ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název: Aplikace pro vizualizaci pojmů a metod z lineární algebry
Student: Martin Chvátal
Vedoucí: Ing. Karel Klouda, Ph.D.
Studijní program: Informatika
Studijní obor: Teoretická informatika
Katedra: Katedra teoretické informatiky
Platnost zadání: Do konce letního semestru 2016/17

Pokyny pro vypracování

1. Seznamte se s knihovnou Qt v jazyce C++ a jejími možnostmi pro zobrazování 2D a 3D objektů.
2. Projděte standardní látku vysokoškolských kurzů lineární algebry a najděte metody vhodné pro vizualizaci; zaměřte se zejména na aplikace těchto metod.
3. Vytvořte knihovnu tříd a funkcí potřebných pro implementaci těchto metod.
4. Vytvořte aplikaci využívající knihovnu z bodu 3 splňující následující požadavky:
a. Implementované metody bude možné aplikovat na uživatelem zadané vstupy (data, matice, parametry atp.).
b. Průběh výpočtů bude v rámci možností názorně vizualizován.
c. V aplikaci bude dostupná sada úkolů, jejichž splnění bude vyžadovat implementaci vybraných objektů a numerických operací lineární algebry (násobení a inverze matic, GEM, apod.) uživatelem.
d. Implementované metody a jejich aplikace budou řádně vysvětleny a popsány.
e. Bude možné přepínat mezi českým a anglickým jazykem.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

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

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 17. prosince 2015

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra Teoretické Informatiky

Bakalářská práce

Aplikace pro vizualizaci pojmů a metod z lineární algebry

Martin Chvátal

Vedoucí práce: Ing. Karel Klouda, Ph.D.

17. května 2016

Poděkování

Chtěl bych poděkovat všem, kteří mi odbornými radami nebo psychickou podporou pomohli při psaní této práce. Zvlášť bych chtěl poděkovat Ing. Karlu Kloudovi, Ph.D., který tuto práci důsledně vedl a upozornil mne na mnohé gramatické a faktické chyby.

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 17. května 2016 .....

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2016 Martin Chvátal. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí, je nezbytný souhlas autora.

Odkaz na tuto práci

Chvátal, Martin. *Aplikace pro vizualizaci pojmů a metod z lineární algebry.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Předmětem této práce je vytvoření výukové aplikace pro lineární algebru, která umožní vyučujícím doplnit výklad o ukázku použití probíraných témat v informatice. Pro studenty je nachystán soubor programovacích úloh, na kterých si osvojené vědomosti mohou procvičit.

Klíčová slova Lineární algebra, Qt, C++, OpenGL, Výukový software

Abstract

The topic of this thesis is creation of an educational software for linear algebra. It allows a teacher to supplement a lecture with example of how the currently studied topic can be used in informatics. There is a set of programming tasks prepared for students, to help them practice what they learned.

Keywords Linear algebra, Qt, C++, OpenGL, Educational software

Obsah

Úvod **1**

1 Cíl práce **3**

2 Literární rešerše **5**
2.1 Překážky ve výuce lineární algebry 5
2.2 Výuka lineární algebry v informatice 6
2.3 Shrnutí 7

3 Analýza **9**
3.1 Funkcionalita 9
3.2 Programovací jazyk 9
3.3 Příklady a cvičení 10

4 Implementace **17**
4.1 Aplikace 17
4.2 Knihovna 20
4.3 Studentův kód 21

5 Návod na rozšíření **25**
5.1 Přidání překladu 25
5.2 Přidání příkladu 26
5.3 Přidání objektu do knihovny 27

6 Budoucí práce **29**
6.1 Aktuální nedostatky 29
6.2 Plánovaná funkcionalita 30

Závěr **33**

Literatura **35**

A Seznam použitých zkratek **37**

B Obsah přiloženého CD **39**

Seznam obrázků

3.1 Korekce perspektivy 14

4.1 Toolbar 17

Úvod

Tato práce se zabývá vytvořením výukové aplikace, která poskytuje pomůcku jak pro vyučující, tak pro studenty lineární algebry. Cílí na studenty informatiky. Ukazuje využití vybraných témat lineární algebry na programovacích úlohách, se kterými je možno setkat se v praxi. Aplikace je navržena tak, aby byla snadno rozšiřitelná a spustitelná na nejpoužívanějších operačních systémech.

Úlohy, které jsou součástí aplikace, cílí kromě procvičení lineární algebry také na zlepšení a procvičení programovacích schopností studenta. Snaží se o motivaci studentů výběrem úloh z atraktivních oblastí informatiky, jako je počítačová grafika, úprava fotografií, data mining a kódování. Úlohy studentům umožnují implementovat logické jádro řešeného problému bez toho, aby se museli zabývat frontendem.

V první kapitole jsou popsány cíle práce. Druhá kapitola zkoumá literaturu týkající se výuky lineární algebry a problémů s výukou spojených. Ve třetí kapitole je popsána funkcionalita programu a jeho jednotlivých součástí. S přihlédnutím k cílům práce je zde diskutován návrh aplikace a technologie zvolené pro implementaci. Čtvrtá kapitola do hloubky popisuje implementaci jednotlivých součástí programu. Sekce této kapitoly se postupně věnují hlavní aplikaci, knihovně pro vykreslování 3D grafiky a knihovně se zdrojovými kódy určenými jako implementační úlohy pro studenty. V páté kapitole je popsány postup pro rozšíření funkcionality programu. V poslední kapitole je nastíněn výhled budoucí práce na aplikaci, včetně aktuálních nedostatků.

Kapitola 1

Cíl práce

Cílem této práce je vytvořit aplikaci pro podporu výuky lineární algebry se zaměřením na studenty informatiky. Aplikace bude ukazovat využití lineární algebry na praktických příkladech a bude tak sloužit k usnadnění pochopení látky a motivaci studentů. Součástí bude také knihovna tříd a funkcí použitých pro vizualizaci vybraných příkladů.

Implementované příklady budou vybrány tak, aby pokryly hlavní oblasti lineární algebry, s důrazem na ty, které se nejčastěji využívají v informatice. Součástí každého příkladu bude text dávající ho do souvislosti s teorií. Některé příklady budou spočívat v implementaci konkrétní funkcionality uživatelem a následného ověření správnosti jeho řešení.

Aplikace bude nabízet možnost změny jazyka. Plánované překlady jsou čeština a angličtina, s možností jednoduchého přídání dalších jazyků dle potřeby. Pro co nejšírší použitelnost bude aplikace napsána za pomoci multiplatformní knihovny Qt, umožňující kompilaci jak pro unixové systémy, tak pro OS Windows.

Kapitola 2

Literární rešerše

Hlavním cílem práce je vytvoření nástroje, který pomůže s výukou a se studiem lineární algebry. První část kapitoly se zaměřuje na překážky, které s výukou lineární algebry souvisejí a na řešení, která byla navržena a vyzkoušena k jejich odstranění. Druhá část této kapitoly zkoumá specifika výuky lineární algebry pro studenty informatiky.

