Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Vizualizace principu hologramu

Plzeň, 2014 Michael Hadáček

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Michael HADÁČEK**
Osobní číslo:	**A11B0359P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informatika**
Název tématu:	**Vizualizace principu hologramu**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování:

1. Prostudujte současnou implementaci simulátoru difraktivní optiky.
2. Doplňte výpočty pozice obrazu založené na cos2/R rovnici.
3. Navrhněte a implementujte infrastrukturu pro optické členy, zejména zrcátko, dělič svazku a čočku.
4. Navrhněte a implementujte podporu 3D výpočtu chodu paprsků.
5. Program otestujte simulací základních sestav pro záznam hologramů.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **doporuč. 30 s. původního textu**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

Rendl, Kamil: Vizualizace principu hologramu. Plzeň, 2012. Bakalářská práce (Bc.). Západočeská univerzita v Plzni, Fakulta aplikovaných věd. Vedoucí práce Petr Lobaz.

Benton, Stephen A.; Bove, V. Michael. Holographic imaging. Hoboken: Wiley-Interscience, 2008. ISBN 978-0-470-06806-9.

Vedoucí bakalářské práce: **Ing. Petr Lobaz**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **14. října 2013**

Termín odevzdání bakalářské práce: **9. května 2014**

Doc. Ing. František Vávra, CSc.
děkan

L.S.

Prof. Ing. Jiří Šafařík, CSc.
vedoucí katedry

V Plzni dne 17. října 2013

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne …….....

Michael Hadáček

Abstract

This work focuses on improving holographic simulator which uses ray model to simulate and research optical sets. This document describes deficiencies of the original version and also how was this deficiencies repaired and improved. At the end there are tests of the improvements on basic optical sets.

Abstrakt

Tato práce je zaměřena na zdokonalení holografického simulátoru. Jde o software pro simulaci optických soustav v paprskovém modelu a zkoumání jejich vlastností. Práce ukazuje nedostatky původní verze, její vylepšení a ověření funkčnosti na základních optických soustavách.

Poděkování

Děkují tímto vedoucímu mé práce a celého projektu panu ing. Petru Lobazovi za jeho vstřícný přístup a přínosné rady při zpracování této práce.

Obsah

1 Úvod

Základem pro mou práci byl program vytvořený Kamilem Rendlem [1]. Jde o výukový simulátor, který by měl pomoci zájemcům, kteří začínají s holografií, pochopit její základní principy. Pro pokročilejší pak např. simulovat, zda je možné vytvořit různé optické soustavy v reálném prostředí a co k tomu bude potřeba. Tento program používá paprskový model světla. Uživatel tak může sledovat chování jednotlivých paprsků na různých optických členech, zaznamenávat a rekonstruovat hologramy a zkoumat různé vztahy během tohoto procesu. Aplikace je realizována jako Java applet, to znamená, že k jejímu spuštění stačí běžný webový prohlížeč s nainstalovanou podporou Javy.

Simulátor ve své původní verzi obsahoval pouze dva objekty – zdroj světla a difrakční mřížku (hologram), se kterými šlo pracovat. S těmito součástmi lze simulovat základní holografické sestavy, jako např. zaznamenání a následná rekonstrukce jednoduchého hologramu a sledování různých posunů obrazu při změnách rekonstrukčního světla. Tento program měl sloužit nejen k výukovým účelům, ale také pro sestavování reálných optických soustav „nanečisto“, aby se dalo předem zjistit, jestli se taková soustava dá sestavit i na skutečném stole. Sestavy pro zaznamenávání hologramů ovšem obsahují daleko více optických členů. Bylo tedy nutné simulátor obohatit o nové členy, jako např. dělič svazku, čočku a zrcátko.

Dále pak holografický simulátor obsahoval několik zásadních chyb při výpočtech směrů paprsků procházejících vytvořeným hologramem. Takové chyby snižovaly výukové kvality programu.

Jak se s tímto simulátorem pracovalo, popisuje kapitola 2. V třetí kapitole jsou popsány některé optické soustavy, ukazující, jaké optické členy jsou nezbytné pro simulace v holografii. Implementaci změn a zlepšení, které byly předmětem této práce, popisuje kapitola 4. Následuje rozbor adresářové struktury a celkové skladby programu (kapitola 5). Předposlední část patří ověření výsledků (kapitola 6) a shrnutí dosaženého postupu v závěru (kapitola 7).

2 Simulátor

Představme si, jak se s holografickým simulátorem v jeho původní verzi pracovalo a co uměl. Nejdříve je však nutné ustanovit pojem, který je zaveden v simulátoru a bude používán v celé této práci. Jde o slovo „mřížka“. Mřížka označuje materiál, do kterého lze zaznamenat interferenční vzor a ten následně i zkoumat. Název je zaveden, protože hologram je vlastně speciální případ difrakční mřížky. V tomto textu tedy difrakční mřížka, hologram a holografická deska jsou synonyma.

Pro zjednodušení a zrychlení výpočtů se v simulátoru používá rozdělení optických členů na segmenty. Z jednoho světleného zdroje pak na povrch prvku dopadá tolik paprsků, kolik je segmentů. Předpokládá se, že ostatní paprsky, které by dopadaly na daný segment, se chovají stejně nebo s minimální chybou.

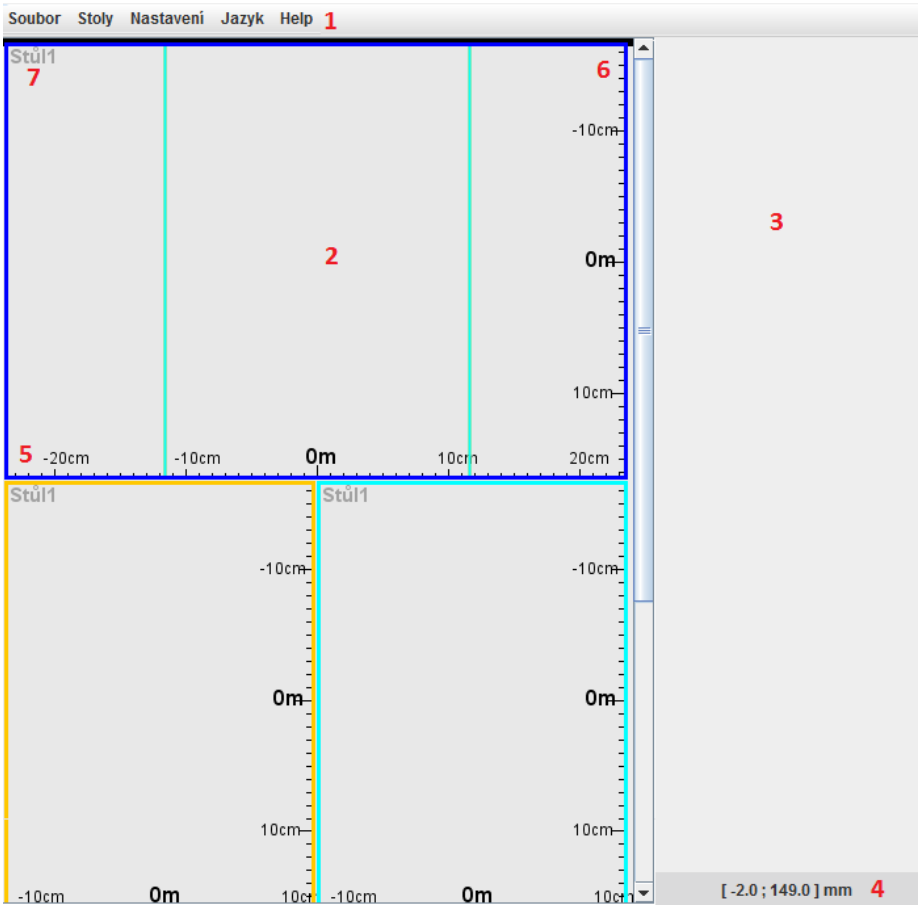
V programu lze vytvářet více stolů. Stůl zde představuje skutečný pracovní stůl, na který se skládají optické členy. Lze tedy vytvářet několik optických soustav najednou a pozorovat jejich chování.

