Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Počítačová podpora výuky optiky

2020

Lukáš Kubík

Vedoucí práce: Mgr. Tomáš Kühr, Ph.D.

Studijní obor: Aplikovaná informatika, kombinovaná forma

Bibliografické údaje

Autor:	Lukáš Kubík
Název práce:	Počítačová podpora výuky optiky
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2020
Studijní obor:	Aplikovaná informatika, kombinovaná forma
Vedoucí práce:	Mgr. Tomáš Kühr, Ph.D.
Počet stran:	33
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Lukáš Kubík
Title:	Software for Teaching of Optics
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2020
Study field:	Applied Computer Science, combined form
Supervisor:	Mgr. Tomáš Kühr, Ph.D.
Page count:	33
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Aplikace Počítačová podpora výuky optiky je pomůckou, jak zábavnou formou podpořit výuku optiky. Umožňuje vkládat do scény různé objekty (čočky, zrcadla, zdroje světla, ...), měnit jejich vlastnosti (velikost, tvar, materiál, ohniskovou vzdálenost...) a zobrazovat aktuální stav lomu (průchody a odrazy) světelných paprsků. Důraz je kladen na názornou prezentaci, přívětivé uživatelské rozhraní a pohodlné ovládání aplikace. Volitelně byl software doplněn také teoretickou a testovací částí.

Synopsis

The Computer Support of Optics Teaching application is a tool for supporting the teaching of optics in a fun way. It allows you to insert various objects (lenses, mirrors, light sources,...) into the scene, change their properties (size, shape, material, focal length ...) and display the current passages and reflections of light rays. Emphasis is placed on visual presentation, friendly user interface and convenient application control. Optionally, the software was also supplemented with a theoretical and test part.

Klíčová slova: zrcadlo; čočka; laser; paprsek; světlo; lom světla; swing

Keywords: mirror; lens; laser; beam; light; refraction; swing

Děkuji vedoucímu mé bakalářské práce Mgr. Tomáši Kührovi, Ph.D, za odbornou pomoc a cenné připomínky, které mi pomohly při vývoji aplikace i psaní dokumentu.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam vět

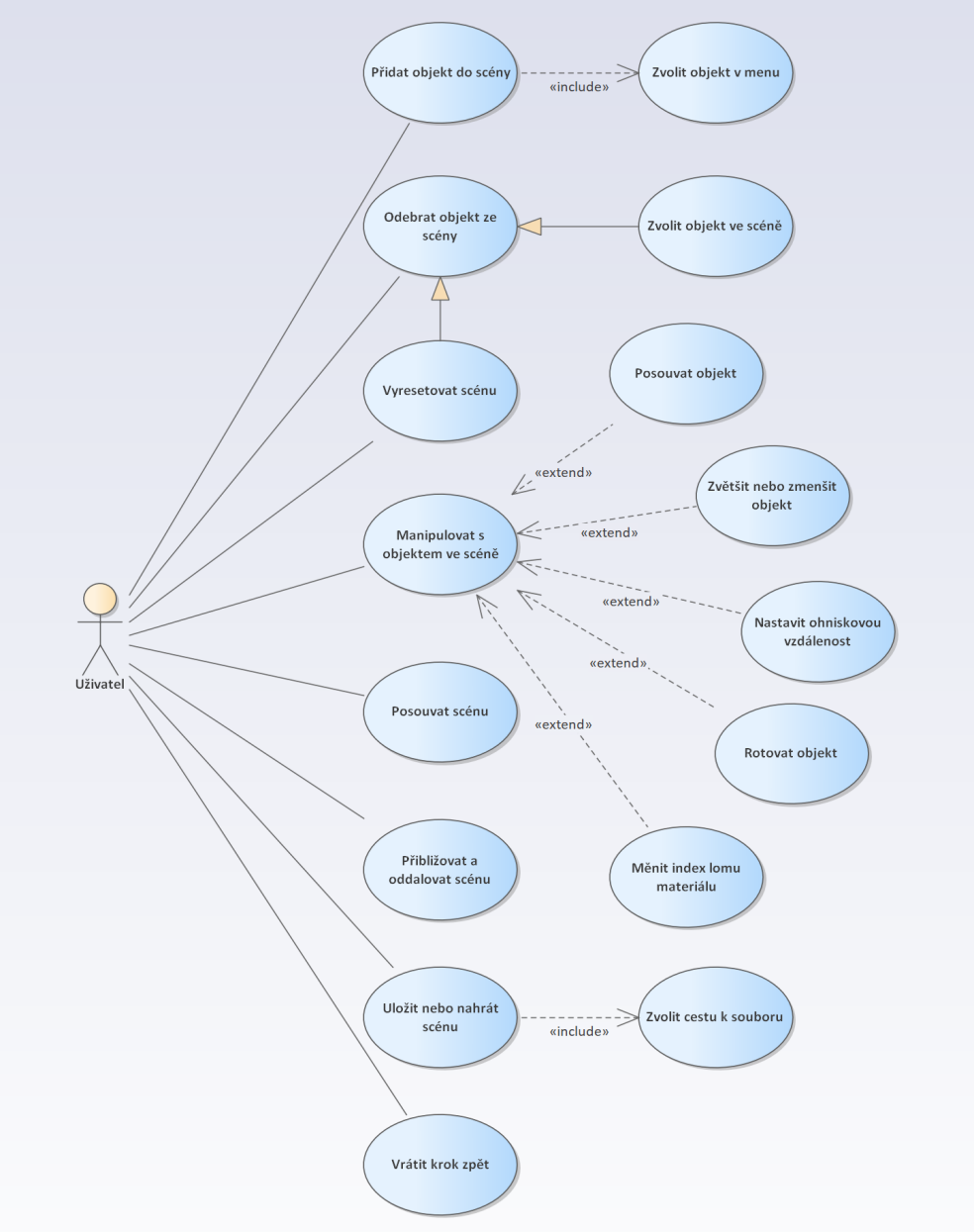
Seznam zdrojových kódů

1 Úvod

Optika [1][2]je obor fyziky zabývající se podstatou světla a jeho šířením v různých prostředích. Světlo je část spektra elektromagnetického záření. Tento software se zaměřuje především na zobrazení lomu světla při průchodu optickými objekty. Například u spojné čočky, kdy se paprsek láme nad či pod jejím středem, anebo když světlo dopadá šikmo na průhledný materiál jako je sklo. Různé materiály zpomalují světlo rozdílně, takže lom nastává vždy pod jiným úhlem. Podíl na rychlosti šíření světla je roven indexu lomu daného prostředí. Ohnisková vzdálenost je vzdálenost čočky nebo kulového zrcadla od jejich ohniska. Ohnisko je místo kde se všechny paprsky střetnou.
Software je rozdělen na tři části, které jsou dostupné z hlavního menu aplikace. První praktickou část tvoří hlavní program se scénou, do které se vloží optické objekty a následně se s nimi manipuluje. Operace související s lomem světla jsou naprogramovány dle Snellova zákona. Druhá je teoretická a seznamuje uživatele s teorií jednotlivých optických objektů přidávaných do scény a třetí testovací prověřuje jeho znalosti formou otázek z optiky.

