ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	GUI pro vizualizaci a editaci struktury krystalů
Student:	Bc. Martin Jelínek
Vedoucí:	Ing. Ivan Šimeček, Ph.D.
Studijní program:	Informatika
Studijní obor:	Systémové programování
Katedra:	Katedra teoretické informatiky
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Nastudujte problematiku popisu krystalu parametry. Dále nastudujte vnitřní architekturu a vstupní parametry programu Jana2006 [1]. Navrhněte a implementujte program, který ze vstupních dat v nativním formátu programu Jana2006 načte strukturu krystalu, zobrazí ji a umožní její editaci. Změny program uloží ve stejném formátu. K implementaci použijte jazyk C/C++ a technologii OpenGL. Navrhněte a implementujte vhodné kódování souborů, které by umožňovalo i kompresi dat (hlavně pro vstupní/výstupní soubory se strukturou a parametry krystalu). Otestujte a porovnejte implementaci s již existujícími programy (hlavně Diamond, případně dalšími: Atoms, Platon, ...).

Seznam odborné literatury

[1] http://jana.fzu.cz/

L.S.

doc. Ing. Jan Janoušek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 8. února 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra teoretické informatiky

Diplomová práce

GUI pro vizualizaci a editaci struktury krystalů

Bc. Martin Jelínek

Vedoucí práce: doc. Ing. Ivan Šimeček, Ph.D.

15. února 2017

Poděkování

Děkuji RNDr. Václavu Petříčkovi, Csc. a vedoucímu své diplomové práce doc. Ing. Ivanu Šimečekovi, Ph.D. za trpělivost a za jejich drahocenný čas, který mi věnovali. Dále děkuji rodině za jejich podporu a trpělivost.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) 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, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 15. února 2017 .....

Odkaz na tuto práci

Jelínek, Martin. *GUI pro vizualizaci a editaci struktury krystalů.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato práce se zabývá zobrazením struktury krystalu ve 3D prostoru a kódováním souboru, který obasahuje vstupní data. Práce má rozšířit dosud používaný program JANA2006 a navrhnout vhodnější kódování pro vstupní soubory JANA2006. Tato práce poskytuje chronologický vývoj statistického kódování a slovníkové komprese a principy, na kterých jsou tyto metody založeny. Dále tato práce popisuje systémy pro tvorbu grafických aplikací a technologie pro tvorbu GUI. V této práci popisujeme navržené kódování pro strukturu krystalu a implementaci programu pro zobrazení molekuly krystalu ve 3D.

Klíčová slova Struktura krystalu, krystalografie, molekula, atom, kódování, komrese, JANA2006

Abstract

This thesis deals with visualisation of crystal structure in 3D and coding of files that provide the structure. The whole project should extend JANA2006 program and improve crystal structure encoding of JANA2006 input files. This paper provides chronological overview of statistical and dictionary methods of data compresion and gives brief summary of the methods. Furthermore, this

thesis provides information about the enviroments for developing graphics applications and the technologies for GUI programming. In addition, this paper provides information about proposed crystal structure encoding and about the implementaion of the program for crystal structure visualisation in 3D.

Keywords Structure of crystal, crystallography, molecule, atom, coding, compression, JANA2006

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Hlavním cílem této diplomové práce je navrhnout a implementovat program, který zobrazí strukturu molekuly krystalu, a navrhnout kódování souborů, které obsahují strukturu molekuly krystalu. V programu bude možné editovat parametry struktury a strukturu uložit ve formátu používaném programem JANA2006. Tato práce dále pojednává o metodách kódování a metodách komprese dat, a také pojednává o vývoji OpenGL a dalších technologií pro tvorbu grafických programů. V práci dále rozebíráme základní cíle strukturní analýzy krystalu a program JANA2006, který je vyvíjen Fyzikálním ústavem Akademie věd České republiky.

Zobrazením struktury krystalu se myslí zobrazení atomů propojených vazbami ve 3D prostoru.

Komprese souborů se stala za poslední léta velmi využívanou technikou. K jejímu rozšíření přispěl exponenciální růst využití počítačů a s tím exponenciální růst množství dat[3]. Ke kompresi se mohou použít statistické metody a slovníkové metody. Existují ovšem i metody, které jsou využívané speciálně ke kompresi zvuku, videí nebo obrázků. Tyto metody ovšem mohou být často ztrátové. V této práci se budeme věnovat hlavně metodám slovníkovým a statistickým.

Strukturní analýza krystalu se zabývá určením poloh atomů v krystalové mřížce na základě experimentálně určených intenzit difrakčních maxim. Ke zpracování naměřených hodnot je možné použít program JANA2006. JANA2006 je program vyvíjený Fyzikálním ústavem Akademie věd České republiky a využívaný mnoha institucemi ve světě. Program JANA2006 zatím neobsahuje možnost zobrazení ve 3D prostoru a pro zobrazení se používají programy třetích stran. Soubory, s nimiž program JANA2006 pracuje, a které obsahují strukturu krystalu, jsou uloženy pouze v textové podobě.

Tato diplomová práce je dále členěna takto: V první kapitole popíšeme čemu se strukturní analýza krystalu věnuje a parametry, s kterými počítá a jaký mají tyto parametry význam. V druhé kapitole pojednáváme o metodách kódování a komprese. Zmiňujeme zde zvláště statistické metody a slov-

níkové metody. Ve třetí kapitole této diplomové práce popisujeme program JANA2006 a externí programy, které se se systémem JANA2006 používají. Ve čtvrté kapitole píšeme o technologiích pro grafické zobrazování a pro tvorbu grafického rozhraní. Jsou mezi nimi technologie OpenGL a framework QT. V páté kapitole ukážeme strukturu souborů, se kterými program JANA2006 pracuje, a dále ukážeme, proč programy třetích stran, které se používají k zobrazení molekuly ve 3D, zcela nevyhovují. V šesté kapitole této diplomové práce navrhneme kódování, které by bylo vhodné pro soubory obsahující strukturu krystalu a popíšeme samotnou realizaci programu pro zobrazení struktury molekuly krystalu a realizaci kódování. V sedmé kapitole shrnujeme splnění zadání a diskutujeme navržené kódování a program pro zobrazení struktury krystalu ve 3D prostoru.

Kapitola 1

Strukturní analýza krystalu

V této kapitole popíšeme čemu se krystalografie věnuje. Dále v této kapitole popíšeme geometrii krystalové buňky a řekneme, co je to operace symetrie a jak symetrii rozlišujeme.

Krystalografie je odvětví vědy studující krystaly. Krystaly se skládají z atomů, molekul a iontů[4], které k sobě pasují v opakujících se vzorcích, které se nazývají základní nebo elementární buňka nebo mřížka. Tato buňka se co do rozměru, obsahu a jeho uspořádání pravidelně opakuje ve třech směrech[5].

Tato kapitola je dále členěna takto. V první části kapitoly popíšeme geometrii krystalové buňky. V druhé části této kapitoly vysvětlíme, co je to symetrie a řekneme, které druhy symetrie rozlišujeme. Ve čtvrté a páté části této kapitoly popisujeme, co je to rentgenová difrakce, a co je to prášková difrakce. V této kapitole čerpáme ze skript [5] a z přednášek [6].

1.1 Geometrie krystalové buňky

Ideální krystalová struktura neboli ideální krystal je dokonalý, neporušený a nekonečně periodický. V ideálním krystalu můžeme vybrat nejjednodušší potřebný motiv - hmotnou bázi (atom, molekulu, skupinu atomů apod.). Pravidelným opakováním hmotné báze o konstantní posun - translaci (viz obrázek 1.1), vybudujeme trojrozměrný periodický vzor. Velikost této translace je ve třech směrech X, Y, Z obecně různá.

Při popisu ideální krystalové struktury nahrazujeme hmotnou bázi jedním bodem. Vzniklá množina, ve které má bod stejné okolí, se nazývá prostorová mřížka. V mřížce jsou všechny uzly ekvivalentní a z každého uzlu můžeme vést množinu translačních vektorů T_i. Z této množiny můžeme vybrat tři nekomplanární základní vektory a, b, c. Tyto vektory nazýváme základní translace a určují tvar základního rovnoběžnostěnu - elementární buňky. Elementární buňka má délky hran a, b, c a úhly jimi sevřené α, β, γ. Tyto parametry se nazývají mřížkové parametry.

K vyjádření polohy atomu, respektive libovolného bodu v buňce, se v krystalografii používají frakční, zlomkové, souřadnice x, y, z (značí se malými písmeny). Tyto souřadnice představují zlomky hran buňky. Například frakční souřadnice $[0.1; 0.5; 0.75]$ znamená, že x má jednu desetinu délky a, y má jednu polovinu délky b a z má tři čtvrtiny délky c. Polohový vektor $\mathbf{r}$ potom spojuje libovolný počátek souřadného systému v buňce s libovolným bodem o souřadnicích x, y, z přičemž platí[5]:

$$\mathbf{r} = \mathrm{x}\mathbf{a} + \mathrm{y}\mathbf{b} + \mathrm{z}\mathbf{c}$$

Pro převedení frakčních souřadnic na kartézské, ortogonální, se používá vztah[6]:

$$\begin{matrix} \mathbf{e}_1 = \mathbf{a}/\left|\mathbf{a}\right| \\ \mathbf{e}_3 = \mathbf{a}\times\mathbf{b}/\left|\mathbf{a}\times\mathbf{b}\right| \\ \mathbf{e}_2 = \mathbf{e}_1\times\mathbf{e}_3 \end{matrix} \implies \begin{pmatrix} \mathrm{x}_o \\ \mathrm{y}_o \\ \mathrm{z}_o \end{pmatrix} = \begin{pmatrix} \mathrm{a} & \mathrm{b}\cos\gamma & \mathrm{c}\cos\beta \\ 0 & \mathrm{b}\sin\gamma & \mathrm{c}\xi \\ 0 & 0 & \mathrm{c}\eta \end{pmatrix} \begin{pmatrix} \mathrm{x} \\ \mathrm{y} \\ \mathrm{z} \end{pmatrix} = \mathbf{T} \begin{pmatrix} \mathrm{x} \\ \mathrm{y} \\ \mathrm{z} \end{pmatrix}$$

kde

$$\xi = \frac{\cos\alpha - \cos\beta\cos\gamma}{\sin\gamma}$$

$$\eta = \sin\beta\sqrt{1 - \xi/sin\beta}$$

Trojúhelníková matice $\mathbf{T}$ pro převod z frakčních souřadnic na ortogonální se odvozuje takto:

$$\begin{pmatrix} \mathrm{x} \\ \mathrm{y} \\ \mathrm{z} \end{pmatrix} = \mathbf{x} = \mathrm{x}_o\mathbf{e}_1 + \mathrm{y}_o\mathbf{e}_2 + \mathrm{z}_o\mathbf{e}_3 = \mathrm{x}\mathbf{a} + \mathrm{y}\mathbf{b} + \mathrm{z}\mathbf{c}$$

Vynásobíme celou rovnici $\mathbf{e}_1$:

$$\mathrm{x}_o = \mathrm{x}\left(\mathbf{a}\cdot\mathbf{e}_1\right) + \mathrm{y}\left(\mathbf{b}\cdot\mathbf{e}_1\right) + \mathrm{z}\left(\mathbf{c}\cdot\mathbf{e}_1\right)$$

$$\mathbf{T}_{1,1} = \mathbf{a}\cdot\mathbf{e}_1 = \mathbf{a}\cdot\mathbf{a}/\left|\mathbf{a}\right| = \mathrm{a}$$

$$\mathbf{T}_{1,2} = \mathbf{b}\cdot\mathbf{e}_1 = \mathbf{b}\cdot\mathbf{a}/\left|\mathbf{a}\right| = \mathrm{b}\cos\gamma$$

$$\mathbf{T}_{1,3} = \mathbf{c}\cdot\mathbf{e}_1 = \mathbf{c}\cdot\mathbf{a}/\left|\mathbf{a}\right| = \mathrm{c}\cos\beta$$

Vynásobíme celou rovnici $\mathbf{e}_2$:

$$\mathrm{y}_o = \mathrm{x}\left(\mathbf{a}\cdot\mathbf{e}_2\right) + \mathrm{y}\left(\mathbf{b}\cdot\mathbf{e}_2\right) + \mathrm{z}\left(\mathbf{c}\cdot\mathbf{e}_2\right)$$

$$\mathbf{T}_{2,1} = \mathbf{a} \cdot \mathbf{e}_2 = \mathbf{a} \cdot \frac{[(\mathbf{a} \times \mathbf{b}) \times \mathbf{a}]}{|\mathbf{a} \times \mathbf{b}|\,|\mathbf{a}|} = 0$$

$$\mathbf{T}_{2,2} = \mathbf{b} \cdot \mathbf{e}_2 = \mathbf{b} \cdot \frac{[(\mathbf{a} \times \mathbf{b}) \times \mathbf{a}]}{|\mathbf{a} \times \mathbf{b}|\,|\mathbf{a}|} = \mathbf{b} \cdot \frac{[\mathbf{b}\,(\mathbf{a} \cdot \mathbf{a}) - \mathbf{a}\,(\mathbf{a} \cdot \mathbf{b})]}{|\mathbf{a} \times \mathbf{b}|\,|\mathbf{a}|} =$$

$$= \frac{\mathrm{a}^2\mathrm{b}^2 - (\mathbf{a} \cdot \mathbf{b})^2}{\mathrm{a}^2\mathrm{b}\sin\gamma} = \frac{\mathrm{b}\,(1 - \cos\gamma^2)}{\sin\gamma} = \mathrm{b}$$

$$\mathbf{T}_{2,3} = \mathbf{c} \cdot \mathbf{e}_2 = \mathbf{c} \cdot \frac{[(\mathbf{a} \times \mathbf{b}) \times \mathbf{a}]}{|\mathbf{a} \times \mathbf{b}|\,|\mathbf{a}|} = \mathbf{c} \cdot \frac{[\mathbf{b}\,(\mathbf{a} \cdot \mathbf{a}) - \mathbf{a}\,(\mathbf{a} \cdot \mathbf{b})]}{|\mathbf{a} \times \mathbf{b}|\,|\mathbf{a}|} =$$

$$= \frac{\mathrm{a}^2\,(\mathbf{b} \cdot \mathbf{c}) - (\mathbf{a} \cdot \mathbf{c})\,(\mathbf{a} \cdot \mathbf{b})}{\mathrm{a}^2\mathrm{b}\sin\gamma} = \mathrm{c}\frac{\cos\alpha - \cos\beta\cos\gamma}{\sin\gamma}$$

Vynásobíme celou rovnici $\mathbf{e}_3$:

$$\mathrm{z}_o = \mathrm{x}\,(\mathbf{a} \cdot \mathbf{e}_3) + \mathrm{y}\,(\mathbf{b} \cdot \mathbf{e}_3) + \mathrm{z}\,(\mathbf{c} \cdot \mathbf{e}_3)$$

$$\mathbf{T}_{3,1} = \mathbf{a} \cdot \mathbf{e}_3 = \mathbf{a} \cdot \frac{\mathbf{a} \times \mathbf{b}}{|\mathbf{a} \times \mathbf{b}|} = 0$$

$$\mathbf{T}_{3,2} = \mathbf{b} \cdot \mathbf{e}_3 = \mathbf{b} \cdot \frac{\mathbf{a} \times \mathbf{b}}{|\mathbf{a} \times \mathbf{b}|} = 0$$

$$\mathbf{T}_{3,3} = \mathbf{c} \cdot \frac{\mathbf{a} \times \mathbf{b}}{|\mathbf{a} \times \mathbf{b}|}$$

čitatel souvisí s objemem elementární buňky. Vyjádříme ho v libovolné kartézské soustavě. Pak platí:

$$\mathbf{c} \cdot (\mathbf{a} \times \mathbf{b}) = \sum_{ijk} \epsilon_{ijk} \mathrm{a}_i \mathrm{b}_j \mathrm{c}_k$$

kde:

1. $\epsilon_{ijk} = 1 \ldots ijk$ je cyklická permutace.
2. $\epsilon_{ijk} = -1 \ldots ijk$ není cyklická permutace.
3. $\epsilon_{ijk} = -1 \ldots i = j \vee i = k \vee j = k$

Tento výraz umocníme na druhou:

$$[\mathbf{c} \cdot (\mathbf{a} \times \mathbf{b})]^2 = \sum_{ijk} \epsilon_{ijk} \mathrm{a}_i \mathrm{b}_j \mathrm{c}_k \sum_{lmn} \epsilon_{lmn} \mathrm{a}_l \mathrm{b}_m \mathrm{c}_n$$

Po upravení dostaneme:

