomto případě ideální způsob, jak tento problém řešit. Dokážeme díky nim odhadnout chování materiálu, aniž bychom výrobek zničili. I přes relativně úspěšné výsledky nám stále nedokáží aktuálně [1] používané metody zodpovědět všechny otázky [2]. Především nedokáže odpovědět na otázku, jaký vliv má změna rychlosti deformace na nukleaci, počáteční velikosti zrna a oscilaci střední velikosti zrna. V posledních 30-ti letech se začíná zvyšovat zájem o fyzikální modely zvané buněčné automaty, které odstraňují nevýhody předchozích řešení.

Počátky buněčných automatů sahají až do 40. let 20. století, kdy se John von Neumann a Stanislav Ulam snažili navrhnout model, který by se sám od sebe reprodukoval [3]. Definovali ho jako prostor rozdělený na jednotlivé buňky, kde je prostor a čas diskrétní. Je složený z mřížky buněk nabývajícího konečný počet stavů v závislosti na simulovaném jevu a definovanými pravidly pro vývoj buněk. Pro každý krok simulace buněčného automatu každá buňka změní svůj stav na základě stavů sousedních buněk a svého stavu. Z toho plyne, že chování je definované na lokální úrovni. Dohromady tvoří celek, který operuje na vyšší úrovni abstrakce reagující na vnější vlivy.

Takovým systémům se říká komplexní systémy. Důležitou vlastností komplexních systému je *emergence*, která popisuje chování systému na vyšší úrovni abstrakce, než na které operují lokální části. Nejmenší jednotkou tohoto systému je *emergent*. Chování není ničím výjimečným v přírodě. Mravenčí kolonie jsou toho důkazem. Každý mravenec je schopný reagovat na 20 až 40 vnějších signálů z okolí, na základě kterých se dále rozhoduje a komunikuje s dalšími mravenci [4]. Dalším příkladem můžou být ptáci poletující v hejnech, kde každý jednotlivý letec si udržuje rychlost a pozici na základě okolních ptáků. Tento jednoduchý mechanismus je vhodný pro simulaci široké škály jevů ve fyzice, chemii, biologii a paralelních výpočtů [5].

Cíle

Tématem bakalářské práce je vytvořit program, který bude simulovat dynamickou rekrystalizaci pomocí buněčného automatu. Díky tomuto programu bude vědcům a zájemcům o studii chování komplexních systémů umožněno nahlédnout do této disciplíny. Aplikace bude schopna během simulace zobrazovat buněčnou mikrostrukturu a další výsledky simulace ve formě grafů. Dále bude možnost vygenerovat inicializační počáteční mikrostrukturu, vložit vlastní předpřipravenou, a také uložení výsledků simulace do souborů ve formě obrázků, konfiguračních souborů nebo dat. Aplikace bude nabízet volnost konfigurace jednotlivých parametrů. Důležitým cílem je napsat program tak, aby byl jednoduše pochopitelný pro fyziky a ukázat jim, jak se dá simulace naprogramovat.

Aplikace bude open source, proto vědci a jiní programátoři budou mít možnost tento program v budoucnu rozšířit.

Struktura práce

První část práce je věnována úvodu do buněčných automatů, komplexních systémů a dynamické rekrystalizace. V druhé kapitole je podrobněji rozebrána simulace dynamické rekrystalizace pomocí buněčného automatu. Ve třetí kapitole je popsána analýza a jsou rozebrány požadavky na software. Čtvrtá kapitola pokračuje návrhem aplikace. Obsahuje popis třídní hierarchie, která se snaží respektovat fyzikální koncepty a návrh uživatelského rozhraní. V páté kapitole je čtenář seznámen s implementací programu a jsou mu představeny časté problémy, na které může při implementaci této domény narazit. Šestá kapitola se zabývá testováním. Je zde popsáno, jak testování probíhalo a na jaké testy bylo zaměřeno. Poslední kapitola obsahuje zhodnocení celé bakalářské práce, diskuze naměřených výsledků a výhled do budoucna.

Kapitola 1

Úvod do buněčných automatů a dynamické rekrystalizace

Tato kapitola slouží jako úvod do oblasti teorie buněčných automatů, teorie komplexních systémů a fyzikálního jevu dynamické rekrystalizace. Pro lepší přehlednost se v práci používá místo pojmu buněčný automat název celulární automat. Tyto pojmy jsou ekvivalentní a lépe se dá na ně odkazovat zkratkami CA (celulární automat) a CAs (celulární automaty). Navíc jsou tyto pojmy zažité v anglickém jazyce od slovíčka *cell* (buňka).

1.1 Komplexní systémy

Neexistuje přesná teoretická definice komplexních systémů, většinou je to jen skladba společných vlastností. Obecně by se dal komplexní systém definovat jako systém, který je složený z mnoha menších lokálně pracujících částí, které jsou schopny mezi sebou interagovat a navenek globálně reagovat s okolím. Koncept komplexních systému byl současně objeven v několika různých disciplínách [6, 7, 8]. To může svědčit o jejich univerzalitě. V následujících podkapitolách se nachází některé společné znaky, které se dají pozorovat.

1.1.1 Samo-organizace

„Samo-organizace se definuje jako spontánní (tj. není řízený nějakým externím systémem) proces organizace, tj. tvorba organizované struktury. Spontánní tvorba 'organizovaného celku' z 'neuspořádané' kolekce interagujících částí, jak o tom svědčí v samo-organizujících systémech ve fyzice, chemii, biologii, sociologii, je základní část dynamické emergence.“ [9]

V úvodu práce byl již jeden typický příklad uveden. U mravenců neexistuje žádný vůdce a komunikace mezi nimi probíhá pouze na lokální úrovni. I přesto jsou schopni společně vytvořit velké mraveniště nebo ochránit královnu.

Druhým příkladem je organizovaně letící ptactvo. Jednotlivec se po dobu letu snaží dodržovat tři jednoduchá pravidla. Snaží se nenarazit do blízkých sousedů, udržuje směr letu a dodržuje vzdálenosti mezi ostatními sousedy. Tímto přeskupováním lze pozorovat neustálé změny formací během letu.

Obrázek 1.1: Organizovaný let ptáků [10]

1.1.2 *Emergence*

Emergence je definován jako výskyt nových entit, které operují na vyšší úrovni abstrakce než samotné lokální pravidla. U našeho příkladu s mravenci je mraveniště výsledkem *emergence* [3].

1.1.3 Hierarchická organizace

Emergence slouží jako propojení mezi jednotlivými úrovněmi v komplexních systémech. Nejlepším příkladem je život na Zemi. Dalším příkladem může být vesmír. Obsahuje jednotlivé struktury atomů, přes molekuly, plyny, hvězdy až do jednotlivých galaxií.

1.2 Celulární automaty

Celulární automat je podle [11] diskrétní, abstraktní výpočetní systém, který je vhodný pro simulování obecných modelů složitosti a reprezentaci nelineární dynamiky napříč vědními obory. Je složený ze čtyř částí: mřížky buněk, jejich přidružených proměnných, množiny sousedů a lokálních pravidel. Prostor a čas je diskrétní. Pro každý krok simulace celulárního automatu každá buňka změní svůj stav na základě stavů sousedních buněk zároveň.

1.2.1 Mřížka

Mřížka je složena z buněk, které jsou uspořádány do jednoho, dvou, tří či více dimenzionálního prostoru. Buňky mohou být čtvercové, trojúhelníkové, hexagonální ve 2D nebo kubické ve 3D mřížce. V případě simulace dynamické rekrystalizace se využívá model složený z 2D mřížky čtvercových uniformních buněk, které jsou všechny aktualizované najednou na základě lokálních pravidel [12].

1.2.2 Proměnné

Celulární automat obsahuje určité množství diskrétních proměnných. Jejich počet závisí na typu simulovaného jevu. Nejjednodušší využívají jednu `boolean` hodnotu, která se dá přeformulovat následujícím způsobem. Buňka je ve stavu `jedna` nebo ve stavu `nula`. Tento druh proměnné lze najít i v CA zvaném *Game of Life*, který je popsán v práci později, viz kapitola 1.2.7.

1.2.3 Diskrétní stavy

Každá buňka v CA se nachází během simulace v nějakém stavu $\sigma \in \sum$. Kde $\sum$ je konečná množina všech možných stavů.

1.2.4 Sousedství/okolí

Vývoj každé buňky je závislý na sousedních buňkách *(angl. neighbours)*. Na základě stavů okolních buněk se vypočítá nový stav aktualizované buňky za použití lokálního pravidla. Nejčastější typy okolí jsou, viz obrázek 1.2.

- ***von Neumann* s poloměrem r**

  Daná buňka je ovlivněna 4 buňkami, které se nachází v jejím okolí ve směru 4 světových stran. Mnohdy se jednotlivé buňky označují právě světovými stranami na základě směru k dané buňce.

  Rovnice popisující sousedy lze definovat následující rovnicí [4], kde poloměr $r = 1$:

$$\begin{aligned} \mathcal{N}_N^r(i,j) &= \{\sigma_{k,l} || i-k| + |j-l| \leq r\} \\ &= \{\sigma_{i,j}, \sigma_{i-1,j}, \sigma_{i,j-1}, \sigma_{i+1,j}, \sigma_{i,j+1}\}. \end{aligned} \tag{1.1}$$

- ***Moore* s poloměrem r**

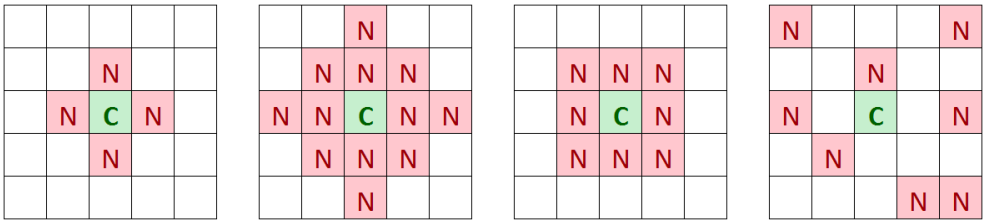
  Oproti *von Neumann* sousedství, které bylo tvořeno čtyřmi sousedními buňkami, které se dotýkaly přes hranu, jsou zde navíc buňky, které se naší buňky dotýkají vrcholem (diagonálně od nás). Daná buňka je tedy ovlivněna 8 sousedními buňkami.

Lze definovat pro poloměr $r = 1$:

$$\begin{aligned}\mathcal{N}_M^r(i,j) =& \{\sigma_{k,l} || i-k| \leq r, |j-l| \leq r\} \\ =& \{\sigma_{i,_j}, \sigma_{i,j-1}, \sigma_{i+1,j-1}, \sigma_{i+1,j}, \sigma_{i+1,j+1}, \\ & \sigma_{i,j+1}, \sigma_{i-1,j+1}, \sigma_{i-1,j}, \sigma_{i-1,j-1}\}.\end{aligned} \tag{1.2}$$

- **Náhodné sousedství**

  Daná buňka je ovlivněna náhodnou množinou okolních buněk.



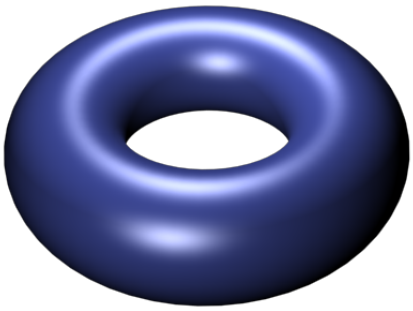
Obrázek 1.2: Typy okolí v mřížce velikosti $5x5$ buněk: (pořadí zleva) *von Neumann* okolí s poloměrem $r = 1$, *von Neumann* okolí s poloměrem $r = 2$, *Moore* okolí s poloměrem $r = 1$, a náhodné okolí. Buňky označené písmenem N jsou sousední pro buňku označenou jako C

1.2.5 Lokální/řídící pravidla

V angličtině možné najít pod názvy *transition rule* nebo *govering rule*. Lokální pravidla definují vývoj CA. Podle toho, jaký jev simulujeme, tak takové má náš CA lokální pravidla. Většinou se vezmou všechny proměnné z sousedních buněk a nad nimi se provede nějaká logická a/nebo aritmetická operace [3]. Hodnoty proměnných všech buněk se mění zároveň/paralelně.

1.2.6 Periodické okrajové podmínky

V celulárním automatu na okraji mřížky můžeme narazit na problém, jak vybírat sousední buňky. Z formální definice [13] celulárního automatu mřížka může být nekonečná nebo s hranicí. Kde na hranici má jiná pravidla než uvnitř mřížky. Na počítačích je požadavek nekonečné mřížky nemožný. Můžeme si však dopomoci trikem, a to tím, že předefinujeme okrajové podmínky na periodické [13, 14]. V 2D prostoru můžeme takto propojit horní a dolní část mřížky, a levou část s pravou částí. Po tomto spojení mřížka dostane tvar podobný toroidu.



Obrázek 1.3: Toroid, který vznikl rotací kružnice. V tomto případě vznikne torus [15]

1.2.7 *Game of Life*

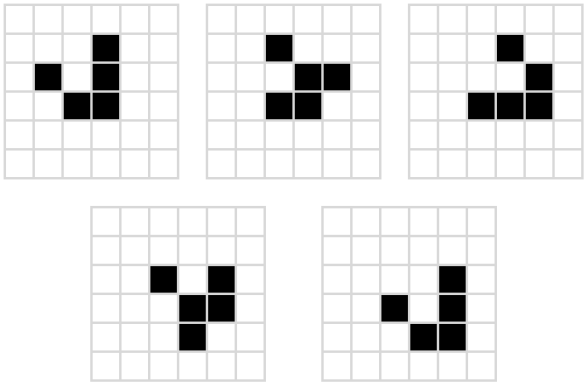
Game of Life vytvořil britský matematik John Horton Conway v roce 1970 [16]. Je založena na celulárním automatu využívající *Moore* okolí. Každá buňka může nabývat jen dvou stavů, může být živá nebo mrtvá. Simulace obvykle začíná náhodnou počáteční konfigurací, kde každý následující krok celulárního automatu je označován jako generace.

Hra obsahuje množinu jednoduchých pravidel.

- Živá buňka, která má méně než dva živé sousedy zemře.
- Živá buňka, která má dva nebo tři živé sousedy zůstává žít.
- Živá buňka s více než třemi živými sousedy zemře.
- Mrtvá buňka oživne právě tehdy, když má okolo sebe tři živé sousedy.

Existuje celá řada celulárních automatů založených na *Game of Life*, které se liší jen rozdílnou lokální funkcí. Předchozí pravidlo by se dalo zapsat jako 23/3. Samostatné první čísla (2 a 3) značí jednotlivý počet sousedů v okolí buňky potřebných pro přežití, druhé číslo za lomítkem značí počet buněk v okolí, aby se buňka narodila.

V *Game of Life* lze u skupiny buněk najít chování, které se dá popsat vzory. Nejtypičtějšími skupinami vzorů jsou: *oscillators*, *still lifes* a *spaceships*. U *spaceships* má smysl hovořit o rychlosti pohybu. Protože buňka je schopná interagovat pouze s nejbližším okolím, není možné posílat informaci na delší vzdálenost než 1 za jeden krok CA. Tato jednotka se označuje *c*.



Obrázek 1.4: *Glider* patřící do skupiny vzorů *spaceships*, zobrazený v jednotlivých krocích CA pohybující se rychlostí $c/4$

1.3 Rekrystalizace

V této části kapitoly bude vysvětlen fyzikální jev zvaný rekrystalizace, se kterým se můžeme setkat u zpracování materiálu. Nejprve budou vysvětleny jednoduché pojmy, abychom mohli správně popsat tento jev. Pro vytvoření následující kapitoly byly využity zdroje [17, 18, 19]. Poslední citovaný zdroj čerpá informace zejména z [20, 21, 22, 23].

1.3.1 Krystalová mřížka

Každá hmota je složena z malých částic zvaných atomy. Tyto atomy se dají fyzikálně popsat a mají své charakteristické vlastnosti. Polohu jednotlivých atomů v krystalu lze popsat pomocí krystalové mřížky. Jedná se o model uspořádání částic v krystalu, kde v ideálním případě jsou částice uspořádané pravidelně. Ve skutečnosti se s takovou strukturou málo kdy setkáme a krystal obsahuje mnoho odchylek od ideálního tvaru. Tyto odchylky se dají dělit do několika kategorií.

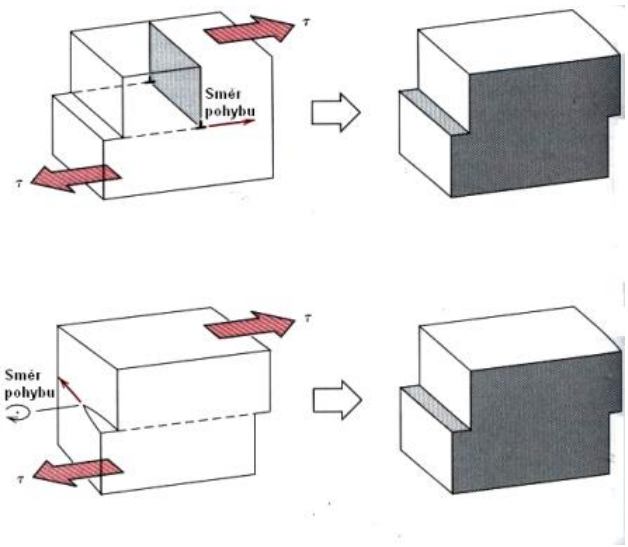
- **Bodové poruchy**

  Na daném místě chybí atom (vakance), atom je nahrazen jiným (substituce) nebo se nachází v blízkosti vložený další cizí atom (intersticiály).