1.1 Diagram případu užití

Diagram popisuje hlavní program.



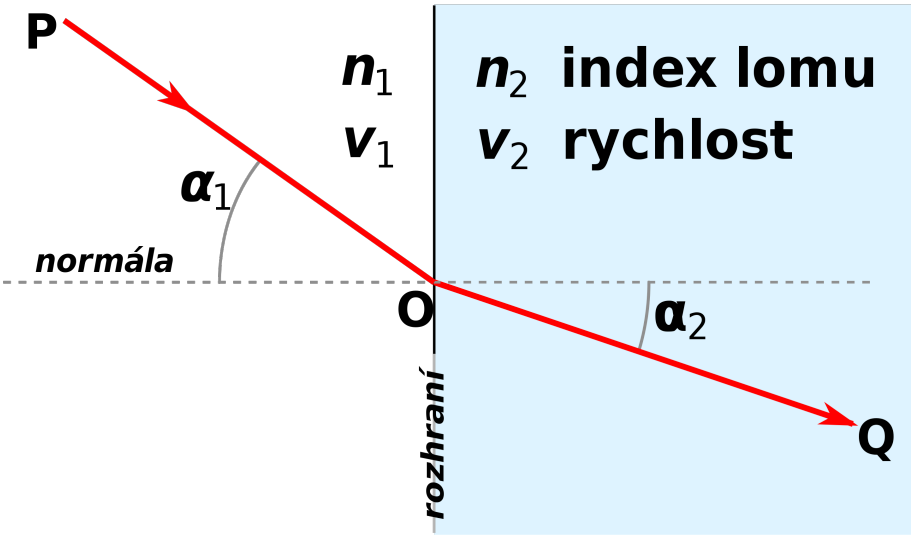
Obrázek 1: Diagram případu užití

1.2 Snellův zákon

Snellův zákon[3] objevený v 10. století popisuje lom paprsku světla tedy elektromagnetického záření. Jde o základní zákon popisující šíření vlnění, které přechází (tzv. lomem) přes rozhraní z jednoho prostředí do jiného, kde mění optické vlastnosti prostředí jako voda aj. Některé operace související s lomem světla v tomto programu jsou naprogramovány dle formulace Snellova zákona.

Definice 1 (Snellův zákon)

$$\frac{\sin \alpha_1}{\sin \alpha_2} = \frac{v_1}{v_2} = \frac{n_2}{n_1}$$



Obrázek 2: Snellův zákon

1.3 Indexy lomu světla

Lom světla u většiny materiálu zobrazeného v tabulce[4] lze nasimulovat v hlavním programu.

Tabulka 1: Indexy lomu světla

Jméno materiálu	Vlnová délka (nm)	Index lomu
Vakuum		1
Vzduch	589.29	1.000293
Hélium	589.29	1.000036
Benzen	589.29	1.501
Petrolej		1.39
Ethanol (ethylalkohol)	589.29	1.361
Aceton		1.36
Voda	589.29	1.333
Diamant	589.29	2.417
Jantar	589.29	1.55
Chlorid sodný	589.29	1.544
Křemenné sklo (čisté)	589.29	1.458
Kapalné hélium		1.025
Led		1.31
Lidská čočka		1.386–1.406
Lidská játra	964	1.369
Rostlinný olej		1.47
Glycerín		1.4729
Cukrový roztok 50%		1.4200
Kamenná sůl		1.516
Okenní sklo		1.52
Bróm		1.661
Safír		1.762–1.778
Oxid zinečnatý	390	2.4
Křemík	1200 – 8500	3.42–3.48
Germanium	3000 – 16000	4.05–4.01

2 Použité technologie

2.1 Java

Java [5] [6] je jedním z nejdůležitějších a nejrozšířenějších počítačových programovacích jazyků na světě. Je navíc součástí revoluce chytrých telefonů, protože se používá při programování pro systém Android. Programování v Javě je v základech většiny moderních světových výpočetních prostředí.
První verze jazyka Java vyšla v roce 1995 pod vedením společnosti Sun Microsystems. V průběhu let se Java rozrůstala a často byla na předním místě ve vývoji programovacích jazyků. První významnou aktualizací Javy byla verze 1.1. Další významné vydání Javy byla verze Java 2, kdy společnost Sun uvedla výsledný produkt jako J2SE (Java 2 Platform Standard Edition) a čísla verzí se začala používat pro tento produkt. Vydání J2SE 1.4 přineslo např. zřetězené výjimky nebo klíčové slovo assert. Vydání J2SE 5 bylo pro Javu také velice významné. Přibyly např. generické typy, výčty, variabilní počet argumentů, anotace a další. Sada pro vývojáře byla pojmenována JDK 5. Java SE 7 byla první hlavní vydání Javy od doby, kdy Sun Microsystems získala společnost Oracle. Java 8 SE přinesla jednu z posledních významných funkcí jazyka: výraz lambda. Lambda výrazy totiž mohou zjednodušit a zredukovat rozsah zdrojového kódu.