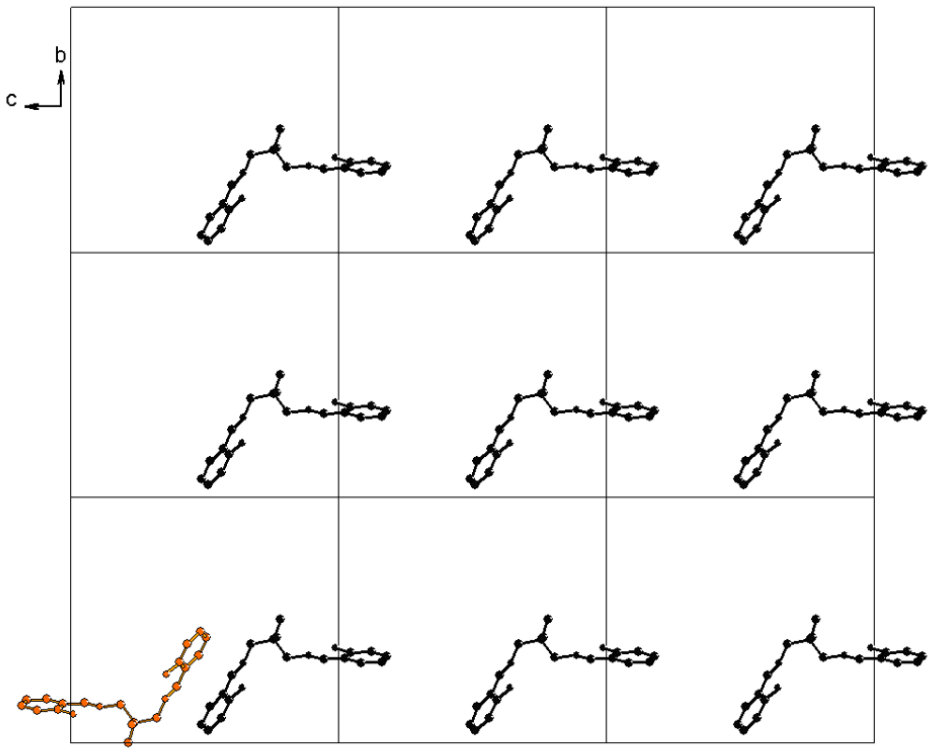
$$\mathbf{c}\cdot(\mathbf{a}\times\mathbf{b})^2 = (\mathbf{a}\cdot\mathbf{a})(\mathbf{b}\cdot\mathbf{b})(\mathbf{c}\cdot\mathbf{c}) - (\mathbf{a}\cdot\mathbf{a})(\mathbf{b}\cdot\mathbf{c})^2 - (\mathbf{b}\cdot\mathbf{b})(\mathbf{a}\cdot\mathbf{c})^2 -$$

$$(\mathbf{c}\cdot\mathbf{c})(\mathbf{a}\cdot\mathbf{b})^2 + 2(\mathbf{a}\cdot\mathbf{b})(\mathbf{a}\cdot\mathbf{c})(\mathbf{b}\cdot\mathbf{c}) =$$

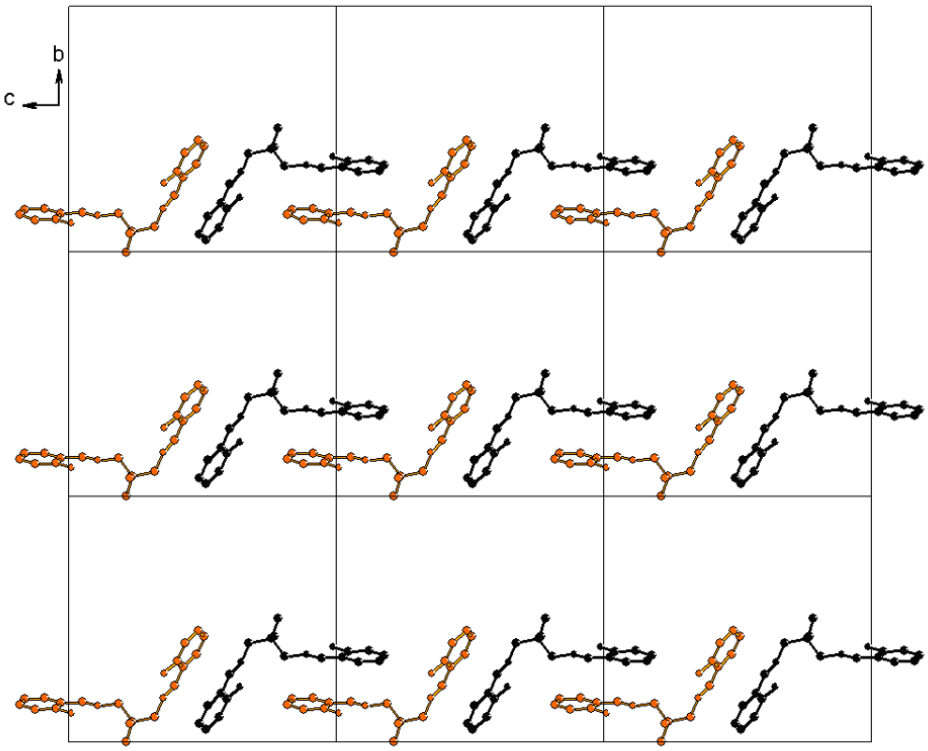
$$= \mathrm{a}^2\mathrm{b}^2\mathrm{c}^2\left(1 - \cos\alpha^2 - \cos\beta^2 - \cos\gamma^2 + 2\cos\alpha\cos\beta\cos\gamma\right)$$

$$\mathbf{c}\cdot(\mathbf{a}\times\mathbf{b}) = \mathrm{abc}\sqrt{1 - \cos\alpha^2 - \cos\beta^2 - \cos\gamma^2 + 2\cos\alpha\cos\beta\cos\gamma}$$

Obrázek 1.1: Na obrázku je zobrazena ukázka translační symetrie. Operace symetrie zobrazila molekuly do sousedních mřížek. Obrázek jsme čerpali z [1].



Obrázek 1.2: Na obrázku je ukázka rotační symetrie. Operace rotační symetrie otočila molekulu v mřížce. Obrázek jsme čerpali z [1].

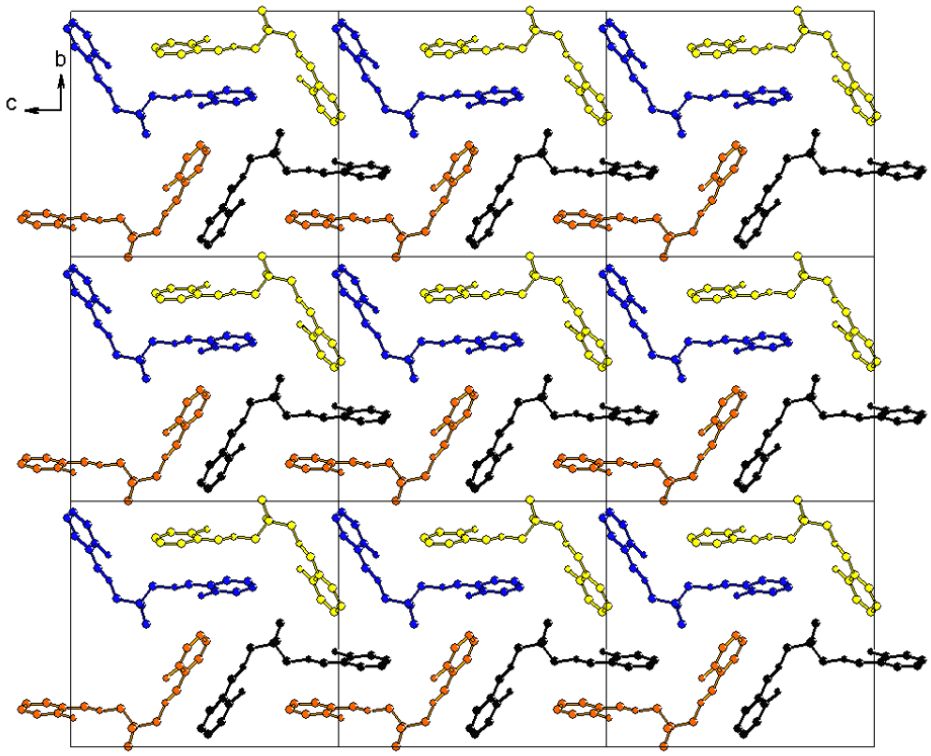


1.2 Symetrie

Ideální krystalická struktura má jednu důležitou vlastnost a to, že je symetrická, souměrná. Symetrie se rozlišuje na translační a rotační. Translační symetrie je pro krystal nutná (viz obrázek 1.1), rotační je pro krystal možná (viz obrázek 1.2 a obrázek 1.3). Pro popis symetrie se zavádí pojmy prvek a operace symetrie. Operací symetrie nazýváme každou geometrickou transformaci, která nemění strukturu krystalu. Operace symetrie zobrazí molekulu do stejné nebo odlišné polohy a přitom nezmění strukturu krystalu. Prvky symetrie nazýváme geometrické prvky (např. přímka, rovina), vzhledem ke kterým se provádí operace symetrie. Transformace lze popsat analytickými rovnicemi, které určují vztahy mezi souřadnicemi bodu P $\begin{bmatrix} \mathrm{x} & \mathrm{y} & \mathrm{z} \end{bmatrix}$ před transformací a bodu P' $\begin{bmatrix} \mathrm{x}' & \mathrm{y}' & \mathrm{z}' \end{bmatrix}$ po transformaci.

Při operaci rotace otáčíme objekt okolo osy, která prochází, nebo neprochází objektem. Operace produkuje ekvivalentní objekt, nebo objekt s odliš-

Obrázek 1.3: Obrázek zobrazuje další možné rotační symetrie. Operace rotační symetrie otočila molekulu v mřížce do různých pozic. Obrázek jsme čerpali z [1].



nou polohou od původního objektu. Četnost rotační osy udává hodnotu n (tzn. $360/n$), která v důsledku translační periodicity ideální krystalické struktury není libovolná. V krystalech se mohou vyskytnout pouze osy jednočetné (rotace o 360°), dvojčetné (rotace o 180°), trojčetné (rotace o 120°), čtyřčetné (rotace o 90°) a šestičetné (rotace o 60°)[5].

Při operaci zrcadlení můžeme vést objektem rovinu tak, že levá polovina tvoří zrcadlový obraz pravé, potom má objekt rovinu symetrie procházející jeho středem. Pokud jsou dva objekty k sobě orientované jako zrcadlové obrazy, potom jsou symetrické podle roviny souměrnosti.

Při operaci inverze se objekt promítne přes střed symetrie. Každému bodu ležícímu na přímce procházející středem objektu, tedy i středem symetrie, musí odpovídat bod na protilehlé straně přímky. Dva objekty jsou vůči sobě orientovány přes střed symetrie, je-li jeden středově symetrickým obrazem druhého. Inverze stejně jako zrcadlení převádí pravé objekty v levé a naopak.

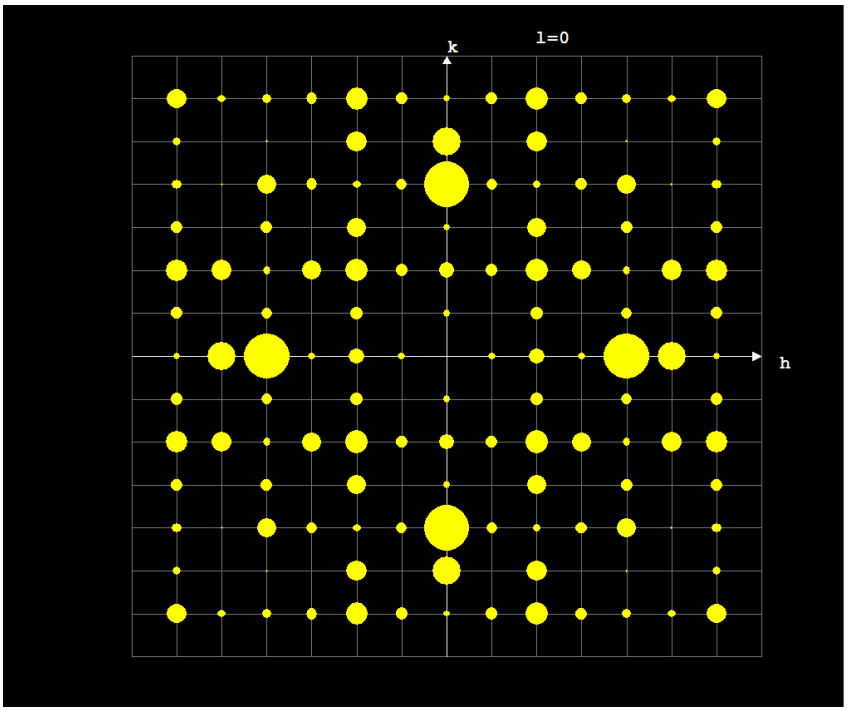
Další operace symetrie jsou operace rotační inverze, kde se kombinuje otá-

čení s inverzí, a operace na skluzné rovině, kde se kombinuje operace zrcadlení. Další operací je operace šroubové osy, kde se kombinuje otáčení se zlomkovou (nemřížkovou) translací. Pro více informací o symetrii viz [7, 5].

1.3 Difrakce

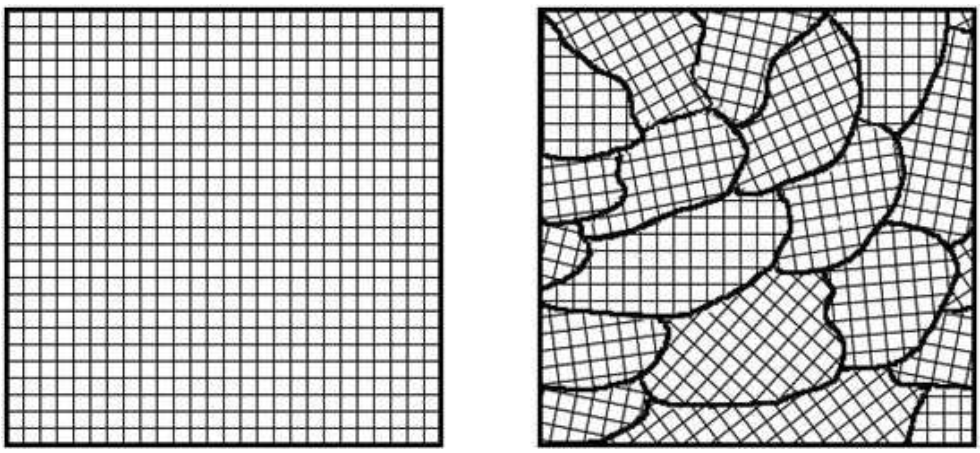
Při interakci rentgenového zařízení s pevnou látkou, mezi vlnami rozptýlenými periodicky trojrozměrnou mřížkou atomů nastává pozorovatelný interferenční jev. Tento jev se projevuje v určitých směrech koncentrací rozptýlené energie – vznikem difrakčního obrazu, který je složen z ostře ohraničených difrakčních stop. V případě, že analyzovanou látkou je monokrystal, difrakční stopy (často nazývané reflexe) jsou pravidelně rozmístěné v prostoru a jejich poloha vzhledem k reciproké bázi je popsána třemi celými čísly h, k, l. V případě zaznamenání na plochou fotografickou desku, dnes nahrazenou plošným detektorem, můžeme získat obrázek 1.4. Každá difrakční stopa má obecně jinou intenzitu a tyto informace jsou výchozím bodem pro řešení krystalové struktury. Difrakční obraz odráží jak translační symetrii krystalu tak i rotační symetrii krystalu. Translační symetrie souvisí s pravidelným umístěním difrakčních maxim v prostoru a rotační symetrie platí i pro difrakční obraz.

Obrázek 1.4: Na obrázku jsou graficky znázorněné difrakční stopy, z kterých se odvozují polohy atomů krystalu. Obrázek jsme čerpali z [1].



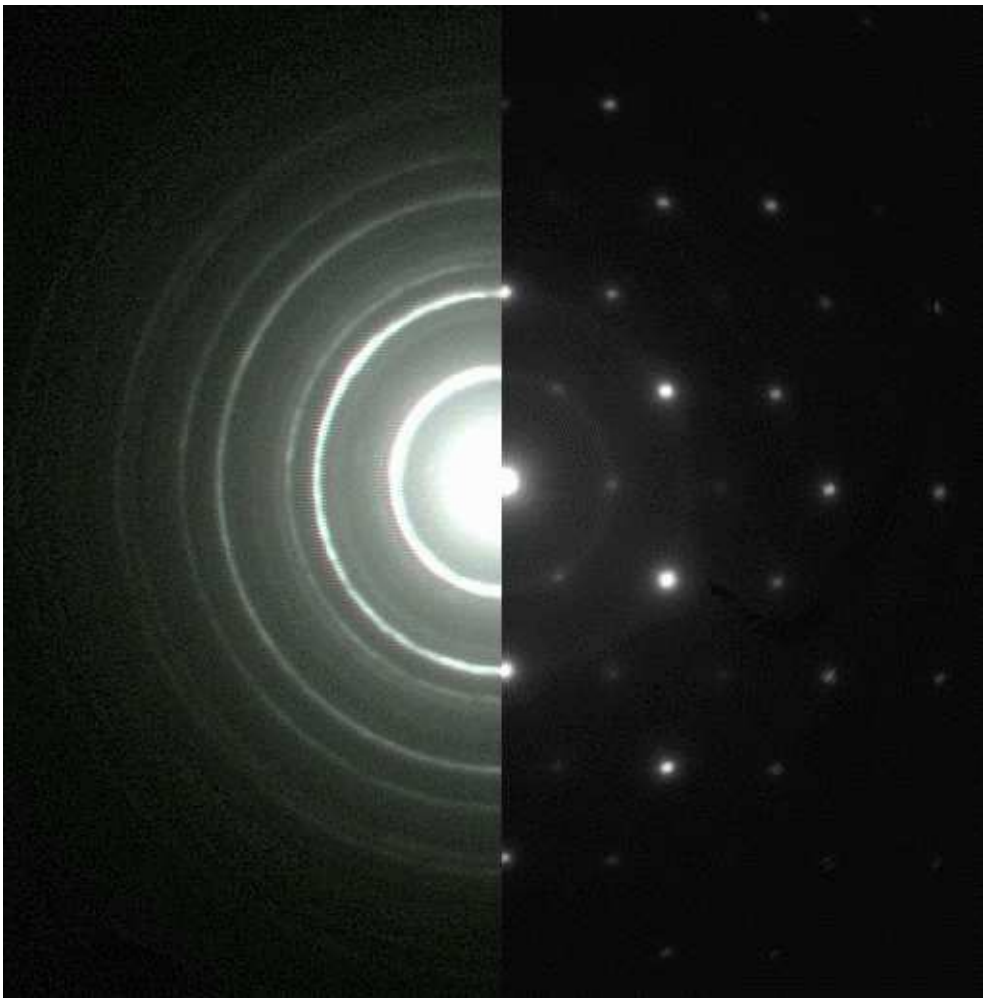
Z analytického hlediska mají největší význam difrakční metody, při kterých rentgenové záření difraktuje na monokrystalickém vzorku. Další difrakční metoda je, při které rentgenové zařízení difraktuje na polykrystalickém vzorku. Tyto metody se souhrnně označují jako práškové nebo polykrystalické, protože vzorek je nejčastěji studován ve formě prášku. Vzorek ovšem může být i kompaktní polykrystalický materiál. Obrázek který při monokrystalické difrakci je bodový, stává se při polykrystalové difrakci čárový (viz obrázek 1.6). Práškový vzorek si můžeme představit z mnoha náhodně orientovaných monokrystalů (viz obrázek 1.5).

