Software pro výpočet proslunění budov a venkovních prostor

Software for the calculation insolation of buildings and outdoor spaces

Bc. Petr Mahdalíček

2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2009/2010

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Petr MAHDALÍČEK**
Osobní číslo:	**A07335**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Software pro výpočet oslunění budov a venkovních prostor**

Zásady pro vypracování:

1. **Vytvořte literární rešerši na zadané téma.**
2. **Popište principy výpočtových metod definované normou ČSN 73 0581.**
3. **Popište potřeby proslunění obytných prostor definované normou ČSN 73 4301.**
4. **Vyberte a popište knihovny použitelné pro tvorbu aplikace určené k řešení těchto norem.**
5. **Navrhněte a vytvořte aplikaci, která bude tyto normy řešit.**
6. **Vytvořte ukázkový model, proveďte výpočty a prezentujte výstupy z aplikace.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **KAŇKA, Jan. ČSN 73 0581 Oslunění budov a venkovních prostor -- Metoda stanovení hodnot.[s.l.] : 2009, 20 s.**
2. **PAPEŽ, Karel, et al. ČSN 73 4301 Obytné budovy. [s.l.] : 2004, 48 s.**
3. **BLANCHETTE, Jasmin, SUMMERFIELD, Mark. C++ GUI Programming with Qt 4. 2nd edition.[s.l.] : Prentice Hall PTR, 2008. 752 s. ISBN 0132354160.**
4. **JUNKER, Gregory. Pro OGRE 3D Programming (Hardcover). 1st edition. [s.l.] : Apress, 2006. 312 s. ISBN 1590597109.**
5. **Ogre 3D [online]. 2000 [cit. 2010-01-27]. Dostupný z WWW: <http://www.ogre3d.org>.**
6. **Qt [online]. 2008 , 2010 [cit. 2010-01-27]. Dostupný z WWW: <http://qt.nokia.com/>.**
7. **Morley, Keith. Realistic Ray Tracing. 2nd edition. [s.l.] : AK Peters, 2003. 235 s. ISBN 1568811985.**
8. **ANTON, Howard. Elementary Linear Algebra. 8th edition. [s.l.] : Wiley, 2000. 608 s. ISBN 0471170550.**
9. **CHURCHER, Clar. Beginning Database Design: From Novice to Professional (Paperback). [s.l.] : Apress, 2007. 300 s. ISBN 1590597699.**
10. **ABRAMENKO, Peter. Buildings: Theory and Applications. 1st edition. [s.l.] : Springer, 2008. 754 s. ISBN 0387788344.**

Vedoucí diplomové práce: **Ing. Pavel Pokorný, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce: **19. února 2010**

Termín odevzdání diplomové práce: **8. června 2010**

Ve Zlíně dne 19. února 2010

prof. Ing. Vladimír Vašek, CSc.
děkan

L.S.

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Cílem mé diplomové práce je seznámit se s problematikou oslunění a proslunění obytných prostor. Porozumět normám, které se těmito problémy zabývají. Prozkoumat existující aplikace umožňující spočítat oslunění bytů. Vybrat vhodné softwarové prostředky a pokusit se vytvořit vlastní aplikaci určenou k těmto výpočtům.

Klíčová slova:

Proslunění, C++, Qt

ABSTRACT

The aim of my thesis is to understand the issue of insolation of living rooms. Understand the standards that address these problems. Explore existing applications designed to calculate insolation of living rooms. Select appropriate software tools and try to create my own application which will be abble to run these calculations.

Keywords:

Insolation, C++, Qt

Tímto bych chtěl poděkovat Ing. Pavlu Pokornému, Ph.D. za vedení při tvorbě této práce. Děkuji také rodině a všem přátelům za podporu během celé doby studia.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými
  subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu
  využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně .....

podpis diplomanta

OBSAH

ÚVOD

Slunce je středem naší sluneční soustavy a je všemi uznávané jako dárce života. Symbolizuje ducha, který stvořil svět a bez kterého bychom nemohli existovat. Slunce je zdrojem 99% veškeré energie na naší Zemi - a veškerý život na této planetě je na něm neodmyslitelně závislý.

Je dokázáno, že sluneční záření má pozitivní vliv na psychiku člověka. Nelze ho tedy považovat pouze za zdroj světla a tepla, ale nutně také za zdroj dobré nálady. Jistě není náhoda, že obyvatelé zemí z oblasti Skandinávie mají často psychologické problémy, které mnohdy končí nejhorším možným způsobem. Naopak národy žijící tam, kde slunce zapadá jen na krátkou dobu jsou všeobecně považováni za vyrovnané a spokojené.

Z tohoto důvodu je v normách stanovujících zásady pro navrhovaní obytných budov nebo částí obytných budov nemalá část věnována problematice oslunění. Tyto normy určují pravidla, podle kterých se hodnotí, jestli sluneční paprsky dopadají do obytných místností. Určují také minimální časové úseky, po které musí být hodnocený prostor prosluněný. Tyto normy je nutno brát v potaz nejen při výstavbě nových bytů a domů, ale také v případě výstavby budov, které by svým tvarem mohly zabraňovat průniku slunečních paprsků do stávající zástavby.

Vyhodnocováním těchto údajů se zabývají lidé z oboru stavebního projektování. Postup zpracování se skládá z několika kroků. Je nutné vytvořit model hodnoceného prostoru včetně okolní zástavby. Dále je nutné zadat lokalitu, ve které se hodnocený objekt nachází. Z těchto údajů je potom možné spočítat polohu slunce a následně dobu, po kterou sluneční paprsky dopadají do místností obytných prostor.

Usnadněním těchto úkolů se zabývá několik aplikací. Od aplikací řešících tuto problematiku obecně, včetně problémů jako je osvícení umělými zdroji světla, až po aplikace určené výhradně k výpočtu doby proslunění. Žádná z těchto aplikací dostupná v české republice však nepodporuje operační systém linux. Navíc byla v loňském roce aktualizována česká norma řešící oslunění. Výsledkem této diplomové práce by měla být aplikace, která bude respektovat aktuální české normy a bude spustitelná na operačním systému linux.

I. TEORETICKÁ ČÁST

1 POPIS PROBLÉMU

1.1 Norma ČSN 73 4301

Tato norma stanovuje zásady pro navrhování obytných budov nebo obytných částí budov platné pro:

- bytové domy
- obytné části v budovách jiného účelu
- nástavby a přístavby budov, jimiž vznikají nové byty
- rodinné domy
- nástavby a přístavby rodinných domů

Pro tyto objekty popisuje:

- umísťování budov do území
- proslunění budov
- minimální rozměry místností
- vybavení budov

Tato diplomová práce se zabývá pouze částí normy, která popisuje oslunění obytných prostor. V této části jsou definovány základní pojmy této problematiky. Způsoby výpočtu parametrů nutných k správnému ohodnocení budovy. Definuje také minimální hodnoty, které musí bytové jednotky splňovat.

1.2 Norma ČSN 73 0581

Tato norma je rozšířením normy ČSN 73 4301. Podrobně vysvětluje pojmy použité v předchozí normě. Poskytuje alternativní způsoby výpočtu různých parametrů. Popisuje mezní situace, ke kterým může dojít.

1.3 Proslunění bytu

K posouzení proslunění bytu se berou v potaz pouze obytné místnosti (ložnice, obývací pokoj...). Aby byl byt považován za prosluněný, je nutné, aby plocha jeho prosluněných obytných místností byla rovna minimálně jedné třetině všech jeho obytných místností.

Místnost je prosluněna, jsou-li splněny všechny tyto podmínky:

- půdorysný úhel slunečních paprsků a roviny okna musí být nejméně 25°
- okno musí být z průhledného, barvu nezkreslujícího materiálu
- nejmenší rozměr okna musí být alespoň 900 mm
- výška slunce nad horizontem musí být minimálně 5°
- při zanedbání oblačnosti musí být dne 1. března a 21. června doba proslunění nejméně 90 minut. Požadovanou dobu proslunění pro den 1. března lze nahradit bilancí, při které je mimo přestupné roky celková doba proslunění ve dnech od 10. února do 21. března včetně 3600 minut (40 dní s průměrnou dobou proslunění 90 minut)
- sluneční paprsky musí dopadat na kritický bod okna [0]

1.3.1 Kritický bod

Tento bod se používá při výpočtu proslunění místnosti. Je to bod ležící v rovině okna ve výšce 300 mm nad středem spodní hrany osvětlovacího otvoru, nejméně však 1200 mm nad úrovní posuzované místnosti.

1.4 Oslunění venkovních zařízení

Venkovní zařízení a pozemky v okolí obytných budov sloužící k rekreaci jejich obyvatel, mají mít alespoň polovinu plochy osluněnou nejméně 3 hodiny dne 1. března.

1.5 Orientace objektu ke světovým stranám

Při stanovení směru poledníku se přihlíží k meridiánové konvergenci C(°) [0]. Velikost meridiánové konvergence je možné stanovit několika způsoby:

- odečtením z mapového podkladu
- je-li situace zpracována v souřadnicovém systému jednotné trigonometrické sítě katastrální výpočtem ze zeměpisné délky λ(°) dané lokality pomocí vzorce:

$$C = \frac{24\,°\,50 - \lambda}{1{,}34} \qquad (1)$$

- dotazem na středisku geodézie a kartografie

Zjištěný nebo vypočítaný úhel se nanese ve směru hodinových ručiček od svislých souřadnicových čar kartografické sítě mapového podkladu, čímž je určen severní směr.

1.6 Výpočet pozice slunce

K určení přesné polohy slunce je nutné definovat několik pojmů.