2.1 Rozložení obrazovky

Po spuštění aplikace se objeví hlavní okno (obrázek 1). To je rozděleno na několik částí. V levém horním rohu je umístěno hlavní menu (1), kde lze měnit vlastnosti aplikace, jako např. přepínat jazyk mezi angličtinou a češtinou (položka **Jazyk**), zapnout si zobrazování popisků anebo ukládat aktuální rozestavení na stolech. Záložka **Stoly** je určena k manipulaci se stoly. Uživatel zde může přidávat nové, mazat stávající a měnit nastavení všech vytvořených stolů.

Největší část obrazovky pokrývají pohledy na pracovní stoly (2). Na ukázce je okno rozděleno do tří náhledů. Okno však lze přizpůsobit rozdělením svisle nebo vodorovně na více pohledů nebo naopak jejich sloučením. Každý stůl má svůj název, nejčastěji složen z klíčového slova „Stůl“ a pořadí, ve kterém byl vytvořen. Název stolu se nachází v levém horním rohu náhledu (7). Pro lepší orientaci v náhledu slouží vodorovná osa x (5) a svislá osa y (6).

V pravé části je vyhrazeno místo pro panel s vlastnostmi vybraného objektu (3). Na stole momentálně není nic umístěno, tak je panel prázdný (více v kapitole 2.2.2). Pravý dolní roh patří aktuálním souřadnicím kurzoru (4).



Obrázek 1: Hlavní obrazovka simulátoru: 1 - hlavní menu, 2 - pracovní stoly, 3 - místo pro panel vlastností, 4 - pozice kurzoru, 5 – osa x, 6 – osa y, 7 – název stolu

2.2 Vytvoření hologramu

Program je určený k simulování záznamu a rekonstrukce hologramů. Ukažme si tedy na jednoduchém příkladu, jak takový hologram vytvořit.

2.2.1 Přidání objektů na stůl

Pro záznam potřebujeme zaznamenávaný objekt, referenční světlo a holografickou desku, do které se záznam uloží (názvosloví referenční a objektový paprsek převzato z [2]). Pro zjednodušení nám místo objektu postačí pouze bodový zdroj světla. Musíme tedy zvolené prvky umístit na stůl. To uděláme pomocí **kontextové nabídky stolu** (obrázek 2), kterou vyvoláme kliknutím pravého tlačítka myši, kamkoliv do náhledu na stůl.

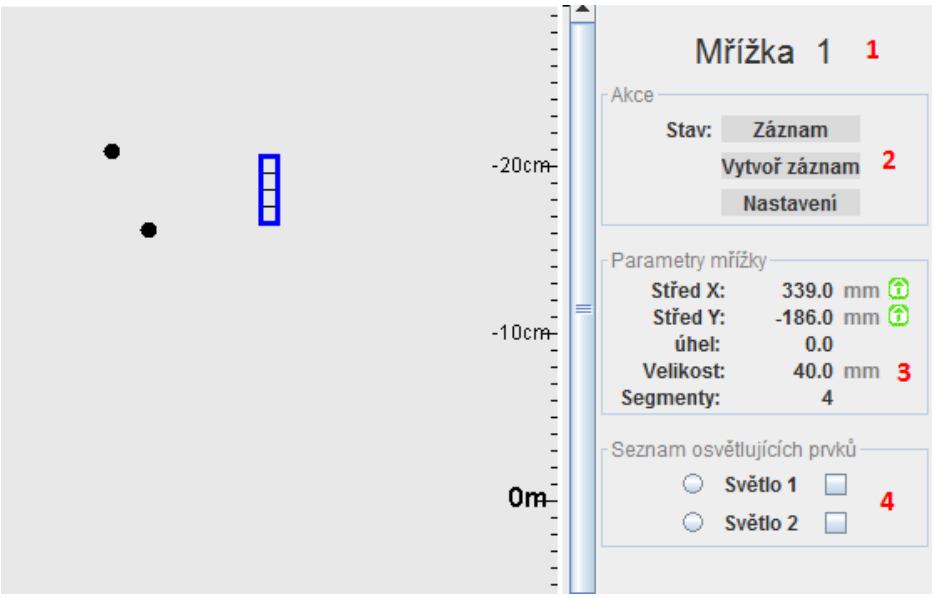
Obrázek 2: Kontextová nabídka stolu

Nabídka obsahuje několik možností. Zapnutím milimetrového papíru se v náhledu zobrazí milimetrová čtvercová síť pro lepší orientaci. **Přidat světlo/mřížku** přidá požadovaný objekt na souřadnice, kde byla vyvolána nabídka. Další blok slouží pro ovládání daného náhledu, **Přepnout stůl** – přepne pohled na zvolený stůl. **Obsluha** ovládá rozložení obrazovky – můžeme zde rozdělit aktuální pohled na více částí. **Nastavení** vyvolá okno s nastavením.

Přidáme vybrané komponenty na stůl vyvoláním kontextové nabídky ve vybraném místě. Potřebujeme dvě světla a mřížku. Po vložení může vypadat rozestavení například jako na obrázku 3, důležité pro záznam v simulátoru ovšem je, aby se oba světelné zdroje nacházely na **stejné straně od mřížky.** S prvky se dá po přidání na stůl manipulovat, a to buď chycením a tažením pomocí myši anebo kliknutím na daný prvek a nastavením x-ových a y-ových souřadnic v jeho panelu vlastností.

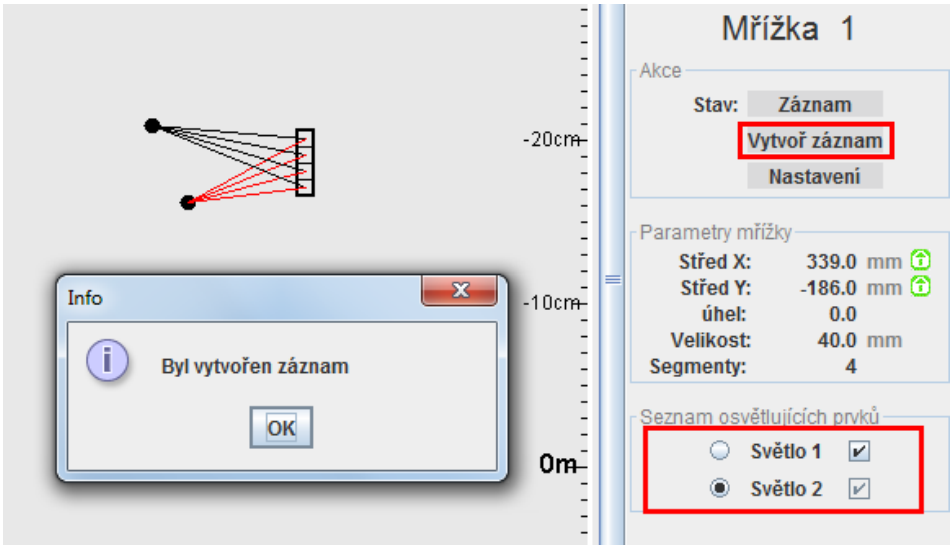
2.2.2 Nastavení mřížky a zaznamenání interferenčního vzoru

Nyní když máme správné rozestavení, nastavíme mřížku. Klikneme na mřížku, zobrazí se její vlastnosti (obrázek 3). Panel obsahuje několik bloků s různými možnostmi. Můžeme si definovat vlastní název mřížky (1). Část nazvaná **akce** (2) obsahuje tlačítka k **nastavení mřížky**, **vytvoření záznamu** a **přepínání režimů zobrazení** mřížky. Třetí část slouží k upravení **pozice**, **velikosti, úhlu a počtu segmentů** holografické desky (3). V posledním bloku (4) si volíme, které **zdroje budou mřížku ovlivňovat** a který z nich bude **referenční**. Kliknutím na zaškrtávací čtvereček říkáme, že daný zdroj osvětluje mřížku, a volbou radiobuttonu vlevo od názvu volíme referenční člen.