Obrázek 1.5: Na obrázku je zobrazen monokrystal a polykrystal.



Dalšími typy záření, která se mohou použít pro difrakci jsou elektronové záření a neutronové záření. Elektronová difrakce podobně jako rentgenová difrakce sleduje elektronový obal, neutronová difrakce sleduje polohy jader atomu. Pro více informací o difrakci viz [7].

Obrázek 1.6: Na obrázku je zobrazen monokrystalová difrakce a polykrystalová difrakce.



Kapitola 2

Komprese a kódování

Komprese a kódování se stalo velmi důležitou disciplínou. Přispěl k tomu vývoj počítačů a jejich rozšíření. Komprese dat byla ještě v šedesátých letech dvacátého století téměř neznámou disciplínou. Komprese dat je založená na teorii informace. Důležitým výsledkem teorie informace je koncept entropie. Výsledkem je, že data nemohou být komprimována pod úroveň entropie. Při kompresi dat se můžeme pouze přiblížit na úroveň entropie[3].

Tato kapitola je dále členěna takto. V první části je vysvětleno, co je to entropie, jak se počítá a jaké typy entropie se používají. V druhé části této kapitoly popisujeme statistické metody kódování a komprese. Ve třetí části této kapitoly popisujeme Slovníkové metody komprese. Čtvrtá část obsahuje shrnutí této kapitoly. V této kapitole čerpáme z přednášek pana prof. Ing. Jana Holuba, Ph.D.[8] a publikace Davida Salmona[3].

2.1 Teorie informace a entropie

Vyčíslení informace je založeno na pozorování, že obsah informace je ekvivalentní množství náhody, nejednoznačnosti, ve zprávě. Teorie informace měří entropii zprávy průměrným počtem bitů, nezbytných k jejímu zakódování při optimálním kódování. Entropie zprávy ze zdroje $X = x_1, x_2 ... x_n$, kde pravděpodobnost zprávy x_i je p_i počítáme takto[9]:

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i.$$

Dále existuje empirická entropie k-tého řádu. Tato entropie závisí na symbolu a v jakém kontextu se symbol nachází. Empirickou entropii 0-tého řádu počítáme jako:

$$H_0(T) = -\sum_{a \in S} \frac{n_T^a}{n} \log_2 \frac{n_T^a}{n},$$

kde $T \in S$ a S je konečná množina zdrojů a n_T^a je počet symbolů a ve zprávě T. Empirická entropie k-tého řádu se počítá takto:

$$H_k(T) = \frac{1}{n} \sum_{w \in S^k} |w_T| \, H_0 \left(w_T \right),$$

kde w je řetězec symbolů náležících do množiny S délky k a w_T je řetězec skládající se ze zřetězení symbolů následujících za řetězcem w ve zprávě T.

2.2 Statistické metody

Každy soubor obsahuje data o určitém statistickém rozdělení. Toto statistické rozdělení dat můžeme použít ke kompresi dat. Pro rovnoměrné rozdělení dat v souboru nedosáhneme žádné komprese. Vyhovující situace nastane, pokud je funkce statistického rozdělení zkosená (rozdělení je nerovnoměrné)[3]. V tomto případě dosáhneme komprese dat. V této sekci popisujeme tři druhy kódování. V první části popisujeme Eliasovi kódy γ, γ‘, δ, ω a ω‘ a s tím související kódy α, β a β‘. V druhé části této sekce popisujeme Huffmanovo kódování a adaptivní Huffmanovo kódování. Ve třetí části této sekce popisujeme aritmetické kódování.

2.2.1 Eliasovy kódy

Autorem Eliasových kódu je Peter Elias. Eliasovy kódy se používají pro reprezentaci celých čísel. Eliasovy kódy minimalizují délku malých celých čísel a jsou jednoznačně dekódovatelné. Mezi Eliasovy kódy patří kódy γ, γ‘, δ, ω a ω‘. Pro reprezentaci čísel se může použít unární kód, kód α. Dále existuje binární kód β. Kód β není ovšem jednoznačně dekódovatelný[10]. V druhé části ukážeme tvorbu binárního kódu β. Ve třetí části ukážeme tvorbu Eliasova kódu γ a γ‘. Ve čtvrté části ukážeme tvorbu Eliasova kódu δ. V páté části ukážeme tvorbu Eliasova kódu ω a ω‘.

2.2.1.1 Unární kód α

Unární kód je jednoduchá reprezentace kladných přirozených čísel. Unární kódování je optimální pro některá exponenciální rozdělení, ale není univerzální[10]. Unární kód α pro číslo i se tvoří takto:

$$\alpha \left(i \right) = 0^i 1$$

kde 0^i značí i nul za sebou (např. 0^2 značí 00). Unární rozdělení je optimální pokud pravděpodobnost čísla i je $p \left(i \right) = 2^{-i}$[8].

2.2.1.2 Binární kód β a β‘

Binární kód β je standardní binární kód. Pro binární kód beta platí:

$$\beta(0) = 0$$

$$\beta(1) = 1$$

$$\beta(2i + j) = /beta(i)\,/beta(j)$$

Například $\beta(5) = \beta(4+1) = \beta(2)\,\beta(1) = 101$. Kód β není ovšem jednoznačně dekódovatelný. Příkladem může být:

$$\beta(7) = \beta(1)\,\beta(1)\,\beta(1) = \beta(1)\,\beta(3) = \beta(3)\,\beta(1)$$

Kód β‘ je kód beta kde chybí první 1. Například kód β‘ $(5) = 01$.

2.2.1.3 Eliasův kód γ a γ‘

Eliasovi kódy γ a γ‘ jsou kombinací unárního kódu α a binárního kódu β. Eliasův kód γ a γ‘ čísla i je definován takto:

$$\gamma\text{‘}(i) = \alpha(|\beta(i)|)\,\beta\text{‘}(i)$$

kde $|\beta(i)|$ je délka binárního kódu čísla i, $\beta(i)$. Eliasovy kódy γ a γ‘ se liší v tom, jak jsou čísla reprezentována. Pro Eliasův kód γ je kód α prokládán kódem β‘ tak, že na lichých pozicích je binární číslice kódu α a na sudých pozicích je binární číslice kódu β‘ (viz tabulka 2.1).

Tabulka 2.1: Tabulka Eliasových kódů γ a γ‘

i	γ‘ (i)	$\gamma(i)$
1	1	1
2	01**0**	0**0**1
5	001**01**	0**00**1**1**
7	001**11**	0**1**0**1**1

Tučným písmem jsou označeny části β‘ kódů.

2.2.1.4 Eliasův kód δ

Eliasův kód δ je kombinací Eliasových kódů γ a β‘. Eliasův kód δ je definován takto:

$$\delta(i) = \gamma(|\beta(i)|)\,\beta\text{‘}(i)$$

kde $|\beta(i)|$ je délka binárního kódu čísla i, $\beta(i)$.

2.2.1.5 Eliasův kód ω a ω‘

Eliasův kód ω je tvořen skupinami binárních kódů β končícími číslem 0. Skupina binárního kódu $\beta(i)$, která je první zprava, je samotné číslo i, kromě případu, kdy se $i = 1$ (viz tabulka 2.2). Algoritmus tvorby Eliasova kódu ω je popsán algoritmem 1.

Input: Číslo i.
Output: Kód Ω.
Result: Eliasův kód ω čísla i.
$j \leftarrow 1$;
while $\lfloor \log_2 i \rfloor \geq 0$ **do**
  zapiš $\beta(i)$ na levou stranu Ω ;
  zkombinuj tyto symboly do binárního podstromu se společným kořenem AB;
  $i \leftarrow \lfloor \log_2 i \rfloor$;
end
return Ω

Algoritmus 1: Algoritmus konstrukce Eliasova kódu ω.

Eliasův kód ω‘ se tvoří podobně. Jednička je v Eliasově ω‘ tvořena dvěma nulami a nejmenší skupina je velká tři binární číslice (až na číslo jedna). Další rozdíl je, že první skupina může začínat nulou (viz tabulka 2.2).

Tabulka 2.2: Tabulka Eliasových kódů ω a ω‘

i	$\omega(i)$	ω‘ (i)
1	0	0 0
2	10 0	010 0
7	10 111 0	111 0
32	10 101 100000 0	101 100000 0

2.2.2 Huffmanovy kódy

Statické Hufmanovo kódování bylo představeno v roce 1952[11]. Adaptivní Huffmanovo kódování je dynamickou verzí statického Huffmanova kódování. Jeden z algoritmů pro tvorbu dynamického Hufmanova kódování byl vytvořen pány Fallerem, Gallagerem a Knuthem (viz algoritmus 3).

Tato část je dále rozdělena takto. V první části ukážeme jak funguje statické Huffmanovo kódování. V druhé části předvedeme, jak funguje algoritmus pro tvorbu adaptivního Huffmanova kódování.

2.2.2.1 Statické Huffmanovo kódování

Vstupem je množina symbolů o n znacích. Pro každý znak v množině máme jeho frekvenci výskytu, nebo jeho pravděpodobnost výskytu. Algoritmus zapisuje kódy znaků do kódového stromu, který generuje. Algoritmus používá četnost, nebo pravděpodobnost, k tvorbě binárního stromu, který po svém dokončení usnadňuje výpočet množiny prefixových kódů pro množinu symbolů. Myšlenkou tohoto algoritmu je, že symbolům s nízkou četností by měly být přiřazeny dlouhé kódy a symbolům s vysokou četností krátké kódy. Symboly s nízkou četností tak budou v binárním stromě níž než symboly s vysokou frekvencí.

Algoritmus začíná tím, že vybere dva symboly s nejnižší četností A a B. Tyto dva symboly jsou vloženy do nejnižšího patra stromu a je z nich vytvořen binární podstrom s kořenem AB. Oba symboly jsou smazány z množiny a jsou nahrazeny dočasným symbolem AB, který má četnost $f(AB) = f(A) + f(B)$. Celý algoritmus je popsán algoritmem 2.

Result: Strom pro tvorbu statického Hufmanova kódu.
$j \leftarrow 1$;
while $1 \leq n$ **do**
  vyber dva symboly z množiny s nejnižší četností a označ je A a B;
  zkombinuj tyto symboly do binárního podstromu se společným kořenem AB;
  přidej do množiny pomocný symbol AB jehož frekvence je $f(AB) = f(A) + f(B)$;
  odstraň symboly A a B z množiny;
  $n \leftarrow n - 1$;
end

Algoritmus 2: Algoritmus konstrukce stromu pro statické Huffmanovo kódování.

Kód je znaku přidělen tak, že se prochází binární strom. Každý posun ve stromě dolů přidává znaku jeden bit. Nulový bit je přidán, pokud se ve stromě odbočí vlevo, jedničkový bit se přidá pokud se ve stromě odbočí vpravo.

Dekódování probíhá podobně. Pokud přečteme nulový bit jdeme ve stromě doleva. Pokud přečteme jedničkový bit, jdeme ve stromě doprava. Pokud jsme v listu stromu, našli jsme znak, jehož kód jsme měli na vstupu.

2.2.2.2 Adaptivní Huffmanovo kódování

U statického Huffmanova kódování máme dvě možnosti. Buď budeme používat stále stejné kódy, což bude rychlé, ale nebude to efektivní, nebo budeme vždy procházet text dvakrát, abychom získali použité symboly a jejich četnosti v souboru, a poté zakódovali soubor. Tento způsob bude efektivní z hlediska kódování, ale bude pomalý. Huffmanovo adaptivní kódování potřebuje k za-

kódování souboru pouze jeden průchod. Kódy se během průchodu souboru mění.

V algoritmu se používá ESC symbol. Tento symbol by měl mít po celý běh algoritmu ve stromě četnost nula. Během běhu algoritmu se může stát, že se strom, který odpovídá definici Huffmanova stromu změní na strom, který neodpovídá definici Huffmanovu stromu. Toto se může stát z důvodu, že se změnila četnost symbolů.

Hufmanův strom definujeme takto: Huffmanův strom je takový binární strom, kde platí tato podmínka. Pokud je strom procházen zespodu nahoru, tak na každé úrovni stromu jsou všechny uzly seřazeny podle četnosti zleva doprava, od nejnižší četnosti po nejvyšší četnost.

Celý algoritmus je popsán algoritmem 3.

Data: |Začínáme s prázdným stromem.
Input: Vstupní text $T = t_1, t_2, ..., t_{|T|}$.
Output: Huffmanův kód H.
Result: Huffmanův kód vytovřený pomocí adaptivního Huffmanova kódování.

```
j ← 1;
while j ≤ |T| do
    načti symbol t_j; if t_j byl načten poprvé then
        zapiš kód c(ESC) symbolu a symbol t_j do H a vlož symbol t_j
        na konec stromu s četností jedna, čímž je symbolu přiřazen kód;
    else
        zapiš kód c(t_j) do H a zvyš četnost výskytu symbolu t_j o jedna;
    end
    uprav strom tak, aby to odpovídal definici Huffmanova stromu;
    j ← j + 1;
end
return H;
```

Algoritmus 3: Algoritmus adaptivního Huffmanova kódování.

Dekódování postupuje podobně jako u statického Huffmanova kódování. Čteme bity, a pokud narazíme na kód $c(ESC)$, tak načteme symbol a vložíme ho do Huffmanova stromu, zároveň ho vypíšeme na výstup a upravíme strom tak, aby to byl Huffmanův strom. Jinak vypíšeme na výstup dekódovaný znak x.

2.2.3 Aritmetické kódování

Aritmetické kódování je statistická metoda, která může kódovat symbol jako racionální číslo. Aritmetické kódování komprimuje řetězec znaků (jejichž pravděpodobnosti známe) tak, že počet bitů na symbol se rovná entropii. Aritmetické kódování komprimuje řetězec symbolů na řetězec bitů. Zkomprimovaná

data si můžeme představit jako číslo v intervalu $[0, 1)$. Každý symbol řetězce náleží podintervalu intervalu $[0, 1)$. Podinterval je vlastně podmnožina množiny, intervalu, $[0, 1)$.

Na vstupu algoritmu je list $S = \{x_1, x_2, ..., x_n\}$ a text $T = t_1, t_2, ..., t_{|T|}$ tokový, že $\forall i : t_i \in S$. Každý symbol x_k má pravděpodobnost $p(x_k)$ a kumulativní pravděpodobnost $cp(x_k)$. Kumulativní pravděpodobnost $cp(x_k)$ je definovánat takto:

$$cp(x_k) = \sum_{i=1}^{k-1} p(x_i)$$

Algoritmus aritmetického kódování je popsán zde 4.

Data: Kumulativní pravděpodobnosti $cp(x_k)$, pravděpodobnosti $p(x_k)$, množina $S = \{x_1, x_2, ..., x_n\}$.
Input: Aritmetický kód B a délku původního textu $|T|$.
Output: Dvojce $(|T|, B)$.
Result: Aritmetický kód B, který nese informce o původním textu.
$LOW \leftarrow 0$;
$RANGE \leftarrow 1$;
$j \leftarrow 1$;
while $j \leq |T|$ **do**
  $LOW \leftarrow LOW + RANGE * cp(t_j)$;
  $RANGE \leftarrow RANGE * p(t_j)$;
  $j \leftarrow j + 1$;
end
$B \leftarrow$ číslo mezi $[LOW, LOW + RANGE)$;
return $(|T|, B)$;

Algoritmus 4: Algoritmus aritmetického kódování.

Pro dekódování aritmetického kódu máme na vstupu list $S = \{x_1, x_2, ..., x_n\}$ a délku původního textu $|T|$ a aritmetický kód B. Pro každý symbol x_k známe pravděpodobnost $p(x_k)$ a kumulativní pravděpodobnost $cp(x_k)$. Algoritmus dekódování je popsaný algoritmem 5.

2.3 Slovníkové metody

Statistické kompresní metody využívají ke kompresi dat pravděpodobnost symbolů, a tak snižují redundanci dat. Slovníkové metody snižují redundanci tím, že hledají identické části dat. Například v anglickém textu se často setkáváme s řetězcem „the “. Slovníkové metody si ukládají takovéto výskyty do slovníku a budoucí výskyt takového řetězce nahrazují pozicí ve slovníku.

První slovníkové metody byly navrženy v sedmdesátých letech dvacátého století pány J. Zivem a A. Lempelem, kteří vyvinuli první slovníkové metody LZ77 a LZ78.

Data: Kumulativní pravděpodobnosti $cp(x_k)$, pravděpodobnosti $p(x_k)$, množina $S = \{x_1, x_2, ..., x_n\}$.
Input: Aritmetický kód B a délku původního textu $|T|$.
Output: Řetězec znáků T.
Result: Řetězec znáků T, který odpovídá znakům původního textu.

```
j ← 1;
LOW ← B;
while j ≤ |T| do
    if j < n then
        porovnej LOW s kumulativními pravděpodobnostmi tak, aby
        platilo cp(t_j) ≤ LOW < cp(t_j), čímž získáš t_j;
    end
    if j = n then
        porovnej LOW s kumulativními pravděpodobnostmi tak, aby
        platilo cp(t_j) ≤ LOW < 1, čímž získáš t_j;
    end
    na konec listu T přidej t_j;
    LOW ← (LOW − cp(t_j)) / p(t_j);
    j ← j + 1;
end
return T;
```

Algoritmus 5: Algoritmus dekódování aritmetického kódu.