1.6.1 Pravý sluneční čas

Pravý sluneční čas (PSČ) [0] vznikne rozdělením časového intervalu mezi dvěma následujícími horními kulminacemi slunce na 24 hodin. K horní kulminaci dochází ve 12 hodin, kdy zároveň v daném dni dosahuje slunce maxima své výšky nad obzorem. V závislosti na pravém slunečním čase se stanoví hodinový úhel τ(°) pomocí vztahu:

$$\tau = 15\left(PS\check{C} - 12\right) \qquad (2)$$

1.6.2 Deklinace

K určení polohy slunce je nejprve nutné spočítat sluneční deklinaci [0]. Deklinace je hodnota závislá na datu výpočtu.

$$\delta = 23{,}45° \sin(0{,}98\,D + 29{,}7\,M - 109)° \tag{3}$$

D – číslo dne v měsíci v datu posuzování

M – číslo měsíce v datu posuzování

1.6.3 Výška

Výška slunce [0] je dána vztahem:

$$\sin(h) = \sin(\varphi)\sin(\delta) + \cos(\varphi)\cos(\delta)\cos(\tau) \tag{4}$$

φ - zeměpisná šířka

δ – sluneční deklinace

τ – hodinový úhel

1.6.4 Azimut

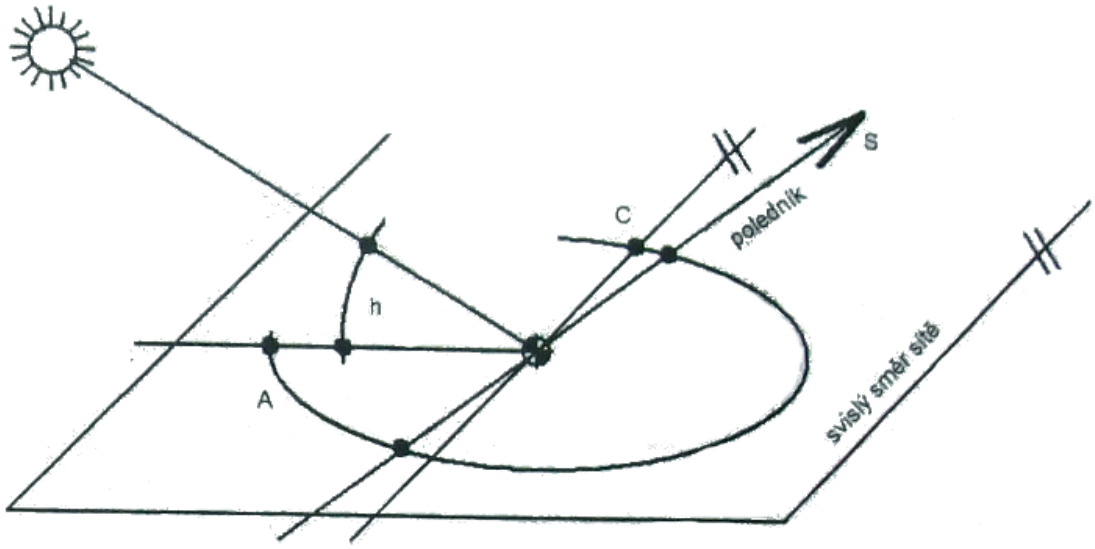
Azimut [0] je úhel nanesený k poledníku ve směru hodinových ručiček.

$$\cos(A) = \frac{\tan(\varphi)}{\cos(h)}\left(\sin(h) - \frac{\sin(\delta)}{\sin(h)}\right) \tag{5}$$

φ - zeměpisná šířka

δ – sluneční deklinace

h – výška slunce



Obrázek 1: Azimut, výška slunce, deklinace

2 C++

C++ [10] je objektově orientovaný programovací jazyk, který vyvinul Bjarne Stroustrup [10] a další v Bellových laboratořích AT&T rozšířením jazyka C. C++ podporuje několik programovacích stylů (paradigmat), jako je procedurální programování, objektově orientované programování a generické programování. Není tedy jazykem čistě objektovým. V současné době patří C++ mezi nejrozšířenější programovací jazyky.

2.1 Standardy

Přestože byl jazyk vyvíjen již od počátku 80. let, první oficiální norma C++ byla přijata v roce 1998, další v roce 2003 (INCITS/ISO/IEC 14882-2003). V roce 2006 a 2007 byly přijaty některé aktualizace.

2.2 Filozofie C++

- je to jazyk, který je stejně efektivní a přenosný jako C
- je navržen tak, aby přímo a komplexně podporoval více programovacích stylů (procedurální programování, abstrakce dat, objektově orientované programování a generické programování)
- je navržen tak, aby programátorovi umožnil zvolit řešení, i když třeba špatné
- je navržen tak, aby byl co možná nejvíc kompatibilní s jazykem C
- neobsahuje funkce, které jsou specifické pro platformu, nebo nemají všeobecné použití
- je navržen tak, aby fungoval bez sofistikovaných programovacích prostředí

2.3 Standardní knihovna

Standard jazyka C++ z roku 1998 se skládá ze dvou částí: popis jazyka a standardní knihovny. Standardní knihovna jazyka C++ obsahuje mírně modifikovanou verzi standardní knihovny jazyka C a Standard Template Library (STL).

STL obsahuje velké množství užitečných datových struktur a algoritmů, jako například vektory (vylepšené pole), spojové seznamy, iterátory, zobecněné ukazatele, (multi)mapy, (multi)sety. Všechny tyto struktury mají konzistentní rozhraní. S použitím šablon je pak možné programovat generické algoritmy schopné pracovat s kterýmkoliv kontejnerem nebo sekvencí definovanou iterátory.

Používání standardní knihovny – například používání std::vector nebo std::string místo polí ve stylu jazyka C – může vést k bezpečnějšímu a lépe škálovatelnému softwaru.

STL byla původně vytvořena a používána firmou Hewlett-Packard a později také Silicon Graphics. Standard se na ni neodkazuje jako na „STL“, ale jen jako na část standardní knihovny, přesto mnoho lidí stále používá tento pojem na odlišení od ostatních částí knihovny.

Většina kompilátorů poskytuje implementaci standardu C++ včetně STL. Existují také implementace standardu nezávislé na kompilátoru (např. STLPort). Jiné projekty také vytvářejí různé zákaznické implementace knihovny jazyka a STL s různými cíli návrhu.

2.4 C++0x

C++0x [11] je neoficiální název plánovaného nového standardu pro jazyk C++. Ten má nahradit stávající C++ standard, ISO / IEC 14882, který byl zveřejněn v roce 1998 a aktualizována v roce 2003. Tyto předchůdci jsou neformálně známé jako C++98 a C++03. Nová norma bude obsahovat několik dodatků k jádru jazyka a rozšíří standardní knihovnu C++. Tato norma však ještě není dokončena, proto se může stát, že název bude nakonec jiný, jelikož 0x z názvu má reprezentovat poslední 2 číslice z roku vydání této normy.

3 QT

Qt [5] ve verzi 4 je svobodná multiplatformní knihovna sloužící primárně (ale nejenom) k vývoji grafických programů. Na první verzi Qt – vyslovuje se jako anglické *cute* (roztomilý), tedy [kju:t] – se začalo pracovat již v roce 1991 a od té doby se framework stále rozšiřuje a zlepšuje. O 3 roky později dva z vývojářů založili společnost Trolltech. V polovině roku 2008 byl Trolltech zakoupen společností Nokia a krátce na to přejmenován na Qt Software.

3.1 Licence

Knihovna Qt je pro vývojáře dostupná ve třech licencích:

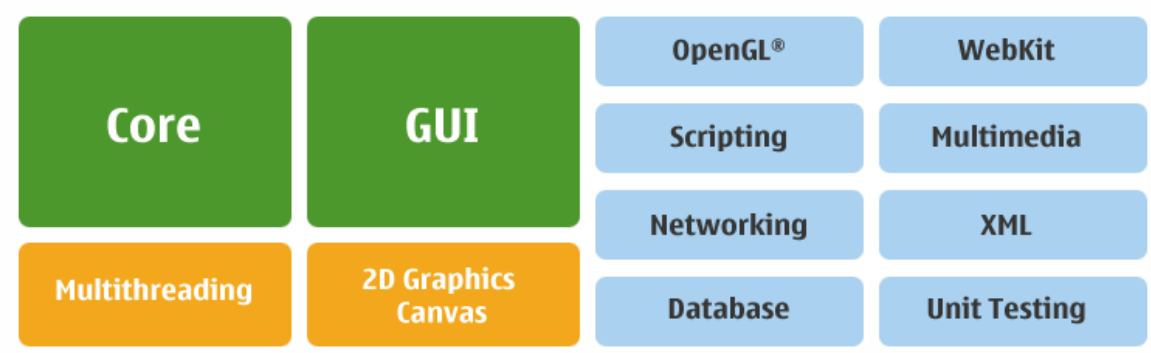
- Komerční licence
- LGPL v. 2.1
- GPL v.3.0

3.2 Charakteristika

3.2.1 Součásti

Knihovna Qt se skládá z několika hlavních částí:

- Jádro
- GUI
- OpenGL
- Skriptování
- Síťování
- Databáze
- WebKit
- Multimedia
- XML
- Unit Testy



Obrázek 2: Struktura knihovny Qt

3.2.2 Kompilace

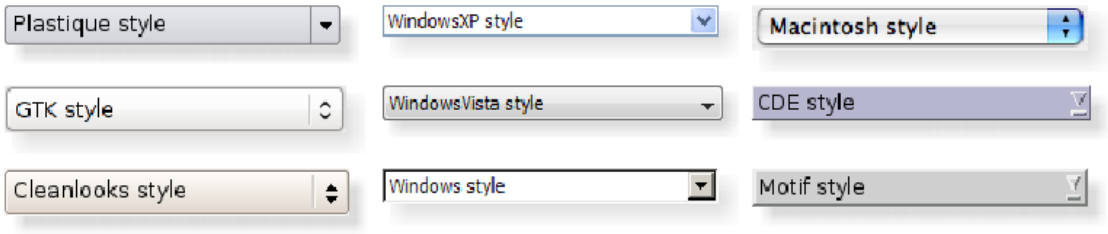
Kompilace Qt projektu se mírně odlišuje od standardního C++ způsobu. Do procesu sestavení programu Qt vkládá tzv. MOC (Meta-Object Compiler) [5]. Toto umožňuje třídám z Qt frameworku používat principy, kterých by za normálních okolností nebylo možné dosáhnout. V praxi to znamená, že před spuštěním kompilace je nutné zavolat příkaz qmake [5].