2.2 Swing

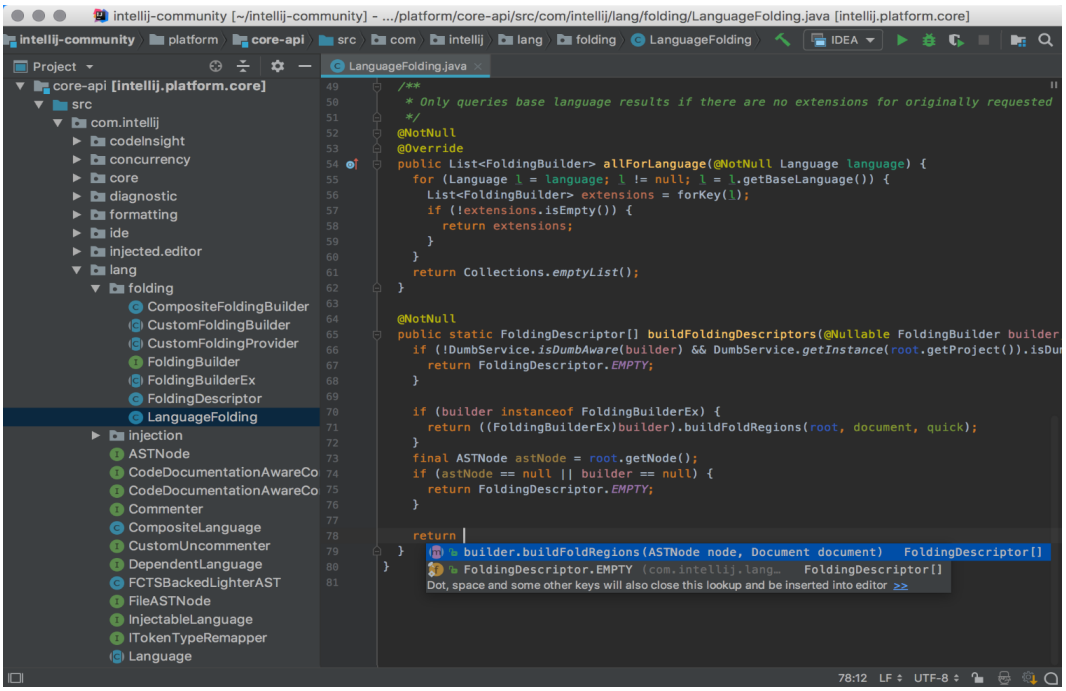
Swing [6] je kolekcí tříd a rozhraní, jež definují vzhled a chování moderního grafického uživatelského rozhraní Javy. V Javě se tato knihovna obvykle používá u desktopové aplikace nebo apletu. Nabízí bohatou sadu vizuálních komponent, jako jsou tlačítka, textová pole, posuvníky, zaškrtávací pole, stromy a tabulky, které si lze přizpůsobit tak, aby byly vhodné pro libovolné potřeby.
Knihovna Swing se skládá ze dvou klíčových prvků: komponent a kontejnerů. Komponenta je nezávislý vizuální ovládací prvek, jako je tlačítko nebo textové pole. Kontejner uchovává skupinu komponent. Veškerá grafická uživatelská rozhraní na bázi knihovny Swing mají alespoň jeden kontejner.

2.3 XML

Značkovací jazyk XML [7] (Extensible Markup Language) přišel v první verzi roku 1998. Jde o otevřený formát umožňující mimo jiné kontextové vyhledávání a transformace textu. XML specifikuje obsah dokumentu. Výhoda XML je, že si jeho uživatelé mohou nadefinovat vlastní tagy, kterými lze přesně vyjádřit význam prezentovaných informací. V přiloženém softwaru slouží k uložení hlavního programu a otázek pro cvičení. Cvičení se díky jazyku XML dají snadno editovat pomocí externího souboru.

2.4 IDE a srovnání s konkurencí

Pro vývoj tohoto softwaru jsem zvolil prostředí IntelliJ IDEA , protože jsem s ním dobře seznámen. Existuje jak placená verze Ultimate, tak i verze Community Edition, která je zdarma a pro desktopové aplikace bez webových frameworků stačí. Já jsem ale díky tomu, že společnost Jetbrains (autoři Intellij IDEA) umožňují studentům PřF UP po dobu studia používat plnou verzi, toho využil a používal plnou verzi.



Obrázek 3: Vývojové prostředí IntelliJ IDEA

Ve srovnání s Eclipse, které je pro Javu [6] nejrozšířenější IDE[1]. a je pro něj spousta rozšíření, tak právě díky nim je často problém se vzájemnou funkčností. Poslední srovnatelné prostředí NetBeans nabízí méně funkcí než IntelliJ IDEA. Na druhou stranu bývá označováno jako nejvhodnější profesionální prostředí pro začátečníky. Koncepce a chování všech uvedených vývojových prostředí je velmi podobná. Proto by neměl být problém například přiložený software upravovat a třeba i rozvést pod jiným z výše uvedených prostředí

[1]IDE (Integrated Development Environment) je program pro vývoj softwaru, který k tomu poskytuje komplexní výbavu.

3 Dokumentace projektu

3.1 Společné třídy

Menu
Třída tvořící hlavní menu celého softwaru.

- Menu

Konstruktor hlavního menu zajistí načtení veškerých grafických záležitostí jako je vytvoření a nastavení parametrů všech tlačítek a popisků. Vytvoří a zobrazí také indikátory postupu a zavolá metodu *getProgress*, která zajistí jejich automatické načtení při spuštění softwaru.

- *getProgress*

Metoda načte postupy jednotlivých cvičení a nastaví je příslušným grafickým indikátorům v atributech *progressBar1*, *progressBar2* a *progressBar3*.

- *deleteProgress*

Metoda otevře soubor s uloženými postupy a všechny je vynuluje. Poté zavolá metodu *getProgress* pro zobrazení vymazaného postupu na indikátorech.

MainWindow
Třída tvoří hlavní okno celého softwaru.

- MainWindow

Konstruktor hlavního okna se stará o nastavení jeho základních hodnot. V tomto okně jsou následně zobrazovány všechny panely programu.

- *thisWindowsClosing*

Metoda zajišťuje správné ukončení programu při zavření okna.

Main
Spouštěcí třída celého softwaru. První vytvoří instanci hlavního okna a nastaví mu přiměřenou velikost dle kontroly rozlišení obrazovky uživatele. Následně do okna přidá panel hlavního menu a zobrazí pomocí metody *setVisible*.

SaveAndLoad
Třída slouží k ukládání, načítání hlavního programu a otázek ke cvičení z a do XML souboru.

- *saveFile*

Metoda slouží k uložení všech objektů ve scéně, včetně zachování nastavených vlastností a také parametrů samotné scény. První se vytvoří objekt typu Writer implementované třídy FileWriter pomocí něhož lze zapisovat do souboru. Dále se vytvoří objekt třídy Document (DOM) a tomu se pomocí metody *createDocument* třídy DocumentHelper přiřadí právě vytvořený XML dokument. Nakonec se vytvoří objekt třídy Element, kterému se budou přidávat data ve formě stringů pomocí jeho dvou metod *addElement* a *addAttribute*.
Samotný algoritmus uložení probíhá formou dvou cyklů. V prvním se prochází seznam vytvořených předaných optických objektů, přičemž každé je nejprve označeno jako „*opticalObjects*“ nebo „*lightSources*“. Následuje uložení souřadnic, které se u daného objektu zjistí zavoláním metod *getX* a *getY* pro oba body objektu zvlášť. Poté se pomocí metody *getType* zjistí typ objektu a uloží stejným způsobem jako předchozí hodnoty. Takto se ukládají i další parametry jako například ohnisková vzdálenost. V druhém cyklu se prochází seznam předaných zdrojů světla. Vše se ukládá stejně jako u optických objektů.
Následuje zápis do souboru, jenž byl vytvořen na začátku této metody. Samotný zápis probíhá metodou *write* třídy XMLWriter. V závěru metody je soubor zavřen. Pro použití tříd Document, DocumentHelper, Element a XMLWriter je potřeba implementovat framework DOM4J.