2.1 Překážky ve výuce lineární algebry

Mnoho vysokých škol a výzkumných skupin na celém světě se zabývá problémy a možnostmi zlepšení výuky lineární algebry. Jednou z takových skupin byla počátkem devadesátých let *the Linear Algebra Curriculum Study Group (LACSG)*, která vydala doporučení k výuce lineární algebry na vysokých školách v USA[1]. V tomto doporučení zdůrazňují, že sylabus a způsob prezentace předmětu musí odpovídat studijnímu programu. Měly by tedy být nějaké rozdíly v tom, jak se látka předloží studentům informatiky, fyziky, matematiky atd., aby se zdůraznil význam pro konkrétní obor. Dále doporučují, aby první kurz lineární algebry byl zaměřen na matice a zdůrazňují přínos užití moderních technologií jako pomůcek při výuce.

Na tato doporučení ve svém článku[2] navázal Guershon Harel z Purdue University. Ve své práci definuje dva pojmy. Definici konceptu, což je verbální definice, jaká se vyskytuje v učebnici nebo je studentovi předložena vyučujícím a představu konceptu, která je jistým mentálním schématem skládajícím se z associací k danému konceptu v mysli osoby a úkonů proveditelných s konceptem, které se člověku vybaví. To může zahrnovat věci jako analogie a vztahy k jiným konceptům, příklady a postupy řešení problémů s konceptem spojených. Mnoho studentů spoléhá pouze na memorizaci definic. To často vede k nesmyslným odpovědím na otázky u zkoušek, pokud student danou definici zapomněl. Dále se takový způsob učení projevuje rychlým zapomenutím látky po absolvování předmětu. Student s ucelenou představou konceptu je mnohdy po zapomenutí definice schopen ji sám znovu sestavit. Student, který

má ucelenou představu, je o konceptu schopen mluvit bez zklouzávání k použití formálních definic. Mezi autorovými doporučeními, jak zlepšit výuku, je použití nějakého programu nebo balíčku pro práci s maticemi, jako je například Matlab.

Problémům s lineární algebrou se věnovali také ve Francii, kde po dobu pěti let probíhal výzkum největších problémů a překážek studentů s tímto předmětem. Hlavními stížnostmi studentů byl častý výskyt formalismů, velké množství definic a nedostatek propojení s dosud nabytými znalostmi.[3] Problém s vizualizací konkrétních konceptů lineární algebry řešili ve svém článku[4] vyučující z Atatürk University v Turecku. Ve svém výzkumu vytvořili dvě skupiny po třiceti studentech. Výuka měla 3 hodiny týdně, z nichž skupina A měla dvě hodiny založeny na geometrické interpretaci látky a jednu hodinu na algebraické reprezentaci. Skupina B měla tento poměr opačný. Na konci kurzu dostali studenti test obsahující otázky zadané jak algebracikým popisem, tak geometrickým popisem. Skupina A, která věnovala víc času výuce za pomocí geometrických struktur, dosáhla lepších výsledků u obou typů zadání.

2.2 Výuka lineární algebry v informatice

Všechny výše uvedené zdroje se schodují v tom, že je třeba látku studentům předložit za pomocí něčeho, co je jim blízké nebo již známe. Jak zdůrazňuje LACSG[1], je nutno přizpůsobit výuku studijnímu programu, v tomto případě informatice. Je tedy nutné se podívat na oblasti informatiky, kde se lineární algebra používá.

V nedávných letech se tomuto tématu věnovala americká komise *2013 ACM/IEEE-CS Computer Science Curricula Joint Task Force* a ve své zprávě [1] zmiňuje jako oblasti využití lineární algebry vyhledávání na internetu, analýzu grafů, strojové učení, grafiku, bioinformatiku, vědecké výpočty, data mining, rozpoznávání hlasu, kompilátory a paralelní programování. Doporučení pro zlepšení výuky lineární algebry rozděluje do dvou skupin. První skupinou jsou doporučení pro informatické předměty, jako například ve větší míře seznamovat studenty s praktickým využitím lineární algebry v jednotlivých předmětech a vkládání konceptů z lineární algebry do předmětů zabývajících se algoritmy a datovými strukturami (algoritmus násobení matic, vyjádření dat maticí sousedností, SVD...). Druhou skupinou jsou doporučení pro výuku matematiky, například používání běžných úloh z oblasti informatiky v hodinách lineární algebry (analýza grafu, 3D transformace, rozpoznávání hlasu...) a kombinování výuky matematiky s použitím a výukou programovacích jazyků (Python, R, Matlab...).

Vůbec nejobsáhlejším dílem posledních let, které se věnuje výuce lineární algebry pomocí programování, je kniha *Coding the Matrix* od Philipa Kleina[5]. Každá kapitola je doplněna programovacím cvičením v jazyce Python, díky kterému si student nabyté teoretické dovednosti ihned vyzkouší

v praxi a vytvoří si asociaci s daným způsobem využití probíraného konceptu. Autor zdůrazňuje význam použití jazyka, který nemá sám o sobě podporu matic a vektorů jako datových typů. Díky tomu je na studentovi, aby nejdřív naprogramoval tyto základní objekty a jejich funkcionalitu a tím upevnil jejich správné pochopení. Základem cvičení je kombinace jejich jednoduchosti a smysluplnosti. Student pracuje na reálně použitelných příkladech a ne pouze na teoretických, na praxi špatně navázatelných úlohách. Dle autorových zkušeností vnímají studenti tuto praktickou část jako nejzábavnější, což zvyšuje jejich motivaci věnovat čas lineární algebře.

2.3 Shrnutí

Nejčastěji zmiňovaným způsobem, jak zlepšit výuku, je zdůraznění významu pro daný obor, v tomto případě informatiku. Často zmiňované je použití moderních technologií pro výuku, v případě studentů informatiky zkombinování s výukou programovacích jazyků. Tato práce se zakládá na těchto doporučeních a!snaží se o!přístup, který studenta neseznamuje nejdřiv s teorií a definicemi, které se později aplikují na učebnicové příklady. Volí přístup opačný. Nejdřív je studentovi ukázán nějaký problém nebo jev, který může student prozkoumat a spojit si tak název daného konceptu lineární algebry primárně s problémem a využitím, místo toho, aby se jako první vytvořila asociace s definicí nebo důkazem.

Jakmile student pochopí, k čemu se dá daný koncept využít, může si vyzkoušet jak ho k danému účelu použít. K tomu slouží sada programovacích úloh, ve kterých se student pokusí implementovat funkcionalitu takovým způsobem, aby odpovídala referenčnímu řešení. Student je oproštěn od složitějších implementačních detailů, jako je GUI aplikace, vykreslování a vizualizace. Stačí, aby naprogramoval klíčovou funkcionalitu související s lineární algebrou. Propojení s výukou programovacího jazyka je řešena tím, že zároveň s postupně se zvyšující náročností lineární algebry se zvyšuje i obtížnost programovací části z hlediska nároků na znalost programovacího jazyka.

Kapitola 3

Analýza

Tato kapitola diskutuje funkcionalitu aplikace, výběr programovacího jazyka a vybrané příklady a cvičení, které jsou součástí aplikace.

3.1 Funkcionalita

Jedním z cílů je vytvořit nástroj pro vyučujícího, pomocí kterého by mohl doplnit výklad vizualizací pobíraného konceptu a na praktických příkladech ukázat využití v různých informatických oborech. Ty samé příklady může student použít při samostudiu.