3.2.3 Signály/Sloty

Každý objekt v Qt4 má nějaké své signály a sloty. Signály, jak již jejich název napovídá, jsou zprávy, které objekt generuje v případě jeho změny nebo při nějaké důležité události (typickým signálem může být například kliknutí myší na tlačítko). Každý signál může být napojen na slot. Slot je pro změnu klasická funkce, která, pokud je připojena k signálu, je zavolána a umožňuje nějakým způsobem měnit objekt.

3.2.4 Widgety

Knihovna Qt obsahuje širokou paletu widgetů. Tyto widgety jsou na každé z podporovaných platforem vykreslovány pomocí nativního API. Díky tomu může multiplatformní aplikace vypadat stejně jako nativní. Lze také používat dialogy z konkrétních platforem.



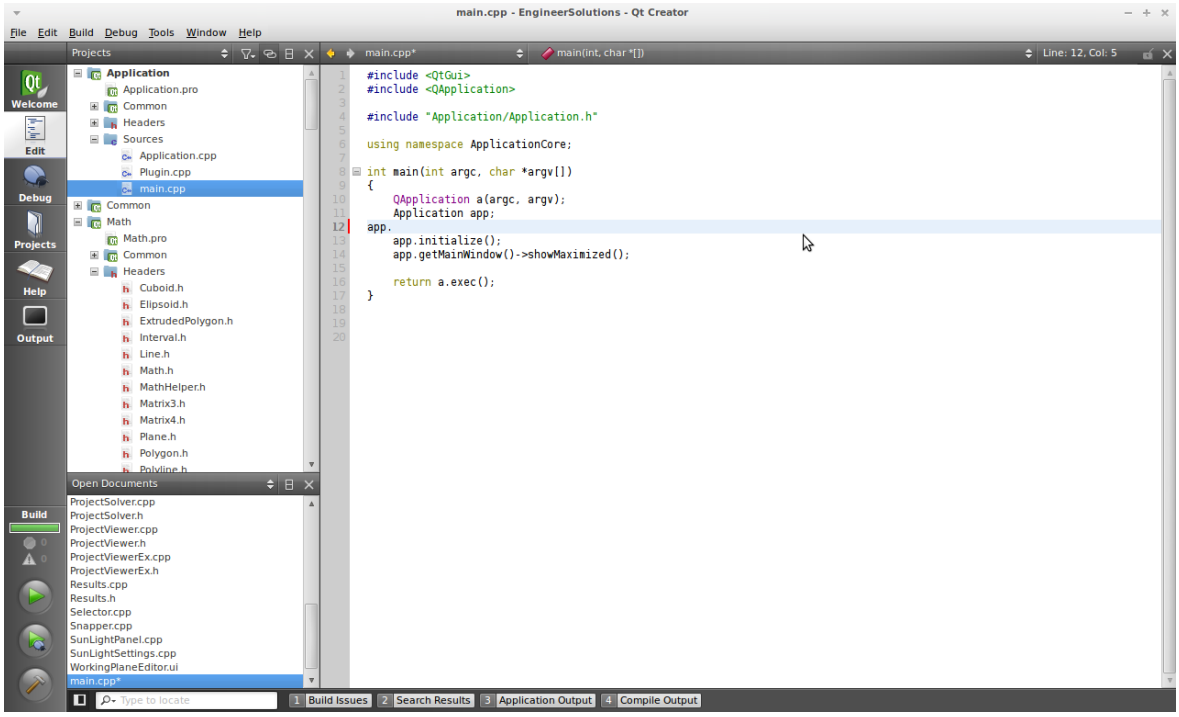
Obrázek 3: Témata vzhledu knihovny Qt

3.3 QtCreator

Qt Creator [5] je relativně nové, ale již velmi schopné IDE určené pro programování za použití C++ a Qt 4. Obsahuje editor s kompletní podporou C++ (doplňování kódu, zvýraznění syntaxe a mnoho dalšího).

Integruje do sebe Qt Designer (Nástroj pro návrh grafického rozhraní. Umožňuje vytvářet grafické rozhraní prostým přetahováním widgetů do okna. Widgetům poté můžete nastavit vlastnosti a snadno je zakomponovat do různých rozložení (layoutů) a následně dle potřeby propojit požadované signály se sloty.)

Kromě toho má i zabudovaný Qt Assistant, takže kompletní dokumentace je vždy po ruce. Tato součást je propojená s editorem tak, že stačí kurzorem myši najet na Qt objekt nebo metodu, zmáčknout F1 a relevantní dokumentace se objeví v panelu hned vedle kódu. Dále je k dispozici debugger (používá se GDB), podpora pro verzovací pluginy (SVN, CVS..)



Obrázek 4: Aplikace QtCreator

3.4 Qt Quick

Qt Quick [5] je vysoko úrovňová knihovna určená k tvorbě uživatelského rozhraní. Umožňuje designérům a vývojářům rychle a snadno vytvářet krásné, perfektní uživatelská rozhraní bez znalosti jazyka C++. Tato technologie bude součástí Qt knihovny od verze 4.7, která bude vydána v první polovině roku 2010.

Platforma Qt Quick se skládá ze 3 hlavních částí:

- QML – QML je deklarativní jazyk určený k popisu uživatelského rozhraní programu, popisuje jak vzhled, tak i chování aplikace
- Qt Creator – vývojové prostředí pro knihovny z rodiny Qt
- QtDeclarative – nový modul knihoven Qt, který umožňuje deklarativní přístup

4 OGRE

OGRE [4] je flexibilní grafický engine napsaný v C++, jehož cílem je umožnit jednodušší a intuitivnější vývoj aplikací využívajících hardwarově akcelerovanou 3D grafiku. OGRE není herní engine, neřeší fyziku, skriptování, logiku. Lze ho ale lehce rozšířit díky množství pluginů, které jsou zpravovány jednak samotnými autory OGRE a jednak širokou komunitou, která kolem tohoto projektu vznikla.

Obrázek 5: Logo knihovny Ogre

4.1 Licence

Od verze 1.7 je OGRE distribuováno pod MIT licencí. Toto je velmi svobodná licence. Jedinou podmínkou kterou klade, je nutnost distribuce textu této licence společně s programem.

4.2 Charakteristika

OGRE lze použít na třech nejrozšířenějších platformách (WINDOWS, LINUX, MAC). Jeho vykreslovací jádro může používat jak OpenGL, tak Direct3D. Zpřístupňuje pro vývojáře všechny funkce potřebné pro tvorbu 3D aplikace:

- načítání shaderů(CG, HLSL, GLSL)
- načítání textur(PNG, JPEG, TGA, BMP nebo DDS)
- načítání geometrie z vlastního formátu
- animace

- prvky scény(uzly, světla, kamery...)
- částicové efekty
- materiály vykreslované pomocí více průchodů

OGRE engine se za dobu své existence stal velmi úspěšným projektem. V dnešní době se jedná o jeden z nejpoužívanějších enginů. Našel své uplatnění jako základ mnoha herních enginů, které používají OGRE pro vykreslování. Je také používán v několika komerčních aplikacích.

5 BOOST

Pod pojmem BOOST [12] se skrývá soubor knihoven určených pro ulehčení programování v jazyce C++. Většina z těchto knihoven je distribuována pod speciální boost licencí. Několik z těchto knihoven bude začleněn do C++0x standartu.

Knihovny jsou navrženy tak, aby umožňovaly použití v široké škále aplikací. Sahají od obecných knihoven jako smart_ptr knihovny, přes abstrakci funkcí operačního systému, až po knihovny zaměřené především na vývoj jiných knihoven.

- Any - obalovací kontejner pro hodnotu libovolného typu. Umožňuje vytvořit pole, v němž budou vedle sebe uloženy proměnné typu int, string atd.
- Date Time - knihovna poskytující datové struktury a metody pro práci s časem
- Exception - knihovna pro práci s výjimkami, umožňuje posílat výjimky napříč vlákny
- Filesystem – knihovna pro práci se soubory a složkami
- Lambda – umožňuje používat malé nepojmenované funkce
- MPL – knihovna určená k meta-programování
- Pointer Container – kontejner pro C++ ukazatele
- Regex – knihovna pro práci s regulárními výrazy
- Serialization – knihovna sloužící k jednoduché serializaci objektů
- Smart Ptr – další knihovna obalovacích kontejnerů pro ukazatele
- System - knihovna poskytující informace o systému
- Uuid – definuje datový typ jednoznačného identifikátoru
- Variant – knihovna podobná knihovně Any, poskytuje kontejner pro libovolný datový typ

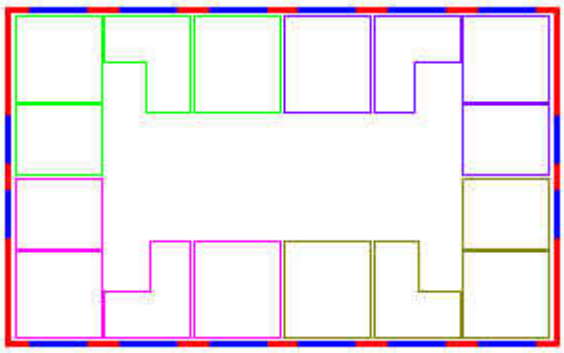
6 SUNLIS

SunList [13] je jedním z programů, které se zabývají prosluněním bytů. Vznikl již v roce 2000. Umožňuje spočítat jak proslunění bytu, tak i venkovních prostor. Tento program podporuje dva způsoby definování scény:

- ručním zadáním souřadnic rohů místností a osvětlovacích otvorů
- propojením s programem AutoCad a definováním souřadnic z tohoto programu

Hodnocení bytů lze provést pomocí dvou kriterií :

- byt musí být prosluněný 90 minut dne 21. března
- byt musí být prosluněný 3600 minut v období od 10. února do 21.března



Obrázek 6: Hodnocený objekt v programu SunLis

II. PRAKTICKÁ ČÁST

7 APLIKACE

V praktické části jsem se pokusil vytvořit aplikaci, která by usnadnila vyhodnocení obytných prostor podle norem řešících problém proslunění.