Obrázek 3: Ukázka rozestavení a vlastností mřížky

Chceme tedy, aby desku osvětlovala obě světla, jedno z nich bude referenční a zbývající objektové. Máme mřížku osvětlenou dvěma svazky paprsků, zbývá už jen vytvořit záznam kliknutím na tlačítko **Vytvoř záznam** (obrázek 4). Jestli se vše povedlo, zobrazí se informační hláška.



Obrázek 4: Zaznamenání hologramu

2.3 Rekonstrukce hologramu

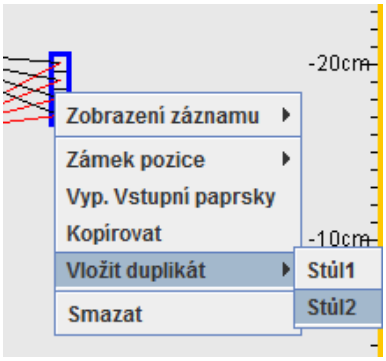
Zaznamenali jsme interferenční vzor na holografickou desku, ta se při nasvícení světlem bude chovat jako difrakční mřížka. Paprsky, kterými jí osvítíme, změní svůj směr podle vztahu, vidíme vzniklá difrakční maxima. (Vzorec viz kapitola 4.1.1 Matematický výpočet výstupních úhlů).

2.3.1 Vytvoření nového stolu

Abychom využili potenciál programu, vytvoříme si nový stůl, na kterém budeme hologram rekonstruovat. To uděláme přes hlavní menu záložka **Stoly** volba **Nový stůl**. Rozdělíme obrazovku na dvě části, abychom nemuseli neustále přepínat mezi pohledy na Stůl 1 a Stůl 2. Vyvoláme kontextovou nabídku možnost **Obsluha** – **Rozdělit vodorovně**. Spodní okno si přepneme na Stůl 2 – ve spodním okně opět zobrazíme kontextovou nabídku, tentokrát zvolíme **Přepnout stůl** a vybereme **Stůl 2**. Na novém stole zatím nic nemáme, potřebujeme tam zkopírovat aktuální rozestavení z prvního stolu. Pro rekonstrukci hologramu nám stačí mřížka s vytvořeným záznamem a rekonstrukční světlo. Jako rekonstrukční světlo použijeme např. referenční světlo, kterým jsme hologram vytvořili. Vytvoříme duplikáty těchto dvou prvků na druhý stůl.

2.3.2 Vložení duplikátů

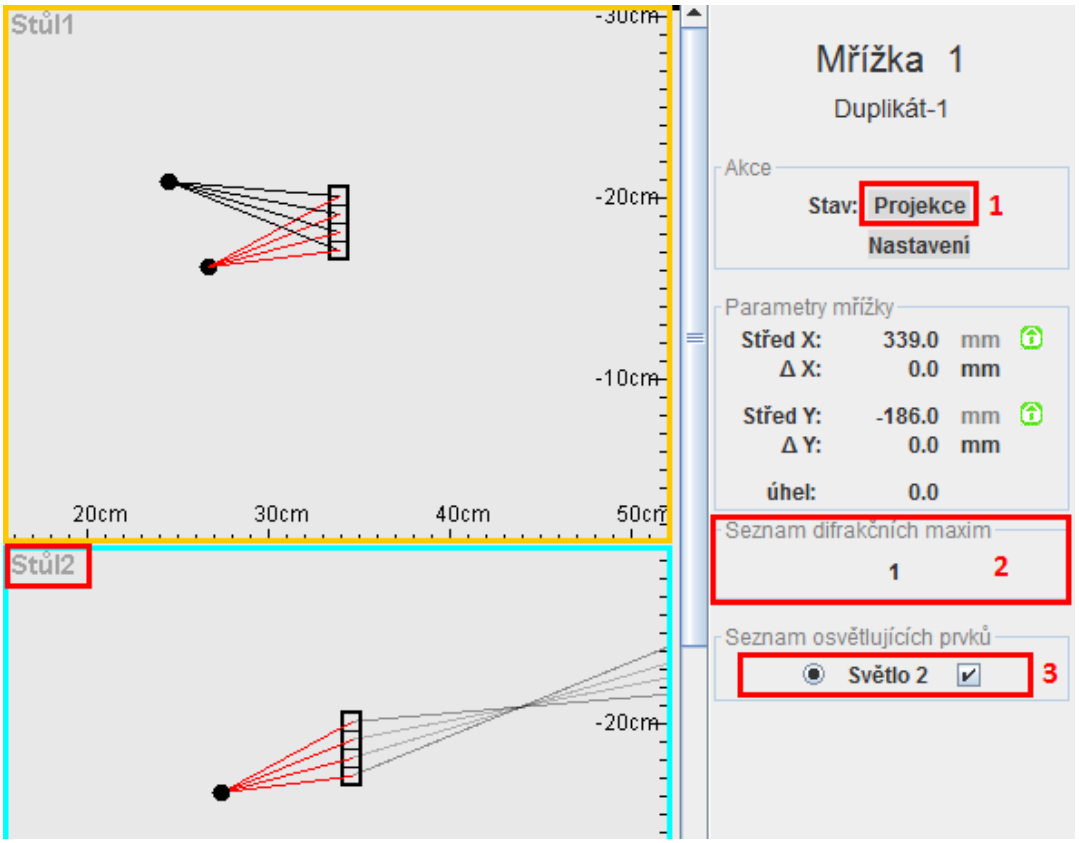
Každý prvek, stejně jako stůl, má svoji kontextovou nabídku, která se vyvolá kliknutím pravého tlačítka myši na daný prvek (obrázek 5). U mřížky můžeme volit různé režimy záznamu přes **Zobrazení záznamu. Zámky pozic** umožňují ukotvit prvek na daném místě. Nás nyní nejvíce zajímá volba **Vložit duplikát**. Duplikát daného prvku má stejné souřadnice a vlastnosti a navíc je spojen se svým rodičem. Když pohneme s mřížkou na jednom stole, její duplikáty na ostatních stolech také změní pozice.



Obrázek 5: Kontextová nabídka mřížky

2.3.3 Rekonstrukce hologramu

Vložíme duplikát mřížky a referenčního světla na Stůl 2. Okno naší aplikace by mělo vypadat asi takto (obrázek 6). Abychom mohli rekonstruovat hologram v simulátoru, musíme přepnout mřížku do režimu projekce, zobrazíme vlastnosti mřížky (kapitola 2.2.2). Kliknutím na tlačítko **Záznam** přepneme mřížku do režimu projekce (obrázek 6). Můžeme si všimnout, že panel s vlastnostmi mřížky se změnil. Nyní už pozorujeme ohyb paprsků na mřížce. Můžeme zkoumat různé vztahy při změně rekonstrukčního světla (3) nebo jeho vlastností anebo **zobrazit další difrakční maxima** (2).