Druhým cílem je vytvoření sady programovacích úloh, v rámci kterých student sám doplní požadovanou funkcionalitu, čímž aplikuje probíranou látku a zároveň si procvičí znalosti objektového programování a daného programovacího jazyka. Základem pro tyto úlohy jsou předpřipravené šablony, kde jsou definovány vstupy, výstupy a pokyny pro vypracování. Zároveň je oddělen jakýkoliv kód, který student nutně nepotřebuje k naprogramování úlohy.

Aplikace umožňuje zvolit jazyk. Základní výběr je mezi anglickým a českým jazykem. Návrh umožňuje snadné přidání nových překladů jednoduchou úpravou kódu, aby bylo možno aplikaci zpřístupnit co nejvyššímu počtu uživatelů. Ze stejného důvodu je aplikace napsaná multiplatformním způsobem, omožňujícím kompilaci pro všechny hlavní desktopové platformy.

3.2 Programovací jazyk

Pro potřeby této práce je vhodné zvolit jazyk, se kterým příjdou studenti v průběhu studia do styku, aby se kromě znalostí lineární algebry procvičil i tento programovací jazyk. Objektové paradigma se zdá být vhodnou volbou. V mnoha studijních programech informatiky se studenti s nějakým objektově orientovaným jazykem setkávají před nebo zároveň s předmětem lineární algebry. Z možných variant se jako nejvhodnější nabízí jazyky Python, Java

a C++. Jazyk Python není na Českých vysokých školách tak rozšířený jako první programovací jazyk jako třeba v USA, kde se stává nejčastějí používaným prvním jazykem ve výuce[6]. Zároveň pro něj již existuje kniha a kurz *Coding the Matrix*[5]. Nakonec byl zvolen jazyk C++, díky vhodným knihovnám a blízké podobnosti s jazykem C, který také bývá častým prvním programovacím jazykem pro studenty vysokých škol. Například na ČVUT FIT je v prvním semestru studia vyučován jazyk C a ve druhém semestru pak C++, společně s lineární algebrou. Cvičné příklady v C++ tedy pomůžou procvičit látku z obou předmětů. K řešení úloh z témat lineární algebry probíraných k začátku semestru je zapotřebí méně pokročilých znalostí C++ a u každého dalšího příkladu roste obtížnost jak programovací, tak matematické části.

3.3 Příklady a cvičení

3.3.1 Použití skalárního součinu pro data mining

Obdobný příklad je zmíněn v [5]. Spočívá v práci nad vhodným souborem dat, jako jsou například záznamy o hlasování jednotlivých poslanců. Údaje o jednotlivých úkonech, či vlastnostech (v tomto případě hlasováních) jsou vyjádřeny diskrétními čísly. Z těchto čísel se pro každý element (poslance) vytvoří vektor a skalárním součinem mezi dvěma takovými vektory se dá zjistit podobnost mezi dvěma elementy (v tomto případě měříme, jak si navzájem odpovídají hlasování poslanců). Výsledek lze vizualizovat například blízkostí bodů na ose.

Data jsou reprezentována jménem poslance a vektorem čísel, které nabývají hodnot -1, které reprezentuje hlasování proti, 1, které vyjadřuje hlas pro a 0, která znamená zdržení se hlasování. Pokud označíme vektor odpovídající hlasování jednoho poslance jako $\mathtt{x}$ a vektor hlasování druhého poslance jako $\mathtt{y}$. Standardní skalární součin je definován jako

$$\mathtt{x} \cdot \mathtt{y} = \sum_i x_i y_i$$

Pokud uvažujeme i-té hlasování, pak je daná část sumy rovna 1, pokud se hlasování obou poslanců shoduje. Je li rozdílné, je část sumy rovna -1 a pokud se některý z nich zdržel hlasování, je výsledkem nula. Čím vyšší výsledek, tím podobnější hlasování obou poslanců bylo.

Příklad je vhodný pro základní procvičení práce s vektory a ukázku, že vektor nemusí být z představitelného geometrického prostoru. Zároveň je vhodný i pro začátečníky v objektovém programování. Jelikož je tento příklad určen hlavně jako úvodní a motivační, tak programovací část pro studenta spočívá pouze v implementaci jedné metody a to skalárního součinu dvou vektorů. Na tomto jednoduchém příkladu si bude student moci vyzkoušet kompilaci a práci s programem, bez toho, aby byl zbytečně zatěžován složitostí úlohy.

3.3.2 Transformace ve 2D prostoru

Velkou a atraktivní oblastí využití lineární algebry je počítačová grafika. Na příkladu pohybu auta ve dvourozměrném prostoru lze ukázat princip použití násobení vektoru maticí za účelem rotace, posunutí, změny měřítka a zkosení. Student dostává možnost si procvičit tyto operace tím, že třídě reprezentující matici přidá tuto funkcionalitu a uvidí, jestli je implementace správná, podle toho, jestli bude auto reagovat na ovládání klávesnicí stejně, jako vzor na druhé polovině obrazovky.

Pro transformace ve 2D prostoru se používá matice o třech sloupcích a třech řádcích. Pro první dvě transformace ze tří, které potřebujeme, rotaci kolem počátku a přiblížení (a oddálení), nám stačí matice o dvou řádcích a dvou sloupcích. Rotace vektoru nebo bodu $\mathbb{u} = (x, y)$ ve směru hodinových ručiček o θ stupňů kolem počátku lze docílit vynásobením vektoru transformační maticí ve tvaru

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}.$$

Přiblížení a oddálení s centrem v počátku s koeficientem a ve směru osy x a koeficientu b ve směru osy y je dosaženo transformační maticí ve tvaru

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}.$$

Rozšířenou matici pak potřebujeme pro třetí operaci, kterou je translace, neboli posunutí. Pokud chceme vektor posunout o m ve směru osy x a n ve směru osy y, pak potřebujeme původní bod $\mathbb{u}$ rozšířit na $\mathbb{u} = (x, y, 1)$ a pro vynásobení použijeme následující matici

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & m \\ 0 & 1 & n \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}.$$

Translace nemá na vektory žádný vliv, proto se pří afinních transformacích, narozdíl od bodů, nerozšiřují o jedničku, nýbrž o nulu. V tomto příkladu nakonec kombinujeme všechny matice do jedné. Z prvních dvou matic uděláme afinní tím, že první dva řádky doplníme nulou a dospod přidáme řádek $(0, 0, 1)$. Nyní jsou matice stejné velikosti a je možno je vynásobit. Je třeba brát ohled na pořadí násobení. Například použití translace jako první by znamenalo následnou rotaci okolo jiného bodu než počátku. Pro výsledek zamýšlený v tomto příkladu je tedy správné pořadí nejdřív přiblížení, pak rotace a nakonec posunutí.

Studentovi je předložen zdrojový kód obashující kostru třídy reprezentující matici. Jeho úkolem je implementovat funkci, která ze zadané rotace v stupních, přiblížení a translací v osách x a y vytvoří matici 2D transformace. Student může v rámci úlohy implementovat libovolné funkce a přidat

nové proměnné do třídy. Úloha vybízí k tomu, aby student pro zjednodušení své práce a vytvoření přehledné implementace využil přetěžování operátorů a procvičil si objektové programování. Referencnčí a studentova matice jsou pro snadné porovnání ukázány na obrazovce.