7.1 Požadavky na aplikaci

Program tedy musí podporovat několik základních operací:

- vymodelování řešeného problému
- označení bytů, místností a oken
- vypočítat proslunění bytů a vyhodnotit podle zmiňovaných norem
- vytvořit protokol o řešení

Z těchto úkonů je pro programátora jistě nejsložitější ten první, tedy modelování. Toto lze v podstatě řešit dvěma způsoby:

1. celou aplikaci napsat jako plugin jednoho z vyspělých modelovacích systémů (AutoCad, BricsCad...)
2. napsat vlastní modelovací prostředí

První řešení je jednoznačně méně náročné jak na čas, tak i na znalosti. Další výhodou by mohlo být, že se uživatel nemusí učit ovládat nové prostředí, pokud tedy již zná použitou hostitelskou aplikaci. Má však i své nevýhody, zejména závislost na aplikací třetí strany. Já jsem se tedy pokusil vytvořit vlastní modelovací prostředí s tím, že podporovány budou pouze nejzákladnější operace.

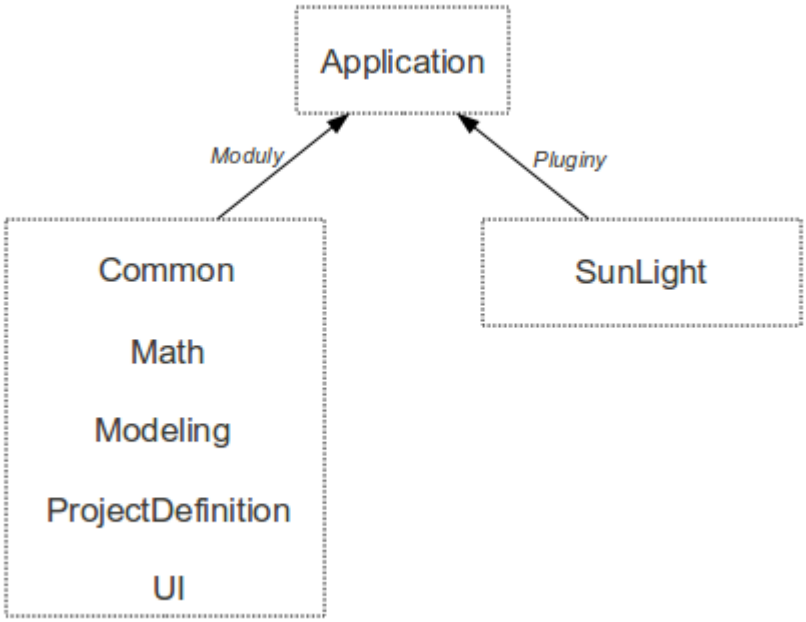
7.2 Použité technologie

Aplikaci jsem programoval s použitím knihoven popsaných v teoretické části v jazyce C++. Knihovnu OGRE ve verzi 1.6.4, Qt ve verzi 4.6, knihovnu Boost ve verzi 1.40. Dále jsem použil knihovnu FastDelegates, která nebyla v teoretické části popsána. Jedná se o knihovnu, která umožňuje použití delegátů v C++.

Vzhledem k tomu, že C++ neposkytuje automatickou správu alokované paměti, použil jsem pro většinu dynamicky alokovaných objektů „chytrý ukazatel“ z knihovny boost shared_ptr. Aplikaci jsem se snažil psát tak, aby byla rozšiřitelná pomocí pluginů. Celý projekt jsem realizoval v prostředí Qt Creator na operačním systému Ubuntu 10.04.

7.3 Moduly aplikace

Aplikace je rozdělena do několika základních modulů. Každý řeší příslušnou oblast.



Obrázek 7: Struktura programu

7.3.1 Common

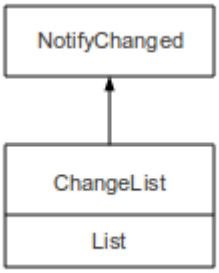
Tento modul obsahuje nejobecnější definice tříd použitelných v mnoha různých situacích.

Mezi nejdůležitější typy z tohoto projektu patří tyto:

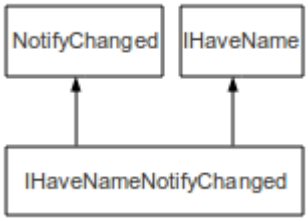
- String / SimpleString – třídy pro práci s řetězci
- Event - třída popisující událost, definuje prvky usnadňující přihlášení a odhlášení z události

- NotifyChanged – základ pro všechny třídy oznamující změny, obsahuje Event pojmenovaný „Changed“
- List – rozšíření std::vector, usnadňuje práci s kolekcemi
- MapEx – rozšíření std::map
- ChangeList – List oznamující změny
- Clonable – abstraktní třída definující klonovací funkce
- Properties – umožňuje uložit pojmenované hodnoty libovolného typu
- Location – reprezentuje pojmenovanou lokalitu
- RegisterEventHandlers – abstraktní třída, definuje metodu, kterou je nutné zavolat po deserializaci tak, aby objekt opět sledoval změny jiných objektů

Obrázek 8: Vybrané třídy



Obrázek 9: Seznam oznamující změny



Obrázek 10: Pojmenovaná položka oznamující změny

Objekt oznamující změny lze vytvořit dvěma způsoby:

1. Skládáním. Tak jako třída ChangeList, která obsahuje List a je odvozena z třídy NotifyChanged, pro přístup k vlastnímu seznamu poskytuje vlastní metody.
2. Odvozením. Například třída IhaveNameNotifyChanged je odvozena z tříd NotifyChanged a IHaveName.

7.3.2 Math

Tento modul obsahuje třídy reprezentující matematické objekty. Místo psaní vlastních tříd pro reprezentaci pozice, transformace a natočení jsem použil třídy použité v knihovně OGRE. Toto je nejen praktické řešení, ale zároveň odstraňuje režii nutnou při převodu z vlastních datových typů. Dále jsou zde definovány třídy reprezentující geometrické objekty:

- Line - úsečka, polopřímka, přímka
- Polyline – skupina navazujících úseček
- Polygon
- Extruded polygon – polygon vytažený do výšky
- Cuboid – reprezentuje kvádr, krychli
- Elipsoid – reprezentuje kouli, elipsoid
- Rovina
- Interval

Tyto objekty jsou jednoduché datové struktury, změny těchto objektů nelze sledovat.

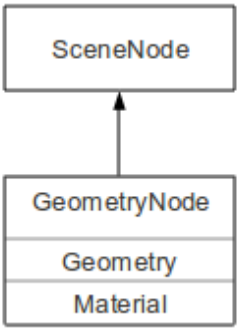
7.3.3 Modeling

V tomto modulu se nachází objekty, ze kterých se skládá scéna reprezentující řešenou situaci. Základní prvek scény je uzel. Z něho je odvozen geometrický uzel, který obsahuje geometrii a materiál.

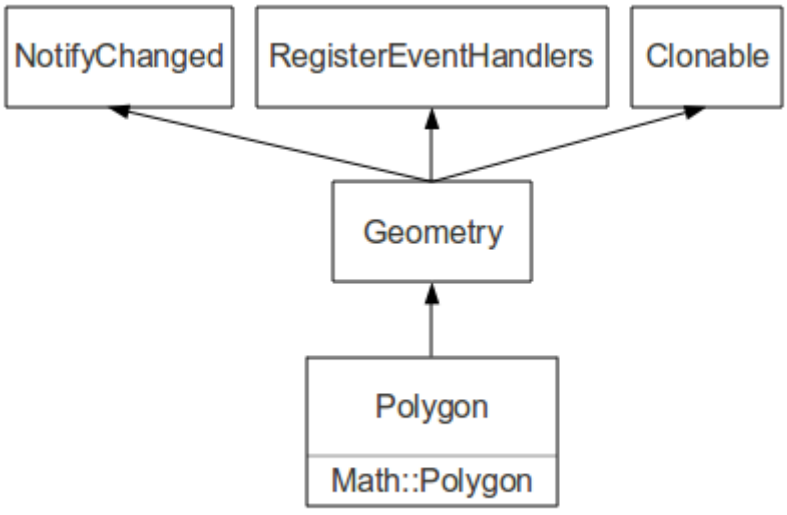
Geometrie je implementována tak, že každá jeho změna je oznámena pomocí události „Changed“. Pro každý matematický objekt z projektu Math je zde implementován objekt typu geometrie. Každý typ geometrie implementuje metodu, která podle předaných parametrů vytvoří síť trojúhelníků reprezentující tuto geometrii. Každá konkrétní implementace třídy geometrie potom obsahuje konkrétní matematickou reprezentaci dané geometrie a zpřístupňuje metody pro editaci tohoto matematického popisu.

Materiál je objekt implementován s oznamováním změn, navíc definuje metodu, která vrací reprezentaci použitelnou pro OGRE engine. Pro jednoduchost je implementován

pouze materiál, kterému lze nastavit barvu a průhlednost. Dále jsou v tomto modulu definovány třídy pracovní roviny, pomocné modelovací třídy a třída „SceneManager“. Tato třída má na starosti převod mezi mými datovými typy a světem OGRE. Dále také sleduje změny uzlů a zajišťuje přegenerování odpovídajících vykreslovaných objektů. Inicializace této třídy provede inicializaci OGRE enginu.