Obrázek 6: Rekonstrukce hologramu

2.4 Nedostatky původní verze

- Jelikož je program velice rozsáhlý, nevyhnul se bezchybnosti. Asi největší problémem byly špatné výpočty úhlů a v závislosti na tom i nekorektní vykreslování výstupních paprsků. V některých případech se pak paprsky chovaly naprosto opačně, než by se dalo čekat.
- Rovněž zobrazování záporných difrakčních maxim nebylo řešeno tak, aby odráželo skutečné chování mřížky. Navíc se v simulátoru zcela chybně vykreslovala i neexistující záporná maxima.
- Program se nevyhnul ani vytvoření špatného záznamu v mřížce, a to v případě, že svazky, jež měli vytvořit interferenční vzor, měly rozdílnou vlnovou délku.
- Původní verze dovolovala práci pouze s holografickou deskou a světelným zdrojem. Jak se později ukáže, k sestavení i základních optických soustav je zapotřebí více členů.
- Dále aplikace obsahovala několik menších či větších nedodělků v grafickém uživatelském rozhraní, např. občasná problematika při přepnutí jazyku, absence exportu do vektorové grafiky i přes možnost zobrazovanou v menu, atp.

2.5 Alternativy programu

K tomuto simulátoru jsem nenašel žádný ekvivalent. Existují programy, které simulují interferenci a difrakci vln [4], [5] a jiné. Specializované aplikace, které ukazují chování zrcadel, čoček atd [6]. Nebo úzce zaměřené programy pro simulaci různých sestav pro příklad třeba simulace pro Michelsonův interferometr [7]. Ovšem univerzální program pro simulaci záznamu a rekonstrukce hologramu pomocí paprskového modelu s možností vytvoření několika stolů nejspíše neexistuje.

3 Požadavky na novou verzi

Původní simulátor v podstatě neumožňuje práci s jinými optickými členy a simulování komplikovanější optické soustavy. V následující kapitole si uvedeme několik vzorových příkladů optických soustav, které se v holografii běžně používají.

3.1 Používané optické soustavy

Poznatky převzaty z [3].

3.1.1 Interferometr

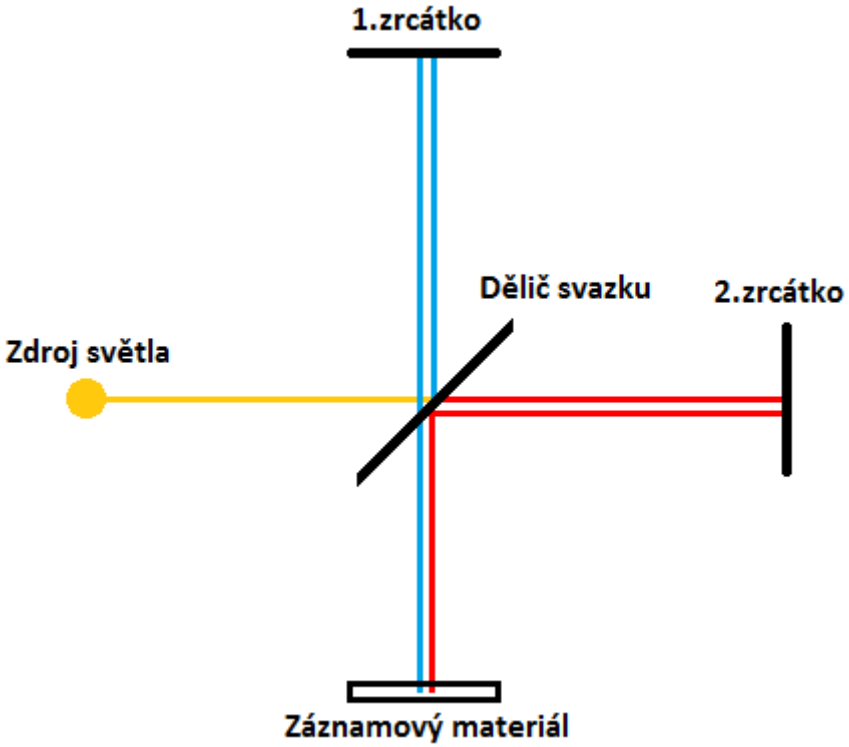
Interferometr je zařízení využívající interference světla pro velice přesné měření vzdáleností, vlnových délek světla a v neposlední řadě také měření indexu lomu u kapalin a plynů.

3.1.2 Michelsonův interferometr

Nejstarší zástupce interferometrů je Michelsonův interferometr, který sestavil A. A. Michelson v roce 1881 původně pro studie světelného éteru (schéma na obrázku 7).

Pro sestavení takového zařízení potřebujeme několik optických členů. V první řadě zdroj rovnoběžného svazku paprsků. V dnešní době se používají lasery. Dříve se umístil bodový zdroj světla do ohniskové vzdálenosti čočky, po průchodu čočkou byly paprsky rovnoběžné. Dále použijeme polopropustné zrcátko (dělič svazku) v úhlu 45° a dvě zrcátka ve stejné vzdálenosti od děliče. Jako poslední a nejdůležitější je zapotřebí člen, který zaznamená interferenci paprsků, použijeme tedy holografickou desku (v terminologii našeho simulátoru jde o mřížku).

Jak lze vidět na schématu, svazek paprsků (znázorněn žlutě na obrázku 7) projde děličem svazku a rozdělí se na dva. Jeden pokračuje směrem k prvnímu zrcátku (červený), druhý se odrazí a směřuje k druhému zrcátku (modrý). Poté, co se paprsky odrazí od zrcátek, směřují zpět k děliči a tam se zachovají stejně jako při prvním průchodu. Výsledkem celého procesu jsou dva svazky paprsků dopadajících na stínítko, a v závislosti na natočení a vzdálenosti zrcátek od děliče vzniká na stínítku interferenční vzor různých vlastností. Z nich je možné usuzovat na kvalitu zdroje světla, natočení zrcátek, apod. Jelikož jde o zcela základní optickou soustavu, je vhodné, aby šlo v simulátoru napodobit její chování.

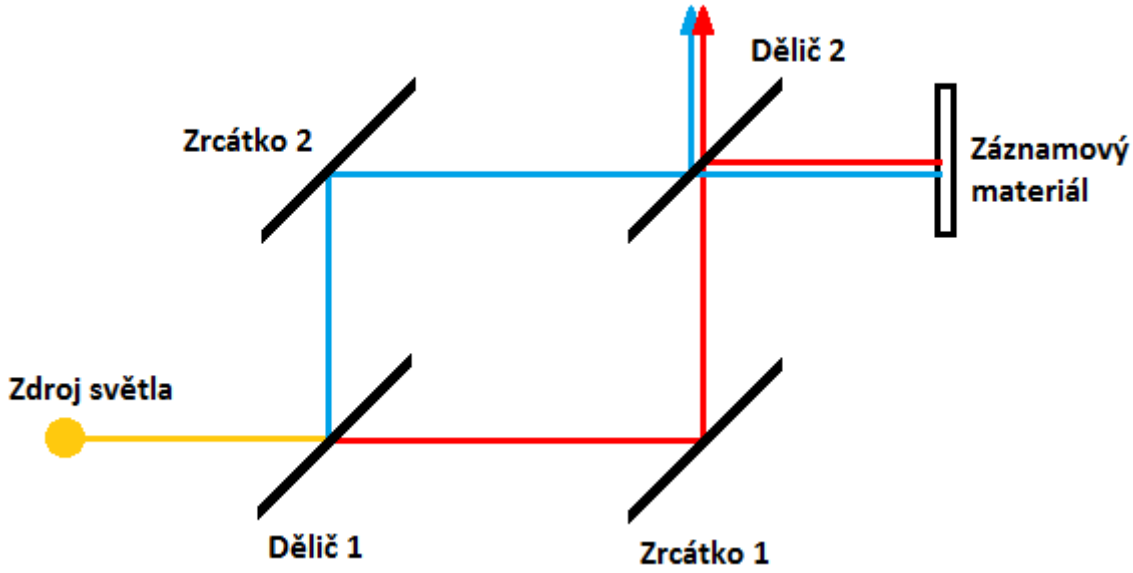


Obrázek 7: Michelsonův interferometr

3.1.3 Mach – Zehnderův interferometr

Toto zařízení zkonstruovali L. Mach a L. Zehnder roku 1891. Používá se zejména ke zkoumání transparentních předmětů. Tento interferometr je vhodný k srovnávacím měřením, v holografii se využívá pro zjištění odolnosti optické soustavy vůči otřesům.