- **Čárové poruchy (dislokace)**

  Čárové poruchy krystalické mřížky vznikají přesunutím (dislokováním) určitého množství atomů při skluzovém pohybu vzhledem k vrstvě sousední. Poruchy se nazývají dislokace. Dislokace jsou nositeli deformace. Dělí se na hranové a šroubové, viz obrázek 1.5. Přítomnost dislokace v krystalu vyvolává pružnou deformaci mřížky. Významnou vlastností

dislokací je schopnost pohybovat se krystalovou mřížkou. Pro pohyb dislokace je nutné dodat zvýšené napětí, aby tuto bariéru překonal.



Obrázek 1.5: Typy dislokací. Na horní části obrázku je ukázka hranové dislokace. Na dolní části se nachází šroubová dislokace [24]

- **Plošné a objemové poruchy**

  U plošných je zajímavá porucha zvaná „hranice zrn“, které vznikají při růstu krystalu mezi jednotlivými zrny. Tyto hranice mezi zrny kov zpevňují.

1.3.2 Hranice zrna

Je to místo, které odděluje různě orientované krystaly. Rozděluje se na:

- maloúhlová, která má rozdíl orientací mezi 10° až 12°, jedná se o subzrna nebo buňky
- velkoúhlová, která má rozdíl orientací větší než 12°, nachází se mezi zrny.

1.3.3 Plastická deformace a deformační zpevnění

Působením dostatečně velkého zatížení mění materiál svůj tvar a rozměry, je deformovaný. Společnými vnějšími činiteli, které modifikují proces deformace, jsou teplota a rychlost deformace. Mezi základní mechanizmy, kterými se realizuje deformace kovů a slitin, patří deformace skluzem, tj. pohybem dislokací ve skluzových rovinách a deformace dvojčatěním. Dvojčatění krystalů se projevuje náhlým přeskupením celého úseku krystalové mřížky.

- **Napětí**

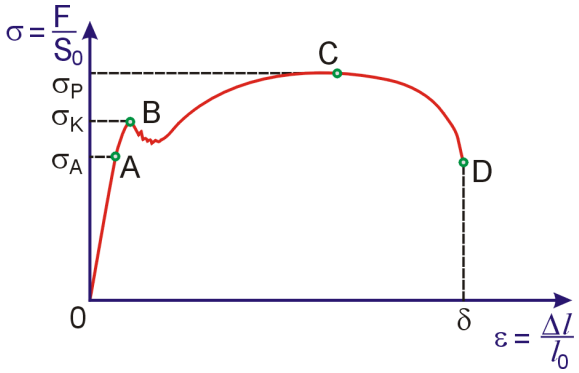
  Působení vnějších sil vyvolá v tělese napětí. To udává množství síly na jednotku plochy. Platí $\sigma = F/S$, kde F je síla deformující těleso, S průřez tělesa kolmý na působící sílu a σ je napětí.

- **Deformace**

  Napětí vyvolané v tělese vnějšími silami způsobí jeho deformaci. Během deformace dochází ke změně tvaru tělesa. Platí $\varepsilon = \Delta l/l_0$, kde Δl je rozdíl v délce materiálu před deformací a aktuální délkou, l_0 je původní délka materiálu, ε je deformace.

- **Plastická deformace**

  Pokud těleso zatížíme nad mez pružnosti, přestává platit přímá úměrnost mezi zatížením a deformací. Závislost má složitější tvar, který je ovlivněn především teplotou a rychlostí zatěžování. Dochází zde k nevratné změně tvaru tělesa. Takovouto deformaci nazýváme plastickou (trvalou) deformací [25].



Obrázek 1.6: Křivka plastické deformace. Bod A značí mez úměrnosti, do kterého platí Hookův zákon, neboli deformace je přímo úměrná napětí. Bod B je mez pružnosti, od kterého v materiálu nastává nevratná deformace. Materiál začíná plasticky téct. Bod C odpovídá maximálnímu možnému zatížení. V bodu D dochází k přetržení materiálu [26]

1.3.4 Tváření materiálu

Během tváření materiálu dochází ke změně tvaru výrobku. U materiálu je překročena mez pružnosti, ale není překročena mez pevnosti. Materiál stále drží pospolu.

1.3.5 Nukleace a růst zrn

Zárodky nových zrn (nukleace) vznikají přednostně v místech s nejvyšší hustotou uložené deformační energie, na hranicích zrn. Nová zrna rostou přemísťováním velkoúhlových hranic, až nová zrna nahradí původní deformovanou strukturu.

1.3.6 Rekrystalizace

Deformace materiálu ve formě dislokací vede ke vzniku termodynamické nerovnováhy deformovaného materiálu. Při nízké teplotě se tento stav může udržet, při zvýšené teplotě nastává proces přechodu do původního rovnovážného termodynamického stavu – vlastnosti se vracejí na původní hodnoty. Tento proces je zvaný rekrystalizace. Hnací silou pro pohyb hranice je uložená energie.

1.3.7 Statická rekrystalizace

Statická rekrystalizace probíhá až po tváření materiálu. Je to proces, během kterého se deformovaný materiál změní v materiál bez deformací. Probíhá zde nukleace, kde jednotlivé nově vytvořené zrna rostou a nahrazují původní deformovaný materiál.

- *Site-saturated nucleation* - všechny nukleace proběhnou zároveň v jednom čase.
- *Continuous nucleation* - nukleace probíhají postupně v čase.

1.3.8 Dynamická rekrystalizace

Nastává během tváření materiálu za zvýšené teploty. Pro zahájení procesu je potřeba větší deformace než u statické (za stejných teplotních podmínek). Pro dynamickou rekrystalizaci je nutná kritická deformace dosažená před mezí pevnosti. Udává se, že nastane přibližně za teploty $T_r = 0{,}7 * T_{tání}[K]$ a vyšší (záleží na materiálu) [18].

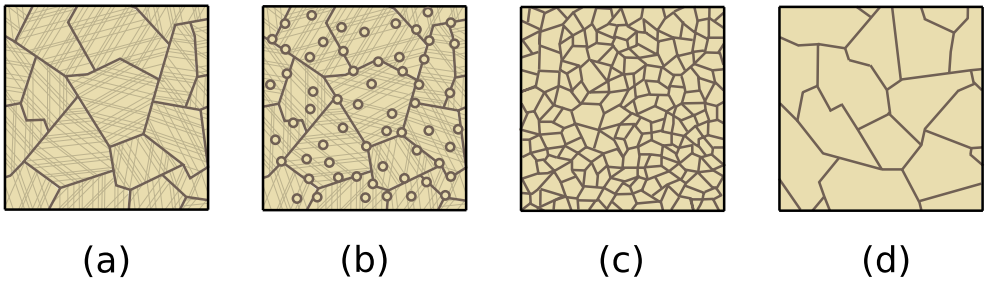
Nové zárodky během dynamické rekrystalizace se tvoří na hranicích původních zrn. Další zárodky pak vznikají na hranicích těchto nových zrn. Tyto zrna poté dále rostou. Jedná se o spojitý proces deformace a nukleace nových zrn. Tato zrna se následně dále deformují. Díky tomu se odstraní zpevnění, a tím pádem dojde k dosažení lepších plastických vlastností.

Podle rychlosti deformace a velikosti teploty lze sledovat dva scénáře.

- **Jeden vrchol *(single peak)* v křivce mechanického napětí.** Dynamická rekrystalizace bude probíhat za nízké teploty s vysokou deformační rychlostí. Rychlost deformace je konstantní. Mikrostruktura bude obsahovat jemná zrna, materiál bude pevnější.

- **Vícenásobný vrchol *(multiple peak)* v křivce mechanického napětí.** Dynamická rekrystalizace bude probíhat za vysoké teploty a nízké deformační rychlosti, která bude v rámci celého experimentu konstantní. Zrna budou naopak hrubší a materiál bude méně pevnější.

Podle [27] bylo zjištěno, že závislost počtu vrcholů v křivce mechanického napětí závisí na původní velikosti zrna. Pro jeden vrchol původní velikost zrna D_0 oproti velikosti zrna v rovnovážně ustáleném stavu D_{SS} je rovna $D_0/D_{SS} > 2$. Pro vícenásobný vrchol platí $D_0/D_{SS} \leq 2$.



Obrázek 1.7: Rekrystalizace materiálu. Na (a) se nachází původní mikrostruktura. Na obrázku (b) byly vytvořeny nukleační zárodky. Na obrázku (c) jde vidět růst zrn nukleačních zárodků. Obrázek (d) obsahuje plně rekrystalizovanou mikrostrukturu [28]

Kapitola 2

Simulace dynamické rekrystalizace pomocí celulárního automatu

V této kapitole bude podrobně popsán algoritmus, který byl navržen v disertační práci [14] a publikován [12]. Autorem je Jiří Kroc, který je taktéž vedoucím této práce. Doplňující informace o tom, jak lze simulovat dynamickou rekrystalizaci pomocí celulárního automatu a ze které moje práce čerpá je následující vědecká práce [2].

2.1 Úvod

Algoritmus byl navržen v roce 1998. V té době bylo nejlepší možností, jak tento jev simulovat za pomocí *Monte Carlo* metody. Bohužel tato metoda nám neposkytuje pro tento problém vhodný aparát [2, 29, 30]. „*Nedává nám odpověď na otázku, proč pro materiál, který má v křivce plastické deformace vícenásobný vrchol maximálního mechanického napětí oproti křivce pro jediný vrchol maximálního mechanického napětí, posunuté toto maximum směrem nahoru*“[12]. Dále nevyřešil v testech zaměřené na změnu rychlosti deformace vliv nukleace, počáteční velikosti zrna a oscilace střední velikosti zrna. Tyto popsané problémy se dají vyřešit využitím celulárního automatu. Na základě publikovaných prací [12, 31] nám dokáže tato metoda na tyto otázky odpovědět.

V současné době se výzkumem v tomto oboru zabývá celá řada vědců např. Huang Shi-quan a kolektiv[32], kteří vymysleli algoritmus pro simulaci 23Co13Ni11Cr3Mo oceli. Dalším příkladem, kteří i citovali práci mého vedoucího jsou Kugler a Turk [33].

2.2 Modifikace celulárního automatu

Pro simulaci dynamické rekrystalizace se využívá dvoudimenzionálního celulárního automatu. Periodické okrajové podmínky jsou zajištěny toroidickým upravením mřížky. Sousedství je využito *Moore* s poloměrem 1, takže se jedná o 8 okolních buněk. Každá buňka má vlastní proměnné. Tyto proměnné pro každou buňku jsou aktualizované paralelně.

2.2.1 Proměnné

Každá buňka má následující trojici proměnných:

- **orientaci zrna** θ *(angl. orientation)* představuje orientaci zrna v materiálu. V simulaci jsou buňky se stejnou orientací obarveny stejnou barvou. Počet možných orientací zrn je 18.
- **dislokační hustotu** ρ *(angl. dislocation density)* je množství dislokací v jednotkovém objemu krystalického materiálu.
- **čekací doba** t_w *(angl. waiting time)* zajišťuje, aby materiál nerekrystalizoval ihned. Tímto se řídí rychlost pohybu hranice. Čím vyšší hodnota, tím déle trvá, než pro buňku může začít rekrystalizace.

2.3 Příprava mikrostruktury pomocí pseudo-statické rekrystalizace

Abychom mohli dynamickou rekrystalizaci simulovat, je nezbytné mít vytvořenou mikrostrukturu. Tato mikrostruktura je vytvářena pomocí algoritmu, který je blízký statické rekrystalizaci (SRX).

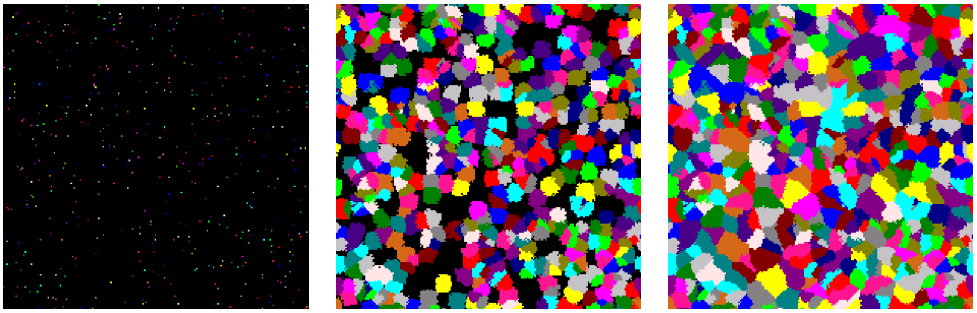
V prvním kroku jsou vygenerovány na náhodných místech mřížky nukleační zárodky. Tento počet se odvíjí podle parametru `embryos count` zadaný v GUI [1]. Jednotlivé nukleační zárodky budou mít přiděleny následující hodnoty proměnných: orientaci θ se přiřadí náhodná hodnota, dislokační hustota ρ se nastaví na 0 a čekací doba t_w se nastaví na maximální možnou (parametr v GUI `max waiting time`).

V dalších krocích se každá daná volná buňka C (nemající orientaci) bude dotazovat všech sousedů v *Moore* okolí [2], jestli náhodou okolní buňky nemají

[1] Ve skutečnosti jejich počet bude o něco nižší. Je to z toho důvodu, že výběr pozice v mřížce je náhodný a pokud již na tomto místě nukleační zárodek je, tak ho přeskočíme. Možností je to dělat tak dlouho, dokud nenajdeme náhodně volné místo. Což by ale při vyšších hodnotách hledalo delší dobu. To, že tento počet nebude úplně přesný, nemá pro simulaci zásadní vliv, proto to můžeme zanedbat.

[2] Toto chování se značí jako *centripetal behaviour*, kdy se snažíme editovat buňku, ve které se momentálně nacházíme. Oproti *centrifugal behaviour*, kde by se jinak editovala okolní buňka, což by v případě paralelismu zbytečně zvyšovalo náročnost implementace.

orientaci. Pokud sousední buňka i orientaci již má, tak buňka C s pravděpodobností $p = 0,5$ převezme hodnoty všech proměnných (ρ, θ, t_w) z buňky i. Tento algoritmus probíhá tak dlouho, dokud se v mikrostruktuře nachází alespoň jedna neorientovaná buňka. Poté je mikrostruktura připravena na simulaci dynamické rekrystalizace.



(a) produkt *site-saturated* nukleace

(b) nedokončená pseudostatická rekrystalizace

(c) konečná mikrostruktura

Obrázek 2.1: Příprava inicializační mikrostruktury

2.4 Algoritmus dynamické rekrystalizace

Algoritmus lze rozdělit na tři části. První část je situace, kdy nastane dynamická rekrystalizace. Druhá část, když dynamická rekrystalizace nenastane. A poslední část, nukleaci. Pro lepší přehlednost je průběh algoritmu znázorněn na vývojovém diagramu 2.3. Dalším diagramem je popsaný jeden krok CA během simulace dynamické rekrystalizace (DRX), viz obrázek 2.2.

Pro provedení dynamické rekrystalizace nad danou buňkou C je zapotřebí splnění všech následujících kroků současně:

- daná buňka C se nachází na hranici zrn[3] *(angl. grain boundary)*
- rozdíl dislokačních hustot buňky C a buňky i náležícímu jinému zrnu je větší nebo rovna kritické hodnotě ρ_{cr}. Neboli platí rovnice $\rho_C - \rho_i \geq \rho_{cr}$
- při nové konfiguraci nevznikne nevhodný tvar [4]
- čekací čas t_w okolní buňky i je roven 0
- zrno s nižší dislokační hustotou vroste do zrna s vyšší dislokační hustotou.

[3]Hranice zrn je takové místo, kde buňka jednoho zrna (jedné barvy) sousedí přímo s buňkou jiného zrna (jiné barvy).

[4]Nevhodný tvar je takový, který vyústí v příliš úzké zrno, nebo část zrna by byla příliš úzká.

1. Pokud jsou všechny předchozí kroky splněné, tak rekrystalizace pro buňku C nastane s pravděpodobností $p_{re} = 0,5$. Tímto zajistíme globulární tvar zrn.

   V tomto případě naše daná buňka převezme orientaci θ z okolního zrna, dislokační hustota ρ se nastaví na 0 a čekací doba t_w se nastaví na maximální hodnotu. Tím algoritmus pro danou buňku končí a může začít pro další buňku od začátku.

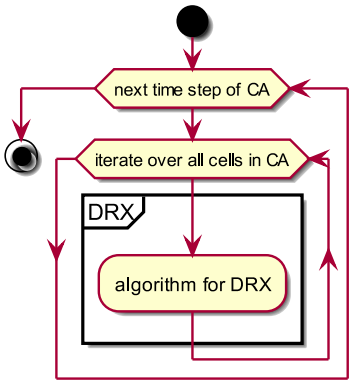
2. V opačném případě, že rekrystalizace nenastane, tak snížíme pro danou buňku čekací dobu t_w o 1 a zvýšíme dislokační hustotu pomocí následující funkce $\rho_C(t) = f(\rho_C(t-1))$.