Tato kapitola je dále členěna takto. V první části této kapitoly popisujeme kompresní slovníkovou metodu LZ77. V Druhé části této kapitoly popisujeme kompresní metodu LZSS, která vylepšuje metodu LZ77. Ve třetí části této kapitoly popisujeme kompresní slovníkovou metodu LZ78. Ve čtvrté části této kapitoly popisujeme slovníkovou metodu LZW, která vylepšuje metodu LZ78.

2.3.1 LZ77

Kompresní metoda LZ77 byla představena v roce 1977. Autory jsou J. Ziv a A. Lempel. Kompresní algoritmus LZ77 je využíván ve formátech .zip, .gzip a .pkzip.

Kompresní metoda LZ77 používá ke kompresi dat okénko, buffer, rozdělené na dvě části. Jsou to look-ahead buffer a search buffer. Look-ahead buffer je buffer na data, která budou zkomprimována a search buffer je buffer na data, která byla zkomprimována a ve kterém se hledá slovo. Search buffer představuje slovník. Grafické zobrazení okénka můžete vidět na obrázku 2.1. Velikost search bufferu je několik tisíc bytu a velikost look-ahead bufferu je několik desítek bytu. Výstupem algoritmu jsou tokeny. Tokeny jsou pro LZ77 trojice (i, j, a), kde i je index v search bufferu, j je počet symbolů ze search bufferu a a je následující symbol na vstupu.

Obrázek 2.1: Obrázek zobrazuje okénko, rozdělené na search buffer a look-ahead buffer

Symboly jsou vkládány ze vstupu do okénka zleva doprava. Každý symbol se posouvá v okénku. Symbol nejdříve projde look-ahead bufferem a pak search bufferem.

2.3.2 LZSS

Kompresní metoda LZSS je vylepšená verze kompresní metody LZ77. Autory jsou J. A. Storer a T. G. Szymanski. Kompresní algoritmus LZSS vylepšuje algoritmus LZ77 takto. Look-ahaed buffer je v cyklické frontě. Search buffer je v binárním vyhledávacím stromě. Pokud se uskuteční posun o k pozic, tak je ze stromu smazáno k uzlů a k uzlů je do stromu vloženo. Výstupní tokeny mají maximálně dvě části místo tří.

2.3.3 LZ78

Kompresní metoda LZ78 byla představena v roce 1978. Autory jsou J. Ziv a A. Lempel.

Kompresní metoda LZ78 používá ke kompresi rostoucí slovník. Slovník je reprezentovaný jako trie. Trie je prefixový strom. Výstupem algoritmu jsou tokeny. Tokeny jsou pro LZ78 dvojice (i, a), kde i je ukazatel do slovníku a a je následující znak.

Symboly jsou vkládány do slovníku, trie, čímž slovník roste. Každá hrana označuje symbol, po kterém se dostaneme do dalšího uzlu ve stromě. Každý uzel ve stromě má index. Při přečtení symbolu ze vstupu jsou dvě možnosti. První možnost je, že pro symbol vede hrana z uzlu, čímž se posuneme o úroveň výš ve stromě. Druhá možnost je, že pro symbol z uzlu žádná hrana nevede. Pak vypíšeme token s indexem uzlu a symbolem na výstup a symbol vložíme do slovníku za uzel, ve kterém jsme skončili hledání. Algoritmus začíná s prázdným slovníkem, to znamená, že strom má pouze kořen, který má index uzlu 0.

Pokud se vyčerpá všechna dostupná paměť z důvodu velikosti slovníku, existují čtyři možnosti, jak postupovat dále. První možnost je, že slovník už dále nerozšiřujeme. Druhá možnost je, že smažeme celý slovník a začneme tvořit slovník od začátku. Třetí možnost je, že smažeme nejméně používané uzly. Čtvrtá možnost je, že slovník dále nerozšiřujeme a monitorujeme kompresní poměr. Pokud se kompresní poměr sníží pod určitou hranici, tak smažeme celý slovník a začneme tvořit slovník od začátku[8].

2.3.4 LZW

Kompresní metoda LZW byla představena v roce 1984. Autorem je T. A. Welch. Kompresní metoda LZW je vylepšená metoda LZ78. Kompresní metoda LZW je oblíbenou metodou a je používána v mnoha aplikacích.

Kompresní metoda LZW používá stejně jako LZ78 rostoucí slovník. Slovník metody LZW je pro všechny symboly použité abecedy inicializovaný. Výstupem kompresní metody LZW jsou tokeny, které obsahují ukazatel do slovníku.

Řetězce ze vstupu jsou porovnávány se slovníkem. Stejně jako u LZ78 i u LZW se vytváří strom, kde hrany označují znaky vložené do slovníku a uzly jsou označeny indexy. Ze vstupu čteme symboly a posouváme se po nich po hranách. Pokud pro symbol z uzlu vede hrana, tak se posuneme do dalšího uzlu o úroveň výš ve stromě. Pokud hrana neexistuje, tak vytvoříme nový uzel a vedeme do něj hranu z uzlu, ve kterém jsme skončili. Tuto hranu označíme symbolem ze vstupu. Na výstup pak vypíšeme token s indexem uzlu.

Pokud se vyčerpá všechna paměť z důvodu velikosti slovníku, můžeme postupovat stejně jako u LZ78. Kompresní metoda LZW se pomalu adaptuje na vstup. Trvá dlouho než se do slovníku dostanou dlouhé řetězce[8].

Kompresní metoda LZW byla patentována v roce 1985. Kompresní metoda LZW je používána ve formátu obrázků .gif. Z důvodů problémů s licencí později vznikl formát .png, aby nahradil formát .gif.

2.4 Diskuse

V této kapitole byly představeny statistické a slovníkové kompresní metody. Existují další kompresní metody, jsou to například kontextové kompresní metody, které využívají to, že se symboly nejčastěji vyskytují v jistém kontextu. Příkladem kontextové metody je metoda PPM (viz [8]). Všechny představené metody v této kapitole jsou neztrátové. Mezi ztrátové kompresní metody patří například formát souboru JPG. Ztrátové metody se používají převážně tam, kde nám nevadí ztráta části informace, například obrázky, nebo zvuky. Tyto metody nejsou vhodné pro kompresi struktury krystalu.

Jednou z nejvíce používaných kompresních metod je metoda LZ77, kterou využívají formáty .zip, .gzip a .pkzip. Další v praxi používanou metodou je metoda LZW, která se používá ve formátu .gif. Pro porovnání v kapitole 7.2 jsme vybrali implementaci metody LZ77, která je použita ve formátu .zip, a implementaci LZW ze zdroje [12].

Eliasovy kódy jsou určeny ke kódování přirozených čísel, ale lze je také použít ke kódování celých čísel. Společnou vlastností Eliasových kódů je, že Eliasovi kódy malých čísel okolo nuly jsou kratší než Eliasovy kódy velkých čísel.

Soubory se strukturou krystalu, používané programem JANA2006, obsahují z 95% čísla s plovoucí řádovou (viz kapitola 5.3).

Eliasovy kódy nejsou vhodné pro kódování struktury krystalu, protože v souboru, ve kterém je popsána struktura krystalu, jsou obsaženy hlavně čísla s řádovou čárkou (viz kapitola 5.3). Pro samotné kódování znaků je vhodnější dynamické Huffmanovo kódování, protože při průchodu souboru počítáme četnost znaků a jsme schopni přiřadit kratší kód pro častěji se opakující znaky.

Slovníkové metody jsou universální metody pro kompresi dat. Tyto metody jsou vhodné pro kompresi souborů se strukturou krystalu.

Nevýhodou všech představených metod je, že soubor je nutný nejdříve dekódovat a potom je tento soubor možné načíst.

Kapitola 3

JANA2006

JANA2006 je volně dostupný systém programů pro řešení a upřesnění klasických, modulovanými i magentickými struktur získaných z rentgenové nebo neutronové monokrystalové/práškové difrakce nebo elektronové difrakce (viz kapitola 1.3). Systém JANA je vyvíjen více než třicet let. Vývoj systému začal jako nástroj pro zpracování modulovaných struktur. V dnešní době program pokrývá požadavky základní i pokročilé strukturální analýzy krystalu. Program je vyvíjen skupinou RNDr. Václavem Petříčkem, Csc., RNDr. Michal Dušekem, Csc. a Dr.rer.nat. Lukáš Palatinusem, Ph.D. z Fyzikálního ústavu Akademie věd České republiky. Systém JANA2006 pracuje hlavně se soubory .m40 a .m50. Strukturu a obsah těchto souborů popisujeme v páté kapitole.

Tato kapitola je dále členěna takto. V první části zmiňujeme historii systému JANA. V druhé části této kapitoly se zmíníme o technických vlastnostech systému JANA2006. Ve třetí části této kapitoly pojednáváme o programu externí programi pro zobrazení struktury krystalu ve 3D prostoru. Ve čtvrté části této kapitoly shrnujeme vlastnosti popsaných externích nástrojů. V této kapitole čerpáme z článku[2].

3.1 Historie systému JANA

JANA 2006 je aktuální verzí sytému JANA. První verze systému JANA byla používána v roce 1984. Původně byl program zamýšlen pro zpracování jednodymenzionálně nesouměřitelných modulovaných struktur. První zpracování modulované struktury pomocí sytému JANA bylo publikováno v roce 1984. Postupně se tento systém měnil z programu pro zpracování modulovaných struktur na obecný krystalografický systém pro analýzu struktury ve tří a více dimenzionálním prostoru.

V roce 1994 se program sloučil s nepublikovaným systémem SDS, také vyvíjeným RNDr. Václavem Petříčkem, Csc. a do systému bylo přidáno grafické rozhraní. Do roku 1998 byl tento systém považovaný za nástroj pro velmi specifické experimenty. Systém JANA1998 byl prvním systémem JANA, který byl

běžně používaný odbornou veřejností, a který měl relativně přátelské grafické uživatelské rozhraním.

Další veze systému JANA byla JANA2000. JANA2000 byla rozšířena o podporu práškových dat. Podpora práškových dat přilákala další uživatele a tím se JANA2000 stala dominantním nástrojem na poli modulovaných struktur.

JANA2006 vyšla v roce 2006. JANA2006 se snaží dosáhnout optimálního kompromisu mezi snadnou použitelností a možností řešit velmi složité struktury. Pro více informací o programu JANA2006 viz [2].

3.2 Technické údaje systému JANA2006

Program JANA2006 je napsáný v jazyce Fortran95 a má přibližně 200000 řádek. Veškerá grafika byla speciálně vytvořena pro systém JANA2006 a nemá žádné požadavky na externí grafické knihovny, jiné než základní funkce API. JANA2006 je distribuována jako zkompilovaný program pro operační systém Windows a může běžet na jakémkoliv počítači. Port pro operační systémy na bázi UNIX je plánovaný, ale není dokončený.

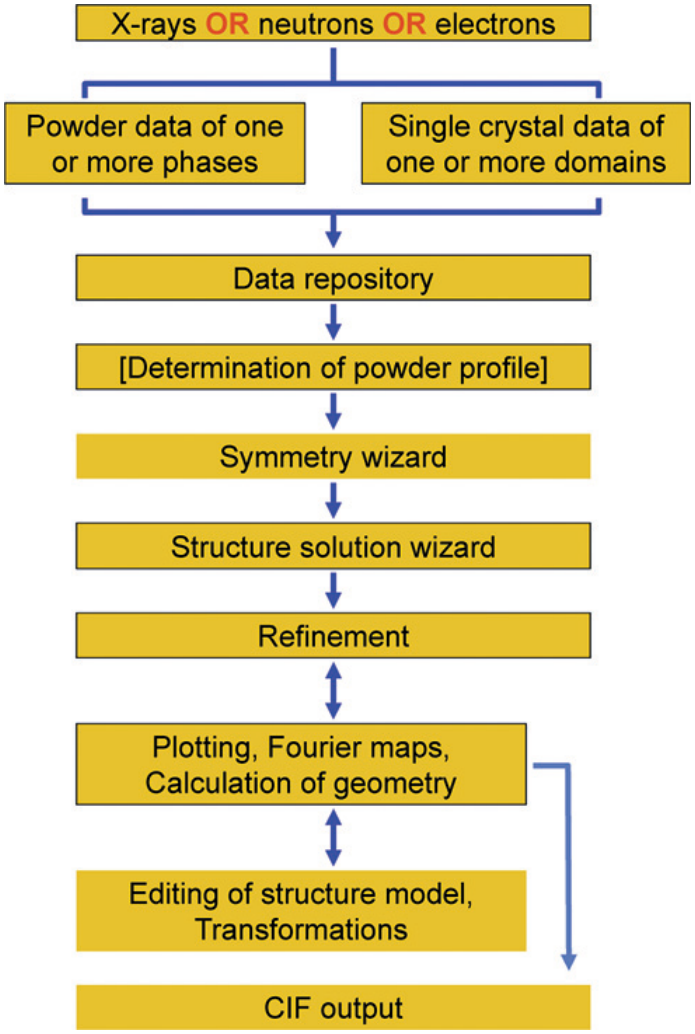
Systém JANA2006 může být aplikován k řešení jednoduchých i složitých struktur, jako jsou například modulované a magnetické struktury. Program nabízí zpracování obvyklých strukturálních parametrů, a také zpracování anharmonických teplotních parametrů(ADP). Program pracuje s nesouměřitelnými i se souměřitelnými modulovanými strukturami až do šesti rozměrů (pro více informací o těchto strukturách viz [13]). Program dále nabízí možnost zpracovat strukturu zároveň proti několika difrakčním data setům, které byly získané buď z monokrystalu, nebo z práškového vzorku. Tato metoda se nazývá *Joint refinement* (česky *Spojené zpracování*) (viz obrázek 3.1). Tato metoda je typicky používána ke kombinaci rentgenových dat monokrystalu a neutronových práškových dat za účelem využití různé citlivosti rentgenového a neutronového difraktometru na různé parametry struktury.

V posledních letech bylo nejvíce úsilí věnováno vyvinutí možnosti popisu magnetické struktury, která umožnila zpracování souměřitelných i nesouměřitelných magnetických struktur s jejich výchozí nukleární strukturou.

3.3 Externí nástroje pro zobrazení v 3D prostoru

JANA2006 neumí zobrazovat strukturu krystalu v 3D prostoru. Pro zobrazení struktury krystalu v 3D prostoru se používají programy třetích stran. Mezi tyto programy patří hlavně program Diamond, který se používá pro zobrazení molekuly v 3D prostoru, a Program Vesta, který se používá pro zobrazení elektronové hustoty struktury krystalu v 3D prostoru. Oba tyto programy jsou používané Fyzikálním ústavem Akademie věd České republiky.

Obrázek 3.1: Diagram zpracování data setů JANA2006 z různých zdrojů. Na diagramu je znázorněn postup zpracování dat. Výstupem je soubor CIF. Obrázek je z publikace [2].



3.3.1 Diamond

Program Diamond je používaný Fyzikálním ústavem Akademií věd České republiky pro zobrazení struktury krystalu v 3D prostoru. Program Diamon je vyvíjen společností CRYSTAL IMPACT. Program Diamond dovoluje číst data ve formátu .CIF. Formát souboru .CIF znamená Crystallographic Information File. Diamon slouží k vizualizaci struktury krystalu. Program Diamond dovoluje modelovat libovolnou část struktury krystalu[14]. Program Diamond dokáže využít symetrii k zobrazení celé struktury krystalu v jedné i více buňkách.

3.3.2 Vesta

Systém Vesta vznikl z dvou programů VICS a VEND. Pro tvorbu programu Vesta je použita technologie wxWidgets. Jedním z autorů programu je Koichi Momma. Program vesta je užíván Fyzikálním ústavem Akademie věd pro zobrazení elektronové hustoty struktury krystalu v 3D prostoru. Program Vesta dovoluje číst data v různých formátech. Jsou to například .CIF a .PDB[15].

3.3.3 Další nástroje

Tato část se věnuje dalším programům třetích stran pro zobrazení molekuly krystalu v 3D prostoru. Mezi tyto programy patří program Atoms a program PLATON.

- Nástroj PLATON je vyvíjen od roku 1980. Autorem nástroje PLATON je A.L. Spek z university Utrecht. Nástroj PLATON pracuje s formáty souborů, které obsahují souřadnice. Jsou .CIF, .PDB, .RES, .FDAT, .SPF. Dále pracuje s reflekčními soubory .HKL a .FCF. Program PLATON dokáže zobrazit strukturu krystalu v 3D prostoru.[16]
- Program Atoms je program na vizualizaci různých typů struktur atomů, jako jsou krystaly, polymery a molekuly. Nástroj Atoms dokáže zobrazit strukturu v 3D prostoru. Program pracuje s různými druhy typů souborů. Mezi tyto soubory patří .CIF a .PDB.[17]
- Program Mercury nabízí obsáhlou sadu nástrojů pro vizualizaci struktury v 3D prostoru. Systém Mercury je vyvíjen organizací CCDC (The Cambridge Crystallographic Data Centre). Program podporuje různé druhy souborů. Jsou mezi nimi například formáty .CIF a .PDB[18].

3.4 Vlastnosti externích nástrojů pro zobrazení v 3D prostoru

Tato část se věnuje shrnutí vlastností externích nástrojů pro zobrazení molekuly krystalu v 3D prostoru. Mezi výhody těchto programů patří to, že to jsou dlouhodobě vyvíjené nástroje s bohatým množstvím funkcí.