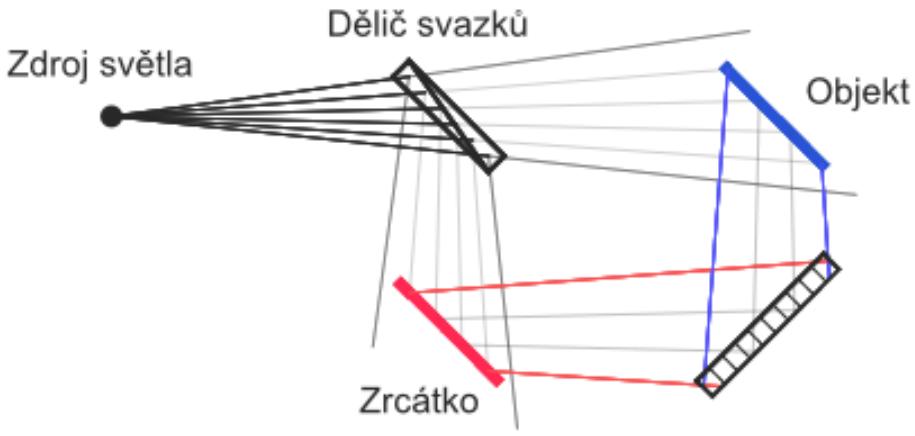
Na obrázku 8 můžeme vidět, že Mach – Zehnderův interferometr se skládá ze dvou zrcátek, dvou děličů svazku a světlocitlivého detektoru – ten lze opět nahradit „mřížkou“. Svazek se od zdroje šíří směrem k prvnímu děliči. Po jeho průchodu se rozdělí na referenční a objektový. Oba svazky se po odrazu od zrcátek dostanou ke druhému děliči. Na něm se svazky rozdělí a odrazí, takže vzniknou dvě dvojice svazků paprsků, které spolu interferují.



Obrázek 8: Mach - Zehnderův interferometr

3.1.4 Off-axis hologram

Metoda off-axis záznamu hologram je založena na tom, že objekt a záznamový materiál nejsou v jedné rovině (off-axis = mimo osu), ale paprsky zdroje jsou optickými členy přesměrovány jinam. Tato metoda se používá pro záznam neprůhledných objektů.

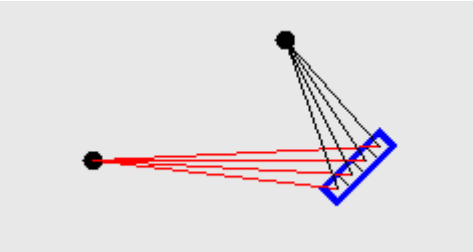


Obrázek 9: Záznam off-axis hologramu

Jak ukazuje schéma (Obrázek 9), svazek paprsků se rozdělí děličem na objektový a referenční. Objektový svazek dopadá na objekt a odráží se na světlocitlivou desku. Referenční svazek se odrazí od zrcátka a také dopadá na holografickou desku, zde spolu objektový a referenční svazek interferují a vytváří hologram.

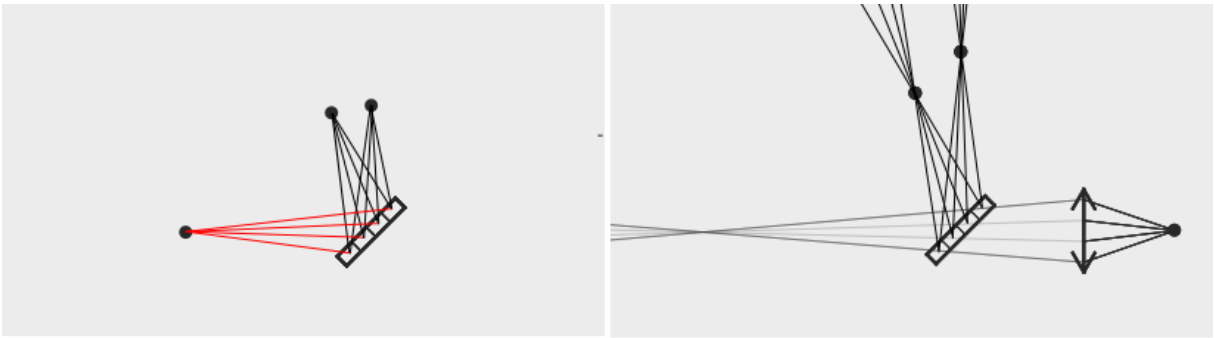
Off-axis hologram v simulátoru

Takovým způsobem s několika úpravami šlo zaznamenat hologram i v původní verzi. Místo paprsků odražených od děliče a zrcátka se použilo referenční světlo a objekt nahradily bodové zdroje světla (obrázek 10).



Obrázek 10: Zaznamenání off-axis hologramu ve staré verzi simulátoru

Při rekonstrukci takového hologramu lze použít původní soustavu pro referenční světlo. Pak vzniká imaginární (před mřížkou) obraz původního objektu. Nebo je možno použít paprsky „opačné“ k referenčním. Při použití těchto paprsků vzniká reálný obraz původního předmětu jako minus první difrakční maximum. Ovšem vzniká zde nepatrný problém, původní paprsky byly rozbíhavé a chceme-li opačně orientované, potřebujeme takový optický člen nebo soustavu členů, na jejímž výstupu budou paprsky sbíhavé. K těmto účelům poslouží spojná čočka a bodový zdroj světla.



Obrázek 11: Záznam (vlevo) a rekonstrukce (vpravo) off-axis hologramu pomocí spojné čočky

3.2 Potřebné změny

Jak ukazují předchozí kapitoly, simulátor potřebuje několik úprav a rozšíření, aby se mohl používat ke svým původním účelům. Rozhodli jsme se spolu s vedoucím tohoto projektu, že:

- Opravíme výpočty a vykreslení výstupních paprsků
- Upravíme výpočet pro záporná difrakční maxima
- Přizpůsobíme program pro práci s optickými členy, které v závislosti na vstupních paprscích generují paprsky výstupní
- Jako příklad optických členů implementujeme rovinné zrcátko, tenkou čočku a dělič svazku
- Zlepšíme zpracování jednotlivých tříd pro jednoduché začlenění nového optického prvku
- Napravíme zbylé menší nedodělky předchozí verze

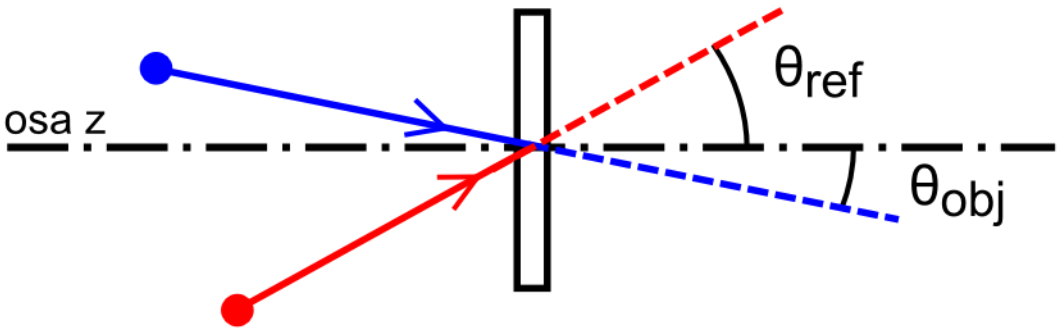
4 Nová verze simulátoru

Tato část je věnována popisu mé práce na holografickém simulátoru. Popíši zde, co jsem do programu přidal a co upravil.