3.3.3 Samoopravné kódy

Příklad slouží hlavně pro ukázku fungování Hammingova kódu. Uživatelem zadaný text je převeden do bitové reprezentace, která je pak zakódována pomocí generující matice ze čtveřic bitů do sedmic. Uživatel může nastavit chybovost přenosu v procentech, která je aplikována na zakódované bity. Stiskem tlačítka je možno barevně zvýraznit chybné sedmice bitů, tak jak je rozpozná kontrolní matice Hammingova kódu. Další tlačítko pak provede opravu kódu. Zároveň se zobrazují i dekódované bity a text. Uživatel tak může porovnat podobu bitů a textu před a po aplikaci chyby přenosu a po opravení pomocí Hammingova kódu.

Hammingův kód (7,4) kóduje řetězce čtyř bitů do sedmi bitů. Původní bity doplňuje třemi paritními bity, které slouží k detekci a opravě chyb. Všechny pozice, jejichž pořadí je rovno některé mocnině dvou, jsou použity pro paritní bity (v tomto případě 1, 2, 4). Ostatní pozice pak odpovídají původnímu kódovanému slovu (3, 5, 6, 7). Pro Hammingův kód (7,4), kde $\vec{a} = (a_1, a_2, a_3, a_4)$ je kódované slovo, tedy platí

$$\vec{u} = (p_1, p_2, a_1, p_3, a_2, a_3, a_4).$$

Paritní bity jsou určeny tak, aby

$$p_1 \oplus a_1 \oplus a_2 \oplus a_4 = 0 \tag{3.1}$$

$$p_2 \oplus a_1 \oplus a_3 \oplus a_4 = 0 \tag{3.2}$$

$$p_3 \oplus a_2 \oplus a_3 \oplus a_4 = 0. \tag{3.3}$$

Na základě těchto poznatků je možno sestrojit generující matici

$$\mathbb{G}_H = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Pro detekci chyb je zpotřebí kontrolní matice, která po vynásobení kódu vrací syndrom $\vec{s} = (s1, s2, s3)$. Pokud nebyla detekována žádná chyba, je syndrom roven nulovému vektoru. Nastala-li chyba, je binární číslo reprezentované tímto vektorem ukazatelem, na které pozici chyba vznikla. Tato matice má podobu

$$\mathbb{H}_H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

Dekódovací matice je triviální. Stačí vrátit třetí, pátý, šestý a sedmý prvek zakódovaného vektoru.

Pro dekódování a opravu chyb lze po přepnutí použít studentovu implementaci místo té referenční. V této úloze se po studentovi požaduje implementace třídy reprezentující matici libovolných rozměrů nad konečným tělesem se sčítáním a násobením modulo 2. Je zapotřebí implementovat funkcionalitu potřebnou ke kódování a dekódování Hammingova kódu, tedy konstruktor pro matici libovolných rozměrů, změnu prvků uvnitř matic a násobení dvou těchto matic, společně s ošetřením správné velikosti vstupních matic.

3.3.4 Soustava lineárních rovnic

Tento příklad spočívá ve vizualizaci soustavy lineárních rovnic v prostoru a jejího výsledku ve formě průniku rovin. Slouží hlavně k ukázce významu lineární závislosti a nezávislosti ve 3D prostoru. Uživatel může přidat libovolné množství rovin definovaných jejich rovnicí. Pro větší přehlednost je možné nastavovat a měnit barvy jednotlivých rovin. Pomocí klávesnice a myši je možné scénu otáčet.

Každá rovina je vykreslována jako čtverec. Tento čtverec má fixní souřadnice a do správné pozice je posunut pomocí transformační matice. Student bude mít za úkol vytvořit funkci, která pro zadanou rovnici roviny vrací správnou transformační matici pro posunutí výchozího čtverce. Příklad je o něco složitější než předchozí příklad na transformace, protože se tentokrát pracuje ve 3D prostoru.

Stejně jako v příkladu na transformaci ve 2D prostoru, je potřeba pro translaci použít matici o jeden řádek a sloupec větší. Body a vektory se rozšíří o čtvrtý prvek, stejným způsobem jako byl ve 2D prostoru přidán třetí prvek. Pro takto vzniklý vektor $\vec{u} = (u_x, u_y, u_z)$ se translace o vektor $\vec{v} = (v_x, v_y, v_z)$ provede pomocí následující transformace

$$\begin{pmatrix} {u_x}' \\ {u_y}' \\ {u_z}' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_x \\ u_y \\ u_z \\ 1 \end{pmatrix}.$$

Rotace kolem obecného vektoru $\vec{w} = (w_x, w_y, w_z)$ o úhel θ, kde $c = \cos\theta$ a $s = \sin\theta$ se provede transformační maticí

$$\begin{pmatrix} w_x^2(1-c)+c & w_x w_y(1-c) - w_z s & w_x w_z(1-c) + w_y s \\ w_y w_x(1-c) + w_z s & w_y^2(1-c)+c & w_y w_z(1-c) - w_x s \\ w_z w_x(1-c) - w_y s & w_z w_y(1-c) + w_x s & w_z^2(1-c)+c \end{pmatrix}.$$

3.3.5 Úprava perspektivního zkreslení

Úprava perspektivy je technika často používaná u strojového čtení textu, snímání kódů čtečkou a při úpravě fotografií. Příklad v aplikaci lze použít na libovolný obrázek vybraný uživatelem. Na tomto obrázku uživatel vybere čtyřúhelníkovou oblast na obrázku, které se po korekci perspektivy stane obdélníkem. Může to být například dům vyfocený z boku (viz obr. 3.1).

Obrázek 3.1: Ukázka úpravy perspektivy

Základem změny perspektivy je projektivní transformace, ve které jsou přímky jedné roviny mapovány na přímky druhé roviny

$$\mathbb{H} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix}.$$

V odborné literatuře je projektivní transformace označována také jako projektivita nebo homografie. Vyjadřuje, jak se mění vjem pozorovaného předmětu při změně pozice pozorovatele nebo úhlu, ze kterého předmět pozoruje. Pro výpočet matice, která odpovídá této transformaci, je třeba znát souřadnice čtyř bodů $((x_1, y_1), \ldots, (x_4, y_4))$ v jedné rovině a souřadnice bodů jim odpovídajícím v druhé rovině $((x'_1, y'_1), \ldots, (x'_4, y'_4))$. Matice $\mathbb{H}$ se získá vyřešením soustavy rovnic

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 & -y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 & -y_2x'_2 & -x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y'_2 & -y_2y'_2 & -y'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x'_3 & -y_3x'_3 & -x'_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y'_3 & -y_3y'_3 & -y'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x'_4 & -y_4x'_4 & -x'_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y'_4 & -y_4y'_4 & -y'_4 \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{pmatrix} = 0.$$

V praxi se často používá více bodů než 4, pro přesnější výsledky. Body musí být zvoleny tak, aby žádné tři neležely na jedné přímce.

Úloha pro studenta spočívá v implementaci funkce, která si na vstupu bere dva čtyřúhelníky a vrací transformační matici, která mění první čtyřúhelník na druhý čtyřúhelník.

Kapitola 4

Implementace

Práce je rozdělena na tři části - aplikaci samotnou, knihovnu se studentovou implementací a knihovnu tříd a funkcí pro vykreslování 3D scény s objekty vizualizujícími různé koncepty z lineární algebry, jako je například rovina reprezentující lineární rovnici v prostoru.