3. Dále může nastat nukleace, pro kterou musí být splněny následující podmínky:

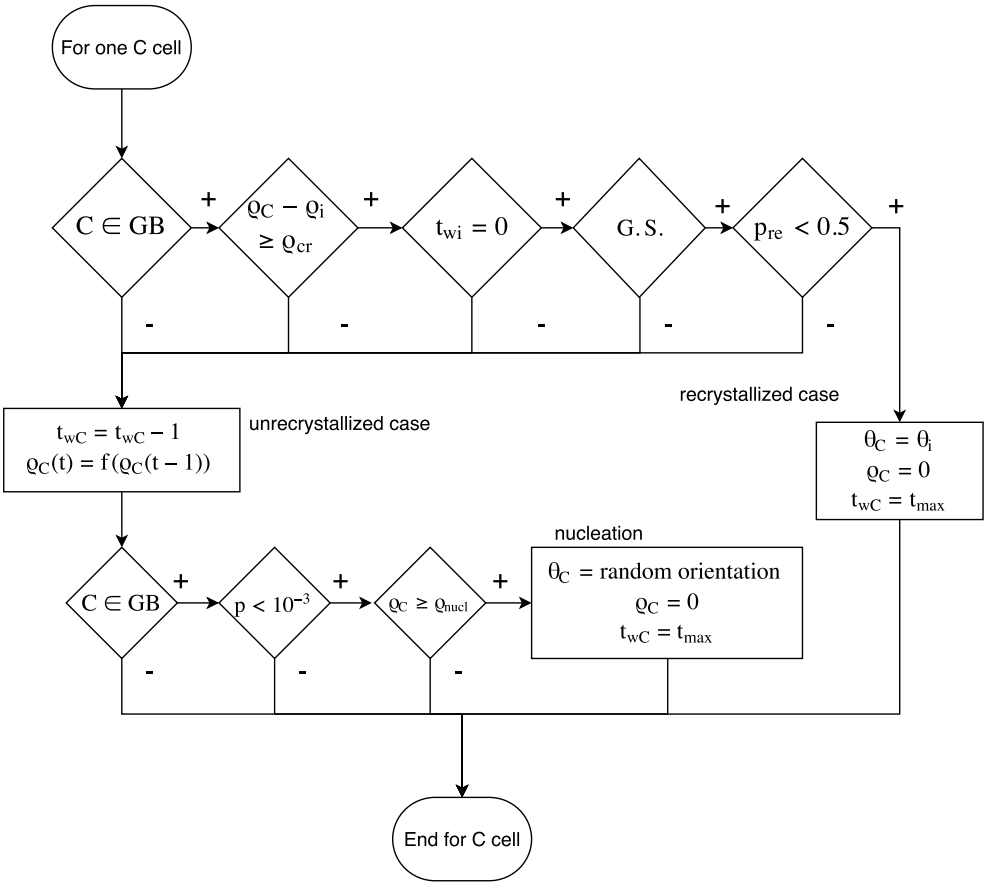
   - daná buňka C se musí nacházet na hranici zrna
   - dislokační hustota dané buňky ρ_C musí být větší než kritická hodnota nukleace. Musí platit následující rovnice $\rho_C \geq \rho_{nucl}$.

   Pokud jsou předchozí body splněny, tak nukleace proběhne s pravděpodobností $p = 0{,}001$. Tuto pravděpodobnost lze upravit v aplikaci. V tomto případě se zrodí nový nukleační zárodek, který dostane náhodnou orientaci θ (z možných orientací), dislokační hustota ρ se nastaví na 0 a čekací doba t_w se nastaví na maximální možnou.

V opačném případě se nic neděje a můžeme pokračovat další buňkou. Algoritmus projde tímto způsobem všechny buňky v mřížce a poté, až doběhne pro poslední buňku je dokončen běh jednoho kroku CA a zobrazí se nám aktualizovaná mikrostruktura.



Obrázek 2.2: Diagram popisující hlavní smyčku simulace. Algoritmus pro dynamickou rekrystalizaci je popsaný na obrázku 2.3



Obrázek 2.3: Vývojový diagram dynamické rekrystalizace na základě [14]. Tento diagram znázorňuje průchod pro jednu buňku v algoritmu. C značí naší buňku, i značí buňku z okolí, $\exists i \in \mathcal{N}(C)$. Proměnné p, p_{re} obsahují náhodnou hodnotu z intervalu $< 0, 1)$. GB je hranice zrn, G.S. značí *Good Shape*

Kapitola 3

Analýza

Analýza algoritmu byla podrobně popsána v předchozí kapitole. V této kapitole bude podrobněji rozebráno stávající řešení a nároky na aplikaci.

3.1 Analýza stávajícího řešení

Stávající řešení je ve formě několika vědeckých programů, kterými je autorem vedoucí práce Jiří Kroc. Tyto programy jsou napsané v jazyce **CellLang**[5] a nemají žádné uživatelské rozhraní. Navíc každá změna simulace musí být provedena přímo v kódu. Ze všech dostupných řešení byl zaslán jeden kód, který nejrozsáhleji popisoval problematiku. Kód obsahuje mnoho optimalizací a triků, které velmi snižují jeho čitelnost. Vývojový diagram představený v předchozí kapitole se v některých částech liší oproti zaslanému kódu.

Proto bylo nutné zvolit alternativní cestu, při které jsem přečetl vybrané kapitoly disertační práce [14] a program se snažil napsat primárně vůči této práci s přihlédnutím ke kódu. To se ukázalo jako mnohem lepší řešení, protože jsem více pochopil souvislosti nabyté v ostatních vědeckých článcích.

3.2 Analýza požadavků

Při domluvě se zákazníkem je potřeba jasně porozumět jeho nárokům na systém, abychom na základě nich mohli udělat návrh, který by splňoval jeho potřeby. K tomuto slouží analýza požadavků, ve které je jasně definované, co systém musí umět. Tyto požadavky se dají rozdělit do dvou hlavních skupin. *Funkční požadavky* popisují funkcionalitu systému a *Nefunkční požadavky*, které kladou omezení na provedení (například na kvalitu, výkonnost). Byla při-

[5] CellLang je jazyk především určený pro simulaci buněk. Engine informace zpracovává paralelně. V současnosti nemá téměř žádné zastoupení. Například vyhledávač neumí požadavek ani indexovat.

dána ještě třetí část *Požadavky na GUI*, kde jsou shrnuté požadavky spojené s uživatelským grafickým rozhraním.

3.2.1 Funkční požadavky

- Aplikace bude zobrazovat mikrostrukturu tvořenou pomocí CA.
- Výsledky simulace bude program zaznamenávat do grafů.
- Uživatelé budou mít možnost definovat parametry simulace.
- Aplikace bude schopna načítat vlastní inicializační mikrostrukturu z konfiguračního souboru.
- Aplikace bude schopna uložit vygenerovanou mikrostrukturu jako konfigurační soubor.
- V aplikaci bude možnost uložit data z grafů ve formě TXT souborů.
- Uživatelé budou moci simulaci zastavit, zrychlit nebo začít novou simulaci.
- Uživatelé budou moci uložit grafy a mikrostrukturu jako obrázek ve formátu PNG.

3.2.2 Nefunkční požadavky

- Aplikace bude možné spustit na operačním systému Windows 7 a novější, a operačním systému rodiny Linux.
- Aplikace bude jednoduše rozšiřitelná a modifikovatelná.
- Třídní návrh a implementace bude navržena tak, aby byla v souladu s fyzikou [6].
- Aplikace bude publikována pod open source licencí a veřejně dostupná na portálech ResearchGate a SourceForge jako příklad komplexních systémů.

3.2.3 Požadavky na GUI

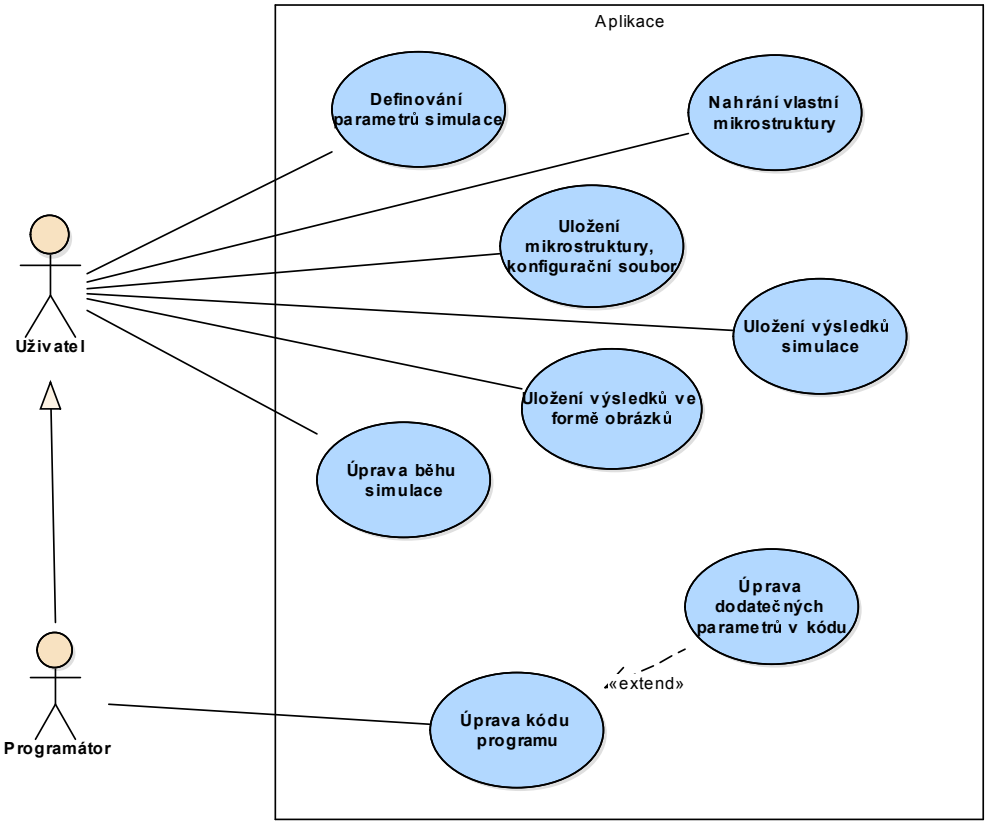
- Konfigurace simulace se dá nastavit v hlavním okně programu.
- Při špatně zadaných parametrech nebo jakékoliv jiné chybě program uživateli zobrazí chybovou hlášku.
- Každé tlačítko a okno zobrazuje vysvětlivky k čemu slouží.

[6] Je to z toho důvodu, že program budou využívat fyzici z celého světa a mnozí se budou dívat i do kódu, který budou případně i rozšiřovat. Proto je nutné mít třídní návrh a kód uzpůsobený této situaci.

3.3 Případy užití

Podkapitola popisuje případy užití, ve kterých je popsána interakce mezi uživatelem programu a samotným programem. Ti, kteří budou s programem pracovat lze rozdělit do dvou rolí. První z nich jsou normální uživatelé, kterým může být kdokoliv, kdo si stáhne spustitelnou verzi programu a bude chtít program používat. Druhou skupinou jsou ti, kteří si stáhnou zdrojové kódy aplikace a budou chtít aplikaci upravovat[7].

Abych neuvedl někoho v omyl, tak slovíčko „mikrostruktura“, které se nachází v nadpisech případů užití je bráno jako konfigurační soubor CA. Je to z toho důvodu, že při vstupu do programu sice konfigurační soubor obsahuje data o CA a všech jeho buňkách, ale ve skutečnosti (fyzice) je to předpřipravená struktura, která je určena k deformaci.



Obrázek 3.1: Diagram případů užití pro skupiny uživatelů, kteří patří buď do role „Uživatel“ nebo do role „Programátor“

[7]Programátor i když nemá aplikaci přímo jako spustitelný soubor, má možnost při psaní kódu aplikaci spustit a provádět úkony jako normální uživatel.

- **Definování parametrů simulace**

  1. Případ užití začíná, když uživatel spustí program nebo program restartoval do počátečního stavu.
  2. Uživatel si vybere, že chce generovat vlastní mikrostrukturu zakliknutím příslušného tlačítka.
  3. Systém přijme požadavek a dá možnost uživateli pokračovat výběrem parametrů.
  4. Uživatel si následně navolí parametry programu.
  5. Systém průběžně vyhodnocuje zadávané parametry a v případě nevhodně zadaných hodnot uživateli nepovolí tuto hodnotu zadat.
  6. Po nastavení parametrů a stisku tlačítka pro start aplikace simulace začne.

- **Nahrání vlastní mikrostruktury**

  1. Případ užití začíná, když uživatel spustí program nebo program restartoval do původního stavu.
  2. Uživatel si vybere, že chce nahrát vlastní mikrostrukturu z konfiguračního souboru (přípona CA) zakliknutím příslušného tlačítka.
  3. Systém přijme požadavek a zobrazí uživateli okno pro výběr souboru.
  4. Uživatel si vybere soubor, ve kterém se nachází konfigurační data.
  5. Systém mikrostrukturu zkontroluje, jestli je ve správném formátu a jestli obsahuje validní data.
     5.1. Pokud uživatel zadal neplatnou mikrostrukturu, systém na skutečnost uživatele upozorní a tok pokračuje od bodu 2.
  6. Uživatel může simulaci spustit stiskem tlačítka pro start aplikace.

- **Uložení mikrostruktury jako konfigurační soubor**

  1. Případ užití začíná, když uživateli běží simulace a chce mikrostrukturu uložit jako konfigurační soubor.
  2. Uživatel si v menu vybere, že chce uložit mikrostrukturu.
  3. Systém zastaví simulaci a nabídne uživateli okno pro uložení souboru s příponou CA.
  4. Uživatel vybere místo a název ukládaného souboru.
  5. Systém konfiguraci celulárního automatu uloží.
  6. Uživatel může buď v simulaci pokračovat dále nebo vytvořit jiný experiment.

- **Uložení výsledků ve formě obrázků**

  1. Případ užití začíná, když uživateli běží simulace a chce mikrostrukturu nebo grafy uložit ve formě obrázků.
  2. Uživatel si v menu vybere, že chce uložit mikrostrukturu nebo grafy ve formě obrázků PNG.
  3. Systém zastaví simulaci a nabídne uživateli okno pro uložení souborů.
  4. Uživatel vybere místo a název souborů.
  5. Systém obrázky uloží na uživatelem vybrané místo. V případě grafů budou obrázky obsahovat postfix podle typu sledovaného grafu.
  6. Uživatel může buď v simulaci pokračovat dále nebo vytvořit jiný experiment.

- **Uložení výsledků simulace ve formě dat**

  1. Případ užití začíná, když uživateli běží simulace a chce data z grafů uložit.
  2. Uživatel si v menu vybere, že chce uložit data z grafů ve formě textového souboru TXT.
  3. Systém zastaví simulaci a nabídne uživateli okno pro uložení souborů.
  4. Uživatel vybere místo a název souborů.
  5. Systém soubory uloží na uživatelem vybrané místo. Textové soubory budou obsahovat postfix podle typu sledovaného grafu.
  6. Uživatel může buď v simulaci pokračovat dále nebo vytvořit jiný experiment.

- **Úprava běhu simulace**

  1. Případ užití začíná, když uživatel má inicializovaný celulární automat buď načtením vlastní mikrostruktury nebo si nechal vygenerovat v programu na základě parametrů.
  2. Uživatel si v GUI změní rychlost simulace, zastaví simulaci, spustí simulaci, resetuje simulaci, nebo upraví parametry.
  3. Systém na základě předchozího požadavku přizpůsobí program.

- **Úprava kódu programu**
  1. Případ užití začíná, když programátor otevře projekt v programu Qt Creator[8].
  2. Programátor má možnost v programu Qt Creator upravovat stávající kód, přidávat vlastní třídy, ***extend*** (Úprava dodatečných parametrů v kódu).
  3. Programátor upraví program a nechá ho přeložit.
  4. Kompilátor program přeloží.
  5. Programátor program spustí, otestuje a změny v programu uloží.
- **Úprava dodatečných parametrů v kódu**
  1. Případ užití začíná ve druhém kroku případu užití „Úprava kódu programu“, pokud programátor chce upravit parametry v souboru.
  2. Programátor mezi zdrojovými soubory najde soubor s názvem `parameters.h` a otevře ho.
  3. Upraví parametry a uloží ho.

[8] `https://www.qt.io/ide/`

Kapitola 4

Návrh

V této kapitole budou popsány technologie, které byly vybrány k implementaci. Na základě předchozích požadavků byl vytvořen návrh uživatelského rozhraní a návrh tříd.

4.1 Použité technologie

4.1.1 Jazyk a framework

Programovací jazyk pro řešení byl vedoucím práce zadán C++ s využitím frameworku Qt [9].

- **C++**

  Volba tohoto programovacího jazyka je jednoduchá. S tímto jazykem se většina vědců již potkala nebo jím aktivně programuje.