Mezi tyto funkce patří například zobrazení atomu, jako elipsoidy v závislosti na teplotních parametrech atomu. Dále animace vibrací atomů v závislosti na teplotních parametrech, či využití symetrie pro zobrazení celkové struktury krystalu v jedné či více buňkách. Některé představené nástroje, jako je program Vesta, dokáží zobrazit elektronovou hustotu struktury krystalu.

Všechny nástroje, představené v této části umí číst strukturu krystalu ve formátu .CIF(Crystallographic Information File). Představené programy neumí zacházet se soubory .m40 a .m50 (o souborech viz kapitola 5), se kterými pracuje program JANA2006.

Kapitola 4

Technologie

Tato kapitola pojednává o technologiích pro tvorbu grafiky a technologiích pro tvorbu grafického uživatelského prostředí. V první části této kapitoly pojednáváme o technologiích pro tvorbu grafiky, v druhé části této kapitoly pojednáváme o technologiích pro tvorbu grafického uživatelského prostředí.

4.1 Technologie pro tvorbu grafiky

V této části popisujeme technologie pro tvorbu grafiky. Pojednáváme zde o technologiích OpenGL, DirectX a Vulkán.

4.1.1 OpenGL

OpenGL je prostředí pro tvorbu přenosných, interaktivních 2D a 3D grafických aplikací. OpenGL bylo představeno v roce 1992. Od té doby se OpenGL stalo nejvíce užívaným a podporovaným 2D a 3D API. OpenGL je podporováno ve všech systémech založených na UNIX a je součástí operačních systémů Windows 95/98/2000/NT a MacOS. V této části čerpáme ze zdrojů [19, 20].

OpenGL bylo vytvořeno jako otevřená alternativa k Iris GL. Iris GL bylo proprietární grafické API. Mark Segal a Kurt Akeley napsali specifikaci OpenGL 1.0, kde se snažili formalizovat definici grafického prostředí a vytvořit platformě nezávislou implementaci. Iris GL mělo definice a fáze připojené pro všechny druhy objektů včetně materiálu, textury, texturovacího prostředí. OpenGL se vyvarovalo těmto objektům a používá postupné změny stavu s myšlenkou, že kolektivní změny mohou být zapouzdřené v zobrazovaném seznamu.

OpenGL prošlo mnoha revizemi. Verze OpenGL 3.0 přinesla koncepci zastaralosti. Tato koncepce označuje některé funkce jako subjekt k odstranění v příští verzi. Mezi funkce, které jsou označeny k odstranění patří například: Color index mode, Shading language 1.10 a Shading language 1.20, bitmapy,

široké čáry a další. Mnoho těchto vlastností je ve verzi 3.1 odstraněno až na funkci široké čáry.

Oficiální verze OpenGL, které vyšly do současnosti můžete nalézt v tabulce 4.1. OpenGL ES je API pro tvorbu grafických aplikací na vestavných systémech. OpenGL je spravované konsorciem KHRONOS GROUP.

Tabulka 4.1: Verze OpenGL[20]

verze	rok	poznámka
1.0	1992	první veze
1.1	1997	bylo přidáno například vertex array, polygon offset a logické operace
1.2	1998	byly přidány například 3D textury, BGRA formáty pixelu
1.2.1	1998	definuje se ARB concept rozšíření
1.3	2001	byly přidány například komprimované textury
1.4	2002	bylo přidáno například automatické generování mipmap
1.5	2003	byly přidány například funkce stínování
2.0	2004	přidává například shader objekty
2.1	2006	přidává například pixel buffer objekty
3.0	2008	přidává koncepci zastaralosti, to znamená, že některé funkce se stanou subjektem k odstranění v pozdějších vezích
3.1	2009	většina vlastností označených jako nesouhlasné byly v této verzi odstraněny
3.2	2009	přidává profil jádra a kompatibility
3.3	2010	přidán například Dual-source blending
4.0	2010	přidán například Shading language 4.00
4.1	2010	přináší například chybějící funkcionalitu z OpenGL ES 2.0
4.2	2011	například povoluje v shaderech rozbalení 16 bitových floatů z 32 bitových integerů bez znaménka
4.3	2012	byly přidány například debagovací zprávy
4.4	2013	bylo přidáno například přímé čištění imige textur
4.5	2014	například přináší kompatibilitu s OpenGL ES 3.1

4.1.2 DirectX

DirectX je skupina technologií designovaná pro běh a zobrazování aplikací s bohatým multimediálním obsahem bohatým na grafiku, video, 3D animace a audio. Prostředí je dostupné pouze na operačním systému Windows. Directx SDK je balíček určený pro tvorbu grafických aplikací, které běhají na technologii DirectX[21].

4.1.3 Vulkan

Vulkan je API nové generace pro tvorbu přenosných grafických aplikací na moderních grafických kartách a přenosných aplikací pro výpočty. Vulkán může být využit na široké škále zařízení od stolních počítačů přes herní konzole a mobilní telefony po vestavné systémy. Vulkan ve verzi 1.0 vyšel v únoru 2016. Vulkan by měl sjednotit technologie OpenGL a OpenGL ES. Vydavatelem tohoto API je KHRONOS GROUP[22].

4.2 Technologie pro tvorbu GUI

V této části pojednáváme o technologiích pro tvorbu grafického uživatelského prostředí. Popisujeme zde technologie Qt, Nana, wxWidgets a GTK+.

4.2.1 Qt

QT je technologie pro tvorbu uživatelského rozhraní pro různé operační systémy. První verze QT, 0.90, vyšla v roce 1995 pro systémy X11/Linux. Původními autory jsou Haavard Nord a Eirik Chambe-Eng. Oba autoři společně pracovali v roce 1990 na C++ databázi pro ultrazvukové obrázky. Vyžadovalo se, aby systém s GUI fungoval na systému UNIX, Macintosh a Windows. V této kapitole čerpáme ze zdrojů [23][24][25].

V roce 1991 Haavard začal psát třídy, které se nakonec staly Qt. Eirik pracoval na designu Qt. V roce 1992 Eirik přišel s nápadem použít tzv. signály a sloty. Tento nápad byl převzat několika dalšími nástroji pro tvorbu GUI. V roce 1993 Haavard a Eirik vytvořili první grafický kernel a byli schopni implementovat vlastní widgety. Koncem roku 1993 Haavard a Eirik založili společnost.

Písmeno *Q* bylo vybráno jako prefix tříd, protože dobře vypadalo ve fontu Haavardova Emacsu. Písmeno *t*, bylo zvoleno protože představovalo *toolkit*; do češtiny přeloženo jako sada nástrojů. Společnost byla založena v roce 1994 nejdříve pod jménem Quasar Technologies, později jako Troll Tech a Trolltech. V roce 1995 Trolltech najala Arnt Gulbrandsena, který během šesti let v Trolltechu vymyslel a implementoval nápaditou dokumentaci systému. V roce 1995 byla první verze, Qt 0.90, nahrána na sunsite.unc.edu. Qt bylo nabízeno k li-

cencování pod dvěmi licencemi. Licence pro komerční použití a volná licence pro open source.

V roce 1996 Trolltech získal prvního zákazníka[24]. Evropská vesmírná agentura si od Trolltechu zakoupila deset komerčních licencí. Během roku vyšla verze 1.0 a koncem roku 1996 vyšla verze 1.1.

V roce 2008 byl Trolltech odkoupen výrobcem telefonů Nokia. V roce 2011 Digia odkoupila práva na komerční licence a v roce 2012 odkoupila zbylá práva Qt. Přehled vývoje frameworku Qt můžete nalézt v tabulce 4.2.

Tabulka 4.2: Vývoj Qt[24][26]

verze	rok	poznámka
Qt 0.90	1995	pro systém X11/Linux
Qt 1.0	1996	podpora Windows
Qt 2.0	1999	QT/X11 open source s Q Public License
Qt 2.2	2000	GPL v2
Qt 3.0	2001	podpora více prostředí databáze, podpora více jazyků, podpora více připojených monitorů, podpora Mac OS X a nový Qt Designer pro tvorbu GUI
Qt 4.0	2005	Celkové přepracování pod komerční licencí a GPL 2.0, nebo pozdějšími, pro všechny platformy i pro systém Windows
Qt 4.5	2009	LGPL v2.1
Qt 4.7	2010	podpora Symbian
Qt 5.0	2012	celkové přepracování Qt 4.x.
Qt 5.1	2013	nové moduly a testování podpory systémů Android a iOS
Qt 5.2	2013	podpora systémů Android a iOS
Qt 5.4	2014	podpora platformy WinRT

4.2.2 Nana

Nana je technologie pro tvorbu uživatelského rozhraní pro systémy Windows a Linux(X11). Projekt Nana je nový projekt. Oficiální stránky projektu [27] byly spuštěny v květnu 2016. Současnou verzí knihovny Nana je verze 1.4.1. Technologie Nana je volný open source projekt s licencí pod Boost Software License. Licence podporuje komerční i nekomerční využití. Nana je knihovna pro moderní C++. Podporuje lambda funkce standardní knihovny a chytré ukazatele (smart pointers)[27].

4.2.3 wxWidgets

Technologie wxWidgets je technologie pro tvorbu uživatelského prostředí pro různé operační systémy. Projekt začal v roce 1992. Původním autorem projektu je Julian Smart z University v Edinburgu. Technologie wxWidgets je open-source technologie současně pod licencí wxWindows Library Licence.

Knihovna wxWidgets je napsaná v programovacím jazyce C++. Knihovnu je možné používat s různými programovacími jazyky. Jsou to například Python, Perl a C#. Knihovna podporuje platformy Linux, Unix, Windows a Mac OS X. V této sekci jsme čerpali z oficiálních stránek wxWidgets[28].

4.2.4 GTK+

Balíček nástrojů GTK+, dříve známý jako GIMP Toolkit, je technologie pro tvorbu uživatelského rozhraní pro různé operační systémy. Technologie GTK+ je součástí GNU projektu. Licence na API GTK+ je pod licencí GNU LGPL, která povoluje používat balíček k vývoji aplikací bez licenčních poplatků.

Technologie GTK+ je napsána v programovacím jazyce C. Technologii GTK+ je možné použít různými programovacími jazyky. Jsou to například Pear a Python. Původní GTK+ bylo vytvořeno pro systém X Window. V dnešní době GTK+ podporuje systémy Linux, Unix, Windows a Mac OS X. V této sekci jsme čerpali z oficiálních stránek GTK+[29].

4.3 Shrnutí a zhodnocení

V této kapitole jsme představili technologie pro tvorbu grafiky a technologie pro tvorbu grafického uživatelského prostředí. V tabulce 4.3 jsou představené technologie a operační systémy, které podporují. Všechny představené technologie podporují programovací jazyk C/C++. Pro samotnou implementaci programu využijeme technologie, které podporují jak operační systém Windows tak operační systém Linux a případně i systémy Mac OS X a Unix. Dalším nemálo významným faktorem je dokumentace technologie.

Tabulka 4.3: Tabulka technologií a podporovaných operačních systémů.

Technologie	Windows	Unix	Linux	Mac OS X
OpenGL	Ano	Ano	Ano	Ano
DirectX	Ano	Ne	Ne	Ne
Vulkan	Ano	Ano	Ano	Ano
Qt	Ano	Ano	Ano	Ano
Nana	Ano	Ne	Ano	Ano
wxWidgets	Ano	Ano	Ano	Ano
GTK+	Ano	Ano	Ano	Ano

Pro samotnou implementaci programu jsme vybrali technologii OpenGl, protože tato technologie je předepsaná zadáním této diplomové práce. Pro implementaci grafického uživatelského prostředí jsme vybrali technologii Qt, která poskytuje rozsáhlou dokumentaci. Tato technologie umožňuje snadnou tvorbu menu a panelu nástrojů. Dále tato technologie umožňuje spolupráci s technologií OpenGl.

Kapitola 5

Analýza

V této kapitole se zabýváme analýzou systému JANA2006 a soubory, ze kterých JANA2006 čte strukturu krystalu a parametry atomů. Dále se v této kapitole věnujeme analýze komunikace mezi programem JANA2006 a programy třetích stran pro zobrazování molekuly krystalu ve 3D prostoru. Samotnému popisu programu JANA2006 se věnuje kapitola 3, kde mluvíme i o programu Diamond. V této kapitole ani v této práci nepopisujeme formát souborů .CIF, protože nejsou pro tuto práci relevantní. Popis struktury souborů .CIF můžete nalézt zde [30].

Tato kapitola je dále členěna takto. V první části této kapitoly se zabýváme soubory, které JANA2006 používá k načtení struktury krystalu. Jsou to soubory s příponou .m40 a .m50. V druhé části se věnujeme souborům .m50. Ve třetí části této kapitoly se věnujeme struktuře souboru .m40, ve kterém jsou uloženy informace o samotné struktuře krystalu. Ve čtvrté části této kapitoly se zabýváme komunikací programu JANA2006 s externími programy pro zobrazení molekuly krystalu ve 3D prostoru. V páté části této kapitoly popisujeme požadované funkce, které by měl program pro zobrazení struktury krystalu ve 3D prostoru obsahovat.

5.1 JANA2006 a soubory s krystalovou strukturou

V této části se věnujeme souborům, z kterých program JANA2006 čte informace o struktuře krystalu. Jsou to soubory .m40, které obsahují samotné parametry atomů molekuly krystalu a soubory .m50, které obsahují informace o krystalové mřížce a symetrii. Samotnému souboru .m40 se podrobněji věnujeme ve třetí části této kapitoly, ve které také ukazujeme strukturu souboru. Program JANA2006 pracuje s dalšími soubory, ale tyto soubory nejsou pro téma této práce relevantní.

Soubory, se kterými program JANA2006 pracuje, jsou textové soubory, ve kterých jsou informace zarovnané pro snadné čtení a snadný zápis uživatelem. Tento fakt ovšem zvyšuje velikost souboru. Při práci s jednou strukturou může

vznikat velké množství verzí souborů .m40, a tím roste spotřeba místa na pevném disku. Parametry uložené v souboru .m50 se tolik nemění, a tak není nutné uvažovat o jejich kódování za účelem komprese souboru.

Soubory .m40 a soubory .m50 se načítají zároveň. V souboru .m40 jsou uloženy parametry krystalu(viz třetí část této kapitoly). Jedním z parametrů jsou frakční souřadnice krystalu. V souboru .m50 jsou informace o mřížce krystalu, o symetrii a typech atomů, které jsou v mřížce.

5.2 Soubory .m50

V souboru .m50 jsou informace o mřížce krystalu, o symetrii a typy atomů, které jsou v mřížce. Strukturu souboru můžete vidět na obrázku5.1. Struktura souboru lze popsat regulárním jazykem.

Pro zobrazení struktury krystalu v prostoru nás zajímají řádky, které začínají slovy „cell“, „symmetry“ a „atom“. Řádek, který začíná slovem „cell“, obsahuje mřížkové parametry struktury. Řádky, které začínají slovem „symmetry“ obsahují informace o použité symetrii a řádky začínající slovem „atom“ obsahují informace, které druhy atomů jsou ve struktuře obsaženy.

Obrázek 5.1: Část souboru .m50. Na obrázku jsou vyznačeny popořadě: parametry mřížky, použitá symetrie a druhy atomů.

```
Version Jana2006
title
cell 15.031 4.9479 19.8615 90 103.29 90
esdcell 0 0 0 0 0 0
spgroup P21/n 14 22
lattice P
symmetry x y z
symmetry -x+1/2 y+1/2 -z+1/2
symmetry -x -y -z
symmetry x+1/2 -y+1/2 z+1/2
unitsnumb 4
chemform S O5 C18 H12
formtab -62
atom S atradius 1.5 color 255255000
atom O atradius 1.5 color 255000000
atom C atradius 1.5 color 200200200
atom H atradius 1.5 color 255255255
lambda 0.71069 radtype 1 lpfactor 1
monangle 6.0818 perfmono 0
roundmethod 1
end
```

Mřížkové parametry, v pořadí za sebou (viz obrázek 5.1), jsou délky stran mřížky a, b, c a úhly jimi sevřené α, β a γ. Tyto parametry jsou důležité pro převod frakčních souřadnic, které jsou uloženy v souboru .m40 do ortogonálních. Z těchto parametrů vytvoříme matici pro převod frakčních souřadnic do ortogonálních (viz první kapitola).

Další informace, která se čte ze souboru .m50 je matice symetrie. Matice symetrie je v souboru .m50 zapsána pomocí 3 rovnic. Matici symetrie a translační vektor pro vstup (viz obrázek 5.1 čtvrtý řádek označený „symmetry“) např.:

$$symmetry \quad x + 1/2 \;\; -y + 1/2 \;\; z + 1/2$$

vytvoříme takto:

$$\begin{pmatrix} x & 0 & 0 \\ 0 & -y & 0 \\ 0 & 0 & z \end{pmatrix} + \begin{pmatrix} 1/2 \\ 1/2 \\ 1/2 \end{pmatrix}$$

Matici symetrie budeme dále označovat $\mathbf{R}$ a translační vektor budeme označovat $\mathbf{t}$. Algoritmus pro získání ortogonálních souřadnic atomů krystalové struktury o n atomech v prostoru je popsán v kapitole 6.3.

Pořadí druhů atomů, ve kterém jsou atomy uloženy v souboru .m50, je používáno dále v souboru .m40, jako index. Z atributů o atomech je nejdůležitější typ atomu. Další informace nejsou podstatné a slouží pouze pro účely programu JANA2006. Jsou to atribut atradius, který říká jaká musí být vzdálenost mezi atomy, aby se vytvořila vazba a atribut color, který říká jakou barvou bude atom zobrazován.