4.1 Aplikace

4.1.1 Hlavní okno

Hlavní okno aplikace, třída CMainWindow dědí ze třídy QMainWindow z Qt framworku, která sama o sobě poskytuje všechnu běžnou funkcionalitu, která je od okna očekávána. CMainWindow se stará o inicializaci knihoven a překlad aplikace. Hlavní okno má dva důležité prvky. Prvním je lišta s tlačítky (viz obr. 4.1).

Obrázek 4.1: Lišta hlavního okna

Na liště je tlačítko zpět, kterém se lze z obrazovek příkladů vrátit zpět na hlavní výběr. Napravo od něj jsou tlačítka pro výběr jazyka. Tlačítko aktuálně vybraného jazyka je deaktivované. Tlačítko s ikonou knih je aktivní pouze na obrazovce jednotlivých příkladů a otevře část učebnice ve formátu PDF, která vysvětluje teorii aktuálně otevřeného příkladu. Poslední je tlačítko, které zobrazí informace o aktuální obrazovce a popis činností, které se na ní dají dělat.

Druhým prvkem hlavního okna je QStackedWidget, prvek GUI, který má uložené všechny zobrazitelné příklady a obrazovky a umožňuje nastavovat, co se má aktuálně zobrazovat. Díky tomuto načtení a uchování obrazovek

v paměti namísto otvírání a zavírání, je GUI svižnější. Výchozí obrazovka je výběr příkladu.

4.1.2 Výběr příkladu

Třída CTaskSelection je výchozí obrazovka hlavního okna a stará se o přepínání na ostatní obrazovky. Jedná se o seznam tlačítek, každé odpovídající jednomu příkladu. Po zmáčknutí tlačítka se vyšle signál hlavnímu oknu a to změní aktivní obrazovku.

4.1.3 Skalární součin

Příklad na skalární součin spočívá v načtení CSV souboru s datasetem, ve kterém jsou poslanci a jejich hlasování o jednotlivých zákonech. Číslo 1 odpovídá hlasování pro zákon, číslo -1 proti a zdržení se je označené jako 0. Dataset je zobrazen v hlavní oblasti obrazovky. Na straně je pak panel s ovládacími tlačítky.

Z číselných hodnot odpovídajících hlasování u jednotlivých poslanců se sestaví vektory. Skalární součin mezi dvěma takovými vektory pak vyjadřuje podobnost hlasování dvou poslanců. Čím vyšší výsledek, tím vyšší podobnost. Po označení řádku lze kliknout na jedno ze dvou ovládacích tlačítek. Jedno vyhledá poslance s nejpodobnější historií hlasování a druhé naopak najde poslance, který je na opačném konci spektra.

Výsledek je vrácen ve formě dialogu, který ukáže jak referenční výsledek, tak ten, který byl získán pomocí studentovy implementace skalárního součinu.

4.1.4 2D Transformace

Tento příklad ukazuje použití transformační matice k pohybu objektu, v tomto případě modelu auta, podobným způsobem, jako u 2D závodních her. Obrazovka příkladu je rozdělena na tři části. První je třída CCarTransformationWidget, což je vykreslovací plocha znázorňující polohu auta za použití referenční transformační matice. Druhá je stejná vykreslovací plocha, která používá studentovu implementaci a poslední částí je boční panel, na kterém jsou vypsány aktuální transformační matice.

Tato třída také kontroluje, která tlačítko jsou aktuálně stisknuta. Tato informace je posílána oběma vykreslovacím plochám, které rozhodují, o kolik se má změnit směr, pozice a přiblížení aut.

Třída CCarTransformationWidget Zajišťuje vykreslování modelu auta a zpracovává vstup z klávesnice. Vykreslování je implementované pomocí časovače, který je nastaven na 16 milisekund, což přibližně odpovídá šedesáti vykreslením za sekundu. Pokud se jedná o instanci, která odpovídá studentově implementaci, tak je pro získání transformační matice volána funkce z knihovny studentova kódu.

4.1.5 Samoopravné kódy

Obrazovka s tímto příkladem názorně ukazuje použití a funkci Hammingova kódu (7,4). V pravém dolním rohu jsou vypsané generující, kontrolní a dekódovací matice. Do řádku *vstup* může uživatel zadat libovolný text, který je převeden do binární podoby, zobrazené hned pod textem a rozdělené do skupin po čtyř bitech, protože s těmito skupinami Hammingův kód pracuje. V další části je pak podoba binárního řetězce po zakódování Hammingovým kódem, která vznikne vynásobením vektoru čtyř bitů (jedné skupiny) generující maticí, ve skupinách po sedmi. V pravé části okna je posuvník, který simuluje nedokonalost přenosu nastavením chybovosti v procentech. Chybovost reprezentuje pravděpodobnost, s jakou bude každý bit přenesen chybně. Po kliknutí na tlačítko *aplikovat chybovost* se zobrazí binární podoba dříve získaného kódu po náhodné negaci bitů s pravděpodobností rovnou navolené chybovosti. Stisknutím tlačítka *dekódovat* se na získané sedmice bitů s aplikovanou chybovostí použije dekódovací matice a spodní části okna se zobrazí jak dekódovaná binární podoba, opět seskupená do čtveřic, tak dekódovaná podoba původního řetězce. Tlačítko *najít chyby* prohledá s pomocí kontrolní matice sedmice bitů v zakódovaném řetezci, na který byla aplikována chybovost. Pokud je výsledkem vynásobení vektoru reprezentujícího tuto sedmici kontrolní maticí nulový vektor, nebyla detekována žádná chyba. Pokud je výsledek jiný, je tato sedmice zvýrazněna červeně. Následující tlačítko, *opravit chyby*, se pokusí opravit sedmice, ve kterých byla detekována chyba.

4.1.6 Soustava rovnic

Příklad *soustava rovnic* slouží k ukázce geometrického významu soustavy lineárních rovnic. Uživatel může zadat libovolnou rovnici roviny a po stisknutí tlačítka *přidat* se vytvoří čtverec reprezentující část této roviny a vykreslí se ve 3D prostoru. Uživatel může nastavit čtvercům libovolnou barvu pomocí tlačítka. Vykreslování je prováděno přiloženou knihovnou, která využívá OpenGL. Scénu je možné otáčet buď šipkami nebo myší.

Třída CLinearEquationWidget reprezentuje UI patřící k jedné rovnici roviny. Zobrazuje koefficienty rovnice, tlačítko na změnu aktuální barvy a tlačítko na odebrání rovnice. Jako proměnou si třída uvnitř drží ukazatel na třídu CGraphPlane z přiložené knihovnu, která reprezentuje samotný vykreslovaný čtverec.

4.1.7 Oprava perspektivy

Tento příklad ukazuje reálnou aplikaci změny báze. Uživatel může načíst libovolný obrázek ve formátu png nebo jpeg, který se pak zobrazí v hlavní části okna. Tlačítkem na ovládacím panelu okna se dá zapnout výběr čtyřúhelníku. Rohy výběru jsou zvýrazněny kružnicemi, které se dají po obrázku libovolně posunovat a tím zvolit čtyřúhelník, který se po transformaci stane

obdélníkem. Pozice rohů jsou zachovány, takže pokud jsou například levé rohy výběru přesunuty doprava, bude výsledný transformovaný obrázek zrcadlově otočen.