- **Qt**

  Dále byl vedoucím vybrán framework Qt. Jedná se o multiplatformní framework pro tvorbu grafických uživatelských rozhraní. Má dobrou dokumentaci. A je kompatibilní s jazykem C++.

4.1.2 Knihovna pro kreslení grafů

Pro vykreslování grafů v aplikaci bylo zapotřebí najít knihovnu, která by šla jednoduše přidat do aplikace a rychle vykreslovala grafy. Vybíral jsem proto mezi programem gnuplot [10] a externí knihovnou QCustomPlot[11].

[9] `https://www.qt.io/`
[10] `http://www.gnuplot.info/`
[11] `http://www.qcustomplot.com/`

- **gnuplot**

  Jedná se o program, který byl původně vytvořen pro vědce a studenty, pro vizualizace matematických funkcí do grafů. Ve vědecké komunitě se těší tento program velké oblibě.

- **QCustomPlot**

  Tato externí knihovna byla vytvořena pro framework Qt. Vytvořil ji Emanuel Eichhammer a je poskytována pod GPLv3 licencí. Knihovna nabízí jednoduché vykreslování grafů a má bohatou dokumentaci.

Při výběru knihovny jsem bral v úvahu, že s programem gnuplot jsem se již setkal v průběhu studia. Další fakt byl ten, že moje aplikace bude nadále rozšiřována ostatními programátory a vědci, kteří často mají s tímto programem také zkušenosti. Bohužel po naprogramování jsem přišel na úskalí gnuplotu, které jsou více popsané v kapitole implementace 5.4.

4.2 Grafické rozhraní

Pro návrh grafického rozhraní byl použit nástroj Wireframe.cc [12]. Při návrhu jsem se snažil všechny důležité ovládací prvky přehledně uspořádat do okna. GUI se dá rozdělit na následující části.

- Pravý panel obsahující vykreslované grafy, které se dají z menu vypnout nebo zapnout.
- Uprostřed je zasazena mikrostruktura, nad kterou se zobrazuje stav simulace.
- Levá část je určena pro konfiguraci programu, která se dále dělí na 4 části.
  - Nahoře jsou zobrazovány informace o běhu programu. Včetně upravování rychlosti běhu.
  - Pod ní se nachází dvojice tlačítek pro výběr nebo generování počáteční mikrostruktury.
  - Následují tři tlačítka, která ovládají běh programu.
  - Poslední část obsahuje konfigurace proměnných simulace.

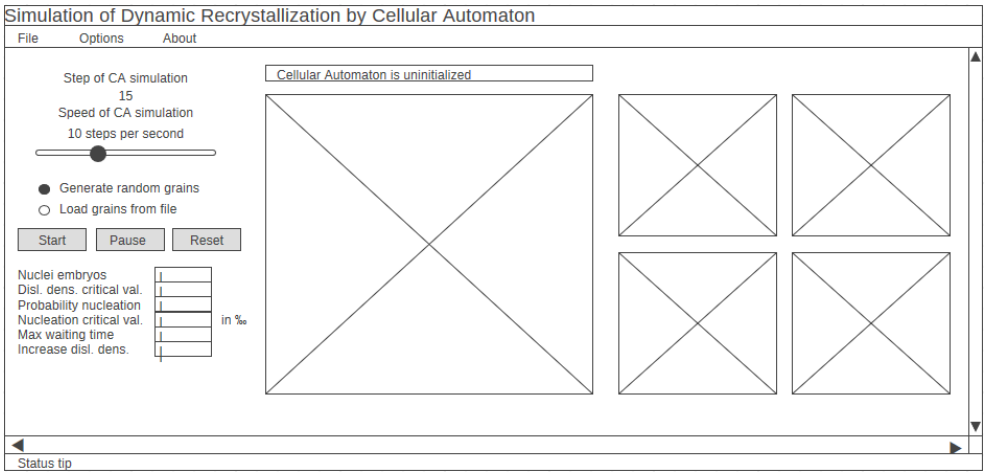
  Při nastavování programu se jednotlivé konfigurační části postupně uživateli zpřístupňují a již vyplněné zamykají[13]. Tím je docíleno, že uživatel

[12] `https://wireframe.cc/`

[13] Nastavování probíhá od vrchu směrem dolů. Mezi konfigurační části jsem ale vložil tlačítka pro ovládání programu, které jsou asi nejpoužívanější. Tím, že jsou ve středu, tak uživatel nemusí myší zajíždět „daleko“ od dění programu a má to i blíž do menu.

intuitivně dokáže v jednotlivých krocích nastavit program, a následně ho i spustit. Pokud uživatel na něco zapomene, tak během pokusu o start programu bude vyzván k nápravě.

- Poslední část je menu, ve kterém se nachází tlačítka pro ukládání a načítání. V záložce **Options** se nachází ovládání zobrazování křivek.



Obrázek 4.1: Návrh uživatelského rozhraní programem Wireframe.cc

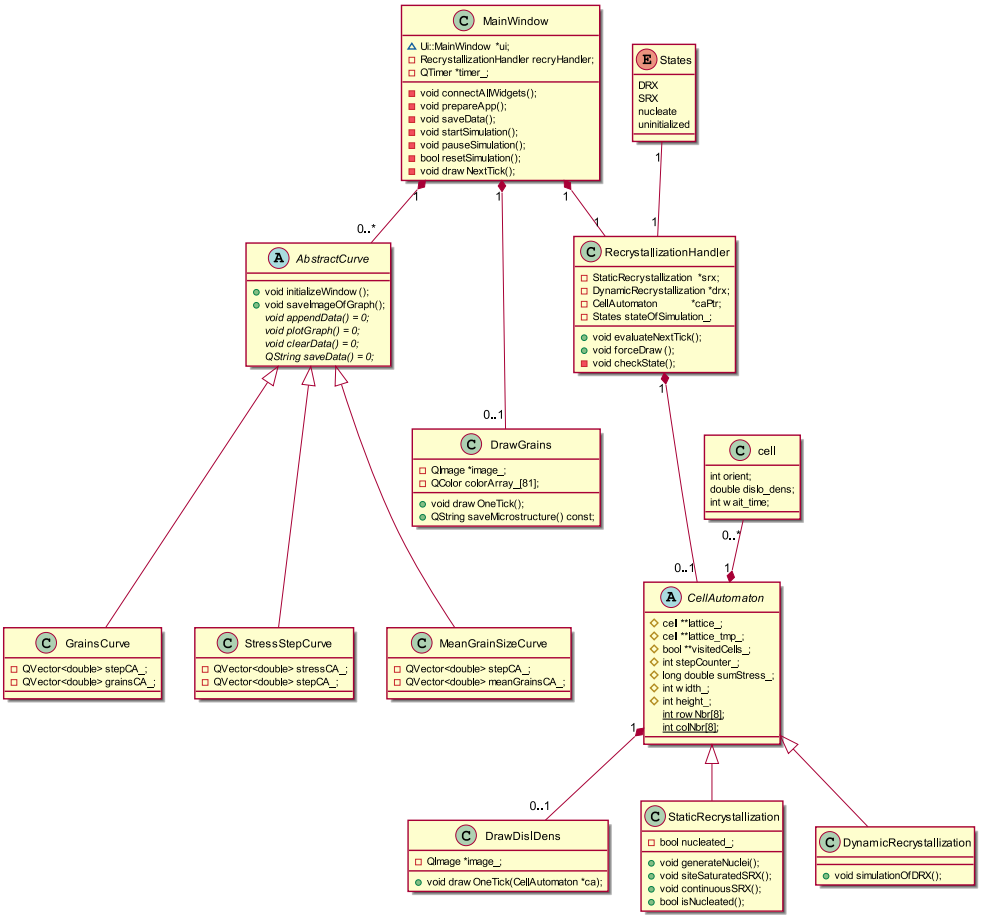
4.3 Model tříd

Diagram tříd byl vytvořen pomocí programu PlantUML [14]. Z důvodu čitelnosti obrázku mají jednotlivé třídy pouze nejdůležitější metody. Diagram tříd obsahující všechny metody lze najít v složce `attachments`. V této složce se nachází i soubory potřebné k vygenerování diagramu.

- `CellAutomaton` Abstraktní třída, která obsahuje celulární automat a metody operující nad tímto CA. Obsahuje hlavně settery a gettery. Simulační metody jsou implementovány v třídách potomků. Navíc obsahuje strukturu `cell`, která představuje jednu buňku v mřížce CA.
  - `StaticRecrystallization` Obsahuje metody pro simulaci statické rekrystalizace, která je použita pro generování inicializační mikrostruktury. Obsahuje navíc metodu i pro *continous nucleation*.
  - `DynamicRecrystallization` Obsahuje metody pro simulaci dynamické rekrystalizace.

[14] `http://www.plantuml.com/`

- `AbstractCurve` Abstraktní třída, která zajišťuje vykreslování grafů. Obsahuje společné metody a zbylé virtuální jsou implementovány v potomcích.
  - `StressStepCurve` Třída, která má za úkol vykreslování křivky mechanického napětí v závislosti na kroku CA.
  - `MeanGrainSizeCurve` Třída, která má za úkol vykreslování střední velikosti zrna v CA.
  - `GrainsCurve` Třída, která má za úkol vykreslování počtu zrn v závislosti na kroku CA.
- `MainWindow` Třída, která reprezentuje hlavní okno aplikace. Tato třída propojuje veškerou komunikaci mezi uživatelem z GUI a samotným programem.
- `RecrystallizationHandler` Třída rozhodující, která funkce v simulaci bude volána nad příslušným objektem. Tato třída je zde kvůli nutnosti rozdělení jedné třídy CA na třídy zabývající se CA, DRX, SRX. Výčtový typ `States` obsahuje jednotlivé stavy, ve kterých se simulace může nacházet. Důvod tohoto rozdělení je rozepsán v kapitole implementace 5.2.
- `DrawDislDens` Třída, která vykresluje dislokační hustotu v CA do okna.
- `DrawGrains` Třída, která vykresluje mikrostrukturu CA, kde různé barvy buněk značí rozdílnou orientaci zrna. Vykreslovaní probíhá do hlavního okna aplikace.



Obrázek 4.2: Návrh modelu tříd

Kapitola 5

Implementace

V této kapitole bude popsána implementace, která vychází ze znalostí, které jsou popsány v předchozích kapitolách.

5.1 Parametrizace a soubor `parameters.h`

V GUI je možné parametrizovat celkem 6 nejčastějších parametrů (počet zrn, kritická hodnota dislokační hustoty, pravděpodobnost nukleace, nukleační kritická hodnota, maximální čekací doba buněk a hodnota o kolik se zvýší dislokační hustota každý krok simulace). Z toho první 4 lze nastavit při startu simulace a zbylé dva lze při pozastavení simulace měnit. Tyto parametry jsou ošetřené proti nesprávným vstupům. To znamená, že lze vložit pouze číslo z validního rozsahu. Tento rozsah je definovaný v souboru `parameters.h`.

Soubor `parameters.h` je rozdělený na 3 části. První část jsou parametry, které jdou změnit. Tyto parametry typicky definují velikosti oken výstupu nebo počáteční parametry. Následuje druhá část parametrů, kde v případě změny může dojít k nereálným výsledkům simulace. Tyto parametry je doporučované měnit jen pro fyziky. Třetí část parametrů obsahuje mezní hodnoty užívané v programech a konstanty pro běh programu. Tyto hodnoty lze měnit jen v případě, když se upraví i program.

Na následujících stránkách se často budu odvolávat na konfigurační konstanty, které pokud není řečeno jinak se nacházejí v souboru `parameters.h`.

5.2 Třídy v souladu s fyzikou

Je potřeba zmínit, že program je napsaný tak, aby vyhovoval potřebám fyziků. A to nejenom z pohledu uživatelského rozhraní, ale i z pohledu rozdělení tříd. Můj původní návrh počítal s třídou `CellAutomaton`, ve které se bude nacházet CA a metody pro simulování statické a dynamické rekrystalizace. Tento návrh je sice jednoduchý, ale z pohledu fyziky není správný. Fyzikální jevy

statické a dynamické rekrystalizace jsou oddělené. Proto, aby práce byla pochopitelná pro fyziky muselo dojít k rozdělení těchto tříd. Došlo k rozdělení třídy `CellAutomaton` na třídy `CellAutomaton`, `StaticRecrystallization` a `DynamicRecrystallization`. Navíc díky tomu jsem už nemohl využívat jednoduše ukazatele z `namespace::Ui`.

Situaci jsem vyřešil tak, že jsem vytvořil další třídu s názvem `RecrystallizationHandler`, která řeší situace kolem volání správné metody nad daným objektem. V této třídě se nachází výčtový typ *(enum)* `States`, který určuje stav CA. Tento výčtový typ obsahuje: `nucleate` ,`uninitialized`, `DRX`, `SRX`. Tento stav je uložený v proměnné `stateOfSimulation_`. Na jejímž základě víme, ve kterém stavu se CA nachází a můžeme podle toho volat příslušné metody nad správnými objekty.

Na přiloženém kódu můžeme vidět funkci, která řeší právě volání jednotlivých metod na základě toho, v jakém stavu se CA nachází [15].

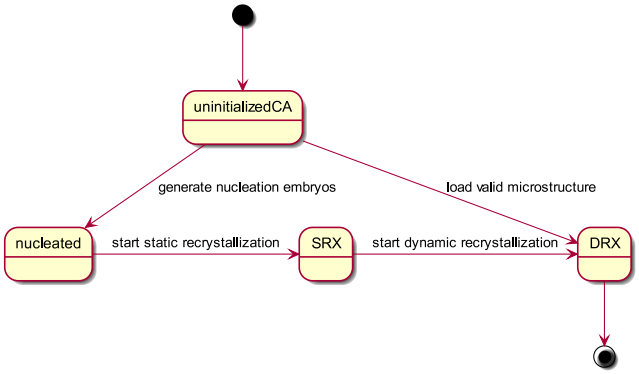
```
checkState(progress);

switch (stateOfSimulation_) {
        case nucleate:
                srx->generateNuclei();
                break;
        case SRX:
                srx->siteSaturatedSRX();
                //srx->continuousSRX();
                break;
        case DRX:
                drx->simulationOfDRX();
                break;
}
caPtr->countNumberOfGrains();

drawStructure->drawOneTick(caPtr);
```

Listing 5.1: Volání metod během simulace

Navíc má tento přepínač *(switch)* názvy, které jsou lépe na první pohled pochopitelné. V metodě `checkState()` probíhá kontrola CA a v případě potřeby změní svůj stav z `SRX` do `DRX`, případně do jiných stavů.

Je vidět, že se v aplikaci nachází vedle objektů `drx` a `srx` ještě `caPtr`. Tento ukazatel neustále ukazuje na správný CA, aby na ostatních místech nemuselo být podobné větvení programu. Je to daň za to, že v aplikaci se musí nacházet objekty rozdělené, ale zároveň bez zbytečného větvení.

[15] Stav `uninitialized`, se v metodě nenachází z důvodu, že v metodě `checkState(progress)` probíhá kontrola stavu simulace. A v případě, že CA je neinicializovaný (a simulace je zapnutá), tak se stav simulace nastaví rovnou na `nucleate`.



Obrázek 5.1: Stavový diagram popisující jeden běh simulace CA

5.3 Řešení simulace

Po spuštění programu máme možnost načíst si vlastní mikrostrukturu nebo si nechat vygenerovat novou v programu na základě námi definovaných parametrů. V případě generování vlastní mikrostruktury zaškrtneme tlačítko **Generate random grains**. Dalším krokem bude definování parametrů pro simulaci. Až nastavíme parametry pro běh a spustíme simulaci, tak dojde k vygenerování nukleačních zárodků. Poté bude probíhat pseudo-statická rekrystalizace, než budeme programem upozorněni o dokončení tvorby mikrostruktury. Pro další pokračování bude potřeba stisknout tlačítko **continue**. Dojde také k resetování počítadla kroků CA, protože tento krok nás zajímá vzhledem k dynamické rekrystalizaci. Během běhu programu můžeme simulaci pozastavit a změnit parametry pro čekací dobu nebo postupného zvyšování dislokační hustoty. Pokud chceme načíst vlastní mikrostrukturu, tak vybereme tlačítko **Load microstructure from file** a v případě validního obsahu budeme moci pokračovat v simulaci.

Data celulárního automatu se budou nacházet v třídě `CellAutomaton` v 2D poli s názvem `lattice_`. Existuje ještě jedno pole s názvem `lattice_tmp_`, které slouží jako dočasné pole pro ukládání mezivýsledků. Tato pole jsou tvořena strukturami `cell`, která představuje jednu buňku. Jsou dynamicky alokovaná a jejich velikost záleží na konstantách `WIDTH,HEIGH`.

Řízení vykreslování jednotlivých kroků CA je řešeno přes `timer` z Qt třídy. Po spuštění simulace se spustí i tento `timer`. Uživatel si může pomocí jezdce v GUI řídit rychlost simulace. Tento `timer` periodicky posílá požadavky pro výpočet dalšího kroku simulace, který je pak uživateli vykreslen.

5.4 Grafy

Pro zobrazování výsledků simulace je využito přehledné vynesení hodnot do grafů. Původní plán, který jsem si na začátku analýzy dal, byl grafy vykreslovat v programu gnuplot. Během implementace jsem zjistil nevýhody tohoto řešení.