Obrázek 11: Hierarchie uzlů scény

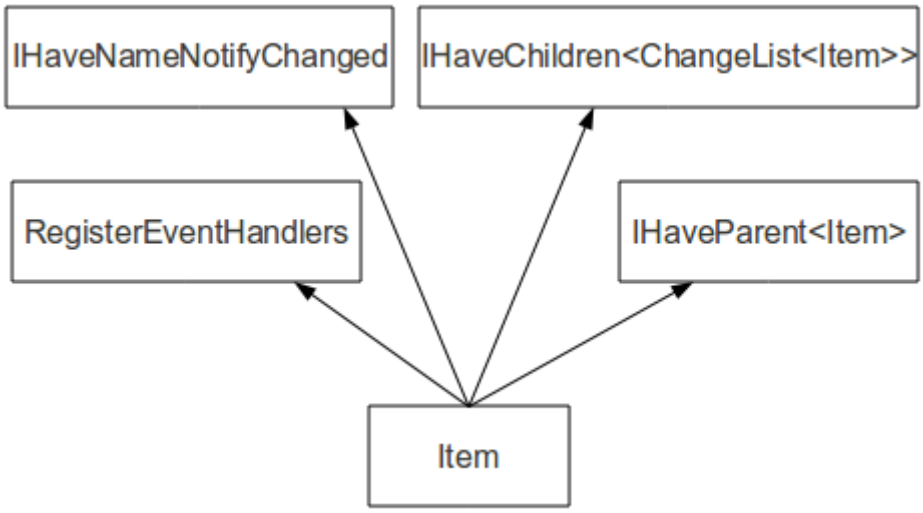


Obrázek 12: Ukázka hierarchie třídy Polygon z jmeného prostoru Modeling

7.3.4 ProjectDefinition

Tento modul obsahuje definici tříd reprezentujících položky projektu. Dále také definici projektu samotného.

- Layer – třída reprezentující vrstvu. Je to zjednodušený ekvivalent vrstvy z CAD systémů. Slouží k zjednodušení změn na skupinách objektů. Umožňuje například hromadně změnit způsob vykreslování (plochy, čáry, body), nebo hromadně přestat vykreslovat objekty náležící této vrstvě.
- Item – základní pojmenovaná položka projektu. Definuje abstraktní metodu vracející geometrickou reprezentaci. Každý objekt typu Item má zároveň seznam podobjektů (dětí) a odkaz na předka. Jedná se tedy o klasickou stromovou strukturu. Tato třída oznamuje změny svých vlastností. Odvozené třídy si mohou sami řídit pravidla vkládaní do stromu projektu tak, aby nedošlo k nesmyslným situacím (byt vložený v okně...)
- StaticItem – nejjednodušší možná implementace třídy Item s lokálně uloženou grafickou reprezentací.
- Project – abstraktní třída definující projekt. Obsahuje seznam položek typu Item. Uchovává informace o autorovi projektu, zadavateli projektu a datu zpracování. Tato třída sleduje změny všech svých vlastností.
- ProjectData – obálka třídy projekt. Obsahuje cestu k souboru a informaci o aktuálnosti tohoto souboru. Při každé změně projektu přenastaví příznak uložení.
- ProjectFactory – abstraktní třída sloužící jako továrna na projekty.



Obrázek 13: Hierarchie třídy Item

7.3.5 UI

V tomto modulu jsou implementovány všechny widgety použité v projektu. Většina z těchto objektů byla definována pomocí visuálního nástroje v QtCreator. Popis několika z nich:

- OgreWidget – objekt zobrazující grafickou reprezentaci projektu. Definuje propojení technologii Qt a OGRE. Neaktualizuje svůj obsah v nekonečné smyčce, ale pouze na vyžádání. Toto je realizováno tak, že třída při vytvoření spustí časovač s nastaveným krokem. Při každé iteraci časovače se ověří příznak, je-li nutno překreslit obsah
- ItemSelector – genericky napsaný widget sloužící jako pomocná kontrolka, určená k výběru jedné položky ze seznamu
- ProjectTree – widget odvozený z QtreeWidget, zobrazuje položky projektu. Aktuální vybraná položka widgetu je synchronizována s vybranými položkami aplikace

- PropertyGrid – tento widget je v UI aplikace použit opakovaně. Jeho hlavním úkolem je zobrazit a editovat vlastnosti přiřazené položky. Její funkčnost je založena na knihovně rozšiřující knihovnu Qt, která ale není součástí základní instalace. Jelikož datové typy projektu nepodporují žádnou formu introspekce, je nutné pro každý objekt editovatelný v property gridu napsat vlastní editor vlastností. Editory vlastností jsou postaveny na mechanismu „správce/továrna“. Například editor hodnoty float se realizuje pomocí správce, který řídí logiku editování, ošetřuje mezní stavy a oznamuje změny hodnoty. K tomuto správci lze poté přiřadit továrnu, která dodá UI prvek. Float hodnotu jde tedy editovat pomocí několika různých widgetů
- ProjectViewer – kontrolka zobrazující projekt. Může obsahovat několik OgreWidget objektů, každý s jiným pohledem. Další obsah je plně konfigurovatelný podle potřeb aktuálně provozovaného pluginu
- Wellcome – widget zobrazující úvodní obrazovku, umožňuje vytvořit nový projekt podle načtených pluginů, nebo otevřít dříve vytvořený projekt
- Projects – zobrazuje seznam otevřených projektů
- About – informace o programu
- Settings – widget zobrazující nastavení

Dále jsou v tomto modulu definovány třídy, které nejsou použity jako widgety, ale mají s nimi přímou souvislost.

7.3.6 Application

V tomto modulu jsou definovány pouze dva objekty. Prvním je třída Plugin, která definuje rozhraní použité pro načtení libovolného pluginu podporovaného aplikací. Dále je pak v projektu definována třída aplikace, tedy třída, která má na starosti chod aplikace. Jejím úkolem je načíst nastavení, načíst dostupné pluginy a vytvořit hlavní okno aplikace. Třída aplikace dále implementuje metody společné všem projektům, jako například ukládání a načítání projektů.

7.3.7 SunLight

Tento projekt implementuje plugin aplikace. Jedná se o rozšíření, které řeší problém proslunění bytů. Implementuje vlastní továrnu na projekt s vlastní implementací ProjectViewer třídy. Dále definuje třídy odvozené z třídy StaticItem projektu ProjectDefinition. Jsou to:

- Apartment – tato třída reprezentuje byt
- Room – třída reprezentující místnost
- Window – třída reprezentující okno

V tomto projektu jsou také implementovány všechny algoritmy nutné pro výpočet proslunění.

Tento úkol lze rozdělit do několika kroků:

1. **Výpočet pozice slunce** – k tomuto výpočtu je potřeba znát několik údajů. Datum a čas, ve kterém výpočet provádíme. Dále zeměpisnou šířku a zeměpisnou délku místa, ve kterém se nachází hodnocený objekt. Z těchto hodnot lze pomocí vzorců popsaných v teoretické části spočítat úhlovou výšku slunce nad obzorem a sluneční azimut.
2. **Ověření geometrie –** v tomto kroku je nutné ověřit velikosti místností a oken. Tyto rozměry musí splňovat podmínky definované normou. Z geometrie okna a geometrie místnosti se také vyvodí kritický bod okna, který se následně použije při výpočtech.
3. **Výpočet proslunění –** nyní je možné provést samotný výpočet. Z vypočtené pozice slunce se vede přímka směřující na kritický bod zvoleného okna. Pokud paprsek dopadne na kritický bod a jsou splněny podmínky omezující úhly slunečního paprsku, lze v tomto čase považovat okno za prosluněné. Tento výpočet provedeme ve zvolených časových krocích.
4. **Vyhodnocení –** provedeme součet všech časů, ve kterých je okno prosluněno. Porovnáním s hodnotami uvedenými v normě lze určit, zda je místnost dostatečně prosluněna.

7.4 Vybrané problémy řešené v aplikaci

7.4.1 Ukládání

K serializaci dat projektu jsem použil knihovnu Boost::Serialization. Tato knihovna umožňuje ukládat v textovém i binárním formátu. Každý datový typ, který je potřeba uložit musí implementovat metody definované touto knihovnou. Pokud je potřeba uložit datové typy, jejichž definici nelze měnit, je možno použít tzv. „volnou“ serializační metodu. Dále je nutné každý datový typ zaregistrovat pomocí speciálního makra. Použití této knihovny značně prodloužilo čas kompilace celého projektu. Zvolené řešení má slabinu v tom, že při otevírání uloženého souboru musí mít aplikace načtena odpovídající plugin, jinak dojde k výjimce. Proto by bylo zřejmě vhodné „obalit“ tento systém ještě jednoduchou textovou vrstvou, kde by byly uloženy informace o verzi, vyžadovaných pluginech atd. Projekty uložené na 64 bitové platformě lze bez problémů načíst na 32 bitovém PC.

7.4.2 Kopírování

V Programu je možné kopírovat již vytvořené objekty. Toto je umožněno rozhraním Clonable, které definuje kopírovací metodu. Objekt vytvořený pomocí této metody je „hlubokým“ klonem původního objektu. V programu ovšem není naprogramována podpora pro kopírování objektů mezi projekty.

7.4.3 Oznamování změn

Aby program mohl reagovat na změnu dat, bylo nutné napsat rozhraní umožňující tyto změny sledovat. Podobný problém řeší mnoho knihoven. Řešení použité v mém programu je založeno na knihovně FastDelegates. Umožňuje snadné „přihlašování“ a „odhlašování“ z událostí. Je také možné automatizovat sledovaní a oznamování změn vnořených objektů.

7.4.4 Pluginy

Pluginy se myslí knihovny dynamicky načítané po spuštění programu. Asi každá větší aplikace dnes podporuje načítání pluginů. Tento úkon lze realizovat mnoha způsoby, kdy

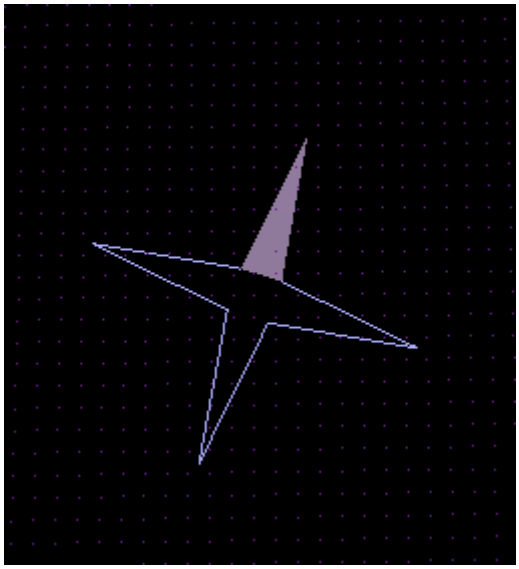
každý má své výhody a nevýhody. Přístup, který jsem zvolil já se dá charakterizovat asi takto:

- každý plugin definuje metody, které ho zaregistrují a odregistrují do hlavní aplikace
- při změně tříd v modulu, na který se plugin odkazuje, je nutné překompilovat i plugin. Toto v podstatě znemožňuje tvorbu pluginů třetích stran, ale zároveň to usnadňuje vývoj pluginů vlastních

V aplikaci není vyřešena definice závislostí pluginů.