- *loadFile*

První se vytvoří objekt třídy SAXReader pomocí něhož lze číst XML soubor. Dále se vytvoří objekt třídy Document a tomu se pomocí metody *read* výše vytvořeného objektu načte předaná cesta. Pak se vytvoří objekt třídy Element, kterému se přiřadí pomocí metody *getRootElement* data z vytvořeného dokumentu.
Vytváření objektů a nastavení atributů probíhá iterativně. Po kontrole, zdali jde o zdroj světla či opticky objekt, se provede příslušný blok kódu s předem danými příkazy. První dojde k vytvoření objektu dle typu. String s typem objektu se předá jako parametr při volání metody *chooseType* a ta vrátí jeho enumerační typ. Následně se pomocí metod *setX*, *setY* pro oba body objektu nastaví pozice a pomocí dalších metod jako *setFocus* přiřadí např. ohnisková vzdálenost objektu.

Nakonec se takto vytvořené a nastavené objekty přidávají do příslušného předaného seznamu zdrojů světla nebo optických objektů. Tyto seznamy přitom byly předány jako argumenty při volání této metody ze scény. Pro použití tříd SAXReader, Document a Element je potřeba přidat kontejner DOM4J[2].

- *chooseType*

Metoda vrací enumerační typ objektu na základě předaného stringu získaného ze souboru.

- *loadExercices*

Tato metoda stejným způsobem jako metoda *loadFile* načítá otázky pro všechny tři cvičení ze souboru XML. Rozlišuje přitom o které cvičení se jedná a zdali jde o správnou či špatnou odpověď, nebo zadání otázky.

I'm writing something here to test several features.

[2]DOM4J je na Javě založený objektový model dokumentu (DOM).
DOM je API umožňující různou práci s XML[7] dokumenty.
API (Application Programming Interface) je naprogramované rozhraní, které je možné rovnou použít pro interakci s řadou funkcí, například pro přístup k hardwaru. Jedná se o abstrakci.

Níže je ukázka kódu načítajícího otázky do cvičení z XML souboru. Vše probíhá stejně jako je popsáno v metodě *load*. Podobným algoritmem se načítají i objekty do scény.

```
1        SAXReader reader = new SAXReader();
2        Document doc = reader.read(path);
3        Element root = doc.getRootElement();
4
5        for (Iterator i = root.elementIterator(); i.hasNext(); ) {
6           Element el = (Element) i.next();
7           switch (el.getName()) {
8              case "Exercise1": {
9                 if (exercise == 1) {
10                   ExerciseQuestion exerciseQuestion = new
                        ExerciseQuestion(el.attributeValue("question")
                        , el.attributeValue("correctAnswer"),
11                         el.attributeValue("wrongAnswer1"), el.
                              attributeValue("wrongAnswer2"), el.
                              attributeValue("wrongAnswer3"));
12                   questionList.add(exerciseQuestion);
13                } else break;
14             }
15             break;
16             case "Exercise2": {
17                if (exercise == 2) {
18                   ExerciseQuestion exerciseQuestion = new
                        ExerciseQuestion(el.attributeValue("question")
                        , el.attributeValue("correctAnswer"),
19                         el.attributeValue("wrongAnswer1"), el.
                              attributeValue("wrongAnswer2"), el.
                              attributeValue("wrongAnswer3"));
20                   questionList.add(exerciseQuestion);
21                } else break;
22             }
23             break:
24             case "Exercise3": // {
25
26             // stejnym zpusobem se vytvori (nacte) i cviceni 3
27             // jednoduse by se zde dalo nacitat cviceni vice a
                  pridat volbu do hlavniho menu
28
29          }
```

Zdrojový kód 1: Načtení otázek pro cvičení

3.2 Třídy hlavního programu

BeamAngle
Třída zajišťuje výpočet a nastavení úhlu lomu (odrazu a průchodu) všech paprsků.

- *setAngleForBeam*

Metoda provádí výpočet úhlu lomu paprsku při jeho odrazu, či průchodu optickým objektem. Přijímá jako potřebné argumenty zdroj světla a optický objekt u kterých došlo k vzájemnému kontaktu. Následně se pomocí předaných objektů kontroluje, jaký má optický objekt úhel a odkud přichází paprsek. Poté se již dle typu předaného optického objektu provede konkrétní výpočet. Výsledkem výpočtu je úhel lomu, který je použit k následnému vykreslování paprsku ve scéně.

Difference
Třida pro výpočet rozdílu.

- *getDiference*

Tato metoda získá vzdálenost, která tvoří rozdíl mezi místem kontaktu a místem lomu paprsku. Argumenty jsou zdroj světla a optický objekt, u nichž došlo ke kontaktu. První se ověří, jestli jde o zrcadlo nebo čočku.
Pokud jde o zrcadlo, tak se nejdřív vypočítá, jak vysoko je paprsek nad středem objektu. Poté se zjistí jeho typ. Nakonec podle zjištěných kritérií proběhne vyhledání příslušné diference pro dané zrcadlo, která se vrátí jako parametr použitý při vykreslování paprsku.
Pokud šlo o čočku, tak se vzdálenost nad středem zjišťovat nemusí, protože paprsek čočkami prochází skrz v jakékoliv výšce. Zjistí se však vzdálenost paprsku do středu čočky, kde se bude paprsek lámat. To se vypočítá odečtením souřadnic zdroje světla od čočky, nebo naopak dle směru paprsku. Takto vypočítaný rozdíl se vrátí a je použit stejně jako u zrcadel.

LightSource
Třída sloužící pro objekty zdrojů světla

- *lightSource*

Konstruktor zdrojů světla přiřazuje výchozí a předané hodnoty atributů při vytváření objektu ve scéně.

- *getColor*

Metoda vrací atribut *color*, který uchovává aktuálně nastavenou barvu paprsku.

- *setlightSourceColor*

Metoda nastavuje atribut *color* na předanou hodnotu a tímto následně dochází ke změně barvy paprsku zdroje světla u kterého je metoda volána.

- *getX*

Metoda vrací souřadnici x uloženou v atributu *x*.

- *getY*

Metoda vrací souřadnici y uloženou v atributu *y*.

- *getX2*

Metoda vrací souřadnici x pro druhý bod objektu uloženou v atributu *x2*.

- *getY2*

Metoda vrací souřadnici y pro druhý bod objektu uloženou v atributu *y2*.

- *getlightSourceWidth*

Metoda vrátí hodnotu atributu *width*. Ten uchovává šířku paprsku zdroje světla.

- *setlightSourceWidth*

Metoda nastavuje atribut *width* na předanou hodnotu a tím mění šířku paprsku.