5.3 Soubory .m40

Soubory .m40 jsou hlavními soubory, se kterými program JANA2006 pracuje. Soubory .m40 obsahují uložené parametry atomů krystalu. Tyto parametry jsou:

1. jméno atomu, identifikátor, který pojmenovává atom
2. druh atomu, v souboru je jako index, který odpovídá pořadí atomu v souboru .m50
3. typ atomu, zda je atom harmonický, nebo izotropní (existuje ještě možnost anharmonický)
4. parametr ai, reprezentovaný, jako číslo s plovoucí řádovou čárkou
5. souřadnice atomu, které jsou v souboru .m40 zadané jako frakční souřadnice, tři čísla s plovoucí řádovou čárkou
6. teplotní parametry, šest čísel s plovoucí řádovou čárkou, které udávají teplotní parametry atomu

7. informace o tom, zda se parametry upřesňují nebo neupřesňují

Soubor dále obsahuje chyby těchto parametrů a další informace, se kterými pracuje program JANA2006. Příklad souboru .m40 můžete vidět na obrázku 5.2, kde jsou vidět samotné parametry atomů molekuly krystalu a na obrázku 5.3, kde jsou vidět chyby jednotlivých parametrů.

Obrázek 5.2: Část souboru .m40. Na obrázku je vyznačena hodnota, která udává počet atomů v molekule krystalu. Dále na obrázku jsou vidět dvojce řádků, které začínají jménem atomu a dále obsahují parametry molekuly krystalu.

```
  36    0    0    0
 2.048393 0.000000 0.000000 0.000000 0.000000 0.000000      100000
 0.000000 0.000000                                          00
 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000      000000
 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000      000000
S1        1  2     1.000000 0.955276-0.209174 0.051923
 0.013188 0.016260 0.010584 0.003197 0.003869 0.000455      0111111111
O1        2  2     1.000000-0.109391 0.972525 0.530930
 0.009205 0.021841 0.016924-0.001206 0.000941-0.005426      0111111111
O2        2  2     1.000000-0.675064-0.674630-0.101259
 0.016497 0.018016 0.013892 0.006513 0.003998 0.000031      0111111111
O3        2  2     1.000000 1.231297 0.083831 0.693964
 0.023431 0.024265 0.013990-0.006978 0.006489-0.001905      0111111111
O4        2  2     1.000000 0.972718-0.314618 0.249951
 0.013552 0.019625 0.011298 0.005516 0.003194 0.002180      0111111111
O5        2  2     1.000000-0.219934 1.165006 0.574125
 0.016341 0.022270 0.017661 0.000096 0.004135-0.003891      0111111111
C1        3  2     1.000000-0.980539 0.302839-0.136565
 0.010671 0.013505 0.013309-0.000340 0.002255-0.000693      0111111111
C2        3  2     1.000000 1.325367 0.328519 0.493859
 0.013229 0.015756 0.010568-0.001429 0.003229-0.000562      0111111111
C3        3  2     1.000000 0.872856 0.032103 0.050998
 0.010472 0.014177 0.013911 0.000772 0.003219-0.001416      0111111111
C4        3  2     1.000000-0.339471 0.614509 0.388312
 0.009424 0.015015 0.011935 0.000494 0.003273 0.000787      0111111111
C5        3  2     1.000000-0.938357 0.195838-0.185970
 0.011653 0.015042 0.011306-0.000870 0.001540 0.002899      0111111111
C6        3  2     1.000000 1.224861 0.034640 0.633320
 0.008909 0.014857 0.014328 0.001715 0.001521 0.001505      0111111111
C7        3  2     1.000000 0.697136 0.512520-0.034010
 0.012310 0.016360 0.010466 0.000074 0.001458-0.002306      0111111111
C8        3  2     1.000000-0.261116 0.840272 0.483978
 0.011418 0.014135 0.010960 0.000044 0.001467 0.000638      0111111111
```

Struktura a zarovnání souboru .m40 je navržena pro snadné čtení. V případě, že vzniká velké množství verzí souboru a soubory se hlavně zpracovávají programem, může tato vlastnost překážet a bylo by vhodnější, kdyby informace tohoto souboru byly zakódovány tak, aby se podstatně snížila velikost souboru. Pro tento účel je dále nutné podotknout, že některé informace jsou v souboru .m40 nadbytečné. Například pokud je atom izotropní, tak se pracuje pouze s jedním teplotním parametrem. Další nadbytečnou informací je opako-

Obrázek 5.3: Část souboru .m40. Na obrázku jsou vidět dvojce řádků, které začínají jménem atomu a dále obsahují chyby parametrů molekuly krystalu.

```
S1                  0.000000 0.000032 0.000099 0.000024
 0.000206 0.000212 0.000195 0.000179 0.000152 0.000187
O1                  0.000000 0.000088 0.000292 0.000072
 0.000593 0.000728 0.000672 0.000526 0.000505 0.000573
O2                  0.000000 0.000093 0.000289 0.000070
 0.000655 0.000717 0.000638 0.000537 0.000515 0.000548
O3                  0.000000 0.000102 0.000317 0.000074
 0.000737 0.000781 0.000676 0.000611 0.000570 0.000590
O4                  0.000000 0.000091 0.000288 0.000068
 0.000606 0.000705 0.000605 0.000547 0.000487 0.000557
O5                  0.000000 0.000097 0.000296 0.000076
 0.000667 0.000774 0.000684 0.000566 0.000538 0.000590
C1                  0.000000 0.000120 0.000394 0.000095
 0.000773 0.000804 0.000802 0.000675 0.000652 0.000727
C2                  0.000000 0.000124 0.000378 0.000096
 0.000809 0.000909 0.000776 0.000681 0.000644 0.000702
C3                  0.000000 0.000121 0.000384 0.000097
 0.000776 0.000872 0.000848 0.000651 0.000657 0.000699
C4                  0.000000 0.000121 0.000384 0.000095
 0.000769 0.000832 0.000838 0.000643 0.000652 0.000679
C5                  0.000000 0.000122 0.000406 0.000093
 0.000774 0.000825 0.000780 0.000682 0.000640 0.000733
C6                  0.000000 0.000120 0.000385 0.000097
 0.000765 0.000861 0.000846 0.000654 0.000647 0.000713
C7                  0.000000 0.000122 0.000407 0.000094
 0.000830 0.000896 0.000804 0.000672 0.000650 0.000690
C8                  0.000000 0.000124 0.000376 0.000096
```

vání jména atomu pro informace o chybách parametrů. Tyto údaje, které jsme uvedli jako nadbytečné, ale slouží k jednoduššímu čtení souboru uživateli.

V této práci dále navrhujeme kódování souboru .m40 za účelem snížení velikosti souboru. Popis kódování naleznete v šesté kapitole této práce (viz 6.1).

5.4 JANA2006 a externí programy pro zobrazení krystalu v 3D prostoru

Jak bylo uvedeno v předchozí části, JANA2006 pro práci s krystaly používá soubory .m40 a .m50. Programy pro zobrazení molekuly krystalu v 3D prostoru s těmito soubory nepracují. Programy, jako je například Diamond a Vesta, pracují s formátem souboru .CIF, Crystallographic Information File. Při volání externího programu z programu JANA2006, JANA2006 převádí informace o struktuře do formátu CIF. a předává je externímu programu jako parametr. Další programy, zmíněné ve třetí kapitole, Atoms a PLATON, neumí pracovat s formáty souborů používaných programem JANA2006.

Tento způsob nemusí být zcela vyhovující pokud chceme ve struktuře něco změnit a máme spuštěný grafický program pro zobrazení molekuly ve 3D. Protože JANA2006 pracuje se soubory .m40 a .m50, a grafický program se soubory .CIF, které mu jsou předávány jako parametr, nebudou změny, které se provedou v externím programu propagovány zpět do programu JANA2006.

Řešením tohoto problému může být vytvoření grafického programu, který bude pracovat se soubory, s kterými pracuje program JANA2006. Popisu vlastní realizace takového programu se věnujeme v šesté kapitole této práce.

5.5 Požadované funkce

Všechny představené grafické programy (viz kapitola 3.3)neumí pracovat s nativními soubory programu JANA2006. Hlavní požadovanou funkcí programu pro zobrazení struktury krystalu ve 3D prostoru je to, aby program dovedl číst strukturu krystalu z nativních souborů systému JANA2006 a zobrazit tuto strukturu ve 3D prostoru. Tyto soubory jsou soubory .m40, obsahující samotnou strukturu krystalu, a .m50, obsahující informace o mřížce a symetrii a informace o druzích atomu (viz sekce 5.3 a sekce 5.2). Program má umět pracovat s izotropními a harmonickými typy atomů.

Další požadovanou funkcí programu je, aby program byl schopen měnit parametry struktury. Tyto parametry jsou souřadnice, teplotní parametry, druhy atomů a typy atomů(izotropní nebo harmonický). Dále by program měl umět změnit textový identifikátor atomu a informace, zda se určitý parametr bude v programu JANA2006 upřesňovat.

Dále se požaduje, aby program by schopen uložit všechny změny v nativním formátu souboru .m40 programu JANA2006.

Pro grafické zobrazení se požaduje, aby v programu bylo možné zobrazenou strukturu otáčet, posouvat a zvětšovat a zmenšovat velikost zobrazené struktury. Další požadovanou funkcí programu je to, aby bylo možné používat klávesové zkratky pro výběr. Jsou to klávesové zkratky CTRL+A pro výběr všech atomů a CTRL+I pro inverzi výběru.

Kapitola 6

Realizace

V této kapitole se věnujeme programu pro zobrazení molekuly krystalu v 3D prostoru, který jsme implementovali a návrhu implementace kódování pro soubor obsahující strukturu krystalu. V této kapitole popíšeme, kterou technologii jsme použili pro implementaci grafického uživatelského prostředí, a kterou technologii jsme použili pro tvorbu 3D grafiky.

Tato kapitola je rozdělena takto. V první části této kapitoly se věnujeme návrhu a implementaci kódování souboru pro strukturu krystalu. V druhé části diskutujeme návrh kódování souboru. V třetí části této kapitoly se věnujeme implementaci programu pro zobrazení molekuly krystalu ve 3D. Ve čtvrté části této kapitoly se věnujeme navrženému grafickému uživatelskému prostředí. V páté části této kapitoly se věnujeme testování.

6.1 Kódování souboru .m40

V této části kapitoly navrhujeme kódování za účelem snížení velikosti dat souboru .m40. Pro pojmenování souboru jsme zvolili .m40k, značící .m40 kódované. Informace ze souboru .m40 jsme se rozhodli převést do binární podoby a část informací ze souboru vynechat, protože jsou redundantní. Dále jsme se rozhodli informace v souboru přeskupit pro snadnější kódování a dekódování samotných informací. Toto kódování jsme navrhli za účelem snížení velikosti souborů. Strukturu kódování jsme takto zvolili, z důvodu relativně snadné implementace, rychlého zápisu a čtení takto navržené struktury souboru a uspokojivému snížení velikosti struktury krystalu uložené v souboru. Takto navržená struktura souboru snižuje velikost souboru s uloženou strukturou až o více než dvě třetiny.

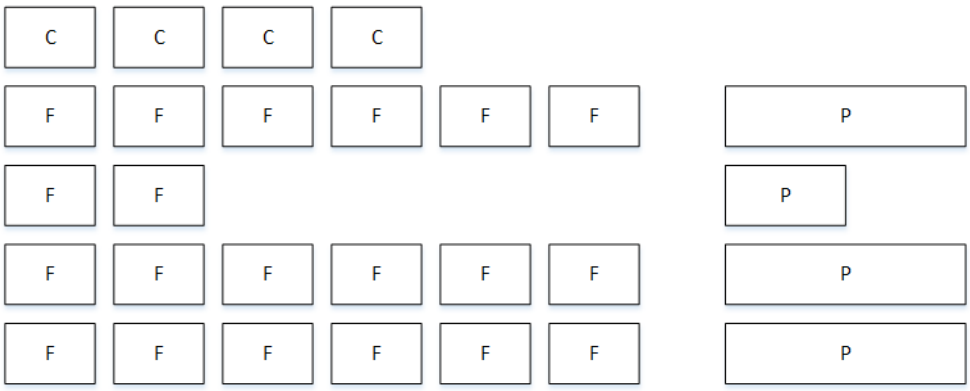
Soubor .m40 jsme si rozdělili na čtyři části. První část souboru .m40 jsou informace obsažené před parametry atomů a před chybami parametrů. Druhá část souboru jsou parametry atomů a chyby parametrů. Třetí část souboru .m40 jsou „save pointy“. Čtvrtá část souboru jsou Furierova maxima a minima, která jsou obsažena za „save pointy“.

Grafické znázornění první části souboru .m40 je na obrázku 6.1. Části, které jsou na obrázku označeny **C** jsou celá čísla. Tyto části jsou reprezentovány každá dvěma byty a jsou uloženy v souboru na prvním místě.

Části označené **P** jsou příznaky. Počet příznaků v řádku je dán počtem buněk označených **F**. Na každý příznak je potřeba jeden bit. Příznaky jsou uloženy na druhém místě v kódovaném souboru a velikost je zaokrouhlena na celé byty.

Části označené **F** jsou čísla s plovoucí řádovou čárkou. Tato čísla jsou uložena na třetím místě a k reprezentaci se používají 4 byty, jednoduchou přesnost. Za tato čísla jsou, uloženy jejich chyby, které mají podobnou strukturu jak je zobrazeno na obrázku 6.1.

Obrázek 6.1: První část souboru .m40. Na obrázku jsou označeny různé části souborů podle reprezentace. Části označené **C** jsou celá čísla. Části označené **F** jsou čísla s plovoucí řádovou čárkou. Části označené **P** jsou příznaky.



Grafické znázornění druhé části souboru .m40 je na obrázku 6.2. Druhá část souboru .m40 je rozdělena na část, ve které jsou parametry atomů a na část, ve které jsou chyby parametrů atomů (na obrázku 6.2 oddělené velkými obdélníky).

Při reprezentaci druhé části souboru, se na první místo za předchozí část, uloží všechny informace, které jsou na grafickém znázornění v druhé části souboru .m40 označeny **C2**. Pro reprezentaci každé této informace jsou potřeba 2 bity. Informace vyjadřuje, zda je atom harmonický nebo izotropní. Existuje třetí možnost, anharmonický atom.

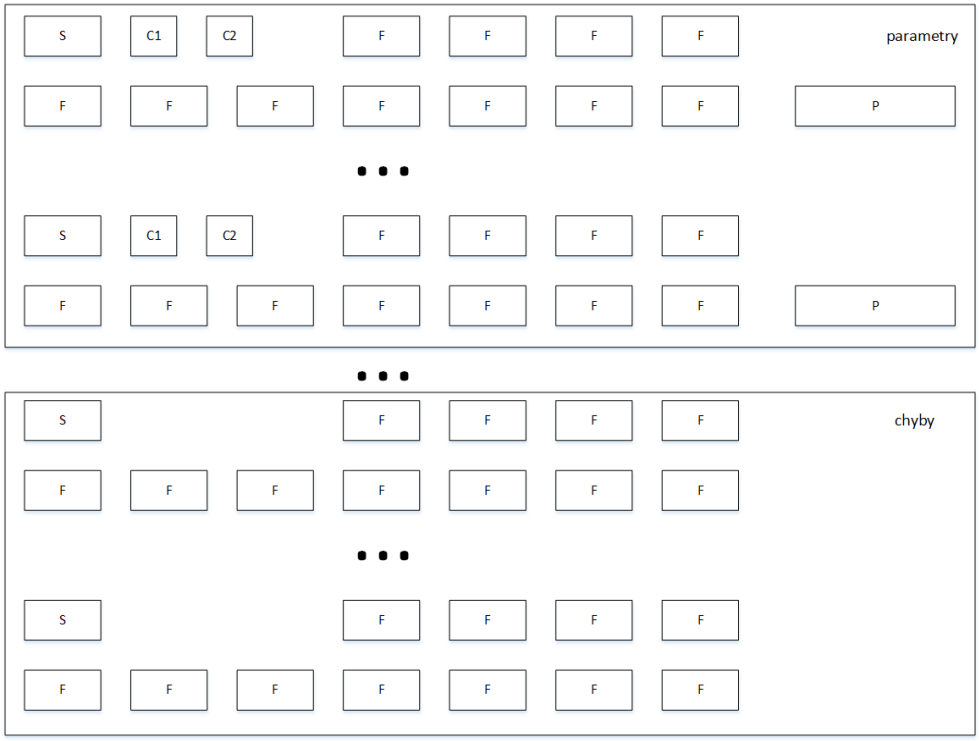
Na druhé místo za předchozí část se uloží v kódovaném souboru informace označené **C1**. K uložení každé této informace je zapotřebí $\lceil \log_2(\textit{počet druhů atomů}) \rceil$ bitů (počet druhů atomů je uložen v souboru .m50). Za tuto část dále uložíme všechny informace označené **P**, které značí zda se jednotlivé parametry v programu JANA2006 upřesňují. Každá tato informace se dá reprezentovat buď deseti bity pokud je atom harmonický (harmonický atom je popsán deseti parametry), nebo pěti bity pokud je atom

izotropní (izotropní atom je popsán pěti parametry). Tyto informace nakonec zaokrouhlíme na celé byty.

Na třetí místo se ukládají všechny informace z části, ve které jsou parametry označené **S**. Informace **S** z chybové části se neukládají, protože jsou redundantní. Každá tato informace je textová a je zakončena osmi nulovými bity pro snadné dekódování.

Dále jsou za předchozí část uloženy všechny informace označené **F** tak, aby nejdříve byly uloženy parametry a chyby parametrů prvního atomu, za ním následovaly parametry a chyby parametrů následujících atomů. Pro izotropní atomy se neukládají všechny teplotní parametry a jejich chyby, ale pouze první teplotní parametr a jeho chyba.

Obrázek 6.2: Druhá část souboru .m40. Na obrázku jsou označeny různé části souborů podle reprezentace. Části označené **S** jsou řetězce znaků. Části označené **C1** a **C2** jsou celá čísla. Části označené **F** jsou čísla s plovoucí řádovou čárkou. Části označené **P** jsou příznaky.