Po stisku tlačítka *opravit perspektivu* se zkontroluje, jestli se strany zvoleného čtyřúhelníku nekříží. Pokud je čtyřúhelník validní, tak se vytvoří transformační matice, která mění bázi takovým způsobem, že se z vybraného čtyřúhelníku stane obdélník. Studentův kód se zavolá pro získání druhé transformační matice. Následně se otevře dialog, který zobrazí vedle sebe dva transformované obrázky. Jeden transformovaný za použití referenční transformační matice a druhý za použití matice vrácené studentovou funkcí.

4.1.8 Změna jazyka

Qt framework výrazně ulehčuje překlad aplikací. Každý textový řetězec, který se zobrazuje uživateli, stačí obalit funkcí tr(). V momentě, kdy je takto obalený řetězec nastavován na nějakém prvku UI, se v souboru s právě nastaveným překladem najde varianta tohoto řetězce a ta se zobrazí. Překlad je uložen ve formě XML v souboru s příponou ts. Tento soubor se kompiluje do binární podoby, která má příponu qm. Ta se pomocí třídy QTranslator nahraje a pomocí funkce installTranslator se dá pokyn aplikaci, aby pro překlad používala právě tento soubor. Pro překlad za běhu programu je potřeba překládat i texty UI prvků, které jsou již zobrazené a tedy mají již překlad načtený. Z tohoto důvodu je každá třída, která obsahuje zobrazitelný text, vybavena funkcí retranslateUI. Ta přeloží všechen zobrazitelný text, který je k dispozici v dané třídě a zavolá retranslateUI na všechny UI prvky touto třídou vlastněné.

4.2 Knihovna

Knihovna pro vykreslování 3D scény využívá API OpenGL. Potřebná verze OpenGL je 3.1 nebo vyšší.

4.2.1 Scéna

C3DScene je klíčová třída knihovny. Jedná se o vykreslovací plochu, která si udržuje seznam všech elementů přidaných do scény. Zpracovává vstup z klávesnice a myši, který mění transformační matice scény. Tato třída také spravuje použití shaderů.

4.2.2 Osa, bod, rovina

CGraphAxis, CGraphPoint a CGraphPlane jsou třídy reprezentující jednotlivé vykreslitelné objekty. Všechny dědí z abstraktní třídy IGraphElement, která definuje společné rozhraní vykreslitelných objektů. V budoucnu bude jednoduché přidat další vykreslitelné objekty pomocí dědění z této třídy.

4.3 Studentův kód

Všechny zdrojové soubory určené pro editaci studentem tvoří vlastní dynamickou knihovnu, která je použita v aplikaci. Projekt je nastaven pro jednoduchou kompilaci pomocí utility qmake, která automaticky generuje makefile. Pomocí qmake příkazu se dá celý projekt i přiložené knihovny snadno znovu zkompilovat.

Každé úloze pro studenta odpovídá jeden zdrojový soubor a k němu přidružený hlavičkový soubor, ve kterých je implementované nutné minimum k příkladu, například hlavičky funkcí a tříd, které se používají pro komunikaci se zbytkem programu. Soubory obsahují komentáře vysvětlující funkcionalitu, kterou má student za úkol implementovat.

4.3.1 dotproduct.cpp

Tento zdrojový kód náleží k prvnímu příkladu, zaměřenému na skalární součin. Jelikož se jedná o úvodní a motivační úlohu, je studentovi předložen jen jednoduchý úkol. Implementovat funkci *dotProduct*, která jako parametry přijímá dva vektory celých čísel a vrací skalární součin těchto dvou vektorů.

4.3.2 transformation2d.cpp

Zdrojový kód k druhé úloze deklaruje jednoduchou třídu reprezentující matici o třech řádcích a třech sloupcích. Třída má dva konstruktory. První přijímá jako parametry 9 čísel, kterými nastaví hodnoty matice. Druhý konstruktor je bez parametrů a vytvoří jednotkovou matici. Funkce *at* pak vrací prvek matice, který se nachází na řádku a sloupci, předané funkci parametry. Funkcí *isImplemented* se ověřuje, jestli se studentova implementace má v aplikaci použít. Od studenta se očekává, že po implementaci změní návratovou hodnotu na true.

Studentovým úkolem je implementovat funkci *setToTransformation*. Vstupem této funkce jsou stupně v celých číslech, o které se má ve směru hodinových ručiček otočit bod vynásobený výslednou maticí kolem počátku. Jako další jsou předána dvě reálná čísla indikující, o kolik se má transformační matice posouvat ve směru osy x a osy y. Posledním parametrem je koeficient zvětšení, ve formě reálného čísla. Je ponecháno na studentově kreativitě, jakým způsobem tuto funkci implementuje. Doporučený postup je přetížit operátor násobení a vytvořit samostatné funkce pro rotaci, translaci a zvětšení, které pak stačí zavolat zasebou.

4.3.3 hamming.cpp

V tomto zdrojovém kódu je studentovým úkolem implementovat matici nad konečným tělesem se sčítáním a násobením modulo 2, která může nabývat libovolných rozměrů. Student měl možnost procvičit si podobný typ úlohy v pří-

kladu na transformace ve 2D prostoru. Z toho důvodu je pro tento zdrojový kód zvolena náročnější forma, kde je známa jen deklarace vnějšího rozhraní třídy. Je na studentovi, aby zvolil třídní proměnné a libovolně implementoval zadané rozhraní, za použití libovolného množství pomocných tříd a proměnných.

Rozhraní třídy obsahuje dva konstruktory. První nastavuje pouze počet sloupců a řádků matice, druhý ještě navíc přijímá jako parametr vektor celých čísel, kterými nově vzniklou matici naplní. Po studentovi se očekává, že součástí implementace bude i kontrola validity parametrů těchto dvou konstruktorů.

Od funkcí *height* a *width* se očekává, že vrátí počet sloupců, respektive počet řádků matice. Funkce insert vloží prvek na specifikované místo v matici. Student má za úkol implementovat dva operátory. Od operátoru násobení se očekává, že bude vracet matici o správných rozměrech a hodnotách korektních pro konečné těleso. Operátor () vrací prvek matice, který se vyskytuje na sloupci a řádku specifikovaných parametry.

4.3.4 linearsystem.cpp

Je zadán čtverec v prostoru s rohy v bodech (1, 1, 0), (1, -1, 0), (-1, -1, 0) a (-1, 1, 0). Studentovým úkolem je implementovat funkci, která dostane na vstupu koeficienty a, b, c, d obecné rovnice roviny $ax + by + cz + d = 0$ a jejím úkolem je vrátit transformační matici, která posune a otočí zadaný čtverec takovým způsobem, aby ležel na této rovině. Ve zdrojovém kódu je poskytnuta deklarace matice, která specifikuje veřejné rozhraní, potřebné k práci s třídou uvnitř aplikace.

Úloha, narozdíl od transformace ve 2D prostoru, vyžaduje zamyšlení nad tím, podle jaké osy a o kolik stupňů zadaný čtverec otočit a jak spočítat velikost posunutí. Student si procvičí prostorovou představivost a znalost analytické geometrie.

4.3.5 perspectivecorrection.cpp

V úloze na rektifikaci perspektivy je deklarována jediná funkce, kterou má student za úkol implementovat. Vstupem této funkce jsou dva čtyřúhelníky, zadané pomocí struktur definovaných v tomto zdrojovém kódu. Úkolem je vracet transformační matici, která přenásobením prvního čtyřúhelníku vytvoří druhý čtyřúhelník. Student si může vytvořit libovolné množství pomocných funkcí. Ve zdrojovém kódu je zahrnuta jednoduchá třída reprezentující matici reálných čísel o čtyř řádcích a sloupcích. Pouze veřejné rozhraní této třídy je specifikováno, a student implementuje zbytek.