- *setAngle*

Metoda nastavuje atribut *angle* na předaný úhel, přičemž kontroluje a případně hned opraví, pokud došlo k přetečení, nebo naopak podtečení. Tímto dochází k nastavení úhlu paprsku při vykreslování.

- *setLocation*

Metoda přesune optický objekt na předané souřadnice a nastaví nové hodnoty atributů *x, x2, y* a *y2*.

- *drawLine*

Metoda volaná při vykreslování paprsku, pomocí které se vytváří 2D grafika zdroje světla uložená v atributu *graphics*.

OpticalObject
Třída slouží pro všechny ostatní optické objekty, které nejsou zdroje světla.

- *OpticalObject*

Konstruktor optických objektů nastavuje obě předané vektorové pozice objektu a výchozí parametry jeho vlastností. Dále nastavuje enumerační typ, předaný jako další argument pro konstruktor, při vytváření instance objektu.

- *getType*

Metoda vrací enumerační typ optického objektu, u kterého je volána.

- *getFocus*

Metoda vrací ohniskovou vzdálenost optického objektu, pokud ji má jinak null.

- *getIndex*

Metoda vrací index lomu optického objektu, pokud ji má jinak null.

- *getX*

Metoda vrací souřadnici x uloženou v atributu *x*.

- *getY*

Metoda vrací souřadnici y uloženou v atributu *y*.

- *getX2*

Metoda vrací souřadnici x pro druhý bod objektu uloženou v atributu *x2*.

- *getY2*

Metoda vrací souřadnici y pro druhý bod objektu uloženou v atributu *y2*.

- *setAngle*

Metoda ukládá předanou hodnotu úhlu. Před uložením hodnotu zkontroluje a případně pokud došlo k přetečení, nebo naopak podtečení, tak ji opraví.

- *setLocation*

Metoda přesune optický objekt na předané souřadnice a nastaví nové hodnoty atributů *x, x2, y* a *y2*.

Scene
Třída tvořící scénu hlavního programu.

- Type

Vytváří enumerační typy pro všechny optické objekty včetně zdrojů světla, které jsou předávány jako argumenty při vytváření nových objektů ve scéně.

- *draw*

Tato metoda se stará o vykreslení paprsků u všech vytvořených zdrojů světla a také všech ostatních objektů ve scéně. Je volána opakovaně přičemž se vždy první odstraní předchozí vykreslení scény. Kontroluje se, zda existuje optický objekt (atribut *opticalObjects* – seznam optických objektů). Pokud existuje, tak se v cyklu budou procházet dráhy všech vytvořených zdrojů světla (atribut *lightSources* - seznam zdrojů světla) a bude se kontrolovat, jestli došlo ke kontaktu s paprskem. Když u zdroje světla ke kontaktu došlo, zavolá se příslušná metoda, které se tento zdroj světla předá a ta už zajistí výpočet úhlu lomu, rozdílu a provede se vykreslení paprsku.
Pokud zdroj světla není s ničím v kontaktu, tak dojde k tomu, jako když žádný optický objekt není vytvořen. To znamená, že dojde v cyklu pouze k vykreslení paprsku u zdroje světla před sebe směrem dle svého úhlu.

- *reset*

Metoda vyprázdní seznamy s objekty v atributech *lightSources* a *opticalObjects*, čímž odstraní všechny objekty ve scéně. Následně ve scéně nastaví výchozí hodnoty jako při prvním spuštění programu.

- *drawRefractionBeam*

Metoda funguje na principu opakovaně běžícího cyklu. Ten se restartuje a běží znovu vždy, když se paprsek předaného zdroje světla ocitne v kontaktu s dalším optickým objektem. Takto může paprsek procházet, anebo se odrazit od libovolného počtu objektů.
První se ověřuje, jestli je na aktuálně kontrolované pozici paprsku další optický objekt a zároveň nejde o stejný objekt jako posledně (to je kvůli správnému vzdálení paprsku od objektu).
Pokud je první podmínka splněna, tak dojde k výpočtu rozdílu pomocí metody *getDifference* třídy Difference a ta je následně použita pro správné vykreslení paprsku. Paprsek se poté nachází na souřadnicích, kde dochází k lomu. Úhel lomu se vypočítá a rovnou pro paprsek nastaví pomocí metody *setAngleForBeam* třídy BeamAngle. Celý cyklus následně pokračuje stejným způsobem první nebo další průchod dle toho, zda došlo k jeho restartu při kontaktu s objektem.

- *thisMouseClicked*

Tato metoda slouží k zjištění kliknutí do scény nebo na tlačítka v menu. První se zjistí, jestli se kliklo na objekt, nebo na tlačítko v menu. U volaných metod se přitom jako argumenty použijí předané souřadnice místa kliku.
Mohou nastat 3 možnosti:

1. Kliknutím na zdroj světla dojde k jeho zvolení a atributu *selectedObject* se nastaví typ aktuálně zvoleného objektu.

2. Kliknutím na optický objekt dojde k jeho zvolení. Pokud má objekt nastavení ohniskové vzdálenosti nebo indexu lomu, tak se podle jeho typu následně zobrazí příslušný posuvník. Atributu *selectedObject* se opět nastaví typ optického objektu dle jeho enumeračního typu získaného pomocí metody *getType*.

3. Kliknutím mimo objekt pravým tlačítkem dojde k odznačení aktuálně zvoleného objektu, pokud není zvoleno v menu jinak. Případně při permanentním stisku tlačítka s posunem dojde k posunu scény. Pokud došlo ke kliku do menu například na objekt k přidání, tak je vybrána příslušná akce pod tlačítkem.

3.3 Třída teorie

Theory
Třída seznamuje uživatele s teorií o optických objektech, které jsou vkládány do scény.

- Theory

Konstruktor třídy první zajišťuje, že při spuštění teorie dojde k vytvoření a nastavení parametrů všech tlačítek a popisků. Potom vytvoří instanci labelu (atribut *labelTheory*) na který se později přidá obsah zvolený v nabídce. Label je následně přidán do atributu *jScrollPane*. Teorie k objektům se přenastaví jednoduše vždy na tomto labelu. To probíhá pomocí událostních metod tlačítek v nabídce teorie.

- *isScrollPaneInFrame*

Metoda kontroluje, jestli byl již atribut *jScrollPane* přidán do okna a případně ho tam přidá a nechá zobrazit. Po kliknutí na tlačítko zpět při opouštění teoretické části dochází následně k jeho odebrání společně s panelem teorie. Atribut *jScrollPane* poskytuje scrollbary pro všechny položky teorie.

3.4 Třídy cvičení

Excersise
Třída sloužící pro testování znalostí formou cvičení.

- Exercise

Konstruktor této třídy zajišťuje, aby při spuštění cvičení došlo k vytvoření a nastavení parametrů všech tlačítek a popisků. Po nastavení grafických objektů zavolá metodu *begin*, která zajistí načtení první otázky a vynuluje statistiky.