Třetí část souboru .m40 obsahuje informaci o počtu uložených „save poitů“, dále značíme **C**, a samotné „save pointy“. Každý „save point“ obsahuje textový identifikátor **S** a tři čísla s plovoucí řádovou čárkou **F**. Do souboru je nejdříve uložena informace **C** a za ni jsou uloženy jednotlivé „save pointy“. Jsou to informace **S** následující informacemi **F**. Každá informace **S** je zakončena osmi nulovými bity pro snadné dekódování.

Čtvrtá část souboru .m40 je rozdělena na dvě stejné části začínající třemi celými čísly. Tato čísla se uloží za předchozí části. Pro každé číslo se použijí dva byty. Každá tato část se uloží zvlášť za předchozí části do kódovaného souboru. Grafické zobrazení třetí části souboru .m40 je na obrázku 6.3.

Každá sekce se ukládá zvlášť. Sekce začínají vždy řetězcem **S**. Sekce se ukládají za sebe. Místo informace **S** se ukládá jeden bit. Pro první část třetí části se ukládá nula pro druhou se ukládá jednička. Za tento bit se ukládají příznaky **P**, které jsou reprezentovány třemi bity. Za příznaky se ukládají informace **C**, které jsou reprezentovány 2 byty. Za tyto informace se ukládají čísla s plovoucí řádovou čárkou **F**, které jsou reprezentována 4 byty.

Obrázek 6.3: Třetí část souboru .m40. Na obrázku jsou označeny různé části souborů podle reprezentace. Části označené **S** jsou řetězce znaků. Části označené **C** jsou celá čísla. Části označené **F** jsou čísla s plovoucí řádovou čárkou. Části označené **P** jsou příznaky.

6.2 Diskuse navrženého formátu

V takto navrženém formát souboru odstraňujeme informace, které se nevyužívají. Tyto informace slouží pro snadné čtení souboru uživatelem. Jsou to teplotní parametry izotropních atomů a chyby těchto parametrů, kde se využívá pouze jeden teplotní parametr a jeho chyba, místo všech šesti parametrů a chyb. Toto kódování snižuje velikost reprezentace informace o atomech v souboru. Takto navržené kódování souboru .m40 (viz kapitola 6.1) významně snižuje velikost souboru a je vhodné pro zálohování těchto souborů. Je také výhodné v případě, kdy vzniká velké množství pracovních verzí.

Hlavním obsahem souborů .m40 jsou čísla s plovoucí řádovou čárkou. Pro reprezentaci čísel s plovoucí řádovou čárkou byla zvolena 4 bytová reprezentace, jednoduchá přesnost (dle IEEE 754 viz [31]). Tato reprezentace je postačující. Soubory .m40 obsahují čísla s plovoucí řádovou čárkou s přesností na šest desetinných míst. Pro tuto přesnost je čtyř bytová reprezentace vyhovující. Entropie informace uložené v čísle s plovoucí řádovou čárkou v souboru .m40 je přibližně mezi $\left(-\log_2 \frac{1}{10^6}; -\log_2 \frac{1}{10^7}\right)$. Entropie se spíše blíží k číslu $-\log_2 \frac{1}{10^6} = 19,9316$, protože desetinná čárka je obsažená většinou za prv-

ním číslem. Čísla s plovoucí řádovou čárkou obsahují přibližně 95% velikosti kódovaného souboru.

6.3 Popis programu pro zobrazení molekuly krystalu ve 3D

6.3.1 Technické údaje

V této části kapitoly se věnujeme našemu programu pro zobrazení molekuly krystalu v 3D prostoru. Pro implementaci byly vybrány technologie OpenGL ve verzi 4.5 pro tvorbu 3D grafiky a technologii QT ve verzi 5.6 pro tvorbu grafického uživatelského prostředí. Pro maticové a vektorové operace související s vykreslením objektů pomocí OpenGL je použita knihovnu GLM (viz [32]). Jedním z požadavků na tento program bylo, aby tento program mohl běhat jak pod systémem Linux tak pod systémem Windows. Obě technologie podporují výše uvedené operační systémy. Technologie OpenGL byla vybrána, protože je určena v zadání této diplomové práce. Pro implementaci GUI bylo vybráno Qt z důvodu obsáhlé dokumentace. Technologie Qt umí využívat OpenGl pro vykreslování grafiky. Dále bychom měli podotknout, že program byl zkompilován pouze pro systém Windows a testován pouze na systému Windows. V programu nejsou použity žádné nativní funkce systému Windows.

6.3.2 Načtení struktury

Program umí načítat strukturu krystalu ze souborů .m40 a .m50 a zobrazit strukturu atomu v 3D prostoru. Při otevírání souborů se kontroluje zda vstup splňuje strukturu souboru .m40 a .m50 a zda informace o atomech jsou validní. Ze souboru .m50 jsou programem načteny informace o mřížkových parametrech, které se používají k převodu frakčních souřadnic uložených v souboru .m40. Dále jsou programem ze souboru .m50 načteny informace o symetrii. Tyto informace jsou použity k určení pozice atomu a vytvoření struktury krystalu. Atom může být zobrazen v různé buňce a v buňce může mít různou polohu (viz kapitola 1.1). Dále jsou ze souboru .m50 načteny: počet druhů atomů a druhy atomů. Tyto informace jsou použity k vykreslení molekuly krystalu a vazeb mezi nimi. Informace o vazbách mezi atomy jsou uloženy v databázi a ze souboru .m50 se nečtou. V databázi jsou dále uloženy informace o velikosti a barvě, a o maximální vzdálenosti mezi atomy takové, aby se vytvořila vazba mezi nimi. Tuto databázi představují dva textové soubory.

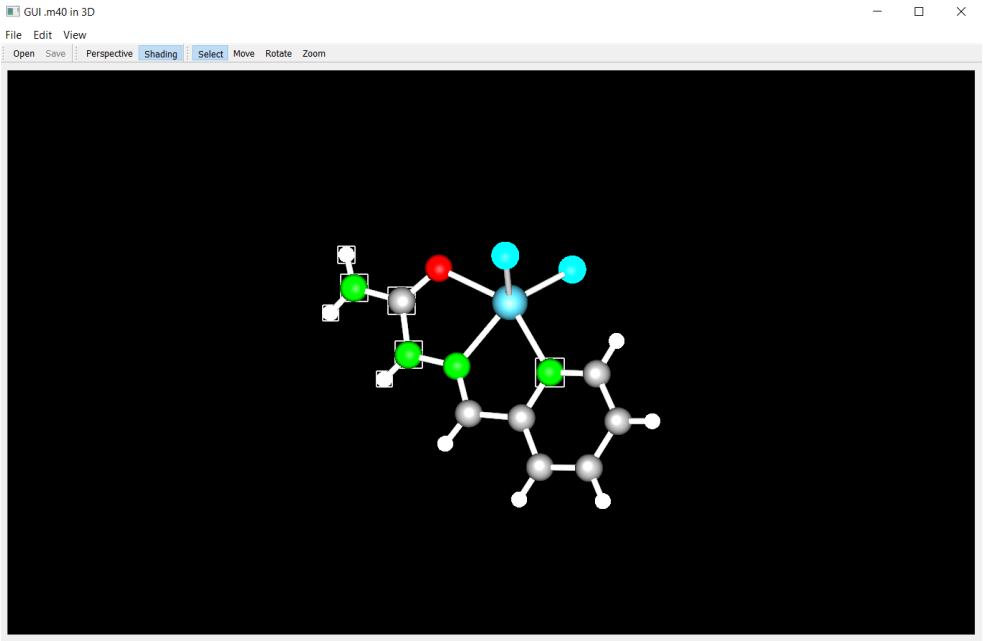
Ze souboru .m40 program čte parametry krystalu a chyby těchto parametrů. Pro vykreslení struktury jsou nejdůležitější frakční souřadnice. Frakční souřadnice jsou v programu převedeny na ortogonální a tyto souřadnice jsou dále využity k samotnému vykreslení struktury krystalu ve 3D prostoru. Pro převod se používá vztah z kapitoly 1.1. Na obrázku 6.4 můžete vidět načte-

nou strukturu ze souboru .m40. Algoritmus pro převod frakčních souřadnic na ortogonální postupuje viz algoritmus 6.

```
Data: Seznam atomů, matice symetrie R, translační vektor t, počet
      atomů ve struktuře n, monožina H všech vektorů h, pro které
      platí h ∈ {0,1}^3, translační matice T.
Result: Ortogonální souřadnice
j ← 1;
k ← 1;
while k ≤ n do
    Ze seznamu všech atomů vyber k-tý atom;
    if k-tý atom není zpracovaný then
        pomocí translační matice spočítej ortogonální souřadnice k-tého
        atomu;
        označ k-tý atom jako zpracovaný;
    end
    while j ≤ n do
        ze seznamu všech atomů vyber k-tý atom;
        if j-tý atom není zpracovaný then
            foreach R do
                x‘ ← Rx + t;
                přičti celočíselný vektor v takový, že frakční souřadnice
                j-tého atomu odpovídají buňce k-tého atomu;
                foreach h do
                    x_o ← T (x‘ + h);
                    if vzdálenost j-tého atomu od k-tého atomu je
                    dostatečně malá, aby se vytvořila vazba then
                        označ j-tý atom jako zpracovaný;
                        ukonči oba FOR cykly;
                    end
                end
            end
        end
        j ← j + 1;
    end
    j ← 1;
    k ← k + 1;
end
```

Algoritmus 6: Převedení frakčních souřadnic na ortogonální

Obrázek 6.4: Zobrazení struktury krystalu načtené ze souboru .m40 v našem programu.



6.3.3 Vykreslování

Vykreslování atomů je implementováno tak, že se tvoří jedna koule pro jeden každý druh atomů zastoupených v struktuře. Tento model je tvořen 1024 body, které jsou propojeny trojúhelníky tak, aby vznikl požadovaný tvar. Takto vytvořený model atomu se zobrazuje na různých pozicích odpovídající souřadnicím.

Pro vytvoření vazby mezi atomy a pro určení pozice atomů se využívá informace ze souboru, kde jsou uloženy maximální vzdálenosti takové aby se vytvořila vazba mezi atomy. Tyto vzdálenosti jsou různé pro různé atomy. Algoritmus pro tvorbu vazeb mezi atomy viz algoritmus 7. Šířka vazeb mezi atomy je konstantní. Vazby mezi atomy se tvoří mezi středem jednoho atomu a středem druhého atomu.

6.3.4 Editace atomů

V programu je implementováno okno, ve kterém lze změnit základní parametry krystalu a změny uložit. Je možné smazat atomy, změnit základní parametry jednotlivých atomů, jako jsou teplotní parametry a frakční souřadnice (viz obrázek 6.5). V okně lze určit, které parametry v programu JANA2006 budou upřesňovány. Dále je v tomto okně možné změnit textový identifikátor atomu, druh atomu a typ atomu.

Data: Seznam atomů s určenými ortogonálními souřadnicemi, počet atomů n.
Result: Seznam vazeb mezi atomy.
$j \leftarrow 1$;
$k \leftarrow j + 1$;
while $j \leq n$ **do**
  Ze seznamu všech atomů vyber j-tý atom;
  while $k \leq n$ **do**
    ze seznamu všech atomů vyber k-tý atom;
    if *vzdálenost mezi j-tým a k-tým atomem je dostatečně malá, tak aby se tvořila vazba* **then**
      Vytvoř vazbu mezi j-tým a k-tým atome a přidej ji do seznamu vazeb;
    end
    $k \leftarrow k + 1$;
  end
  $k \leftarrow j + 1$;
  $k \leftarrow k + 1$;
end

Algoritmus 7: Algoritmus pro vytvoření vazeb mezi atomy.

6.4 Navržené GUI programu pro zobrazení molekuly krystalu ve 3D

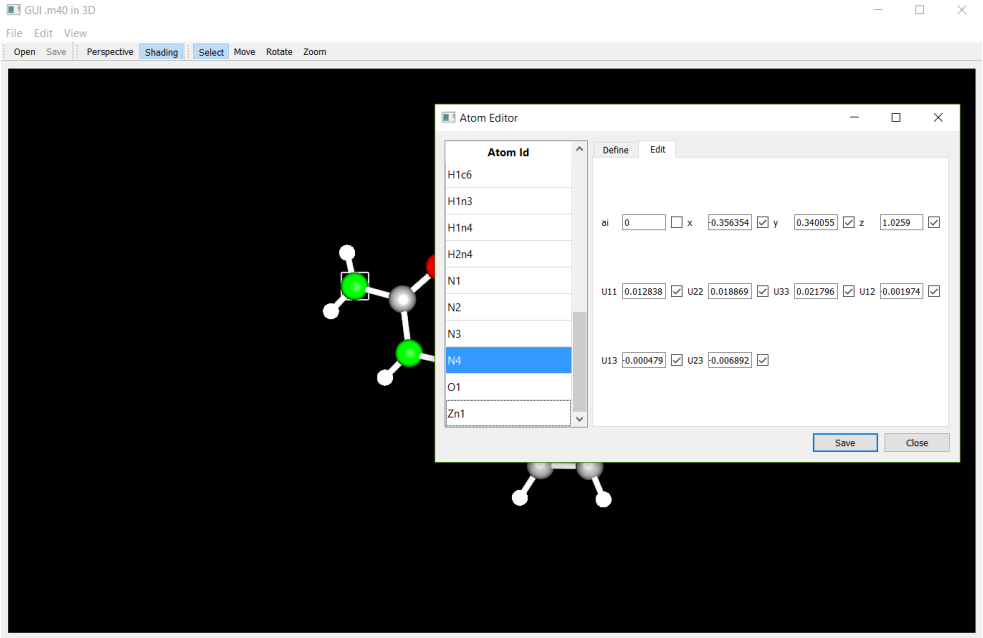
V této části kapitoly se věnujeme grafickému uživatelskému prostředí našeho programu pro zobrazení struktury krystalu ve 3D prostoru. Grafické uživatelské prostředí je navrženo tak, aby funkce tohoto programu byly snadno dostupné. Další naší snahou bylo, aby prostředí bylo intuitivní. Pro snadné použití tohoto programu jsou implementované zkratky. Například to jsou zkratky: CTRL+A – pro výběr všech atomů, CTRL+I – pro výběr všech neoznačených atomů a CTRL+S – pro uložení struktury.

V programu je implementováno tradiční menu, které bylo navrženo tak, aby odpovídalo zvyklostem uživatelů (viz obrázek 6.6). Toto menu je rozděleno na „File“ menu, „Edit“ menu a „View“ menu. Ve „File“ menu jsou hlavně funkce pro načítání a ukládání souborů. V „Edit“ menu jsou funkce pro výběr atomů, pro mazání atomů a pro editaci atomů. Ve „View“ menu jsou funkce toho, jak bude model molekuly zobrazen.

Pro snadnější přístup je dále implementován panel nástrojů. V panelu nástrojů jsou implementovány základní funkce pro práci se soubory, pro zobrazení obrázků a pro manipulaci s obrázkem.

Pro načítání struktury krystalu se používá funkce „Open“, kterou je možné nalézt ve „File“ menu, nebo pro rychlý přístup v panelu nástrojů. K načtení samotné struktury krystalu je potřeba nalézt umístění souborů .m40 a .m50.

Obrázek 6.5: Okno pro změnu parametrů krystalu.



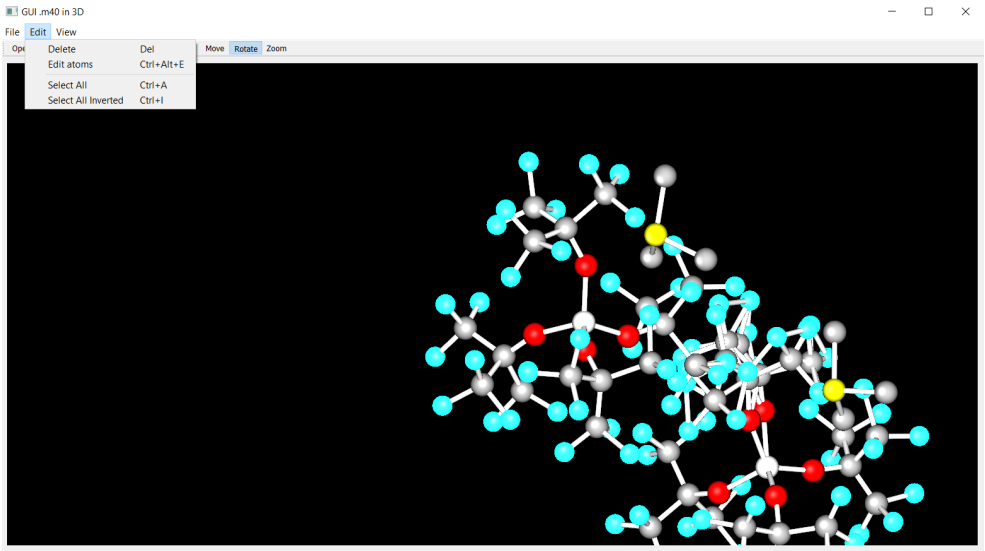
Program počítá s tím, že soubory .m40 a .m50 mají stejný název.

V „Edit“ menu jsou funkce pro editaci struktury krystalu. Jsou to funkce „Delete“ pro mazání a funkce „Edit atoms“ pro otevření okna na editaci atomů struktury. Pro otevření editačního okna je možné použít pravé tlačítko, čímž se otevře menu, ve kterém je možnost Edit. Tato možnost otevře okénko pro editaci atomů. Pro manipulaci s obrázkem se používají tlačítka v panelu nástrojů „Select“ – pro výběr atomů, „Move“ – pro pohyb se strukturou, „Rotate“ – pro otáčení struktury a „Zoom“ – zvětšení struktury krystalu.