7.4.5 Orientace souřadnicového systému

Při vytvoření nového projektu se zobrazí okno, které umožňuje nastavit vlastnosti projektu. Jednou z těchto vlastností je lokalita řešeného problému. Podle této lokality se pak spočítá meridiánová konvergence. Na základě této hodnoty se poté zobrazí směrová hvězdice ukazující na sever.



Obrázek 14: Hvězdice ukazující sever

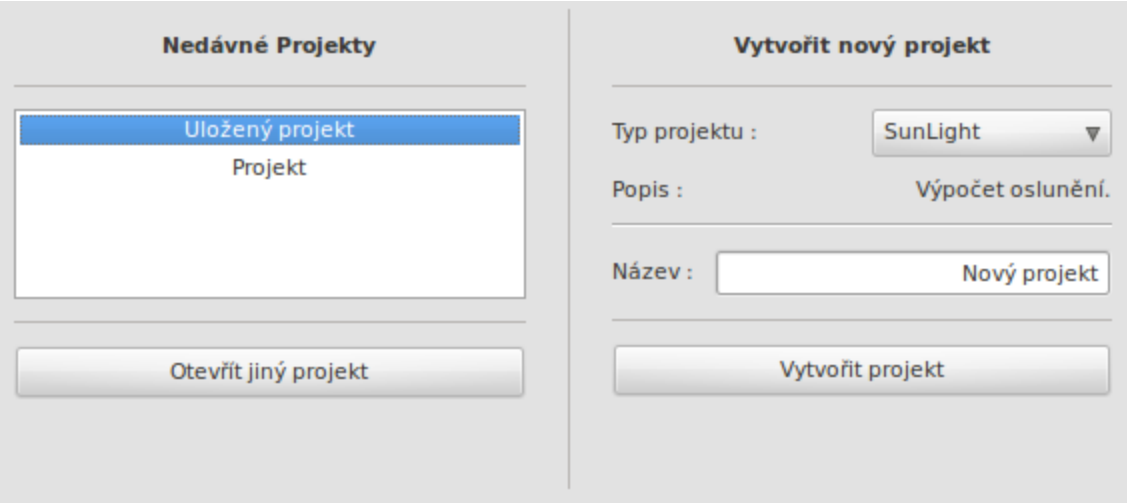
7.4.6 Zobrazení výsledků

K zobrazení výsledků slouží tlačítko vyhodnotit. Pokud byl již výpočet proveden, zobrazí se výsledky, pokud ne, dojde k výpočtu. Okno zobrazující výsledky ukazuje několik dat

- jestli byt vyhovuje normě
- celkovou dobu proslunění každé místnosti
- časové intervaly, v nichž je místnost prosluněna
- celkovou dobu, po kterou na okno dopadají sluneční paprsky
- časové intervaly, v nichž na okno dopadají sluneční paprsky

7.5 Program pohledem uživatele

7.5.1 Úvodní okno

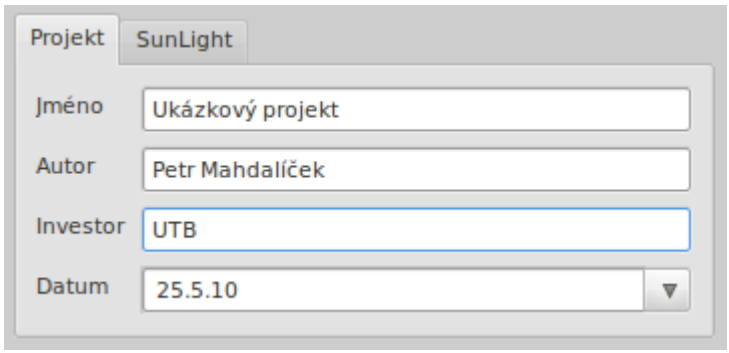


Obrázek 15: Úvodní okno aplikace

Ihned po spuštění aplikace je zobrazeno úvodní okno. Zde je možno otevřít nedávno používané projekty, jiné již vytvořené projekty a také vytvořit projekt nový. Seznam typu projektů je ve výchozím stavu prázdný. Aplikace při spuštění prohledá příslušnou složku a

načte z ní pluginy. Každý načtený plugin implementující nový typ projektu bude zobrazen v tomto okně.

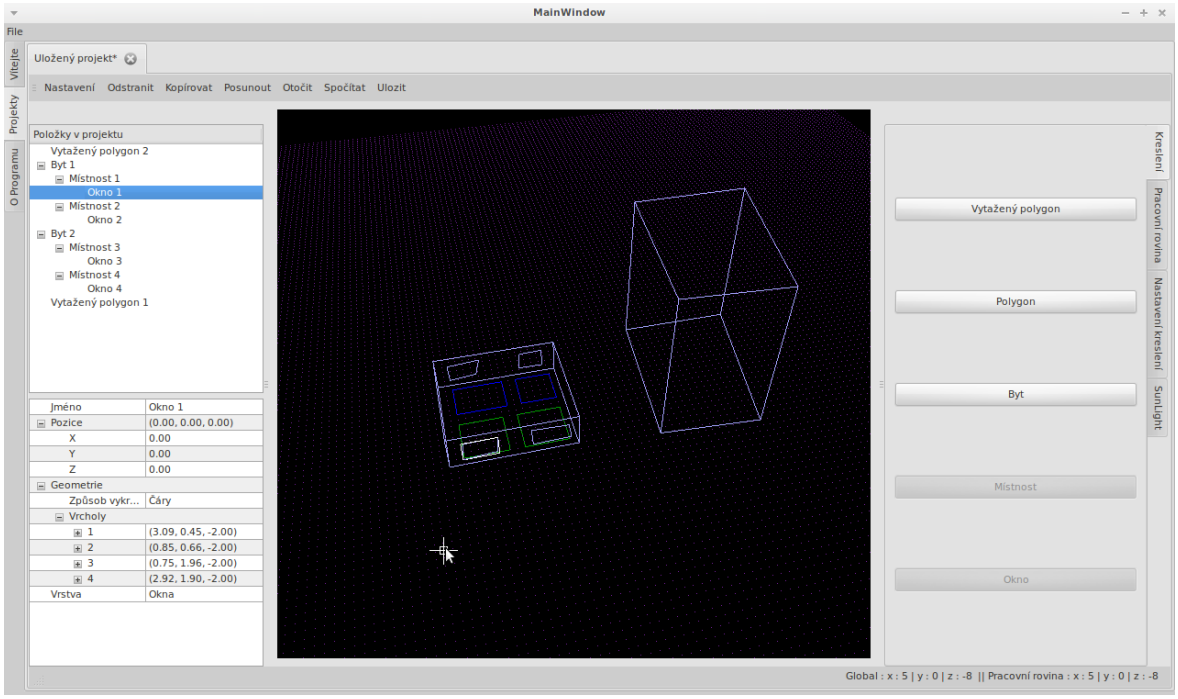
7.5.2 Nastavení projektu



Obrázek 16: Nastavení projektu

Ihned po vytvoření projektu se uživateli zobrazí okno umožňující nastavit různé vlastnosti projektu. Základní implementace podporuje zadání názvu, autora, investora a data zpracování. Toto menu je opět rozšířitelné pomocí pluginu. Plugin SunLight přidává vlastní záložku na, na které je nutné vybrat lokalitu v níž se nachází objekt hodnocený v projektu. Toto menu lze poté kdykoliv zobrazit pomocí tlačítka „Nastavení projektu“.

7.5.3 Hlavní okno programu



Obrázek 17: Hlavní okno programu

Po nastavení vlastností projektu se zobrazí hlavní okno aplikace. To se skládá z několika částí. Levý panel, kreslící plocha, pravý panel a lišta tlačítek.

V levém panelu se zobrazuje strom položek projektu. Pod stromem položek se zobrazuje tabulka vlastností vybrané položky(název, pozice, vrstva...). Tyto vlastnosti se zobrazují pokud je vybrána pouze jedna položka.

Kreslící plocha je dominantní prvek celé aplikace. Ve výchozím stavu zobrazuje pracovní rovinu a směrovou hvězdici, ukazující na sever. V tomto okně se během práce zobrazuje průběh modelování. Poskytuje vizuální přehled a kontrolu zadaných dat.

V pravém panelu se nachází několik záložek:

- Kreslení – umožňuje vkládat do projektu položky (geometrie, byty, místnosti)
- Pracovní rovina – umožňuje měnit vlastnosti pracovní roviny
- Nastavení kreslení – mění nastavení vlastností vrstev

- SunLight – panel dodaný pluginem SunLight

V okně je ještě zobrazena lišta tlačítek, která umožňuje uložit projekt, zkopírovat nebo odstranit položku projektu, změnit nastavení projektu a zobrazit vyhodnocení projektu.