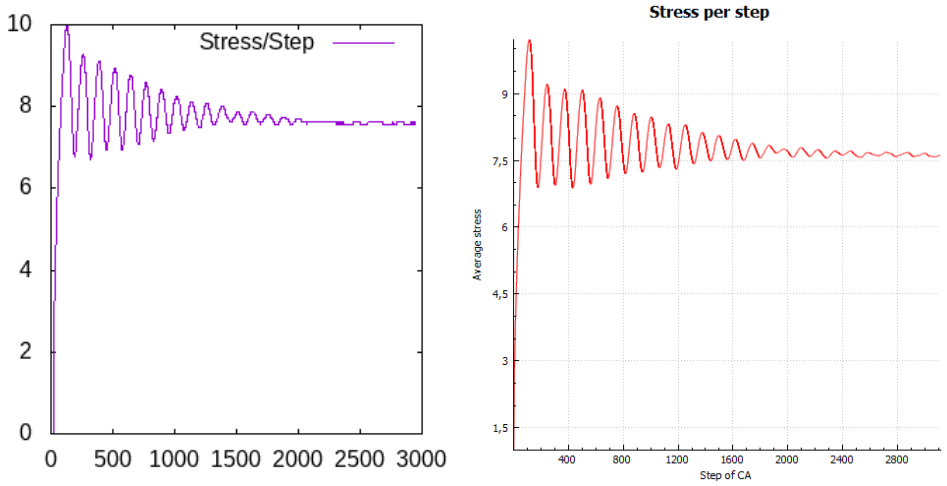
(a) Vykreslování nebylo dostatečně rychlé [16].

(b) Nutnost vyžadovat instalaci tohoto programu a přidání cesty do proměnné `PATH`.

(c) Během simulace byl neustále zatížen pevný disk, jak se ukládaly data, ze kterých se vykreslil soubor, který byl následně zobrazen v aplikaci.

(d) V případě ustálení křivky docházelo k splývání čáry a křivka byla hrbolatá, viz obrázek 5.2.

Na základě těchto důvodů jsem se rozhodl, že nakonec třídy přepíši a místo programu gnuplot využiji externí knihovnu QCustomPlot.



(a) gnuplot graf (b) QCustomPlot graf

Obrázek 5.2: Porovnání výsledků z grafu vytvořený v programu gnuplot a knihovny QCustomPlot. Rozměry grafu byly stejné

[16] Testování rychlosti probíhalo pro stejné parametry na velké mřížce mezi kroky CA 400 až 500. QCustomPlot těchto 100 kroků zvládl za 21 s, kdežto gnuplotu to trvalo 46 s.

5.5 Vizualizace výsledků

Zobrazování výsledků simulace se dělí na tři části.

5.5.1 Vizualizace mikrostruktury

První z nich je hlavní okno `DrawGrains`, které zobrazuje mikrostrukturu. Tato mikrostruktura zobrazuje jednotlivé buňky a jejich orientaci, na základě které je buňka obarvena příslušnou barvou. Počet možných orientací v simulaci je 18. Je zde předpřipravena možnost simulace s až 80 možnými orientacemi. Stačí jen přiřadit konstantě `MAX_ORIENT` až číslo 80. Barvy jsou uložené v poli `colorArray_`, jehož prvky jsou typu `QColor(R,G,B)`, kde proměnné `R,G,B` jsou hodnoty barvy ve formátu barevného modelu RGB z rozsahu (0-255). Velikost vizualizačního okna je definovaná konstantou `DRAW_SIZE`. Proměnná `pixelSize_` se na základě velikosti vizualizačního okna a menší z dvojice šířky nebo výšky mřížky CA vypočte, na kolik pixelů se zobrazí jedna buňka.

5.5.2 Vizualizace grafů

Zobrazování grafů je zajištěno pomocí externí knihovny QCustomPlot. Jednotlivé grafy mají svoji vlastní třídu, která má společného rodiče, třídu `AbstractCurve`. V této třídě dojde k základnímu nastavení okna pro vykreslovaní. Tyto grafy se dají vypnout nebo zapnout v menu. V aplikaci se nacházejí tři grafy a jeden, který je v kódu předpřipravený pro budoucí rozšíření.

- ***Grains Curve***

  Pro výpočet celkového počtu zrn v CA bylo použito algoritmu *Depth-first search*, který byl upraven pro počítání počtu zrn na toroidicky zakřivené mřížce.

```
for (int k = 0; k < 8; ++k){
  DFS(((i + rowNbr[k]) % height_ + height_ ) % height_ ,
      ((j + colNbr[k]) % width_  + width_  ) % width_);
}
```

Listing 5.2: Ukázka volání algoritmu Depth-first search pro všechny sousedy dané buňky na toroidicky zakřiveném 2D poli

- ***Mean Grain Size Curve***

  Průměrná velikost zrna D se vypočítá podle vzorce $S = \pi \cdot (D/2)^2$, kde $S = (m \cdot n)\,/numberOfGrains$, kde je m je šířka mřížky CA a n je její výška.

- ***Stress Step Curve***

  Je křivka zobrazující mechanické napětí v závislosti na čase. Výpočet podle [14] probíhá následovně.

1. Nejprve pro všechny buňky spočítáme průměrnou dislokační hustotu
$$\bar{\rho} = \frac{\left(\sum_{i,j=0}^{m,n} \rho_{i,j}\right)}{m \cdot n}.$$
2. Mechanické napětí σ je přibližně rovno odmocnině $\bar{\rho}$
$$\sigma \propto (\bar{\rho})^{1/2}.$$

Výsledkem je křivka, která vyjadřuje průměrné mechanické napětí na jednu buňku v materiálu.

- ***Flow Stress Curve***

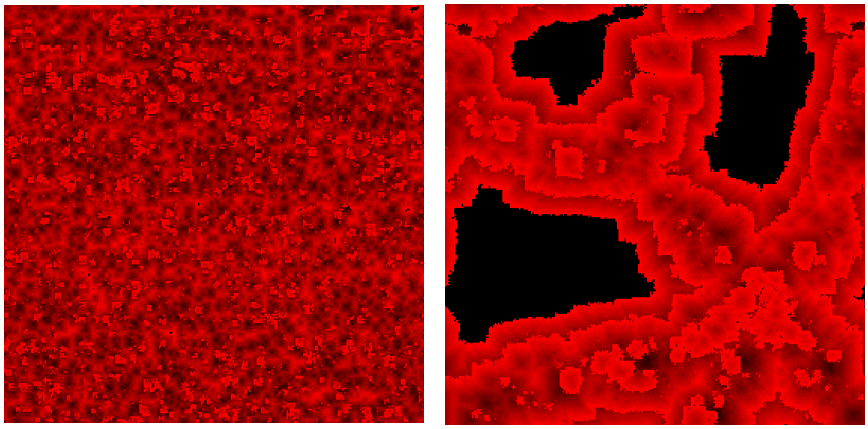
  Tato část je nad rámec zadání, proto zobrazování této křivky nese sebou menší omezení. Pro výpočet by bylo nutné přidat další proměnnou do buňky nebo využít proměnnou pro dislokační hustotu ρ, která již slouží pro výpočet křivky mechanického napětí. Využil jsem druhou možnost, protože většinou uživatel bude chtít vidět křivku mechanického napětí a tato křivka slouží jen jako ukázka, že lze provést výpočet i této křivky.

  Vývoj deformace lokální dislokační hustoty ρ se dá podle [14] definovat jako
$$\frac{d\rho}{d\varepsilon} = A\rho^{1/2} - B\rho,$$
  kde A a B jsou konstanty a ε je deformace. Konstanta A představuje únavu vzniklou deformací a B představuje obnovu dislokací. Tyto konstanty jsou v programu pod jmény `STRAIN_HARDENING` a `RECOVERY_DISLOCATIONS`. Vyjádření je numericky pomocí Eulerovy metody pro obyčejné diferenciální rovnice. Kde po úpravách se dostaneme do následujícího tvaru
$$\rho_{i+1} = \rho_i + h(A\rho^{1/2} - B\rho),$$
  kde h je krok intervalu o velikosti 0,05.

  Pro vizualizaci této křivky místo křivky mechanického napětí je nutné upravit kód a v místě volání funkce `increaseDislocationDens()` volat funkci `calculateDislocationDensity()`. Tato část s konstantami není úplně vyladěna a je laskavě přenechána dalším vědcům.

5.5.3 Vizualizace dislokační hustoty

Poslední okno zobrazuje dislokační hustotu. Čím vyšší je tato hodnota, tak tím je tmavší. Naopak nízké hodnoty mají barvu sytě červenou. Toto okno se zobrazuje v pravé části GUI vedle ostatních grafů.



Obrázek 5.3: Obrázky dislokační hustoty

5.6 *Centripetal* styl

V programu je využit tzv. *centripetal* styl. To znamená, že v případě buněk edituji jen tu buňku, ve které se momentálně nacházím a okolní neměním. Je díky tomu snadnější paralelizace. Rozdíl mezi *centripetal* a *centrifugal* stylem lze najít v třídě `StaticRecrystallization` a metodou `neumannSRX()` a `siteSaturatedSRX()`. V `neumannSRX()` je použit *centrifugal* styl, kde pro danou buňku C budu editovat okolní buňky. Toto metoda zůstává v práci jako příklad, kterému je dobré se vyhnout.

5.7 Validace vstupů

Program je napsaný tak, aby uživatel a případně programátor měli velkou volnost v nastavování různých parametrů a ovládání programu. To si vyžádalo zvýšenou obezřetnost při programování. Bylo nutné náležitě ošetřit všechny vstupy a případně upravit program tak, aby dokázal situaci vyhodnotit a v případě nezávadných vstupů se dokázal i sám přizpůsobit. Například při načítání mikrostruktury, která má velikost 250x250 buněk do programu, který má definovaný v parametrech velikost mřížky 100x100 buněk na velikost zobrazovacího okna 500x500 (konstanta `DRAW_GRAINS`) pixelů došlo k přeškálování velikosti pixelu z 5 pixelů na buňku na 2 pixely na buňku a nahrála se správná velikost mřížky.

Vedle tohoto přizpůsobování aplikace obsahuje velké množství testů. Před startem aplikace jsou nejprve validovány proměnné v souboru `parameters.h`. V případě některé nevhodné hodnoty je na tuto hodnotu uživatel upozorněn a aplikace se vypne. Další ošetření je v místě, když nastavuji parametry. Uživatele nepustí zadat hodnotu, která je větší než maximální. Další série testů je v případě načítání vlastní mikrostruktury. Je dbáno na to, aby hodnoty, které obsahuje soubor, měly přesně definovanou strukturu. V případě nevhodné mi-

krostruktury dostane uživatel upozornění, že tato mikrostruktura nelze do programu načíst a přímo mu řekne, kde je chyba.

Poslední rozsáhlá série testů patří pro kontrolu uživatelských akcí. V případě nevhodné akce je uživatel programem upozorněn různými vyskakovacími okny. Díky tomu, je uživatel nasměrován správným směrem.

5.8 Ukládání dat

Uživatel si může uložit jak výsledky simulace, tak konfigurační soubor, ze kterého lze sestavit znovu mikrostrukturu a CA. Všechny následující operace lze vyvolat stiskem příslušného tlačítka v menu, kde si pak uživatel může vybrat místo uložení a název, případně vyhledat daný konfigurační soubor pro načtení dat.

5.8.1 Konfigurační soubor

Konfigurační soubor má přesně danou strukturu:

```
šířka_mřížky_CA
výška_mřížky_CA
počet_zrn
krok_CA
orientace_zrna, dislokační_hustota_zrna, čekací_doba
```

Listing 5.3: Ukázka konfiguračního souboru CA

Poslední trojice proměnných oddělených čárkou definují jednotlivé buňky, které jsou uloženy z CA po řádcích zleva doprava. Jejich počet je roven počtu buněk v CA.

5.8.2 Data grafů

Data z grafů budou uložena po řádcích ve formátu:

```
hodnota_na_ose_X, hodnota_na_ose_Y
```

Listing 5.4: Ukázka dat výsledků

Počet řádků je roven počtu kroků CA. Data všech grafů jsou uložena najednou a každý z nich obsahuje postfix podle typu grafu. Jedná se o `_stress`, `_grains`, `_mean_grain_size`.

5.8.3 Obrázky mikrostruktury a grafů

Poslední funkcionalitou je ukládání mikrostruktury a grafů ve formě obrázků ve formátu PNG. Data grafů mají stejný postfix jak v předchozím případě. Velikost obrázků je definována konstantou `IMAGE_GRAPH_SIZE_WIDTH` a `IMAGE_GRAPH_SIZE_HEIGHT`.

5.9 Načítání konfiguračního souboru

Načíst konfigurační soubor lze, když program vrátíme do počátečního stavu tlačítkem **reset**. Poté buď vybereme v GUI tlačítko **Load microstructure from file**, nebo přímo z menu. Po vybrání konfiguračního souboru s příponou `CA` bude následovat sada testů, která ověří správnost mikrostruktury. Validní mikrostruktura je taková, která má všechny buňky již orientované, počet buněk je roven šířce·výšce mřížky a všechny buňky mají správnou strukturu tj. 3 čísla oddělené od sebe čárkou. Pak dojde ke zkontrolování maximální povolené velikosti mřížky `DRAW_SIZE` a pokud to projde i tímto testem, tak dojde k upravení velikosti mřížky, nahrání dat a přeškálování velikosti pixelu pro jednu buňku. Po stisku tlačítka **start** můžeme pokračovat v simulaci.

5.10 Realizace GUI

Grafické uživatelské rozhraní bylo implementováno podle návrhu z kapitoly 4.2. Uživatel má možnost parametrizovat simulaci přímo v hlavním okně. V menu se pak nachází možnost zapnutí nebo vypnutí jednotlivých grafů.

5.11 Dokumentace a ostatní

Program obsahuje dokumentaci napsanou v programu Doxygen [17]. Tato dokumentace se nachází ve složce `doc`. Další možností je dokumentaci si vygenerovat pomocí příkazu `doxygen Doxyfile`. Tuto dokumentaci lze prohlížet přes libovolný internetový prohlížeč po otevření souboru `index.html` ve složce `doc`.

Aplikace obsahuje dále třídu `CellularAutomaton`, která obsahuje simulaci *Game of Life*. Pro počáteční pochopení celulárních automatů a frameworku Qt jsem začal s touto jednoduchou simulací. Třídu nechávám přibalenou k aplikaci jako ukázku typického zástupce CA.

V kódu se vedle komentářů nacházejí zakomentované nebo nepoužité dvě větší části. První z nich již tady byla zmíněna, byla to metoda `neumannSRX()`, která není vyhovující kvůli použití *centrifugal* stylu. Další část je kolem dotazování na sousedy. Momentálně jsou použita dvě statická pole, která obsahují 8 prvků představující pohyb po sousedních buňkách. Nad tímto statickým polem je cyklus, který postupně projde všechny prvky/směry v sousedství. Původní (zakomentovaná) implementace toto řešila přes 8 větví. Díky tomu byl kód delší a hlavně původní řešení, které řešilo přetečení na toroidické mřížce, muselo mít pro každý případ vlastní podmínku.

[17] `http://www.stack.nl/~dimitri/doxygen/`

Kapitola 6

Testování

V této kapitole bude popsán způsob testování aplikace. Dále bude popsán testovací scénář, který museli jednotliví testeři splnit. Na závěr bude zhodnocení.

6.1 Testy zaměřené na správnost simulace

Správnost algoritmu a výsledků experimentů byla konzultována a předvedena vedoucímu práce. Dále výsledné grafy a mikrostruktura se dala ověřit proti výsledkům vědeckých prací [2, 12, 14] a křivek pro měď.

6.2 Testy zaměřené na spolehlivost

Další část testů, které probíhaly během celé doby implementace se zaměřovaly na mezní hodnoty a nečekané vstupy. Aplikaci jsem se nad rámec zadání snažil maximálně parametrizovat, proto si to vyžádalo náležité ošetření nevalidních vstupů. Testování spočívalo ve vytvoření souborů, které měly nevalidní obsah a testoval jsem, jestli program vstup odmítne a uživateli zahlásí hlášku se správným popisem problému.

Bylo nutné otestovat vykreslování mikrostruktury pro toroidickou mřížku. To se dá jednoduše simulovat při vytvoření dvou nukleačních zárodků a pozorovat chování na krajích zobrazovaného okna, viz obrázek 6.1.

Dále aplikace procházela testem pro správu paměti programem Valgrind [18] a bylo využito syntaktického analyzátoru kódu cppcheck [19].

6.3 Testovací scénář

Pro tvorbu kvalitního software je důležité výsledek náležitě otestovat nezávislými testery. Proto se do testování mojí aplikace zapojilo celkem 11 testerů.

[18] `http://valgrind.org/`

[19] `http://cppcheck.sourceforge.net/`

Obrázek 6.1: Test toroidického zakřivení mřížky

Testování probíhalo za mojí asistence na jejich počítačích (Windows i Linux, kde na Linuxu verze knihovny Qt byla 5.7.0. a 5.8.0.), ať už to bylo naživo nebo pomocí sdílené obrazovky přes program Skype [20].