6.5 Testování

Během vývoje byl tento program průběžně testován panem RNDr. Václavem Petříčkem, Csc.z Fyzikálního ústavu Akademie věd České republiky. Byla testována funkčnost programu, zda program zobrazuje správně strukturu krystalu a uživatelská přívětivost GUI. Většina takto nalezených nedostatků byla odstraněna během samotné implementace. Program je specializovaný a tak se od programu neočekává masové testování.

Obrázek 6.6: V programu je implementováno tradiční menu.



Kapitola 7

Vyhodnocení a diskuse

Tato kapitola je věnována vyhodnocení zadání této diplomové práce. V této kapitole dále diskutujeme navržené kódování souboru a samotný program pro zobrazení struktury krystalu ve 3D prostoru.

7.1 Splnění zadání

Prvním cílem této diplomové práce bylo nastudovat problematiku krystalu. Této problematice se věnujeme v kapitole 1. V této kapitole popisujeme, co je to krystalová mřížka a mřížkové parametry, co jsou frakční souřadnice, co je symetrie a převod frakčních souřadnic na ortogonální. Ortogonální souřadnice se využívají pro vykreslení pozic atomů. Dále v této kapitole popisujeme jak se měří pozice atomů v krystalu.

Dalším bodem zadání bylo nastudovat vnitřní architekturu a vstupní parametry programu JANA2006. O programu JANA2006 pojednává kapitola 3. V této kapitole také popisujeme externí programy, které se používají pro zobrazení struktury krystalu. Vstupní soubory programu JANA2006 jsou popsány v kapitole 5, kde je popsána komunikace mezi programem JANA2006 a programem Diamond, který používá Fyzikální ústav Akademie věd České republiky pro zobrazení struktury krystalu ve 3D prostoru.

Dále bylo zadáním této diplomové práce navrhnout a implementovat program, který ze vstupních dat v nativním formátu programu Jana2006 načte strukturu krystalu, zobrazí ji a umožní její editaci. Změny program uloží ve stejném formátu. K implementaci je použitý jazyk C/C++ a technologii OpenGL. Tento program je popsán v kapitole 6.3. Diskuse tohoto programu je v kapitole 7.4. Popis technologie OpenGL a dalších technologií je v kapitole 4. V této kapitole popisujeme technologie pro tvorbu grafiky a technologie pro tvorbu GUI.

Zadáním této diplomové práce bylo také navrhnout a implementovat vhodné kódování souborů, které by umožňovalo i kompresi dat, hlavně pro vstupní a výstupní soubory se strukturou a parametry krystalu. Toto kódovaní je na-

vrženo v kapitole 6.1. Popisu různých metod kódování a komprese se věnuje kapitola 2. Navržený formát souboru diskutujeme níže v 7.2.

Testování a porovnání implementovaného programu pro zobrazení struktury krystalu ve 3D prostoru se věnujeme v kapitole 6.5 a 7.3. Program porovnáváme s programem Diamond, Vesta, Atoms a Platon.

7.2 Diskuse navrženého kódování

Cílem navrženého kódování je snížit velikost souborů obsahujících velikost struktury krystalu. Navržený formát souboru navržený v kapitole 6.1 uspokojivě snižuje velikost souborů. Výhodou takto navrženého formátu souboru .m40 je, že velikost souboru se sníží až o více než dvě třetiny (viz tabulka 7.2). Další výhodou tohoto kódování je relativně snadná implementace takto navrženého kódování a rychlost čtení a zápisu takto navržené struktury souboru. V neposlední řadě výhodou tohoto formátu oproti dalším metodám uvedeným v kapitole 2 je, že data ze souboru je možné okamžitě číst bez předchozího dekódování (například v porovnání s LZ77). Nevýhodou takto navrženého formátu souboru .m40 je, že veškeré informace jsou reprezentovány binárně a pro běžné uživatele je tak tento soubor těžko čitelný oproti souboru .m40. Při návrhu formátu souboru byl hlavně důraz kladen na rychlost čtení a zápisu navrženého formátu.

Rozdíl velikosti mezi komprimovaným souborem .m40 metodou ZIP, založenou na LZ77 (viz kapitola 2.3.1), a námi navrženým formátem není příliš výrazný. Rozdíl se pohybuje na testovaných datech mezi jednou čtvrtinou a jednou polovinou ve prospěch kompresní metody ZIP 7.2 (procentuální rozdíl mezi velikostí souboru A a velikostí souboru B počítáme $(A - B)/A$). Pokud je na námi navržený formát použita kompresní metoda ZIP, je v průměru nižší než velikost komprimovaného souboru .m40 (viz tabulka 7.1) (Implementace metody LZW pochází z [12]).

Soubor .m40 o velikosti 11409 bytů, který obsahuje informace o 36 atomech, má po zakódování touto metodou velikost 3499 bytů (viz tabulka 7.1). Tato velikost se může měnit. Záleží hlavně na poměru harmonických atomů a izotropních atomů. Tento faktor může zvýšit velikost takto navrženého souboru až o stovky bytů. Dalším, ale ne tak významným faktorem, je délka textového identifikátoru. Délka textového identifikátoru je v souboru .m40 omezena na 8 bytů. Toto může zvýšit velikost takto navrženého souboru o desítky bytů. Pro výše uvedenou velikost souboru by se tímto způsobem rozsah souboru zvýšil přibližně o 180 bytů.

Pro dosažení lepší komprese souboru by bylo vhodné navrhnou jiné kódování čísel s plovoucí řádovou čárkou. Tato čísla zabírají přibližně 95% velikosti kódovaného souboru. Další možností je použít pro kódovaný soubor některou kompresní metodu, která představenou v kapitole 2 této diplomové práce. Při použití komprese ZIP, která využívá kompresní metodu LZ77, na soubor .m40

Tabulka 7.1: Tabulka velikosti souborů. V tabulce jsou porovnány velikosti souborů v bytech.

Velikost souboru .m40	komprese LZ77 .m40 (ZIP)	komprese LZW .m40	Velikost .m40k	komprese LZ77 .m40k (ZIP)	komprese LZW .m40k
11409B	2630B	3392B	3499B	2499B	3107B
10449B	2338B	3095B	3162B	2220B	2747B
8289B	2035B	2591B	2534B	1895B	2211B
23030B	4747B	5954B	9119B	4102B	5954B
33347B	10216B	10881B	13033B	10176B	13496B

Tabulka 7.2: Tabulka rozdílu velikostí mezi soubory. V tabulce je procentuální srovnání.

Velikost souboru .m40	Procentuální rozdíl mezi .m40 a .m40k	Procentuální rozdíl mezi .m40k a ZIP(.m40)	Procentuální rozdíl mezi .m40k a LZW(.M40)	Procentuální rozdíl mezi ZIP(.m40) a ZIP(.m40k)
11409B	69,3%	24,3%	3,1%	5,0%
10449B	69,7%	26,1%	2,1%	5,0%
8289B	69,4%	19,7%	-2,2%	6,9%
23030B	60,4%	47,9%	34,7%	13,6%
33347B	60,9%	21,6%	16,5%	0,4%

se velikost souboru sníží z 11409 bytu na 2630 bytů. Při použití kompresní metody ZIP na námi navrženou strukturu souboru, se velikost souboru sníží z 3499 bytů na 2499 bytů (viz tabulka 7.1).

7.3 Popis funkcí a porovnání s programy třetích stran

Základními funkcemi našeho programu je zobrazení struktury krystalu a editace parametrů krystalu. V programu je možné zobrazenou strukturu rotovat kolem středu, nebo okolo vybraného atomu. Dále je možné strukturu zvětšit či zmenšit a posunout. Na obrázek je dále možné použít stínování nebo perspektivu.

Výhodou námi navrženého programu je, že umí načíst strukturu krystalu z dvojce souborů .m40 a .m50 a umí uložit změněné parametry struktury do souboru .m40. Toto žádný ze zmíněných programů třetích stran neumí. V programu je dále implementováno čtení a zápis námi navrženého formátu souborů pro strukturu krystalu, představený v kapitole 6.1. Program umožňuje edito-

vat frakční souřadnice atomů, teplotní parametry atomů, určit, zda je atom izotropní nebo harmonický a smazat atom ze struktury. Program dále umožňuje změnit textový identifikátor atomu, který používá program JANA2006 a určit, které parametry se v programu JANA2006 budou upřesňovat.

V porovnání s programem Diamond náš program neumí zobrazit atomy, jako elipsy v závislosti na teplotních parametrech atomů. Námi implementovaný program dále neumí zobrazit obsah celé buňky krystalu za použití symetrie. V porovnání s programem Diamond náš program dále neumí například exportovat obrázek ve formátu JPG nebo načítat struktury v jiném formátu než v nativním formátu programu JANA2006. V porovnání s dalšími nástroji třetích stran námi navržený program neumí zobrazit elektronovou hustotu struktury, nebo například neumí zobrazit animaci vibrace atomů.

Programy třetích stran obsahují zmíněné a další funkce, které bychom chtěli do budoucna do našeho programu přidat, aby tento program mohl plně konkurovat všem těmto programům.

7.4 Diskuse programu

Popsaný program z kapitoly 6.3 splňuje všechny kladené požadavky (viz kapitola 5.5). Cílem programu pro zobrazení struktury krystalu ve 3D prostoru je zobrazit strukturu krystalu z dat v souborech .m40 a .m50 s kterými pracuje program JANA2006. Dalším cílem toho programu je možnost editovat parametry struktury krystalu. Takto navržený program umožňuje editovat všechny základní parametry struktury, kterými jsou souřadnice, teplotní parametry, informace o tom zda je atom izotropní nebo harmonický, a dále umožňuje určit parametry, které program JANA2006 bude upřesňovat. Program také umožňuje smazat atomy ze struktury a změny uložit zpět do souboru .m40.

Výhodou tohoto programu je, že pracuje se soubory s kterými pracuje program JANA2006 a umí tyto soubory editovat. Programy Diamond, Vesta, Atoms, PLATON a Mercury s těmito soubory neumí pracovat[14, 17, 16, 15, 18] a v programu JANA2006 lze využít pouze omezené funkce těchto programů. Oproti programům třetích stran, tento program zatím umožňuje práci pouze s jednoduchými strukturami krystalu a neumí pracovat s jinými soubory než se soubory .m40 a .m50. V současné době jsou v našem programu uvažovány pouze dva typy atomů, izotropní a harmonický. S anharmonickými atomy se v současné době v programu nepočítá. Po programu se vyžaduje, aby uměl pracovat s izotropními a harmonickými typy (viz kapitola 5.5).

Do budoucna bychom chtěli tento program dále rozšiřovat o další funkce. Tento program bychom chtěli rozšířit tak, aby mohl pracovat se složitějšími strukturami, aby v budoucnu pokrýval potřeby pokročilé analýzy struktury krystalu, a rozšířit funkce tohoto programu, například o možnost vrátit se o krok zpět. Dále bychom do programu chtěli přidat možnost zobrazit strukturu krystalu v buňce za využití symetrie a možnost zobrazit více buněk krys-

talové mřížky. Program bychom chtěli rozšířit tak, aby byl schopný konkurovat programům Diamond, Vesta, Atoms a Platon.

Závěr

Úkolem této diplomové práce bylo navrhnout a implementovat kódování souborů se strukturou krystalu, který používá program JANA2006. Dalším cílem práce bylo implementovat program pro zobrazení struktury molekuly krystalu v 3D prostoru, který by mohl měnit parametry atomů.

V práci je analyzována struktura souborů, ve kterých je uložena struktura krystalu, a komunikace mezi programem JANA2006 a programy třetích stran pro zobrazení atomu krystalu ve 3D prostoru. Ukázali jsme zde, že tento model nevyhovuje pro práci se soubory .m40. Dále jsme navrhli a popsali implementaci kódování souboru obsahující strukturu krystalu. Velikost souboru se díky tomuto kódování snížila až o dvě třetiny původní velikosti.

Dále jsme v této práci popsali program pro zobrazení molekuly krystalu v 3D prostoru, který jsme implementovali, porovnali jsme ho s programy třetích stran. Uvedli jsme zde, jak bychom tento program chtěli rozšiřovat. Dále jsme v této práci zdůvodnili, proč tento program vznikl. Současné programy pro zobrazení molekuly krystalu v 3D prostoru nepracují se soubory .m40 a .m50, s kterými pracuje program JANA2006. Jejich využití je proto limitováno pokud jsou volány z programu JANA2006. Věříme, že tento nástroj usnadní práci se strukturou krystalu, která je zároveň zpracovávána v programu JANA2006.

Literatura

[1] Petříček, V.; Dušek, M.; Palatinus, L.: Regular structures from single crystals. Workshop Lviv, 2016.

[2] Petříček, V.; Dušek, M.; Palatinus, L.: Zeitschrift für Kristallographie - Crystalline Materials. kapitola Crystallographic Computing System JANA2006, De Gruyter, 2014, s. 345–352. Dostupné z: `https://www.degruyter.com/view/j/zkri.2014.229.issue-5/issue-files/zkri.2014.229.issue-5.xml`

[3] Salomon, D.: Handbook of Massive Data Sets. kapitola Data Compression, Norwell, MA, USA: Kluwer Academic Publishers, 2002, ISBN 1-4020-0489-3, s. 245–309. Dostupné z: `http://dl.acm.org/citation.cfm?id=779232.779241`

[4] Crystallography - Cristalografia. `http://www.xtal.iqfr.csic.es/Cristalografia/index-en.html`, accessed: 2016-12-20.

[5] Kratochvíl, B.: *Chemie a fyzika pevných látek I.* Praha: Vysoká škola chemicko-technologocká v Praze, druhé vydání, 1994, ISBN 90-7080-196-4.

[6] Petříček, V.: universitní přednáška MFF UK.

[7] Giacovazzo, C.; Monaco, H. L.; Viterbo, D.; aj.: *Fundamentals of Crystallography.* OXFORD UNIVERSITY PRESS, druhé vydání, 1992, ISBN 0-19-855579-4.

[8] Holub, J.: Data Compresion. universitní přednáška ČVÚT FIT, 2015.

[9] Lórencz, R.: Bezpečnost. universitní přednáška ČVÚT FIT, 2011.

[10] Elias, P.: Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, ročník 21, č. 2, Mar 1975: s. 194–203, ISSN 0018-9448, doi:10.1109/TIT.1975.1055349.

[11] Huffman, D.: A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE*, ročník 40, č. 9, Sept 1952: s. 1098–1101, ISSN 0096-8390, doi:10.1109/JRPROC.1952.273898.

[12] LZW download | SourceForge.net. `https://sourceforge.net/projects/lzw/?source=typ_redirect`, accessed: 2016-1-26.

[13] van Smaalen, S.: *Incommensurate Crystallography*. OXFORD UNIVERSITY PRESS, první vydání, 2007, ISBN 978-0-19-857082-0.

[14] Diamond - Crystal and Molecular Structure Visualization. `http://www.crystalimpact.com/diamond/Default.htm`, accessed: 2016-1-6.

[15] VESTA. `http://jp-minerals.org/vesta/en/`, accessed: 2016-1-17.

[16] THE PLATON HOMEPAGE. `http://www.cryst.chem.uu.nl/spek/platon/`, accessed: 2016-1-8.

[17] Shape Software. `http://www.shapesoftware.com/00_Website_Homepage/#anchor_files`, accessed: 2016-1-8.

[18] Mercury - The Cambridge Crystallographic Data Centre (CCDC). `https://www.ccdc.cam.ac.uk/solutions/csd-system/components/mercury/`, accessed: 2016-1-17.

[19] `https://www.opengl.org/about/#5`, accessed: 2016-1-4.

[20] History of OpenGL - OpenGL Wiki. `https://www.khronos.org/opengl/wiki/History_of_OpenGL`, accessed: 2016-1-4.

[21] Download DirextX End-User Runtime Web Installer from Microsoft Download Center. `https://www.microsoft.com/en-us/download/details.aspx?id=35`, accessed: 2016-1-6.

[22] Vulkan - Industry Forge. `https://www.khronos.org/vulkan/`, accessed: 2016-1-6.

[23] `https://www.qt.io/about-us/`, accessed: 2016-1-4.

[24] `https://www.qt.io/qt20/`, accessed: 2016-1-4.

[25] Blanchette, J.; Summerfield, M.: *C++ GUI Programming with Qt 4*. Prentice Hall, první vydání, 2006, ISBN 0-13-187249-4.

[26] `https://wiki.qt.io/Releases`, accessed: 2016-1-4.

[27] Nana C++ Library - a modern C++ GUI Library. `http://nanapro.org/en-us/`, accessed: 2016-1-7.

[28] wxWidgets: Cross-Platform GUI Library. http://www.wxwidgets.org/, accessed: 2016-1-7.

[29] GTK Project. https://www.gtk.org/, accessed: 2016-1-7.

[30] (IUCr) File syntax. http://www.iucr.org/resources/cif/spec/version1.1/cifsyntax, accessed: 2016-1-17.

[31] IEEE: IEEE Standard for Floating-Point Arithmetic. http://ieeexplore.ieee.org/document/4610935/, 2008.

[32] OpenGL Mathematics. http://glm.g-truc.net/0.9.8/index.html, accessed: 2016-1-15.

Příloha **A**

Seznam použitých zkratek

GUI Graphical user interface

CIF Crystallographic Information File

PDB Protein Data Bank

Příloha **B**

Obsah přiloženého CD

```
readme.txt ..................................... stručný popis obsahu CD
data ........................................... adresář s testovacími daty
exe ......................... adresář se spustitelnou formou implementace
src
  impl ....................... zdrojové kódy implementace, VS projekt
  thesis ..................... zdrojová forma práce ve formátu LaTeX
text ..................................................... text práce
  thesis.pdf ............................ text práce ve formátu PDF
  thesis.ps ............................. text práce ve formátu PS
uživatelská příručka ......... instalace, spuštění a práce s programem
```