7.5.4 PropertyGrid

Jméno	Válec 2
⊟ Pozice	(-5.74, 3.50, -17.82)
x	-5.74
y	3.50
z	-17.82
⊟ Natočení	(0.00, 0.00, 0.00)
Kolem osy x	0.00
Kolem osy y	0.00
Kolem osy z	0.00
⊟ Materiál	
Průhlednost	100
⊟ Geometrie	
Způsob vykres...	Čáry
⊞ Krok	(0.50, 0.50, 0.50)
⊞ Podstava	(0.15, 0.15)
Výška	2.00
Vrstva	Výchozí

Obrázek 18: Property grid

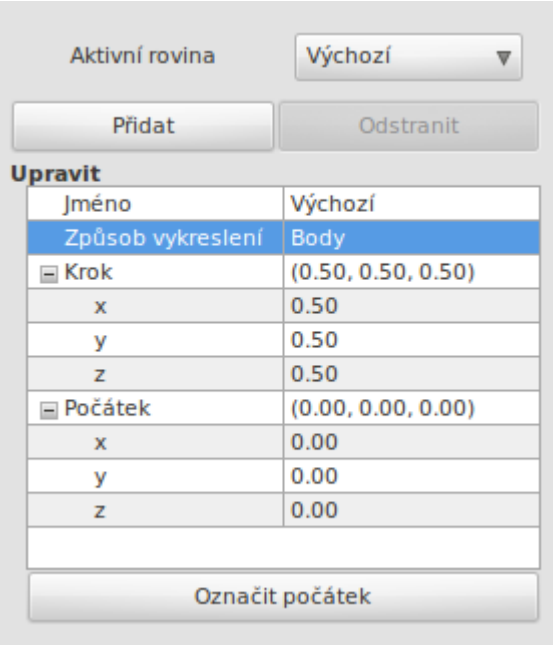
Tento widget slouží k zobrazovaní vlastností vybrané položky projektu. Pro každý datový typ, který je potřeba zobrazit v property gridu musí existovat editor vlastností. Tyto editory se při inicializaci aplikace zaregistrují globálního seznamu. V aplikaci jsou implementovány editory základních datových typů (int, double, string, vector..). Implementace editoru vlastností pro složitější objekty poté spočívá ve skládání editorů dílčích vlastností.

Obrázek 19: Editor průhlednosti

Obrázek 20: Editor výšky

Pro jeden datový typ lze napsat více editorů tak, aby co nejvíce vyhovovaly povaze hodnoty.

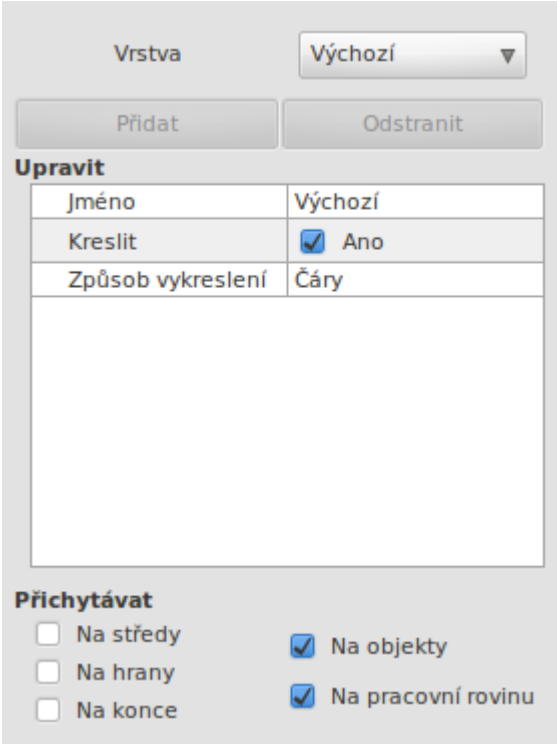
7.5.5 Panel nastavení pracovní roviny



Obrázek 21: Panel nastavení pracovní roviny

Na tomto panelu lze vybrat aktuálně používanou pracovní rovinu. Je také možné přidat rovinu vlastní. Rovinu lze vykreslit pomocí bodů, čar, nebo jako souvislou plochu. Hustotu mřížky lze libovolně nastavit. Pracovní rovina má nastavitelný počátek, nelze ní však rotovat.

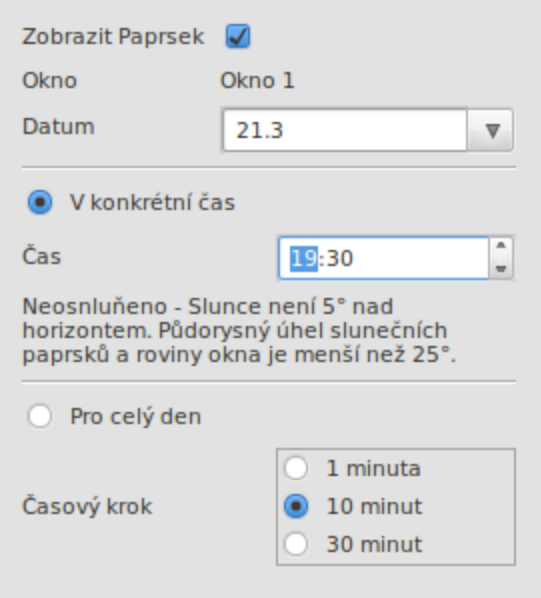
7.5.6 Panel nastavení modelování



Obrázek 22: Panel nastavení modelování

Tento panel umožňuje editovat vlastnosti vybrané vrstvy. Umožňuje přidávat vrstvy nové. Ve spodní části lze zvolit způsob přichytávání kursoru.

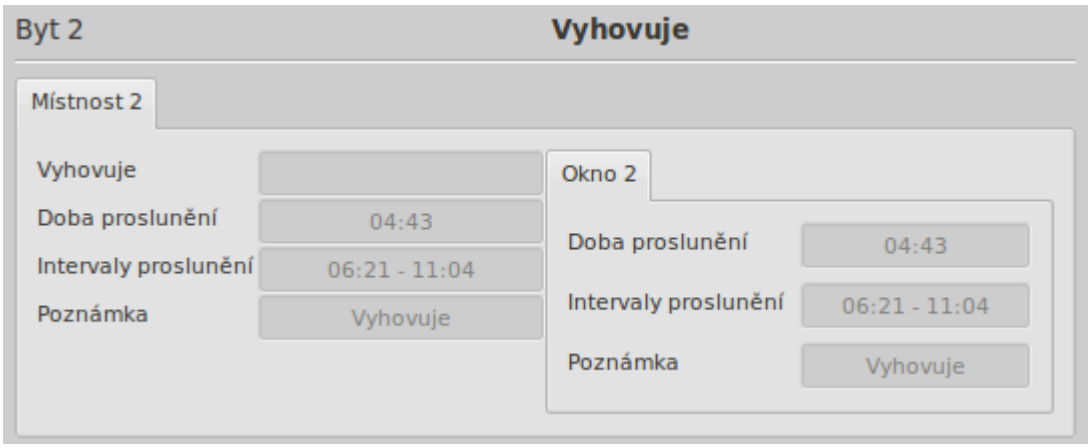
7.5.7 SunLight panel



Obrázek 23: Panel pluginu SunLight

Tento panel umožňuje uživateli zobrazit sluneční paprsek. Lze zobrazit buď paprsek v konkrétním čase, nebo v rozmezí celého dne ve stanoveném časovém kroku. Pokud je zvolena první volba, panel uživatele informuje, je-li okno v daný moment osluněno.

7.5.8 Panel zobrazující výsledky výpočtu



Obrázek 24: Panel zobrazující výsledek výpočtu

Po stisknutí tlačítka „Vyhodnotit“ se zobrazí okno s výsledky. Pro každý z hodnocených bytů se zobrazí tabulka obsahující data pro místnost a data pro každé okno v místnosti. Samozřejmostí je informace o tom, zda daný byt vyhovuje normě.

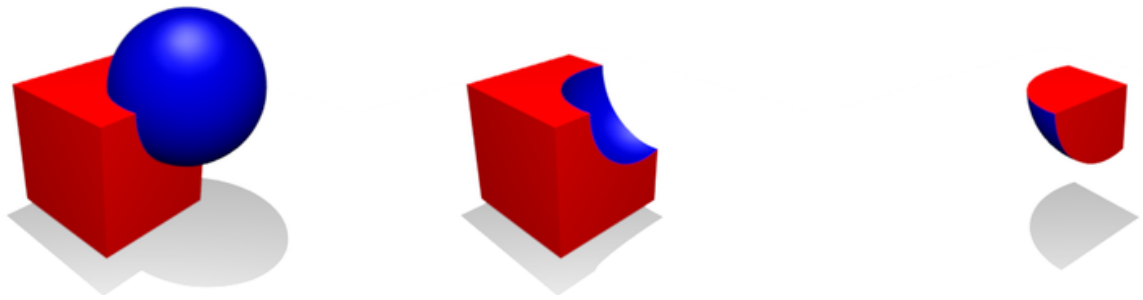
8 NAVRHOVANÉ VYLEPŠENÍ

8.1 Oslunění venkovních prostor

Normy popsané v teoretické části definují podmínky oslunění venkovních prostor určených k rekreaci obyvatel. Výsledná aplikace neumožňuje tyto prostory zadat.

8.2 Pokročilé modelování

Pro jednoduchou a efektivní práci s programem by bylo vhodné rozšířit možnosti modelování. Jednou z největších slabin současné implementace je to, že nepodporuje modelování pomocí konstruktivní geometrie těles (CSG). Reprezentace modelů pomocí konstruktivní geometrie těles je založena na analytickém popisu těles jejich objemem, tj. podmnožinou trojrozměrného prostoru ležícího uvnitř tělesa.



Obrázek 25: Použití konstruktivní geometrie

8.3 Instalační balíček

K aplikaci by bylo vhodné dopsat instalační balíček, který by ošetřil instalaci všech knihoven potřebných k běhu, vytvořil odkazy na aplikaci v příslušných částech operačního systému. Při odinstalování by také měl odstranit veškerá data aplikace. V současném stavu je nutné závislosti doinstalovat ručně.

ZÁVĚR

Cílem této diplomové práce bylo porozumět normám, které stanovují minimální hodnoty proslunění obytných prostor. Seznámit se se způsoby výpočtu proslunění bytů a prozkoumat existující aplikace ulehčující tyto úkony. Na základě těchto znalostí vybrat vhodné technologie a pokusit se vytvořit aplikaci, která bude vyhovovat nejnovějším normám a poběží na operačním systému linux.

Potřebami proslunění obytných prostor se zabývají normy popsané v kapitole 1. Tyto normy definují minimální hodnoty proslunění. Vysvětlují význam pojmů nutných pro pochopení problematiky proslunění. Definují způsoby výpočtu pozice slunce a příslušných parametrů. Z těchto hodnot lze poté spočítat skutečné hodnoty proslunění pro zvolený datum a čas. Porovnáním spočtených hodnot proti kritériím určeným v normě lze rozhodnout, zda hodnocený objekt vyhovuje.