Testeři by se dali rozdělit do dvou kategorií. Jedni, kteří znají fyzikální jev dynamické rekrystalizace ze školy. Tato skupina dobře simuluje situaci, když k programu přijde vědec, který zná tento jev, ale setkává se s programem poprvé. A druhá skupina, kteří jsou povětšinou studenti jiných než strojařských oborů a nemají tyto znalosti.

Testovací scénář byl sestaven tak, aby pokryl všechny funkční požadavky programu. A navíc ukázal, jestli dokáží uživatelé, kteří vidí tento program poprvé ho správně ovládat.

Na začátku testování jsem poskytl testerům základní informace o tom, co se v mojí aplikaci odehrává. Pak dostali následující instrukce a bez mých dalších zásahů se snažili dané úkoly splnit.

Testovací scénář začínal po spuštění programu a skládal se z 12 úkolů.

(1.) Vygenerování mikrostruktury v programu s počátečním počtem zrn o hodnotě 500.

(2.) Zrychlit běh simulace.

(3.) Kolem 150 kroku CA zastavit program.

(4.) Nastavit zvyšování dislokační hustoty na hodnotu 5 a pokračovat v simulaci.

[20] `https://www.skype.com/cs/`

(5.) Kolem kroku 300 zastavit simulaci a vytvořit novou mikrostrukturu, která bude mít:

   (a) 150 počátečních zrn
   (b) 100 kritickou hodnotu dislokační hustoty
   (c) 80 kritickou hodnotu nukleace
   (d) 5‰ pravděpodobnost, že proběhne nukleace
   (e) 25 čekací doba.

(6.) Spuštění programu s předchozími parametry.

(7.) Kolem 100 kroku simulace DRX uložit konfiguraci CA, uložit obrázek mikrostruktury, uložit data z grafů.

(8.) Spustit simulaci a nechat doběhnout do 200 kroku.

(9.) Uložit obrázky grafů do souborů.

(10.) Načíst dříve uložený konfigurační soubor CA.

(11.) Spustit simulaci nad načtenou mikrostrukturou a zkusit vypnout zobrazování všech grafů.

(12.) Vypnout aplikaci.

6.4 Výsledky testování

Během testování se vyskytly celkem tři problémy. První spočíval v nepřesném pojmenovaní názvů tlačítek v menu. Tento problém jsem vyřešil lepším pojmenováním zaměnitelných tlačítek. Druhý problém nastal v případě generování mikrostruktury, která má pouhé dvě zrna. V tomto případě došlo na systému Windows k zaplnění zásobníku během volání rekurze v metodě zjišťující počet zrn. Tuto situaci jsem vyřešil navýšením zásobníku. Poslední problém spočíval v generování názvu souborů na Windows. Místo `image_grain.png` to vytvořilo soubor s názvem `image.pdf_grain.png.png`. Opět jsem upravil kód, aby soubory i na Windows byly pojmenovány správně.

Dalo by se říct, že jinak skupina tvořena studenty strojařských škol neměla žádné další problémy. U druhé skupiny byl menší problém při zadávání jednotlivých parametrů, kdy jim trvalo delší dobu, než se v programu zorientovali.

Jeden uživatel měl problémy s rozeznáním konfiguračního souboru CA a souborů z grafů, které obsahují data.

Kapitola 7

Výsledky experimentů

V této kapitole budou popsány některé výsledky experimentů. Mimo jiné budou rozepsány problémy, se kterými jsem se během implementace setkal a které jsem se snažil nad rámec práce vyřešit.

7.1 Úvodní zhodnocení a budoucnost implementace

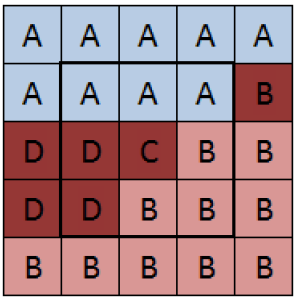
Náplní práce byla implementace CA simulující dynamickou rekrystalizaci, který poslouží vědcům jako dobrý příklad simulace komplexních systémů. Díky rozdílnému obsahu disertační práce a kódu v jazyku CellLang nebylo jednoduché napsat správný algoritmus. Hlavním cílem bylo zopakování výsledků grafů z disertační práce, což se povedlo, a jsou diskutované v druhé části této kapitoly.

I přesto, že původní náplní mojí práce nebylo tento algoritmus odladit a dovést k dokonalosti, tak mě tato část práce natolik motivovala a zajímala, že jsem se snažil některé problémy vyřešit. Této části práce jsem věnoval značné úsilí. Modifikací bylo víc a některé z nich vedly na hodně zajímavé chování (v rámci CA obecně). Není v možnostech, a hlavně v rozsahu této práce, všechny pozorované jevy zadokumentovat, zvlášť když se jedná o obecné chování CA. Proto zde budou zmíněny jen možné úpravy v rámci dynamické rekrystalizace.

Celulární automaty jsou obecně velmi náchylné na chaotické chování, které v tomto případě může vzniknout nevhodně zadanými parametry. Některému nežádoucímu chování byly schopny moje úpravy vzdorovat, ale bohužel jsem je nemohl použít. Důvody jsou uvedené v této kapitole.

7.2 Problém s tvorbou čtverců

Můj první návrh spočíval ve výběru náhodného zrna v okolí, a až poté se testovalo, jestli nastane pro danou buňku rekrystalizace.



Obrázek 7.1: Chování algoritmu na hranicích zrn. Nacházejí se zde různě orientovaná zrna A, B a zrno D. Pro danou buňku C patřící zrnu B je okolí zvýrazněné tučným ohraničením. Modrá barva představuje nízkou dislokační hustotu uvnitř buňky, kdežto červené barvy představují vyšší dislokační hustotu

Díky této modifikaci se v rámci celého algoritmu tvořila více globulární zrna. Z pohledu fyziky něco takového ale nemůžu udělat. Tato situace je znázorněna na obrázku 7.1. Nacházím se v buňce C, která je momentálně orientovaná jako zrno B, kde bych správně (podle fyziky) měl vybrat buňku náležící vedlejšímu zrnu A [21], která má navíc dislokační hustotu takovou, že splní podmínku $\rho_C - \rho_i \geq \rho_{cr}$. Takže v tomto případě by s pravděpodobností $p = 0{,}5$ buňka C od teď náležela zrnu A. Tento algoritmus ovšem generuje čtvercová zrna. Je to z toho důvodu, že podmínky budou velmi pravděpodobně splněny i pro ostatní buňky náležící na této hranici. A pokud je pravděpodobnost $p = 0{,}5$ a t_w roven třeba 25, tak buňky se budou postupně „nabalovat“ a tím pádem zaujmou čtvercový tvar, který se při další příležitosti rekrystalizace „obalí“ o další „slupku“. Protože pro další růst musí čekat, dokud t_w nebude roven 0. Umělým trikem může být například pro pravděpodobnost $p = 0{,}5$ nastavení t_w na hodnotu 2.

Moje řešení tedy bylo vybrat náhodnou buňku z okolí a až poté testovat jednotlivé proměnné, jestli dynamická rekrystalizace vznikne. V našem případě bych si mohl vybrat náhodnou buňku náležící zrnu D, která má vysokou dislokační hustotu, takže by podmínka nebyla splněna a bylo by rozhodnuté, že dynamická rekrystalizace nenastane.

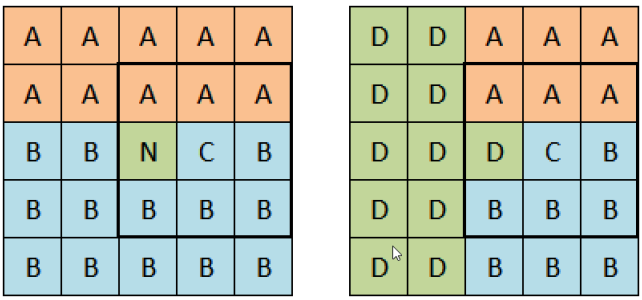
Druhou modifikací byla úprava pravděpodobnosti rekrystalizace. Místo $p = 0{,}5$ jsem uvažoval $p = 1/8$. V tomto případě tvar zrn také nebude čtvercový

[21] Tak jako třeba kapka vody se snaží zaujmout kulovitý tvar, protože je to z hlediska fyziky nejlepší energetický tvar, tak se buňky snaží podobným způsobem růst do míst, které jsou pro ně nejlépe energeticky výhodné.

a bude se blížit globulárnímu tvaru, ale jen v případě nižší hodnoty t_w.

7.3 *Good shape*

Dalším omezením je *Good Shape*. To znamená, že pokud by v případě rekrystalizace vzniklo zrno, které by mělo úzkou část delší nebo rovna dvě buňky, tak tuto situaci nepovolíme. Jedná se o situaci na obrázku 7.2.



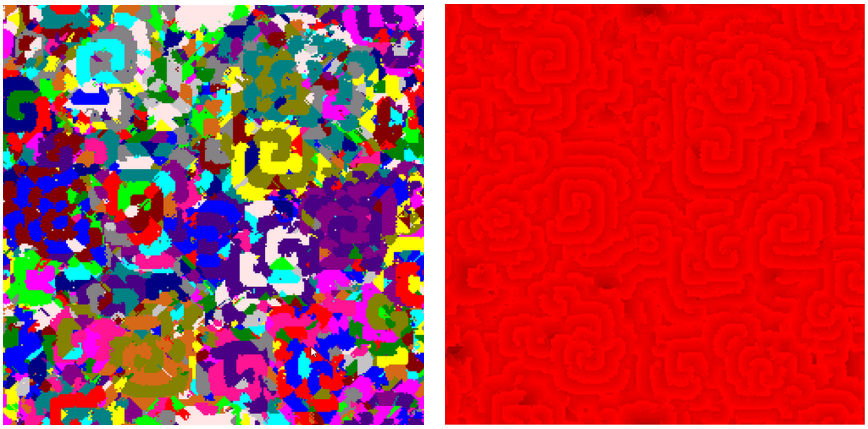
Obrázek 7.2: *Good shape* detekce. Pro buňku C je okolí zvýrazněné tlustější barvou. Na levém obrázku je nukleační zárodek. Na pravé straně by v případě růstu buňky D, která je nalevo od buňky C vzniklo úzké zrno

V levé části obrázku je nukleační zárodek, který může růst. Naopak v situaci na pravé straně nechceme povolit růst zrn. Vznikla by příliš úzká část zrna. Zrno by v tomto případě šlo do energeticky nevhodného stavu. Tento jev z pohledu fyziky taktéž není obvyklý.

Z informatického hlediska existuje řešení. V případě, že sousední buňka i má celkový počet buněk stejného zrna v okolí pro buňku C roven 1, tak bych následně otestoval kolik sousedů má tato sousední buňka i ve svém okolí. Pokud vyjde 0, víme že je to nukleační zárodek a můžeme růst, pokud by počet sousedů byl 1, který by byl naproti zrnu C, víme, že růst nemůžeme. Z pohledu fyziky je ale tento postup neobhajitelný.

7.4 Spirály

Dalším problémem, který můj náhodný výběr buňky do určité míry řešil, byly spirály, které jsou viditelné na obrázku 7.3. Díky mojí náhodnosti dokázal algoritmus předcházet těmto jevům, kterým trpí aktuální implementace při nevhodně zadaných parametrech. Toto chování bylo pozorováno i u vedoucího v disertační práci.



Obrázek 7.3: Spirály, které vzniknou při nevhodně nastavených parametrech. Vlevo mikrostruktura, kde jednotlivým buňkám přísluší jejich dislokační hustota

7.5 Ostatní možné odchylky

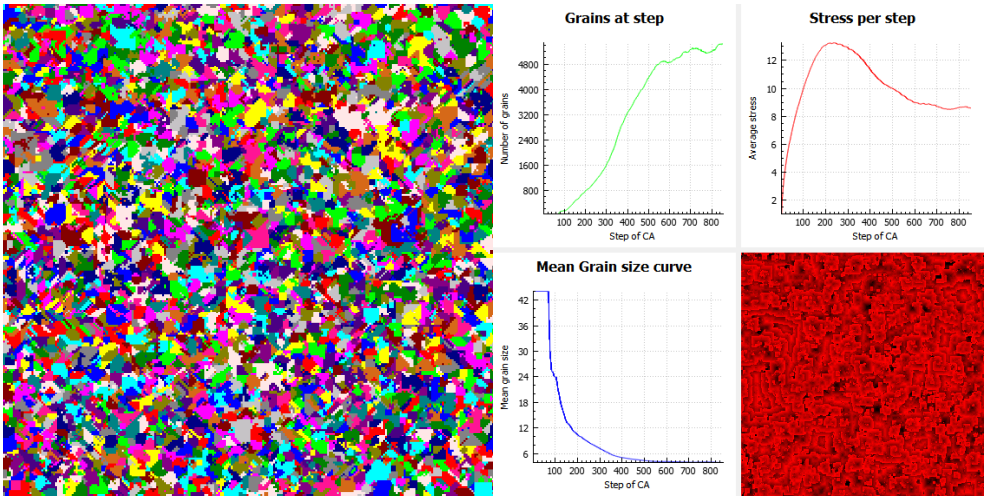
Mezi další odchylky patří implementace, která probíhala nad reálnými čísly. V době implementace předchozího algoritmu nebyly přístupné všechny prostředky, a proto simulace probíhala nad omezeným množství hodnot. Díky tomu bylo zapotřebí dělat velké množství optimalizačních triků, které mohly mít (a nejspíše měly) vliv na výsledný běh algoritmu. Poslední možností může být díky paralelnímu zpracování pro engine CellLang, který dělá navíc další kroky, o kterých já ani vedoucí nevíme, zvlášť bez dokumentace.

7.6 Zopakování výsledků podle disertační práce

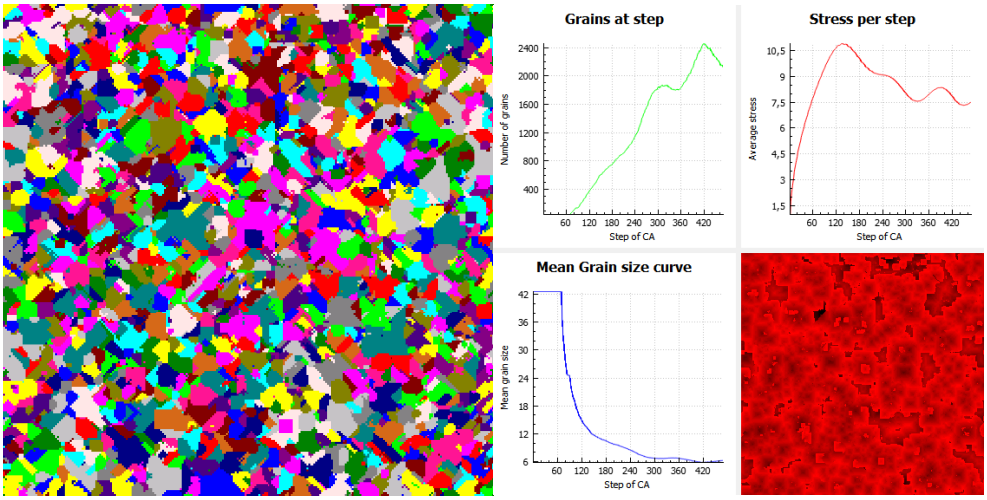
V této části naopak budou představeny výsledky, které se již povedly napodobit. Velikost mřížky byla pro všechny experimenty nastavena na 256 v obou směrech. V testech na obrázcích 7.4 a 7.6 došlo ke změně počáteční velikosti zrna. Rozdíl mezi testy na obrázcích 7.4 a 7.5 je v jiné hodnotě t_w, čímž měníme rychlost pohybu hranice.

7.6.1 *Single peak* v křivce mechanického napětí

Pro *Single peak* chování následují dva obrázky. Oproti prvnímu obrázku 7.4 je na druhém obrázku 7.5 v křivce mechanického napětí vidět, že probíhá další vlna dynamické rekrystalizace dříve, než ta předchozí skončila.