- *begin*

- Metoda pomocí již implementované metody *shuffle* zamíchá atribut *questionList* s otázkami, vynuluje statistiky, nastaví neutrální ikonku úspěšnosti a zavolá metodu *addAnswers*.

- *addAnswers*

Tato metoda zajišťuje načítání všech otázek a odpovědí u cvičení. Pokud nedošlo k vyčerpání otázek, tak tato metoda na základě náhodného čísla nastaví jednomu tlačítku správnou odpověď a ostatním ty špatné. Z atributu *questionList* pak už jen odebere použitou otázku i s odpověďmi.
Pokud již byly vyzkoušeny všechny otázky, provede se otevření souboru s uloženým pokrokem, porovná se počet dosažených správných odpovědí a případně se přepíše za vyšší. Provede se nastavení statistik a změní se barvy tlačítek na výchozí hodnoty. Tato metoda se volá jako událost spojená s kliknutím na tlačítka odpovědí, kdy dojde k vyhodnocení včetně změny barvy a aktualizací statistik.

ExerciseQuestions
Třída sloužící pro vytváření otázek ke cvičení.

- ExerciseQuestions

Konstruktor bere jako argumenty zadání otázky a odpovědi na ni, kterými rovnou nastaví všechny atributy otázky. Tyto atributy jsou *question*, *correctAnswer*, *incorrectAnswer1*, *incorrectAnswer2* a *incorrectAnswer3*.

- *getQuestion*

Metoda vrací zadání otázky v atributu *question*.

- *getCorrectAnswer*

Metoda vrací správnou odpověď otázky v atributu *correctAnswer*

- *getIncorrectAnswer1*

Metoda vrací první špatnou odpověď otázky v atributu *correctAnswer1*.

- *getIncorrectAnswer2*

Metoda vrací druhou špatnou odpověď otázky v atributu *correctAnswer2*.

- *getIncorrectAnswer3*

Metoda vrací třetí špatnou odpověď otázky v atributu *correctAnswer3*.

- *loadQuestions*

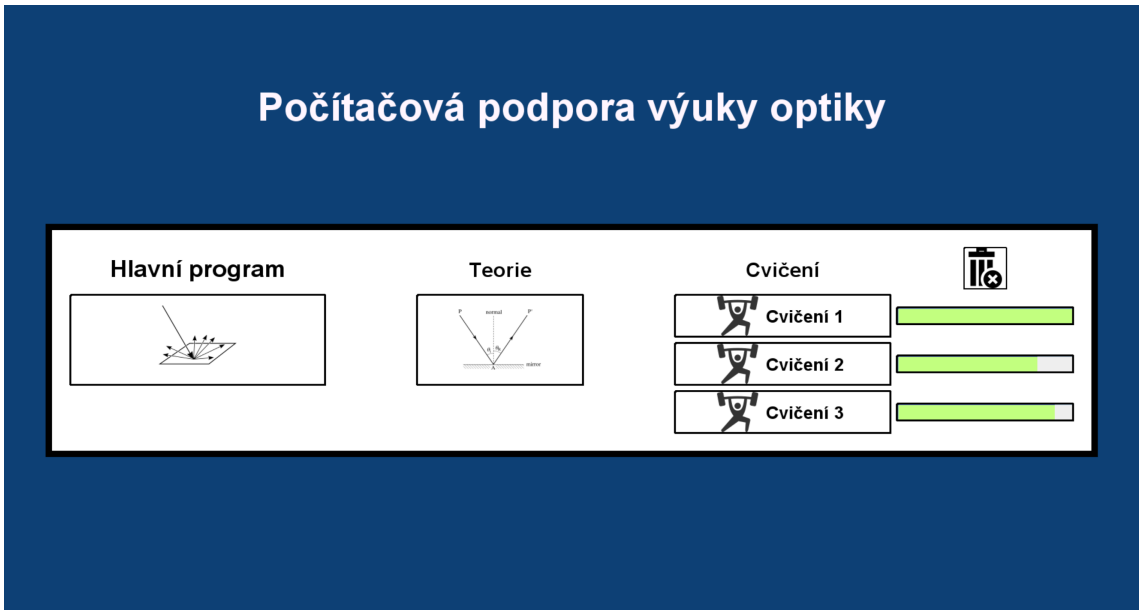
Metoda slouží k načtení všech otázek pro jednotlivá cvičení uložených v souboru questions.xml. Volána je automaticky hned při každém spuštění programu.

4 Popis softwaru

4.1 Instalace a spuštění softwaru

Ke spuštění softwaru je třeba mít nainstalované JRE.[3]
odkaz ke stažení: https://www.java.com/en/download
Software se spouští jednoduše kliknutím na soubor Optika.

4.2 Hlavní menu

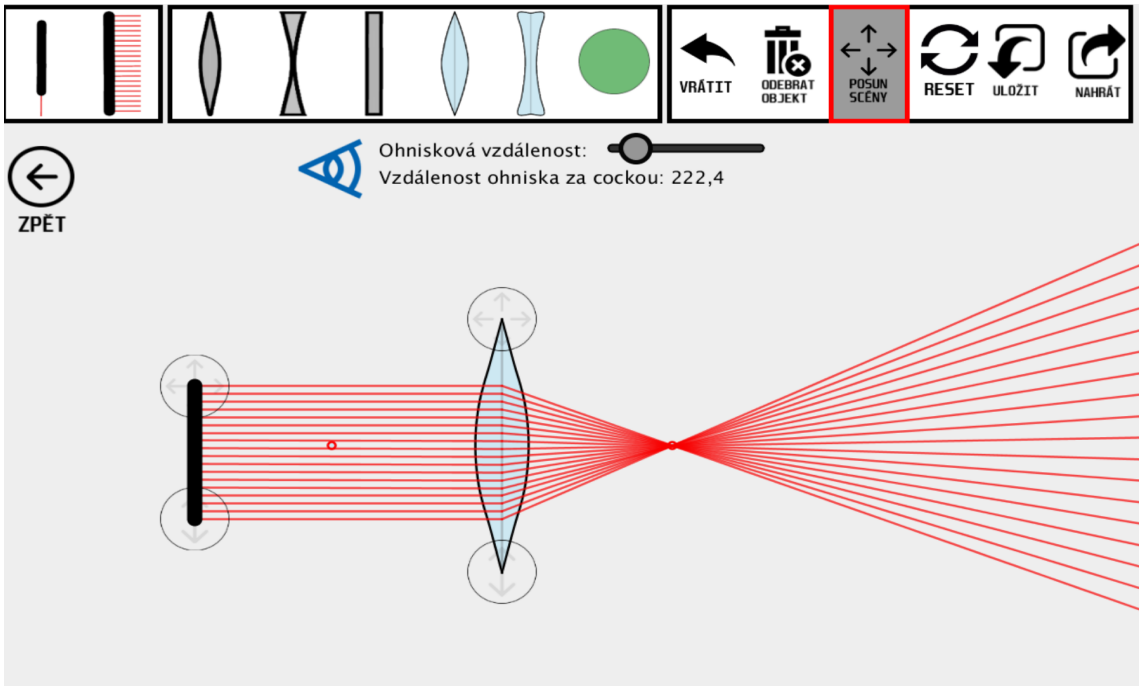


Obrázek 4: Hlavní menu

Zde se uživatel ocitne po spuštění softwaru. Hlavní menu umožňuje přepínat mezi jednotlivými částmi. Je tady také zobrazen postup jednotlivých cvičení.