4.1 Výpočet úhlů a souřadnic paprsků

4.1.1 Matematický výpočet výstupních úhlů

Jedním z nejviditelnějších neduhů původní verze programu byla chyba při výpočtu úhlů a následného vykreslování paprsků při záznamu a projekci difrakční mřížky. Ukažme si matematicky, jak vypadá vytvoření a rekonstrukce záznamu holografické desky. Pro záznam potřebujeme dva svazky paprsků stejné vlnové délky λ_z, které spolu budou interferovat na desce. Paprsky dopadají na mřížku pod nějakým úhlem (např. jako na obrázku 12). Úhly se měří od kladné poloosy „z“; tj. v ilustraci má θ_{ref} kladné znaménko, θ_{obj} záporné znaménko.



Obrázek 12: Paprsky vstupující do mřížky

Podle [2] tyto paprsky vytvoří v holografické desce interferenční vzor o frekvenci

$$f = \frac{\sin\theta_{obj} - \sin\theta_{ref}}{\lambda_z}$$

Takto jsme vlastně vytvořili jednoduchý hologram. Nyní, když ho osvítíme světlem vlnové délky λ_r o úhlu θ_{ill} vzniká několik paprsků (difrakčních maxim) očíslovaných celočíselným indexem m. Tyto paprsky opouští desku pod úhlem θ_{out} podle vztahu:

$$\sin\theta_{out} = m\,\lambda_r\,f + \sin\theta_{ill}$$

Dosazením za f získáme jediný vztah pro výpočet výstupních úhlů:

$$\sin\theta_{out} = m\,\frac{\lambda_r}{\lambda_z}\left(\sin\theta_{obj} - \sin\theta_{ref}\right) + \sin\theta_{ill}$$

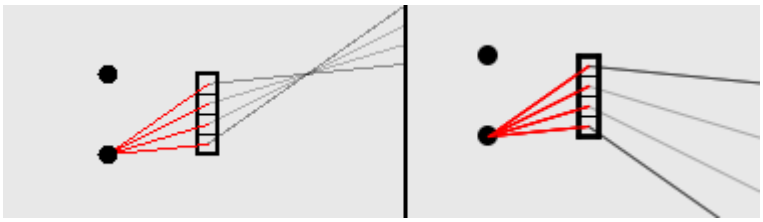
4.1.2 Zobrazení paprsků modelové situace

Uvažujme nyní jednoduchou modelovou situaci, kdy zaznamenáme hologram pomocí bodového zdroje světla a referenčního světla v určitých úhlech. Dále chceme tento hologram rekonstruovat za použití světla o stejné vlnové délce a stejném úhlu jako bylo světlo referenční. Pak se nám rovnice pro výstupní úhel pro první difrakční maximum ($m = 1$) zkrátí na tvar:

$$\sin\theta_{out} = m\,\frac{\lambda_z}{\lambda_z}\left(\sin\theta_{obj} - \sin\theta_{ref}\right) + \sin\theta_{ref}$$

$$\theta_{out} = \theta_{obj}$$

Vidíme, že výstupní paprsek bude opouštět mřížku pod stejným úhlem, jako do ní vstoupil paprsek objektový (od zdroje světla představující objekt). Je tedy pokračováním původního objektového paprsku.



Obrázek 13: srovnání vykreslení paprsků ve staré (vlevo) a nové verzi

4.1.3 Zobrazování záporných difrakčních maxim

V souvislosti s výpočtem výstupního úhlu docházelo k dalšímu neduhu při zobrazování záporných difrakčních maxim. Ta byla vykreslována jako osově převrácený obraz jejich kladných protějšků, což vedlo jednak k nesprávnému vykreslování, ale také k vykreslování záporných difrakčních maxim, která se vykreslovat vůbec neměla. Zkontrolujme na příkladu: pro objektový paprsek zvolme úhel 30°, pro referenční -30°, rekonstrukční světlo je stejných parametrů jako referenční. Víme, že pro 1. difrakční maximum platí:

$$\theta_{out} = \theta_{obj} = 30°$$

První difrakční maximum existuje. V původní verzi se ovšem vykresluje jak první, tak i minus první maximum. Zkusme si ale spočítat, jaký výstupní úhel má paprsek -1. difrakčního maxima.

$$\sin\theta_{výstupi} = -1\,(sin(30°) - sin(-30°)) + sin(-30°) = -\frac{3}{2}$$

Zjistili jsme, že takový paprsek nemůže být vykreslen, protože funkce sinus pro hodnotu -1,5 neexistuje. Musel jsem tedy změnit výpočetní proceduru tak, aby se počítala jak kladná tak záporná maxima, a tomu přizpůsobit i výpočet souřadnic jednotlivých paprsků.

4.2 Nové optické členy

Po změnách zmíněných v předchozí podkapitole simulátor fungoval, jak měl a bylo možno vytvářet soustavy pro zaznamenání hologramu za pomoci světelných zdrojů a difrakční mřížky. V úvodu a teoretické části jsou popsány sestavy, které si s takovým výběrem prvků nevystačí. Rozhodli jsme se, že do simulátoru přidáme zrcátko, dělič svazku a čočku.

4.2.1 Přidání třídy

Začal jsem s přidáním zrcátka. Ovšem takový zásah do struktury programu nebyl tak jednoduchý, jak se na počátku mohlo zdát. Celá aplikace byla v jejím původním stavu přizpůsobená pouze pro práci s mřížkou a světlem. Zrcátko jsem tedy vytvořil zkopírováním struktury a obslužných tříd použitých pro mřížku. Co se týče spolupráce nově přidaného členu a stolů (pláten), zde už vše prošlo hladce, poněvadž už v původní verzi byla použita abstraktní třída **Tvar**. Tak tedy **Zrcatko** a obslužné třídy představovaly správnou funkčnost zrcátka. Následně jsem hledal atributy a metody, které má společné s mřížkou a se světelným zdrojem. **Zrcatko** a **Mrizka** mají několik společných vlastností, jako např. počet segmentů, úhel natočení, aj. a také spoustu metod s těmito atributy spojených.

Bylo by vhodné sjednocení do nějaké supertřídy, abych při přidávání dalších optických členů nemusel vše opisovat znovu. **Tvar** jsem ovšem nemohl rozšířit o tyto vlastnosti a chování, protože tato abstraktní třída slučuje prvky, které mohou být vykresleny na plátně. Do této kategorie patří i světla a náhledy, a ty nedefinují úhly, segmenty ani podobné věci jako optické členy.

Vytvoření abstraktní třídy OptickyClen

Pro zjednodušení práce s dalšími přidanými prvky jsem tedy definoval novou abstraktní třídu **OptickyClen**, která představuje všechny nově přidané optické členy a už definovanou mřížku. Optické členy jsou ale stále tvary, **OptickyClen** musí dědit od **Tvaru** (UML viz Obrázek 19). Nyní už nic nebrání vložení i ostatních členů, ty jsou potomky třídy **OptickyClen**, tak vlastně i **Tvaru.**

4.2.2 Změny v grafickém uživatelském rozhraní

Sjednocení do třídy TvarGUI

Nové optické členy s sebou přinášely i nové požadavky na grafické uživatelské rozhraní (GUI). Bylo to podobné jako při definování **OptickyClen** a **Tvar.** Také zde bylo spoustu komponent stejných pro všechny a nějaké specializované pro daný člen. Logicky z toho vycházelo, že vytvořím abstraktní třídu, která sloučí společné atributy a metody. Třída **TvarGUI** obsahuje metody pro vytvoření co nejvíce tlačítek, panelů, seznamů a jiných komponent. Pro vytvoření jednotlivých uživatelských rozhraní stačí pouze zavolat metody z rodičovské třídy a uspořádat je podle potřeby. V případech, kdy tyto atributy a metody nestačí, dodělají se pouze specializované ovládací prvky.