Obrázek 7.4: Jeden vrchol v křivce mechanického napětí. Počáteční počet zrn 50, $\rho_{cr} = 80$, $\rho_{nucl} = 70$, $t_w = 25$, $p_{nucl} = 0{,}001$. Zvyšování dislokační hustoty každý krok probíhalo o 1

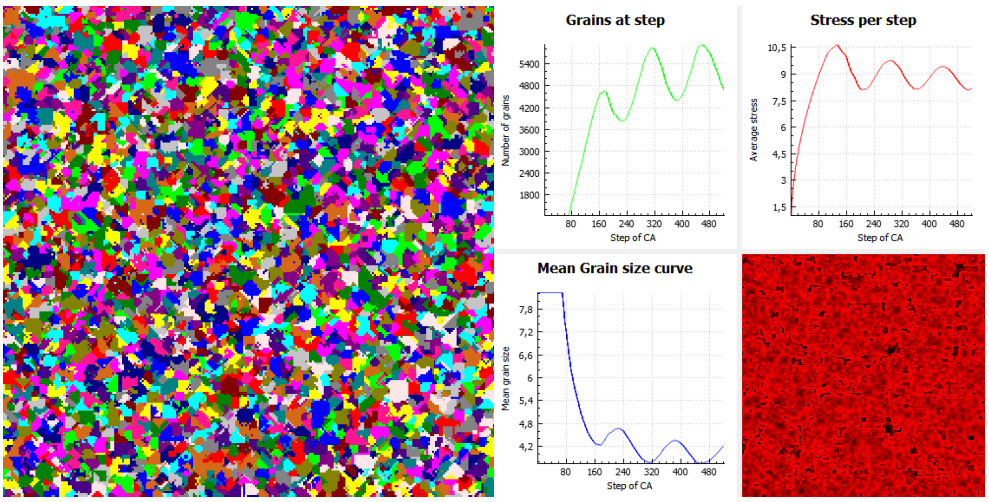


Obrázek 7.5: Jeden vrchol v křivce mechanického napětí, ve kterém lze pozorovat další vlnu dynamické rekrystalizace dřív, než předchozí doběhla. Oproti předchozí konfiguraci došlo k úpravě čekací doby pro DRX. Kde $t_w = 10$

7.6.2 *Multiple peak* v křivce mechanického napětí

U vícenásobného vrcholu je to zajímavější. Na křivce mechanického napětí a křivce střední velikosti zrna lze vidět oscilace, které jsou posunuté o 1/4 fáze, viz obrázek 7.6. Toto chování již bylo publikováno „*Minima and maxima of the grain size curve are phase shifted by approximately 1/4 of a period before the corresponding maxima and minima of the flow stress curve during*

the multiple peak behaviour[12]“.



Obrázek 7.6: Více vrcholů v křivce mechanického napětí. Počáteční počet zrn 1500, $\rho_{cr} = 80$, $\rho_{nucl} = 70$, $t_w = 25$, $p_{nucl} = 0{,}001$. Zvyšování dislokační hustoty každý krok probíhalo o 1

Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat celulární automat simulující dynamickou rekrystalizaci a udělat k němu vhodné grafické uživatelské rozhraní. Výsledek práce bude sloužit vědcům a ostatním zájemcům jako vhodný příklad komplexních systémů. Implementaci ztěžovalo rozdílný popis simulace algoritmu v práci vůči obsahu kódu. Nakonec se implementace povedla a program je schopný zobrazovat mikrostrukturu a křivky mechanického napětí, střední velikosti zrna a počtu zrn. Nad rámec původního zadání bylo přidané okno zobrazující dislokační hustotu. Velká míra parametrizace zaručuje simulaci v různých podmínkách.

Protože některé věci v disertační práci nejsou dokumentované a poskytnutý kód obsahuje rozdílné řešení problému, tak jsem se některé věci snažil vyřešit sám. Jednalo se zejména o tvorbu čtvercových tvarů zrn, které bylo pozorováno i v disertační práci. I když několik nápadů je v práci popsáno, který tento problém řeší, tak z pohledu fyziky jsem nemohl tyto úpravy zahrnout do programu. I přesto dokáže program počítat správné výsledky, které jsou sledovány v disertační práci.

Program splňuje všechny požadavky, které byly vyjmenovány v kapitole věnované analýze. Aplikace je napsaná v souladu s fyzikou, aby mohla být vědci v budoucnu jednoduše rozšiřovatelná.

Aplikace je cílena do vědecké komunity a bude vyvěšena na portálech ResearchGate a SourceForge. Proto byly přeloženy příručky do angličtiny a vyvěšeny na portály spolu s programem. Tyto přeložené příručky lze najít na přiloženém CD.

Budoucí práce

I když byly všechny požadavky splněny a bylo přidáno několik věcí nad rámec původního zadání, tak stále je možnost aplikaci vylepšit.

- Díky *centripetal* stylu by dalším krokem mohla být paralelizace.

- Návrh je udělaný tak, aby nebyl žádný problém s přidáním dalšího grafu. Taktéž díky třídě `RecrystallizationHandler` a stavů simulace není problém přidat další jev.
- Další možností je úprava algoritmu, aby netvořil čtvercová zrna. Nápady na řešení jsou popsány v kapitole Problém s tvorbou čtverců.
- Mezi poslední možnost rozšíření patří detekce úzkých zrn, která je popsána v kapitole *Good shape*.

Literatura

[1] HARRISON, Robert L., Carlos GRANJA a Claude LEROY. Introduction to Monte Carlo Simulation [online]. In: . s. 17-21 [cit. 2016-12-03]. DOI: 10.1063/1.3295638. Dostupné z: http://scitation.aip.org/content/aip/proceeding/aipcp/10.1063/1.3295638

[2] KROC, Jiří a Václav PAIDAR. Modelling of Recrystallization and Grain Boundary Migration by Cellular Automata. Materials Science Forum [online]. 2003, 426-432, 3873-3878 [cit. 2016-12-02]. DOI: 10.4028/www.scientific.net/MSF.426-432.3873. ISSN 1662-9752. Dostupné z: http://www.scientific.net/MSF.426-432.3873

[3] KROC, Jirí a Peter M.A. SLOOT. Complex Systems Modeling by Cellular Automata. Encyclopedia of Artificial Intelligence [online]. IGI Global, 2009, s. 353 [cit. 2016-12-02]. DOI: 10.4018/978-1-59904-849-9.ch054. ISBN 9781599048499. Dostupné z: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-59904-849-9.ch054

[4] HOEKSTRA, Alfons G., Jiří KROC a Peter M.A. SLOOT. Introduction to Modeling of Complex Systems Using Cellular Automata [online]. s. 1 [cit. 2016-12-02]. DOI: 10.1007/978-3-642-12203-3_1. Dostupné z: http://link.springer.com/10.1007/978-3-642-12203-3_1

[5] TOFFOLI, Tommaso. Cellular automata as an alternative to (rather than an approximation of) differential equations in modeling physics. Physica D: Nonlinear Phenomena [online]. 1984, 10(1-2), 117-127 [cit. 2017-04-08]. DOI: 10.1016/0167-2789(84)90254-9. ISSN 01672789. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/0167278984902549

[6] BAK, Per. How nature works: The science of self-organized criticality. New York: Springer-Verlag, 1996. ISBN 0-387-94791-4.

[7] FISHWICK, Paul A. Handbook of dynamic system modeling. Boca Raton: Chapman & Hall/CRC, c2007. ISBN 9781584885658.

[8] RESNICK, Mitchel. Turtles, termites, and traffic jams: explorations in massively parallel microworlds. Cambridge: Bradford Book, 1997. ISBN 0-262-18162-2.

[9] HEYLIGHEN, Francis. Self-organization: emergence and the architecture of complexity, 1989. Free University of Brussels, Brussels.

[10] HUMPHREYS, Owen. An amazing photo of a murmuration of starlings after sunset. In: Mudfooted [online]. Science Photo Library, 2011 [cit. 2017-04-14]. Dostupné z: http://images.mudfooted.com/murmuration-bird-flock.png

[11] BERTO, Francesco a TAGLIABUE, Jacopo. The Stanford En- cyclopedia of Philosophy: Cellular Automata [online], 2012. [cit. 207-04-16] Dostupné z: https://plato.stanford.edu/entries/cellular-automata/

[12] KROC, Jiří. Application of Cellular Automata Simulations to Modelling of Dynamic Recrystallization [online]. s. 773 [cit. 2016-12-02]. DOI: 10.1007/3-540-46043-8_78. Dostupné z: http://link.springer.com/10.1007/3-540-46043-8_78

[13] Boundary conditions. JCASim: Cellular automata simulation system [online]. [cit. 2017-04-16]. Dostupné z: http://www.jcasim.de/main/node6.html

[14] KROC, Jiří. Simulation of Dynamic Recrystallization by Cellular Automata [online]. Praha, 2001. Disertační práce. Matematicko-fyzikální fakulta Univerzity Karlovy. Vedoucí práce Pavel Lukáč [cit. 2017-4-12]. Dostupné z: https://www.researchgate.net/publication/311733496_Simulation_of_Dynamic_Recrystallization_by_Cellular_Automata

[15] CONTRIBUTORS OF WIKIPEDIA. Torus2.png [online]. [cit. 2017-04-11]. Dostupné z: https://cs.wikipedia.org/wiki/Torus

[16] GARDNER, Martin. The fantastic combinations of John Conway's new solitaire game "life". Scientic American [online]. 1970, 223, 120-123 [cit. 2016-12-03]. Dostupné z: https://www.researchgate.net/publication/238757964_The_Fantastic_Combinations_of_John_Horton_Conway's_New_Solitaire_Game_Life'

[17] KRAL, Robert. Fyzika materiálů II Zotavení a rekrystalizace 1.část, přednáškové slidy [online]. [cit. 2017-04-22]. Dostupné z : http://material.karlov.mff.cuni.cz/people/janecek/studenti/Fyzika_materialu2/Rekrystalizace_2h_1cast.ppt

[18] KRAL, Robert. Fyzika materiálů II Zotavení a rekrystalizace 2.část, přednáškové slidy [online]. [cit. 2017-04-22]. Dostupné z : http://material.karlov.mff.cuni.cz/people/janecek/studenti/Fyzika_materialu2/Rekrystalizace_2h_2cast.ppt

[19] BENEŠ, Libor. Nauka o materiálech, přednáškové slidy [online]. [cit. 2017-04-22]. Dostupné z: http://users.fs.cvut.cz/libor.benes/vyuka/mattech/01_Materialem%20letem%20svetem.pdf

[20] PLUHAŘ, Jaroslav. Nauka o materiálech: Celost. vysokoškol. učebnice. Praha: SNTL, 1989.

[21] PTÁČEK, Luděk. Nauka o materiálu I. 2., opr. a rozš. vyd. Brno: Akademické nakladatelství CERM, c2003. ISBN 8072042831.

[22] PTÁČEK, Luděk. Nauka o materiálu II. 2. opr. a rozš. vyd. Brno: CERM, 2002. ISBN 80-7204-248-3.

[23] HORÁČEK, Jaroslav. Nauka o materiálu. Praha: Česká zemědělská univerzita, 2000. ISBN 80-213-0397-2.

[24] Fyzikální základy vědy o materiálu. [online]. [cit. 2017-04-22]. Dostupné z: http://www.ped.muni.cz/wphy/fyzvla/

[25] Teoretické základy nauky o materiálu. [online]. [cit. 2017-04-22]. Dostupné z: http://umi.fs.cvut.cz/wp-content/uploads/2014/08/2_teoreticke-zaklady-nauky-o-materialu.pdf

[26] Kurzy pro fyziky, online materiály. Matematicko-fyzikální fakulta Univerzity Karlovy. [online]. [cit. 2017-04-23]. Dostupné z: http://physics.mff.cuni.cz/kfpp/skripta/kurz_fyziky_pro_DS/display.php/kontinuum/obrazky/image040.gif

[27] SAKAI, T., AKBEN, M.G., JONAS J.J. Dynamic recrystallization during the transient deformation of a vanadium microalloyed steel. Acta Metall, 1983. 631–642.

[28] CONTRIBUTORS OF WIKIPEDIA. Ricristallizzazione_e_accrescimento.svg [online]. [cit. 2017-04-26]. Dostupné z: https://upload.wikimedia.org/wikipedia/commons/9/98/Ricristallizzazione_e_accrescimento.svg

[29] ROLLETT, A.D., M.J. LUTON a D.J. SROLOVITZ. Microstructural simulation of dynamic recrystallization. Acta Metallurgica et Materialia [online]. 1992, 40(1), 43-55 [cit. 2016-12-02]. DOI: 10.1016/0956-7151(92)90198-N. ISSN 09567151. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/095671519290198N

[30] PECZAK, P. a M.J. LUTON. A Monte Carlo study of the influence of dynamic recovery on dynamic recrystallization. Acta Metallurgica et Materialia [online]. 1993, 41(1), 59-71 [cit. 2016-12-02]. DOI: 10.1016/0956-7151(93)90339-T. ISSN 09567151. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/095671519390339T

[31] HESSELBARTH, H.W. a I.R. GÖBEL. Simulation of recrystallization by cellular automata. Acta Metallurgica et Materialia [online]. 1991, 39(9), 2135-2143 [cit. 2016-12-02]. DOI: 10.1016/0956-7151(91)90183-2. ISSN 09567151. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/0956715191901832

[32] HUANG, Shi-quan, You-ping YI, Peng-chuan LI a Hai-lin HE. Simulation of dynamic recrystallization in 23Co13Ni11Cr3Mo steel using a modified cellular automaton. Journal of Central South University [online]. 2014, 21(2), 454-459 [cit. 2017-04-14]. DOI: 10.1007/s11771-014-1959-7. ISSN 2095-2899. Dostupné z: http://link.springer.com/10.1007/s11771-014-1959-7

[33] KUGLER, G., TURK, R. Study of the influence of initial microstructure topology on the kinetics of static recrystallization using a cellular automata model. Computational Materials Science [online]. 2006, 37 (3) , pp. 284-291 [cit. 2017-05-07]. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0927025605002740

Příloha **A**

Seznam použitých zkratek

2D Two-dimensional space

CA Cellular automaton

CAs Cellular automata

DRX Dynamic recrystallization

GUI Graphical user interface

PNG Portable Network Graphics

RGB RGB color model

SRX Static recrystallization

TXT Text file

Příloha B

Uživatelská příručka

V této příručce budou popsány základní informace k ovládání a možnostem programu, který je výsledkem bakalářské práce s názvem „Návrh a implementace buněčného automatu simulujícího dynamickou rekrystalizaci“.

B.1 Obecné informace

Aplikace zobrazuje mikrostrukturu, dislokační hustotu a vizualizuje výsledky simulace uživateli ve formě grafů. Jedná se o tyto křivky: křivka mechanického napětí, křivka zobrazující průměrnou velikost zrna a křivka zobrazující celkový počet zrn v simulaci. Příprava inicializační mikrostruktury je prováděna pomocí pseudo-statické rekrystalizace. Uživatel může výsledky libovolně ukládat ve formě obrázků, konfiguračních souborů nebo souborů obsahující data.

Volitelnost parametrů zaručuje simulaci materiálu deformovaných v různých podmínkách.

B.2 Spuštění programu

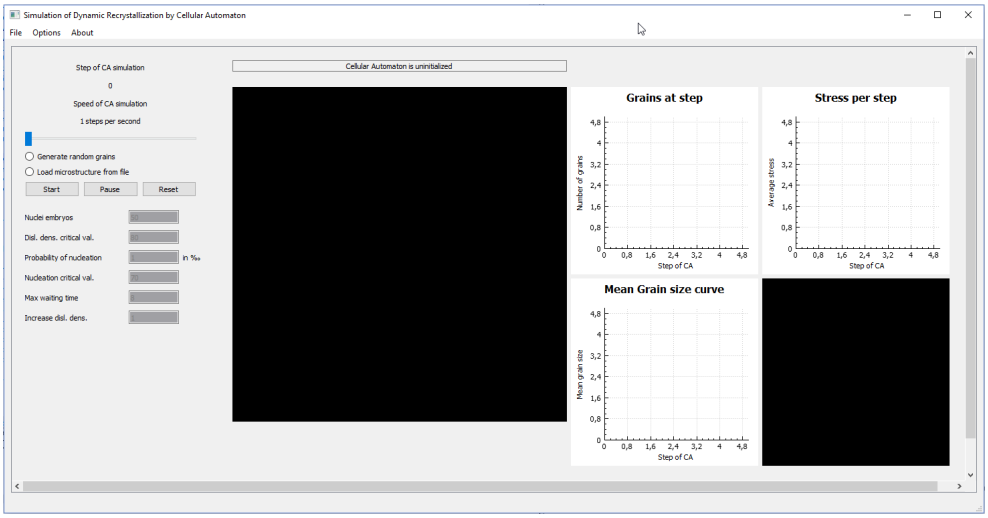
Na systému **Windows** stačí jít do složky `/exe`, kde se nachází spustitelná verze programu včetně potřebných knihoven pro běh. Spuštění probíhá otevřením souboru `DRX.exe`.

Na systému **Linux** je potřeba mít nainstalovaný Qt 5.7.0., qmake a g++. A spustit v adresáři `/src/DRX` následující trojici příkazů:

```
$ qmake
$ make
$ ./DRX
```

B.3 Okno programu

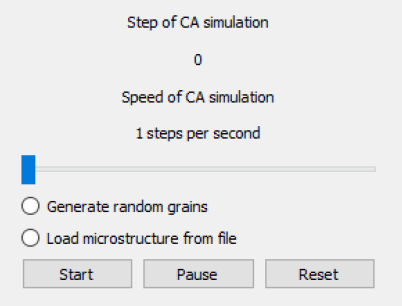
Po spuštění programu můžeme vidět hlavní obrazovku B.1.



Obrázek B.1: Hlavní obrazovka programu

Obrazovka je složena z šesti částí:

- levá horní část slouží k ovládání běhu simulace a zobrazování informací okolo běhu programu
- levá dolní část slouží k nastavení parametrů
- prostřední část obsahuje mikrostrukturu
- v pravé části se nachází vykreslované grafy
- v horní části je menu
- dolní panel zobrazuje dodatečné informace.