[3]JRE (Java Runtime Environment) je běhové prostředí potřebné pro používání programů naprogramovaných v Javě.

4.3 Hlavní program



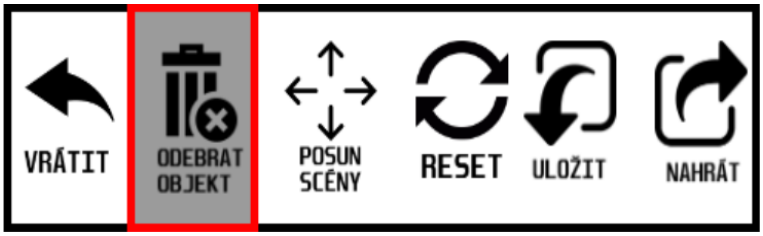
Obrázek 5: Hlavní program a scéna

Hlavní program tvoří praktickou část, v níž se nachází scéna. Do scény lze přidávat z menu různé vizuálně zobrazené optické objekty včetně zdrojů světla. Uživatel může následně s objekty ve scéně provádět různé operace, jako nastavení ohniskové vzdálenosti, rotace, změna velikosti a další. Změna vlastností objektů se zobrazuje v reálném čase a to tak, že jsou paprsky vykresleny plynule.

4.3.1 Ovládání scény

Posun objektů ve scéně probíhá jednoduše pomocí drag & drop v horní části objektu. Rotace nebo změna velikosti objektu probíhá stejně, ale v dolní části objektu. Pro operace jako je nastavení ohniskové vzdálenosti se zobrazí příslušný posuvník. Scénu je možné kdykoliv přiblížit či oddálit kolečkem myši, nebo posouvat stejně jako objekty.

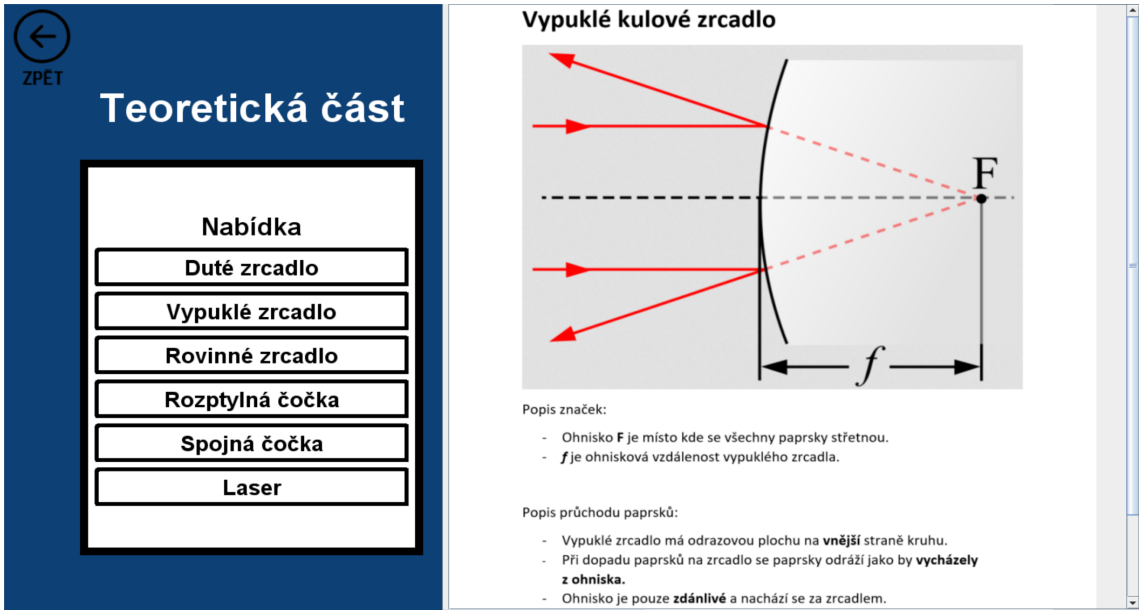
4.3.2 Pravá část menu



Obrázek 6: Pravá část menu hlavního programu

Možnost vrátit vrací poslední akci. Položka odebrat objekt slouží k odstranění jednotlivých objektů. K odstranění všeho ve scéně slouží reset, který uvede scénu do stavu nově spuštěného programu. Posun scény posouvá všechny objekty jednoduše drag & drop. Položky uložit a nahrát otevírají nové okno, kde si lze klasicky zvolit cestu kam se soubor uloží nebo odkud se načte. Pokud je položka v menu aktivní, tak je zdůrazněna červeným rámečkem. Zrušení vybrané položky se provede opakovaným kliknutím.