Listenery

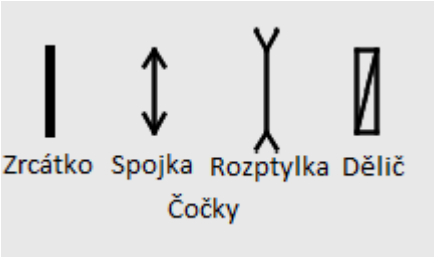
K obsluze událostí typu zmáčknutí tlačítka, zaškrtnutí položky atd., se v Javě používají Listenery (posluchači). Jde o speciální druh třídy, ve které se definují procedury spouštěné při konkrétní akci. V původní verzi simulátoru byly posluchači pro každý kousek grafického rozhraní odděleny, mřížka měla své vlastní, světlo také atd. Díky změnám a zavedením **TvarGUI** se jednotlivé části rozhraní vytvářejí na jednom místě a není předem určeno, jestli budou použity v rozhraní pro mřížku, zrcátko nebo světlo. Zároveň díky zavedené abstrakci (**OptickyClen**, **Tvar**, **TvarGUI**) lze téměř všechny Listenery předělat do univerzální podoby. Založil jsem pro takové účely nový balík **Listenery** ve složce GUI. Tento balík obsahuje univerzální posluchače, které lze použít pro více ostatních tříd. Jejich duplicitní verze u každého optického členu jsem vymazal. Avšak GUI jednotlivých optických členů obsahují i různé odlišené prvky, sloužící jedině pro účely jednoho konkrétního prvku. Ty se nachází ve složkách jednotlivých členů (ukázka uspořádání adresářové struktury viz obrázek 20).

4.2.3 Grafika

Vzhled

Stejně jako uživatelské rozhraní bylo třeba vytvořit grafiku pro nové členy. Určit jejich vzhled, jak budou vypadat vstupní/výstupní paprsky atp. Třídy pro grafiku patří do balíku **Grafika.**

Nejdříve jsem definoval vzhled jednotlivých členů, aby šly na plátně rozeznat.



Obrázek 14: Zobrazení jednotlivých optických členů

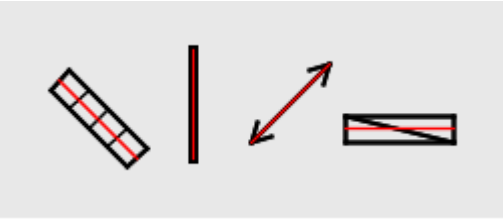
Vstupní paprsky

Další přišly na řadu vstupní paprsky. Všechny optické členy mají určitý počet segmentů, stejně jako mřížka. Paprsky tak ze světelných zdrojů směřují do středu každého segmentu na členu. Vykreslování paprsků je u každého členu stejné, zavedl jsem opět abstraktní třídu. **Grafika** slouží jako místo koncentrace všech společných metod a atributů a jako předpis pro grafické třídy jednotlivých prvků.

Výpočet výstupů

V této fázi jsem mohl umístit optický člen na stůl a nechat osvítit světlem. Scházela nejdůležitější část: vypočítat, jak se paprsky změní při dopadu na člen – jaké budou výstupní paprsky. Každý prvek se chová jinak, a tak v každém prvku musím definovat jeho chování, to předepisuje abstraktní metoda **vypocitejVystupy** třídy **Grafika**. Pro ukládání vypočítaných výstupů jsou určeny atributy v rodičovské třídě optických členů. Uchovávají se nejen paprsky, ale také jejich průsečíky, a je-li třeba, tak i prodloužení paprsků. Prodloužení nejsou skutečné paprsky, vykreslují se např. pro představu, odkud paprsky vychází, jsou-li rozbíhavé při výstupu z optického členu. Metody pro vykreslení výstupních paprsků, průsečíků i prodloužení se nachází, stejně jako kreslení vstupních, ve třídě **Grafika.** Mřížka fungovala tak, jak má ještě před zavedením rodičovské třídy pro grafiku a nebylo nutné její algoritmy měnit.

Takto jsem docílil pouze vypočítání výstupů, když vstupem je světlo, ale cílem bylo, aby člen mohl být osvícen jakýmkoliv jiným. K interakci mezi členem a jeho osvětlujícím objektem slouží seznam výstupních paprsků osvětlujícího prvku. Při zpracování se prochází každý paprsek a zkoumá se, zda protíná osu (úsečku procházející středem viz obrázek 15) osvíceného optického členu. Jestliže ano, zařadí se tento paprsek do nově přidaného seznamu – **seznamVstupu** třídy **Grafika.** Při výpočtech tedy členy používají tento seznam a výstup se vypočítává pro každý paprsek zvlášť.



Obrázek 15: Ukázka členů s červeně vyznačenými osami

Výpočty pro mřížku

Výstupy v difrakční mřížce se dříve zjišťovaly pouze pro zdroj světla nebo jinou mřížku. Oba způsoby navíc byly odděleny. Takže zaznamenání a rekonstrukce hologramu sice fungovalo korektně, ale tento způsob již nebyl dále použitelný. Přesunul jsem algoritmus na výpočet výstupních úhlů do **MrizkaGrafika** – **vypoctiVystupy** tak, jak je to u ostatních optických členů a přizpůsobil ho, aby počítal s každým paprskem. Ale i přes to jsou třeba pro mřížku speciální seznamy a postupy, proto její grafická třída překrývá téměř všechny funkce.

Při záznamu difrakční mřížky interferují pouze paprsky stejné vlnové délky. V původní verzi nebylo obtížné zjistit vlnovou délku paprsků dopadajících na mřížku, poněvadž se pracovalo pouze se světly. Struktura **Svetlo** obsahuje metody udávající vlnové délky. To už však po přidání nových optických členů neplatí. Zjištění, z jakého zdroje se světlo přes soustavu optických členů šíří, by bylo neefektivní. Založil jsem tedy strukturu **Paprsek** podle návrhového vzoru přepravka. Přepravky slouží k úschově a přenášení informací rozličných datových typů. **Paprsek** přechovává vlnovou délku a čáru představující paprsek na plátně. Tak tedy lze určit u každého paprsku, jaké je vlnové délky.

Pořadí výpočtů a vykreslování jednotlivých členů

Jestliže jsou ke zjištění výstupů potřeba seznamy výstupních paprsků osvětlujících členů, musí se výpočty provádět v určitém pořadí. Nejlépe tak, jak se světlo šíří prostorem – postupně od zdrojů. Při přidávání optických členů a zdrojů světla a volením osvětlujících prvků se vlastně utváří orientovaný graf(y). Hrana v takovém grafu znamená, že jeden člen osvětluje druhý. Jednotlivé optické členy obsahují seznam osvětlujících prvků, to jsou vlastně seznamy sousedů v pomyslném grafu. Znám místo, od kterého chci začít vypočítávat výstupy a seznamy sousedů vrcholů v grafu, můžu tedy využít nějaký z grafových algoritmů, abych prošel graf, jak jsem si navrhl. Přímo se nabízí algoritmus průchodu grafu do šířky od zadaného vrcholu. *Breadth-first search (bfs)* používá mechanismus fronty, do ní uloží sousedy aktuálně zpracovávaného vrcholu a zároveň tento vrchol zapíše do seznamu navštívených. Po zpracování jednoho bodu vezme další z fronty a postup se opakuje. Seznam navštívených prvků představuje průchod grafem a tedy i pořadí, jak se budou počítat výstupy.