Obrázek B.2: Informace o celulárním automatu a ovládání běhu programu

B.3.1 Informační a ovládací část

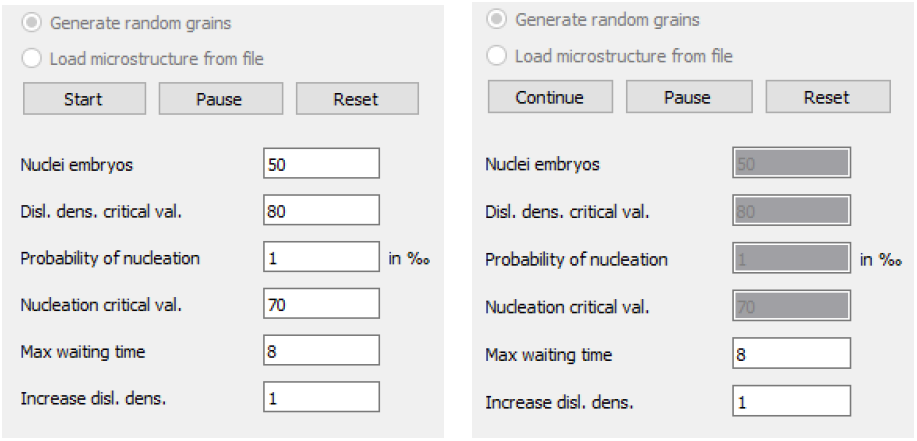
Tato část zobrazuje některé informace kolem běhu celulárního automatu a poskytuje uživateli základní ovládání programu.

- **Krok CA** pod titulkem ukazuje, v kterém kroku simulace se nacházíme.
- **Rychlost běhu** představuje počet kroků za sekundu.
- **Jezdec nastavování rychlosti** slouží pro nastavení rychlosti běhu.
- **Výběr inicializační mikrostruktury**
  - **Generování mikrostruktury** podle zadaných parametrů v programu - nastavení proběhne v dalším kroku.
  - **Načtení vlastní mikrostruktury** ze souboru s příponou *.ca. Jedná se o konfigurační soubor CA.
- **Tlačítka pro ovládání programu**
  - **Start/Continue** spustí simulaci.
  - **Pause** pozastaví simulaci.
  - **Reset** vrátí celulární automat do původního stavu. Pro vytvoření nové mikrostruktury, nebo při načítání vlastní mikrostruktury je nutné celulární automat tímto tlačítkem „připravit“.

B.3.2 Konfigurační část

V této části můžeme měnit parametry simulace. Na obrázku B.3a je zobrazena volitelnost v případě generování mikrostruktury v programu, naopak na obrázku B.3b jde vidět možnosti úpravy parametrů během pozastavení simulace.

- **Počet nukleačních zárodků** vytvořených během generování inicializační mikrostruktury.



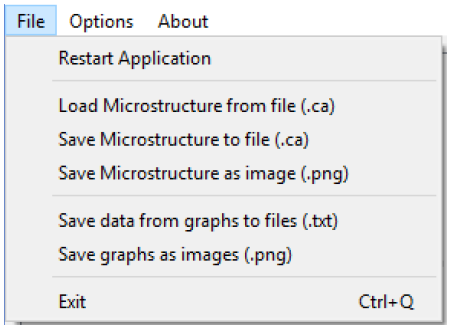
(a) Nastavení parametrů (b) Úprava parametrů za běhu

Obrázek B.3: Nastavení parametrů

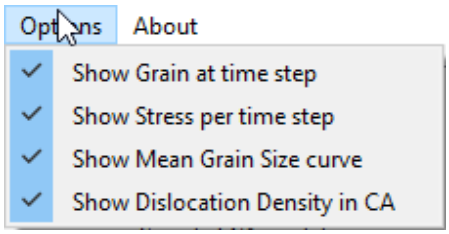
- **Kritická hodnota dislokační hustoty** pro dynamickou rekrystalizaci, která udává jak velký rozdíl dislokační hustoty mezi dvojicí buněk na hranici zrn je potřeba pro spuštění dynamické rekrystalizace.
- **Pravděpodobnost nukleace** v jednotkách promile.
- **Kritická hodnota nukleace** pro nukleaci, která udává jak velká je potřeba dislokační hustota v buňce, aby mohla proběhnout nukleace.
- **Maximální čekací doba** pro dynamickou rekrystalizaci, řídí se tím rychlost pohybu hranice.
- **Zvýšení dislokační hustoty** představuje o kolik se každý krok simulace pro buňky, které nebyly rekrystalizované zvýší dislokační hustota.

B.3.3 Menu

Menu se nachází v horní části. Obsahuje tlačítka pro načítání a ukládání, případně tlačítka určené pro vypnutí a zapnutí zobrazování jednotlivých křivek.



(a) Nabídka v menu File



(b) Nabídka v menu Options

Obrázek B.4: Menu

- **Restart Aplikace** vrátí aplikaci do původního stavu.
- **Načtení mikrostruktury ze souboru** s příponou *.ca. Jedná se o konfigurační soubor CA. Pro správné načtení je potřeba načíst validní mikrostrukturu.
- **Uložení mikrostruktury do souboru** s příponou *.ca. Jedná se o konfigurační soubor CA, ze kterého jde poté zpětně nahrát do programu mikrostrukturu.
- **Uložení mikrostruktury jako obrázek** s příponou *.png. Uloží mikrostrukturu jako obrázek.
- **Uložit data z grafů do souborů** s příponou *.txt. Jednotlivé grafy budou mít následující postfixy `_stress`, `_mean_grain_size` a `_grains`.
- **Uložit grafy jako obrázky** ve formátu *.png. Jednotlivé grafy budou mít následující postfixy `_stress`, `_mean_grain_size` a `_grains`.
- **Konec** - tlačítko vypne program.
- **Zobraz křivku počtu zrn** po kliknutí vypne/zapne zobrazování grafu.
- **Zobraz křivku mechanického napětí** po kliknutí vypne/zapne zobrazování grafu.
- **Zobraz křivku průměrné velikosti zrna** po kliknutí vypne/zapne zobrazování grafu.
- **Zobraz dislokační hustotu** po kliknutí vypne/zapne zobrazování okna.
- **O aplikaci** zobrazí okno, které obsahuje základní informace o programu.

B.3.4 Mikrostruktura

Uprostřed je zobrazena mikrostruktura, která je obarvena na základě orientace zrna. Nad ní se nachází titulek, který zobrazuje aktuálně probíhající jev v simulaci.

B.3.5 Grafy

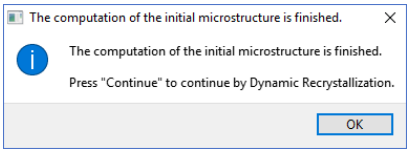
V pravé části jsou vykreslovány grafy. Jedná se o graf zobrazující počet zrn (zelená křivka), průměrnou velikost zrna (modrá křivka), křivku mechanického napětí (červená křivka) a poslední okno zobrazuje dislokační hustotu. Čím tmavší je barva v tomto posledním okně, tak tím větší je v těchto buňkách dislokační hustota.

B.3.6 Stavový řádek

Součástí GUI je také stavový řádek vlevo dole, který po najetí na libovolný graf, okno, titulek zobrazí nápovědu. V aplikaci se nachází i vertikální a horizontální posuvníky.

B.4 Průběh simulace

Pro začátek experimentu v levé části vybereme, jestli chceme vygenerovat vlastní mikrostrukturu **Generate random grains** podle zadaných parametrů nebo již máme nějakou vlastní předpřipravenou **Load microstructure from file**. Pokud jsme vybrali načtení vlastní mikrostruktury, tak se nám otevře okno, ve kterém vybereme příslušný konfigurační soubor obsahující data o mikrostruktuře. Pokud soubor je validní, tak budeme moct spustit program tlačítkem **Start**.

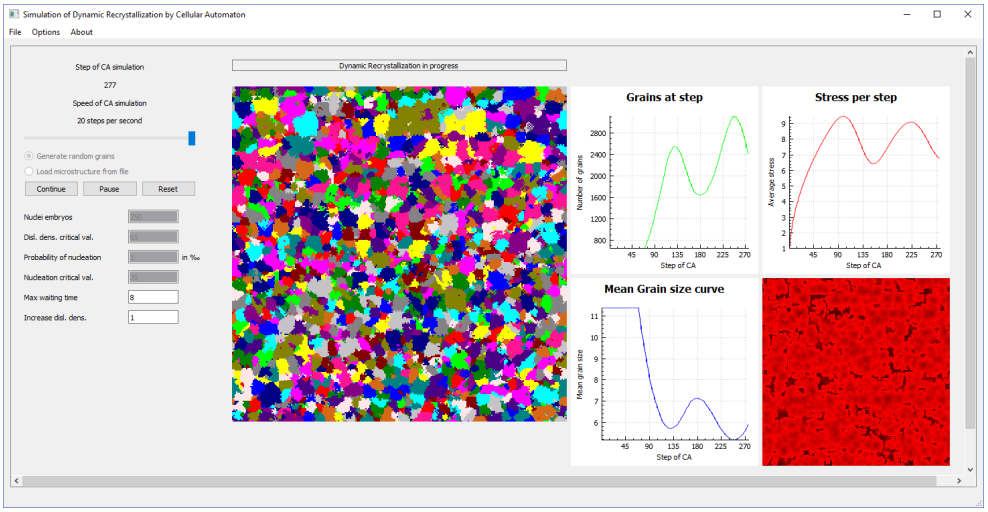


Obrázek B.5: Okno informující o dokončení přípravy mikrostruktury

V případě generování vlastní inicializační mikrostruktury se nám zpřístupní nastavování parametrů simulace, které lze vidět na obrázku B.3a. Po nastavení parametrů spustíme aplikaci tlačítkem **Start**.

V simulaci se vygenerují nukleační zárodky a poté bude probíhat pseudostatická rekrystalizace, která vygeneruje inicializační mikrostrukturu. Po vygenerování budeme programem upozorněni, viz obrázek B.5 a program bude připravený pro simulaci dynamické rekrystalizace, během které dojde k vy-

nulování čítače kroků [22]. Tlačítkem **Continue** (na původním místě, kde bylo tlačítko **Start**) odstartujeme simulaci.



Obrázek B.6: Průběh simulace

Je možnost simulaci pozastavit tlačítkem **Pause** a upravit některé parametry, viz obrázek B.3b. Po nastavení můžeme v simulaci s upravenými podmínkami pokračovat.

V menu v záložce **File** je možnost uložit jednotlivé výsledky experimentů. V záložce **Options** je možnost vypnout nebo zapnout zobrazování grafů.

Pro vytvoření nového experimentu je potřeba vrátit celulární automat do původního stavu tlačítkem **Reset**. Budeme vyzváni o uložení výsledků experimentu, které jinak budou ztraceny.

[22] Pro experimenty je lepší mít krok CA vzhledem k dynamické rekrystalizaci.

Příloha C

Programátorská příručka

V této příručce budou popsány základní informace kolem programování, které je vhodné při rozšiřování této aplikace dodržet. Program je výsledkem bakalářské práce s názvem „Návrh a implementace buněčného automatu simulujícího dynamickou rekrystalizaci“.

C.1 Technologie

Pro tvorbu byly využity následující technologie:

- jazyk **C++**
- framework **Qt**
- dokumentace v programu **doxygen**
- externí knihovna pro vykreslování grafů **QCustomPlot**.

Pro vývoj je vhodné použít třeba programu Qt Creator. Aplikace je dodávána pod GPLv3 licencí.

C.2 Dokumentace

Nejedná se o běžný program. Proto doporučuji věnovat dostatečné úsilí dokumentaci zdrojového kódu, hlavně kolem simulace nebo kolem na první pohled nestandardních rozhodnutí. Tyto části kódu jsou ve větší míře okomentované přímo v kódu.

Každá třída obsahuje základní popis k čemu slouží. Pro každou metodu je zaznamenaný stručný popis (bude zobrazený v dokumentaci), dále delší popis chování. V případě vstupních parametrů jejich názvy a vlastnosti. Výstupní parametr je okomentovaný včetně toho, co přesně vrací.

```
/*!
 * \brief Method for loading data from file into drx object.
 *
 * It also checks if microstructure is valid. When
 * microstructure isn't complete it won't load data
 * and tells you what is wrong.
 *
 * \param fileName path to configuration file
 *
 * \return retuns true if it succesfully loaded microstructure
 * otherwise it return false
 */
bool loadIntoCA(QString fileName);
```

Listing C.1: Ukázka okomentování metody

```
int stepCounter_;           ///< Step of CA
```

Listing C.2: Ukázka okomentování proměnné

Pro vytvoření dokumentace stačí spustit příkaz

```
$ doxygen Doxyfile
```

a dokumentace se vygeneruje do složky `doc`. Je k tomu zapotřebí mít tento program nainstalovaný[23]. Pro tvorbu grafů závislostí je zapotřebí mít nainstalovaný program Graphviz[24]. Pro zobrazení dokumentace v této složce vyhledáme soubor `index.html` a spustíme v libovolném prohlížeči.

C.3 Konvence

C.3.1 Třídy

Jednotlivé třídy jsou rozdělené do `*.h` a `*.cpp` souboru. V `*.h` souboru se nachází deklarace jednotlivých metod, proměnných a jejich dokumentace. V souboru `*.cpp` se nachází jejich definice.

Privátní třídní proměnné končí postfixem `_`, např. `bool nucleated_;`. Díky tomu je jednoduché poznat, pokud pracujeme s privátní proměnnou dané třídy. Názvy tříd začínají velkým písmenem a pokračují malým. Každé další slovo v názvu třídy je taktéž velkým, např. `class CellAutomaton`.

C.3.2 Metody

Metody začínají malým písmenem a každé další slovo je velkým, např. `int getNumberOfGrains() const;`. Settery začínají prefixem `set`, gettery prefixem `get`. Je vhodné názvy metod zkracovat, ale ne na úkor pochopení toho,

[23] `http://www.stack.nl/~dimitri/doxygen/`
[24] `http://www.graphviz.org/`

co metoda dělá. V případě metody, která vrací nějaký typ se snažím jméno pojmenovat podle toho, co vrací.

C.4 Rozdělení tříd

Třídy a jejich chování se dá najít v dokumentovaném kódu a v práci v sekci návrh. Z pohledu dalšího rozšíření o grafy je důležitá třída `AbstractCurve`, což je abstraktní třída pro všechny grafy. Pro přidání dalšího grafu je zapotřebí vytvořit novou třídu, která bude potomkem právě této třídy. V případě více stavů simulace (mimo SRX,DRX,nucleated) stačí přidat v třídě `RecrystallizationHandler` nový stav do výčtového typu `States`, a v této třídě v metodě `evaluateNextTick()` přidat další větev podle nového stavu. Celulární automat se nachází v třídě `CellAutomaton`.

C.5 Kompilace a nasazení

C.5.1 Windows

V případě, že pro vývoj je použito programu Qt Creator, je kompilace jednoduchá. V programu otevřeme projekt vybráním `drx.pro` souboru. Tento soubor slouží pro specifikaci programu `qmake` pro jednotlivé platformy. Po otevření můžeme začít programovat. Když budeme chtít aplikaci nasadit, tak přepneme mód sestavování z **Ladění** na **Nasadit**. Po sestavení programu se nám vytvoří nová složka, která bude mít v názvu `*release*`. V této složce se bude nacházet sestavená verze programu. Abychom mohli tuto aplikaci spustit, je zapotřebí přibalit k ní všechny používané knihovny. Toho lze jednoduše docílit programem `windeployqt.exe`, který se nachází v složce `„Qt\version-of-Qt\compiler-type\bin\"`. Tento program je nejjednodušší zkopírovat do složky s programem a spustit v příkazovém řádku jako

```
windeployqt.exe .
```

a dojde k přidání všech potřebných Qt knihoven. Je však zapotřebí přidat dodatečné knihovny. Které jsou v případě prostředí MinGW `LIBSTDC++-6.DLL, LIBWINPTHREAD-1.DLL, LIBGCC_S_DW2-1.DLL`. Tyto knihovny se dají ve stejné složce jako `windeployqt.exe`. Pro bližší informace doporučuji stránku [25].

[25] `http://doc.qt.io/qt-5/windows-deployment.html`

C.5.2 Linux

Pro linux je situace jednodušší. Pokud máme nainstalovaný Qt ve verzi 5.7.0. a vyšší, qmake a g++, tak stačí následující sekvence příkazů:

```
$ qmake
$ make
$ ./DRX
```

Druhou možností je využít programu **linuxdeployqt**[26] a postupovat podle návodu na stránkách [27].

[26] `https//github.com/probonopd/linuxdeployqt`

[27] Tuto možnost jsem ale nestihl plně otestovat. Chci tím ale ukázat, že je i jiná možnost nasazení.

Příloha D

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `attachments` ..... adresář obsahující doplňující soubory
  - `User's manual` ..... uživatelská příručka v anglické verzi
  - `Programmer's manual` ..... programátorská příručka v anglické verzi
- `doc` ..... adresář s dokumentací
  - `index.html` ..... hlavní stránka dokumentace
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `DRX` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `BP_Tkac_Jakub_2017.pdf` ..... text práce ve formátu PDF
  - `zadani-BP_Tkac_Jakub_2017.pdf` ..... zadání práce ve formátu PDF