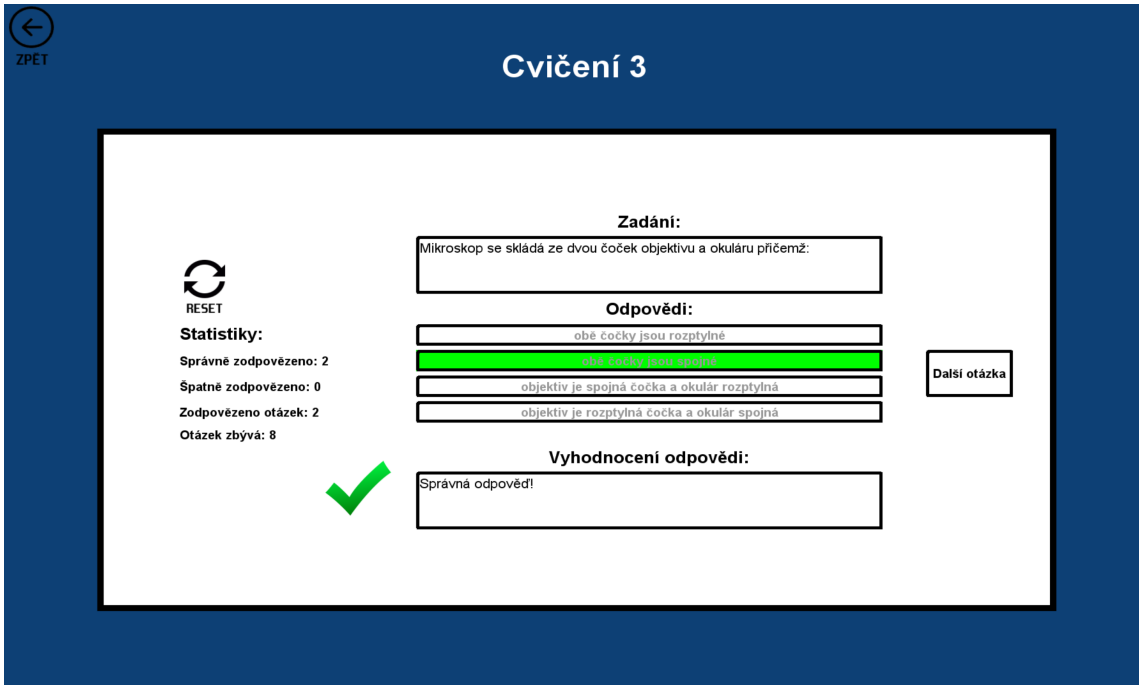
4.4 Teorie



Obrázek 7: Teoretická část

Teoretická část seznamuje uživatele s teorií jednotlivých optických objektů, které jsou vkládány do scény. Graficky i textově je zde zobrazeno, kde má daný objekt ohnisko, nebo informace o převráceném obrazu a jiné. Uživatel také zjistí, kde se objekty užívají v praxi. Aby se nestalo, že se zobrazovaný obsah nevejde uživateli na obrazovku, tak je teoretická část vybavena scrollbary.

4.5 Cvičení



Obrázek 8: Testovací část

Třetí část programu slouží k testování znalostí o optice formou cvičení s otázkami. Je rozdělena do tří částí, přičemž je možné mezi nimi volit bez omezení. Každá část má svoje vlastní statistiky týkající se zodpovídaných otázek. V případě špatné odpovědi se zobrazí i ta správná. Po zodpovězení otázky se zpřístupní položka posunu k další otázce. Odpovědi na zodpovězenou otázku již nejsou aktivní a došlo k aktualizaci statistik. Nejvyšší dosažený pokrok je následně uložen do souboru a zobrazen na progressbaru v hlavním menu. Všechny otázky lze jednoduše editovat prostřednictvím externího souboru questions.xml.

Závěr

Původním plánem bylo vytvořit jeden hlavní program, jenž poskytne uživateli podporu při výuce optiky prováděním pokusů s lomem světla zábavnou formou. Ve výsledku byl vytvořen software rozdělený na tři části ,kdy se hlavní program tvořící praktickou část rozšířil o část teoretickou a testovací. Mezi částmi lze jednoduše přepínat.
Hlavní část obsahuje scénu do které lze přidávat různé optické objekty, včetně zdrojů světla. Uživatel ve scéně může s objekty provádět různé operace jako je nastavení ohniskové vzdálenosti, rotace, změna velikosti a další. Provádění operací jsem se snažil udělat uživatelsky přívětivé. Tam kde je to praktické lze s objekty manipulovat pomocí drag & drop a jinde se používá posuvník. Operace související s lomem světla jsou naprogramovány dle Snellova zákona. Průchody paprsků ve scéně jsou vykreslovány v reálném čase a jsou zcela plynulé. Scénu s objekty je možné celou posunout, nebo si nastavit její vzdálenost. Ke konci vývoje do hlavní části přibyla možnost scénu uložit, anebo udělat krok zpět.
Volitelná teoretická část umožňuje uživateli zjistit bližší informace o tom co dělají a kde se užívají objekty vkládané do scény. Poslední třetí část testuje znalosti uživatele formou otázek z optiky. Do testovací části je možné jednoduše přidávat další otázky prostřednictvím externího souboru.

Conclusions

The intention was to create one main part of the program, which will provide the user with support in teaching optics by performing light refraction experiments. As a result, software was created divided into three parts, where the main program forming the practical part was expanded by a theoretical and testing part. You can easily switch between parts.
The main part contains a scene to which various optical objects can be added, including light sources. The user in the scene can perform various operations on objects such as setting the focal length, rotation, resizing and more. I tried to make operations user friendly. Where practical, objects can be manipulated using drag & drop and elsewhere a slider is used. Operations with refraction are programmed according to Snell's law. The beam passages in the scene are plotted in real time and are completely smooth. It is possible to move the whole scene with the objects or to set its distance. At the end of the development, the option to save the scene or take a step back was added to the main part.
The optional theoretical part allows the user to find out more information about what they do and where the objects inserted into the scene are used. The last third part tests the user's knowledge in the form of questions from optics. Additional questions can be easily added to the test section using an external file.

A Obsah přiloženého CD/DVD

bin/
: Program Optika, spustitelný přímo z CD/DVD. Adresář obsahuje i všechny runtime knihovny a další soubory potřebné pro bezproblémový běh instalátoru a programu z CD/DVD.

doc/
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu, vložené obrázky, apod.

src/
: Kompletní zdrojové texty programu Optika se všemi potřebnými (příp. převzatými) zdrojovými texty, knihovnami a dalšími soubory potřebnými pro bezproblémové vytvoření spustitelných verzí programu.

readme.txt
: Instrukce pro instalaci a spuštění programu Optika, včetně všech požadavků pro jeho bezproblémový provoz.

Navíc CD/DVD obsahuje:

install/
: Instalátory aplikací, runtime knihoven a jiných souborů potřebných pro provoz programu Optika, které nejsou standardní součástí operačního systému určeného pro běh programu.

Literatura

[1] LEPIL, Oldřich. *Fyzika pro gymnázia – Optika*. 2015. 104 s.

[2] BOUCHAL, Zdeněk. *Optika: Učební pomůcka pro studenty*. 108 s. Dostupný také z: ⟨http://optics.upol.cz/userfiles/file/OPT_PO_CAST_I.pdf⟩.

[3] Snellův zákon. Dostupný také z: ⟨https://en.wikipedia.org/wiki/Snell%27s_law⟩.

[4] List of indices. Dostupný také z: ⟨https://en.wikipedia.org/wiki/List_of_refractive_indices⟩.

[5] SCHILDT, Herbert. *Java 8: Výukový kurz*. První vyd. Brno): Computer Press, 2016. 696 s.

[6] PECINOVSKÝ, Rudolf. *Java 9: Kompletní příručka jazyka*. První vyd. Praha): Grada, 2018. 552 s.

[7] XML. Dostupný také z: ⟨https://cs.wikipedia.org/wiki/Extensible_Markup_Language⟩.

.