Jedná se o nejsložitější přiloženou úlohu. Funkce implementovaná studentem není triviální a má více kroků. Zároveň se od studenta očekává i implemen-

tace pomocných tříd a struktur, které bude k vyřešení potřebovat a k úspěšnému vyřešení úlohy je zapotřebí hlubší pochopení probírané látky.

Kapitola 5

Návod na rozšíření

Tato kapitola se věnuje popisu toho, jak se dá aplikace a knihovna rozšířit o novou funkcionalitu. Aplikace je určena a navržena pro snadnou rozšiřitelnost, aby mohla pružně reagovat na zpětnou vazbu od vyučujících a studentů a přizpůsobovat se budoucím změnám v sylabu lineární algebry. Návod předpokládá aspoň základní znalost jazyka C++. Je doporučena alespoň zběžná znalost frameworku Qt a jeho základních principů, ale díky dokumentaci Qt, která je na velmi vysoké úrovni, by měli jednoduché úpravy zvládnout i uživatelé bez předchozí znalosti tohoto frameworku. Doporučeno je k práci s projektem použít prostředí Qt Creator, které je uzpůzobené k práci s projekty založenými na qmake a má integrovanou dokumentaci ke Qt frameworku.

5.1 Přidání překladu

5.1.1 Vytvoření souboru s překladem

Jako první je zapotřebí získat seznam všech přeložitelných textových řetězců v projektu, v souboru formátu ts. Na systému je nutné mít nainstalovaný framework Qt, který poskytuje konzolový příkaz lupdate. Soubor s textovými řetězci z qmake projektu se získá příkazem *lupdate -pro název-projektu.pro*. Vytvoří se jeden xml soubor s příponou ts pro každý překlad definovaný v projektovém souboru visual-lingebra.pro. Do proměnné *TRANSLATIONS* stačí přidat další název souboru pro každý jazyk, pro který chceme příkazem lupdate vygenerovat soubor. Překladové soubory se dají vygenerovat i z protředí Qt Creator z menu *Tools* → *External* → *Linguist*.

Vygenerovaný soubor lze nejsnadněji upravovat pomocí nástroje Qt Linguist, případně se dají soubory upravovat v libovolném textovém editoru. Po dokončení překladu se použije *lrelease*, druhý příkaz poskytovaný nástrojem Qt Linguist. Jako parametr se příkazu zadá cesta k projektovému souboru a všechny překlady definované v proměnné *TRANSLATIONS* jsou ze

XML souborů převedeny do binárního formátu qm který je vstupem pro třídu QTranslator, která překlad zajišťuje.

5.1.2 Načtení souboru s překladem

Jako první se přidá vytvořený binární soubor do translations.qrc, aby se při kompilaci přidal do binárního souboru aplikace. Do třídy CMainWindow se jako proměnná přidá QTranslator, který v sobě bude držet načtený soubor s překladem. V konstruktoru třídy se pomocí funkce třídní funkce load do této nově vytvořené proměnné načte překladový soubor z translations.qrc.

Pokud se má nově implementovaný jazyk přepínat tlačitkem stejně jako již implementované jazyky, je třeba přidat ikonu tohoto tlačítka do souboru icons.qrc a následně přidat do horní lišty programu instanci třídy QAction s touto ikonou. Posledním krokem je implementace slotu, který po vzoru již existujících slotů nainstaluje v aplikaci daný překlad, a napojení tohoto slotu se signálu stisknutí pro tento překlad vytvořené instance třídy QAction.

5.2 Přidání příkladu

Pro přidání nové obrazovky s příkladem je třeba přidat nový typ do výčtu CTasks v hlavičkovém souboru taskselection.h. Ve stejném souboru se do třídy CTaskSelection přidá deklarace ukazatele na nové tlačítko a slotu, který bude toto tlačítko obsluhovat. V konstruktoru třídy se tlačítko alokuje a následně je třeba ho přidat do proměnné m_vLayout, což tlačítko zobrazí v seznamu na obrazovce s výběrem úloh. Tlačítko se následně propojí s vytvořeným slotem. Implementace slotu je triviální, pouze vyšle signál taskSelected s položkou výčtu CTasks odpovídající vytvářenému příkladu.

Třída reprezentující vytvářený příklad musí dědit ze třídy QWidget. Instance třídy se použije v CMainWindow, kde se v konstruktoru přidá do m_centralWidget. Posledním krokem je rozšíření funkce onTaskSelected, která se volá při přijetí signálu taskSelected ze třídy CTaskSelection, přidáním funkcionality, která při přijetí signálu o výběru nově vytvořeného příkladu tento příklad nastaví jako aktuální widget pro m_centralWidget, čímž se nový příklad zobrazí.

Aby v nově implementovaném příkladu fungovala nápověda a otevření učebnice, je třeba upravit funkce onTextbookSelected a onInfoSelected ve třídě CMainWindow přidáním nové volby, s cestou k přidanému souboru. Pro učebnici jsou použity soubory ve formátu pdf, pro info textové soubory. Nově přidávaná pdf se vkládají do balíčku pdfs.qrc, info soubory do info.qrc.

5.3 Přidání objektu do knihovny

Nekteré budoucí příklady mohou vyžadovat vykreslení jiných 3D objektů než které poskytuje knihovna v aktuální podobě. Pro přidání nového objektu je třeba, aby nově tvořená třída dědila z IGraphElement. Třída musí implementovat funkci draw, která pomocí příkazů knihovny OpenGL nově implementovaný objekt vykreslí.

Kapitola 6

Budoucí práce

6.1 Aktuální nedostatky

Hlavním nedostatkem je absence příkladů na některé koncepty lineární algebry, jako jsou Gaussova eliminační metoda, ortogonalizace a vlastní čísla. Možnými příklady vhodnými k budoucí implementaci je například strojové učení, s testováním hypotézy pomocí metody nejmenších čtverců, implementace PCA nebo implementace pagerank algoritmu za použití vlastních vektorů.

Dalším nedostatkem je, že některé obrazovky aplikace mohou vypadat špatně na obrazovkách s menším rozlišením, například některé větší matice mohou mít z důvodu nedostatku místa zalomené řádky. V budoucí práci bude třeba nastavit minimální šířku jednotlivých komponent UI a pod učité velikosti zmenšovat jen některé části UI. Tlačítka, hlavně ta, která mají pouze ikonu a žádný text, potřebují pro zvýšení čitelnosti a snadnosti použití nastavit nápovědu, která se objeví při podržení myši nad tlačítkem.

V aktuální podobě se chybové hlášku a debugovací výpisy zobrazují do konzole. Součástí budoucí práce je přesunout tyto zprávy so samostatných dialogů, které budou pro uživatele zřetelnější a umožní výběr akce při chybách.

Příklad na korekci perspektivy vrací výsledný obrázek s ne zcela přesným poměrem stran. V budoucnu je třeba vylepšit algoritmus na výpočet rozměrů nového obrázků. Další možností vylepšení tohoto příkladu je možnost zvolit si jak vstupní, tak výstupní čtyřúhelník, místo aktuální implementace, ve které je vybraný čtyřúhelník vždy převáděn na obdélník.