V teoretické části práce jsou také popsány knihovny zvolené pro tvorbu vlastní aplikace. Pro uživatelské prostředí byla zvolena knihovna Qt, která toho ovšem zvládne mnohem víc. Pro zobrazování jsem použil knihovnu OGRE. V programu jsou také použity různé části knihovny boost. V aplikaci jsou dále použity menší knihovny, obvykle řešící specifický úkol. Program jsem se rozhodl psát v prostředí QtCreator v jazyce C++.

Výsledná aplikace je spustitelná na operačním systému linux. Umožňuje modelovat řešené prostory. Po zadaní tvarů místností, oken a polohy řešeného objektu lze provést výpočet doby proslunění a porovnání výsledků s požadavky danými normami. Program umožňuje uživateli zobrazit polohu slunce vzhledem k řešenému objektu. Program umožňuje ukládat a načítat již vytvořené projekty.

V závěru praktické části jsem popsal nedostatky vytvořeného programu. V této kapitole je také naznačen směr dalšího vývoje. Hlavním nedostatkem aplikace je zřejmě nedostatečně robustní modelovací nástroj.

RESULT

The aim of this thesis was to understand the norms, which determines the minimum value of insolation of living rooms. To get acquainted with methods of calculating insolation of dwellings and find some existing applications that makes it easier to examine these acts. Based on this knowledge, select appropriate technologies and try to create an application that will meet the latest standards and runs on Linux.

Needs of insolation living areas are described in standards outlined in Chapter 1. These standards define the minimum value of room insolation. They also explains the meaning of the terms necessary for understanding this discipline. Norms defines methods for calculating the position of the sun and the relevant parameters. From these values can be calculated the real value of insolation for the selected date and time. By comparing computed values against the criteria specified in the standard can be decided whether the rated object meets requirements defined by the norm.

In the theoretical part are also described libraries chosen to build my own application. For the user interface was chosen Qt, which of course can do much more. To display data was used Ogre library. The program also uses various parts of the Boost libraries. The application also uses some smaller libraries, usually addressing a specific task. I decided to write the program in QtCreator in language C++.

The resulting application can run on Linux. Allows to create data model of rated objects. After definning the rooms and windows shape, and location of the rated object program will calculate the insolation time and compares the result with the requirements of norms. The program allows user to view the sun's position due to the treated object. The program allows you to store and retrieve an already established project.

In the end of the practical part, I described the deficiencies of created program. This chapter also outlined the direction of further development. The main drawback seems to be a lack of robust modeling tool.

SEZNAM POUŽITÉ LITERATURY

[0] KAŇKA, Jan. ČSN 73 0581 Oslunění budov a venkovních prostor – Metoda stanovení hodnot.[s.l.] : 2009, 20 s.

[1] PAPEŽ, Karel, et al. ČSN 73 4301 Obytné budovy. [s.l.] : 2004, 48 s.

[2] BLANCHETTE, Jasmin, SUMMERFIELD, Mark. C++ GUI Programming with Qt 4.

2nd edition.[s.l.] : Prentice Hall PTR, 2008. 752 s. ISBN 0132354160.

[3] JUNKER, Gregory. Pro OGRE 3D Programming (Hardcover). 1st edition. [s.l.] : Apress, 2006. 312 s. ISBN 1590597109.

[4] Ogre 3D [online]. 2000 [cit. 2010-01-27]. Dostupný z WWW: <http://www.ogre3d.org>.

[5] Qt [online]. 2008 , 2010 [cit. 2010-01-27]. Dostupný z WWW: <http://qt.nokia.com/>.

[6] Morley, Keith. Realistic Ray Tracing. 2nd edition. [s.l.] : AK Peters, 2003. 235 s. ISBN 1568811985.

[7] ANTON, Howard. Elementary Linear Algebra. 8th edition. [s.l.] : Wiley, 2000. 608 s. ISBN 0471170550.

[8] CHURCHER, Clar. Beginning Database Design: From Novice to Professional (Paperback). [s.l.] : Apress, 2007. 300 s. ISBN 1590597699.

[9] ABRAMENKO, Peter. Buildings: Theory and Applications. 1st edition. [s.l.] : Springer, 2008. 754 s. ISBN 0387788344.

[10] STROUSTRUP, Bjarne . *Stroustrup : C++* [online]. 2001, 2010 [cit. 2010-06-01]. Stroustrup : C++. Dostupné z WWW: <http://www2.research.att.com/~bs/C++.html>.

[11] C++0x. In *Wikipedia : the free encyclopedia* [online]. [cit. 2010-06-01]. Dostupné z WWW: <http://en.wikipedia.org/wiki/C%2B%2B0x>.

[12] *Boost.org* [online]. 1998 [cit. 2010-06-01]. Boost.org. Dostupné z WWW: <http://www.boost.org/>.

[13] *Astra92* [online]. 2001 [cit. 2010-06-01]. SunLiS. Dostupné z WWW: <http://www.astra92.cz/Software/Produkty/SunLiS/tabid/97/Default.aspx>.

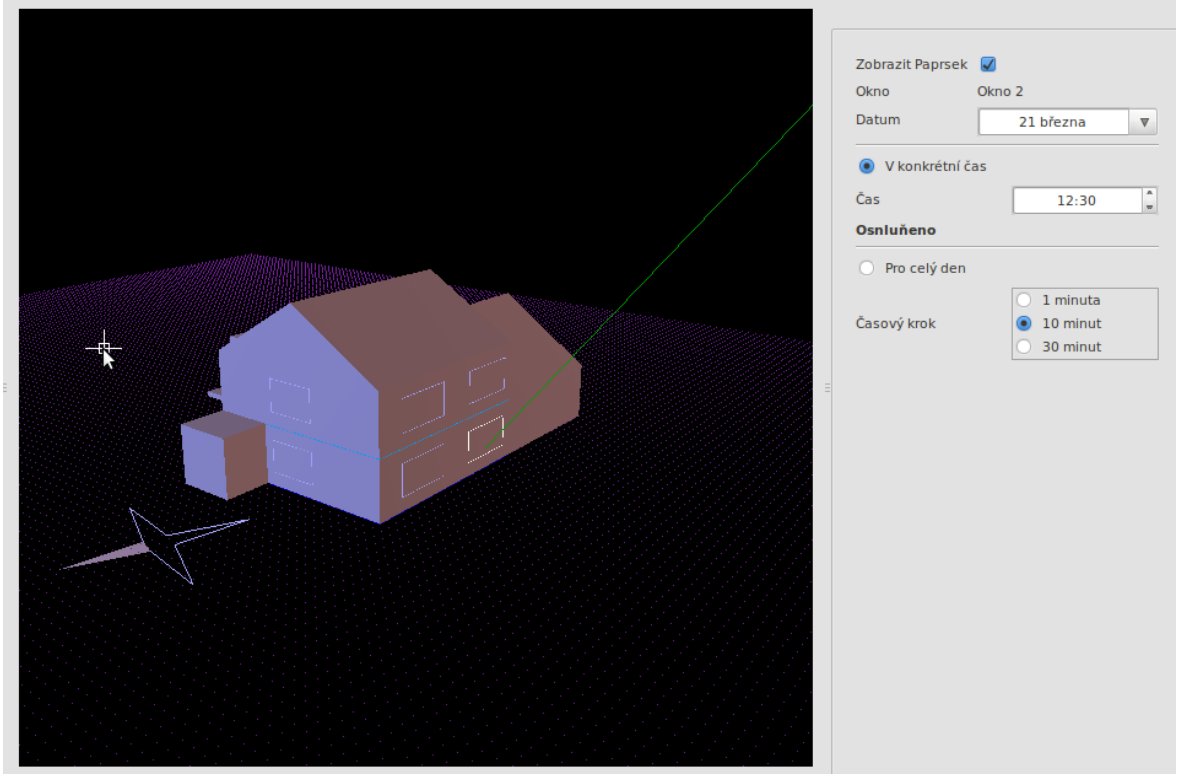
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

C	Meridiánová konvergence
δ	Sluneční deklinace
A	Sluneční azimut
h	Sluneční výška
τ	Hodinový úhel
φ	Zeměpisná šířka
λ	Zeměpisná délka
C++	Programovací jazyk
Widget	Prvek uživatelského rozhraní

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

PŘÍLOHA P I: ZOBRAZENÍ SLUNEČNÍHO PAPRSKU



PŘÍLOHA P II: TISK REPORTU

SunLight

Projekt :	Nový projekt			
Autor :	Petr Mahdalíček			
Investor :	Fai			
Datum :	po 5 31 2010			
Lokalita :	Uherský Brod (Uherské Hradiště)			
Byt v 1. patře	Vyhovuje			
Ložnice	Doba oslunění	05:30	Intervaly oslunění	12:07 - 17:38
Obývací pokoj	Doba oslunění	05:29	Intervaly oslunění	12:08 - 17:38
Dětský pokoj	Doba oslunění	00:00		
Byt v přízemí	Vyhovuje			
Pokoj 1	Doba oslunění	05:30	Intervaly oslunění	12:07 - 17:38
Pokoj 2	Doba oslunění	05:29	Intervaly oslunění	12:08 - 17:38
Ložnice	Doba oslunění	00:00		

PŘÍLOHA P III: REGISTRACE PLUGINU

```cpp
#include "Application/Application.h"
#include "ProjectActivator.h"

using namespace ApplicationCore;
using namespace SunLight;

extern "C" bool registerPlugin(Application & app)
{
    app.getProjectFactory().getActivators().push_back(
            ProjectDefinition::ProjectActivatorPtr( new ProjectActivator(app)));
    return true;
}

extern "C" bool unRegisterPlugin(Application & app)
{
    Common::List<ProjectDefinition::ProjectActivatorPtr> & list =
            app.getProjectFactory().getActivators();
    for(size_t i = 0; i < list.size(); ++i)
    {
        if(list[i]->getId() == L"SunLight")
        {
            list.removeAt(0);
            return true;
        }
    }

    return false;
}

extern "C" void getPluginDescription(Plugin::PluginDescription & pluginDescription)
{
    pluginDescription.Name = L"SunLight";
    pluginDescription.Description = L"Oslunění bytů.";
}
```