Momentálně je v programu funkcionalita zobrazení učebnice pouze v českém jazyce. Anglická verze bude dodána po nalezení vhodného materiálu na téma lineární algebry, který má dostatečně permisivní licenci použití, k použití v opensource programu.

6.2 Plánovaná funkcionalita

Příklad na skalární součin by v budoucnu měl obsahovat víc funkcionality, včetně zakreslování výsledků do grafu. Například porovnání všech vektorů a jejich zakreslení na ose, včetně proměnlivých odstíný podle vzdálenosti od konců osy.

Úloha a příklad na transformace ve 2D prostoru by v budoucnu měly mít podobu kompletní hry, která by šla v libovolný okamžik pozastavit a výběrem jakéhokoliv objektu ve hře by se zobrazila jeho transformační matice, rozdělená na elementární transformace (rotace, přiblížení, translace). Zároveň by byl příklad rozšířen o kratší úvod do programování her, což by mohlo studenty úspěšně namotivovat k zájmu o příklad.

Vizualizace práce Hammingova kód v aktuální verzi přijímá pouze text zadaný přímo z klávesnice. V rámci budoucí práce bude implementována možnost přijmout vstup ze souboru jak v textové, tak binární podobě a výstup po zakódování nebo dekódování zpět uložit do zvoleného souboru. UI tohoto příkladu je třeba zrevidovat a deaktivovat tlačítka, která v daném stavu nemá smysl použít, například dekódování, pokud ještě nebyl zadán nebo zakódován vstupní text.

V rámci knihovny pro vykreslování za pomoci OpenGL bude v budoucnu implementována třída umožňující načíst libovolný model ze souboru a vložit do scény. V úvahu připadají další úlohy pro studenty, zaměřující se na vykreslování modelů a funkcí s tím spojených, jako je odraz světla, a větší množství úloh na programování některých komponent her, jako je grafika a umělá inteligence, například vyhledávání nejkratších cest pomocí maticových algoritmů.

Na základě zpětné vazby od uživatelů budou prováděny opravy a úpravy aktuální funkcionality, případně implementace nové funkcionality.

V úvahu připadá implementace podobných vizualizačních příkladů a úloh i pro jiná odvětví matematiky. Teorie grafů se dá předvést atraktivním způsobem, zároveň s některými algoritmy umělé inteligence, například maticovými algoritmy pro vyhledávání nejkratších cest v grafu, nebo prohledávání nedosažitelných podgrafů. Stejně tak v oblasti počítačových sítí se používají algoritmy postavené na maticích, na kterých by se daly postavit příklady a úlohy pro studenty, na kterých by si procvičili nejen znalosti matematiky a programování, ale zároveň i třetího oboru. Nemusí se však jednat pouze o vysokoškolskou matematiku. Například vykreslování ve 3D prostoru se dá využít pro vysvětlení klasických základoškolsých příkladů na prostorovou představivost, které spočívají ve tvarech poskládaných z krychlí nebo jiných geometrických útvarů. Středoškolská matematika nabízí další oblasti vhodné k vizualizaci, například analytickou geometrii.

Vizualizace pomocí počítačové grafiky, jako pomůcka k výuce, se dá využít i v jiných oblastech, než je samotná matematika. Některé jevy ve středoškolské fyzice by bylo možné poměrně snadno předvést pomocí animací s vysvětlivky, ať už ve 2D nebo 3D prostoru. Pak je zde přímo výuka počítačové grafiky,

pro kterou by bylo snadné naimplementovat vizualizační příklady po rozšíření nynější verze knihovny pro vykreslování v trojrozměrném prostoru.

Závěr

Cílem bylo vytvořit aplikace, která by byla využitelná v hodinách lineární algebry a při jejím procvičování. Obsahuje soubor úloh pro studenty, ale bude zapotřebí zpětné vazby od uživatelů, aby se případně upravila obtížnost předložených úloh, pokud se ukáže být příliš nízká, nebo naopak příliš vysoká.

Některá hlavní témata lineární algebry mezi implementovanými příklady chybí. Jedná se o pokročilejší koncepty a metody, které by mohly být příliš složité na implementaci pro studenta v prvním semestru. Pokud se v budoucnu ukáže, že studenti mají potřebné matematické i programátorské schopnosti a zároveň dostatek času během semestru, budou přidány pokročilé úlohy.

V současnosti je aplikace připravena na testování ve výuce. V případě pozitivní reakce uživatelů by příklady s potenciálem k rozšíření, jako například soustava rovnic a transformace ve dvourozměrném prostoru, byly přepracovány a vylepšeny. Aplikaci je snadné rozšířit a upravit, což umožní flexibilně reagovat na připomínky a návrhy vyučujících, kteří se rozhodnou aplikaci využívat.

Aplikace momentálně umožňuje kompilaci na různých operačních systémech, ale je třeba instalovat framework Qt, což může na některých platformách být zdlouhavé a zabírat zbytečně velké množství místa na disku, zejména pokud chce uživatel použít pouze vizualizaci a nemá zájem implementovat programovací úlohy. Pro toto použití bude třeba vytvořit instalační balíček, který bude obsahovat všechno potřebné, bez nutnosti instalovat na systém cokoliv dalšího.

Literatura

[1] Carlson, D.; Johnson, C.; Lay, D.; aj.: The Linear Algebra Curriculum Study Group recommendations for the first course in linear algebra. *College Mathematics Journal*, ročník 24, 1993: s. 41–46.

[2] Harel, G.: The Linear Algebra Curriculum Study Group Recommendations: Moving Beyond Concept Definition. *Resources for Teaching Linear Algebra MAA Notes*, ročník 42, 1997: s. 107–126.

[3] Robert, A.; Robinet, J.: Quelques résultats sur l'apprentissage de l'algèbre linéaire en première année de DEUG. *Cahier de Didactique des Mathématiques*, ročník 53, 1989: s. 328–342.

[4] Konyalioğlu, A. C.; İpek, A. S.; Işik, A.: On the Teaching Linear Algebra at the University Level: The Role of Visualization in the Teaching Vector Spaces. *Journal of the Korea Society of Mathematical Education Series D*, ročník 7, March 2003: s. 59–67.

[5] Klein, P. N.: *Coding the Matrix: Linear Algebra through Applications to Computer Science*. Newtonian Press, první vydání, 2013, ISBN 9780615856735, 530 s.

[6] Guo, P.: Python is Now the Most Popular Introductory Teaching Language at Top U.S. Universities [online]. July 2014, [Cited 2015-12-12]. Dostupné z: `http://cacm.acm.org/blogs/blog-cacm/176450`

Příloha **A**

Seznam použitých zkratek

GUI Graphical user interface

XML Extensible markup language

UI User interface

API Application program interface

PDF Portable document format

OS Operační systém

LACSG Linear Algebra Curriculum Study Group

Příloha **B**

Obsah přiloženého CD

```
readme.txt ..................................... stručný popis obsahu CD
bin ....................... adresář se spustitelnou formou implementace
src ..................................... zdrojové kódy implementace
thesis ..................................................... text práce
  thesis.pdf ........................... text práce ve formátu PDF
  thesis.tex ................... zdrojová forma práce ve formátu LaTeX
```