Vznik cyklů

Při popisovaném zpracování ale může dojít k nepříjemnému jevu, který by celý postup mohl znehodnotit. Mohla by nastat například následující situace: uživatel chce mít na stole dvě zrcátka a jeden zdroj světla. Nechá osvítit jedno zrcátko světlem a druhé zrcadlo tím prvním. Paprsky se odrazí od zrcátka jedna k zrcátku dva a dále. Kdyby ale uživatel zaškrtl u zrcátka jedna možnost osvětlení jak světlem, tak zrcátkem druhým, nastala by situace, kdy se paprsky do nekonečna odráží mezi zrcátky. Takový scénář by mohl vést ke dvěma závěrům. První varianta, méně nebezpečná, by skončila nevykreslením odrazu od jednoho nebo druhého zrcátka, a to v závislosti na průchodu *bfs* algoritmem. Nebo by se mohlo stát, že se paprsky spočítají a vykreslí správně, to však znamená uložení, výpočet a vykreslení obrovského množství paprsků. Toto je nebezpečné z hlediska stability appletu, a je nutné to ošetřit. Takové vzájemné osvětlení by se v grafu projevilo jako cyklus. Pro zjištění cyklu v grafu jsem použil metodu prohledávání grafu do hloubky. Dá se to však udělat i použitím průchodu do šířky a jinými způsoby.

Prohledávání do hloubky *depth-first search* (*dfs*) je velice podobné dříve popisovanému algoritmu *Breadth-first search*. Místo fronty se zde používá zásobník. Při zpracování vrcholu se přesuneme vždy na prvního nenavštíveného souseda a uložíme zpracovávaný vrchol do zásobníku. Není-li se kam dál z vrcholu posunout, uložíme ho do výstupního seznamu a vezmeme poslední ze zásobníku. Jde vlastně o to, posunout se co nejdále po dostupných sousedech, a jestliže to dál nejde, tak se vrátit na předchozí. Cyklus pomocí *dfs* zjistíme tak, že při průchodu do hloubky narazíme na vrchol, který jsme už navštívili.

Předcházení cyklům

Zjištění cyklu není nadále problém. Problémem ovšem je, že při vzniku cyklu a vykreslení na plátno už je pozdě. Tomu se musí předcházet. Takže program nesmí dovolit uživateli zvolit si optický člen, který by v grafu vytvořil cyklus. To jsem zajistil při vytváření GUI pro označený prvek. Když uživatel chce změnit parametry nějakého optického členu, klikne na něj myší a zobrazí se grafické rozhraní, které v jedné části obsahuje všechny prvky, které tento člen můžou osvětlovat. Takže při vytváření tohoto seznamu se „na zkoušku“ přidá každý potenciálně osvětlující prvek do seznamu určeného členu a projde se pomocí dfs, jestli v takové konfiguraci neobsahuje cyklus. Když ano, znemožní se uživateli tuto volbu zaškrtnout.

Zamezením cyklů se na druhou stranu znemožní snadná konstrukce některých důležitých optických soustav, např. Fabry-Perotova interferometru. Na druhou stranu jsou výhody zamezení cyklů v současnosti větší než nevýhody. Korektní práce s cykly tedy zůstává námětem pro rozšiřování simulátoru.

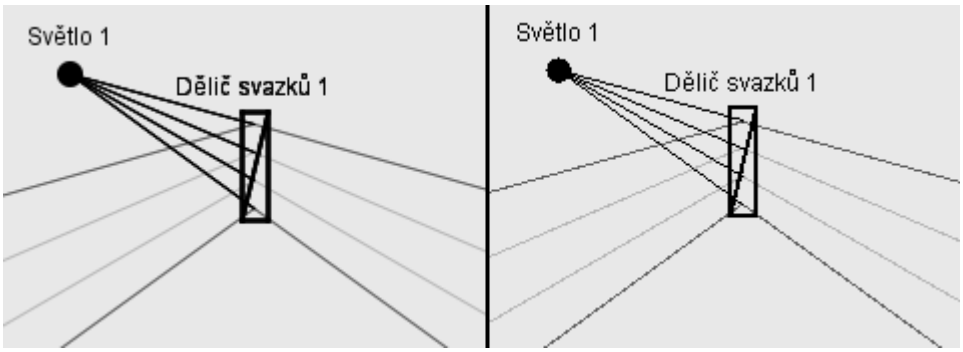
4.3 Další změny v grafickém rozhraní

4.3.1 Drobné úpravy dialogů nastavení

Po dříve popsaných změnách by měla být hlavní funkčnost programu hotová. Během práce jsem však zjistil několik menších problémů v uživatelském rozhraní. Nejvíce byla poznat absence potvrzovacích tlačítek v mnoha oknech nastavení a všechna se musela ukončovat pouze křížkem. To sice nemělo žádný vliv na funkčnost, ale bylo matoucí, že po nastavení hodnot člověk musel okno vypnout, místo aby stiskl uložit, popř. zrušit.

4.3.2 Vyhlazování hran

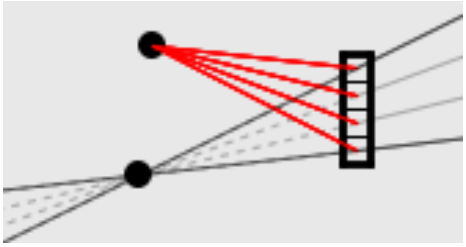
Další vylepšení se týkalo celkového vykreslování okna. Přidal jsem do nastavení možnost zapnout/vypnout vyhlazování hran. Jde sice o minimalistický zásah do programu, ale grafický výstup díky tomu vypadá o mnoho lépe.



Obrázek 16: Rozdíl mezi vykreslování se zapnutým (vlevo) a vypnutým (vpravo) vyhlazováním hran

4.3.3 Paprsky prodloužení

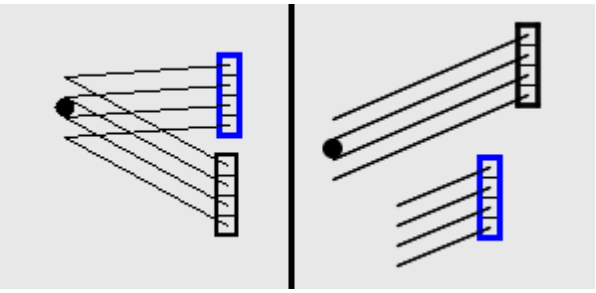
Při rekonstrukci záznamu určitých parametrů na difrakční mřížce se paprsky po průchodu mřížkou rozchází. Sbíhají se jakoby před hologramem, vytváří tedy virtuální obraz. V simulátoru se ovšem nemůžeme podívat z jiného pohledu než shora a virtuální obraz bychom tedy neměli možnost vidět, pouze si můžeme domýšlet, kde by se mohl nacházet. K větší orientaci a vyřešení tohoto problému v původní verzi sloužilo prodloužení paprsků směrem před mřížku, jestliže se protínaly. Taková možnost je užitečná, ale ne vždy si jí přejeme zobrazovat. Do dialogových nabídek jsem přidal možnost tyto paprsky vypnout nebo zapnout. To však nebylo úplně jednoduché, protože stará verze appletu tyto paprsky společně s výstupními vykreslovala dohromady, vždy jako jednu čáru. Zavedl jsem nový seznam do třídy **OptickyClen**, do kterého se zapisují paprsky prodloužení. To mi umožnilo další užitečnou věc, mohl jsem graficky odlišit paprsky prodloužení a výstupů.



Obrázek 17: Nová grafická podoba paprsků prodloužení

4.3.4 Světelné zdroje a jejich chyby

Světelné zdroje se také nevyhnuly úplné bezchybnosti. V nastavení světla se nachází volba mezi bodovým a kolimovaným (rovinná vlna) zdrojem světla. Z kolimovaného zdroje dopadají všechny paprsky pod stejným úhlem – jsou rovnoběžné. V původní verzi to platilo pouze, když světlo dopadalo na jednu mřížku, jinak ne. Přidal jsem do uživatelského rozhraní pro světlo výběr „směrodatného“ prvku. Pak na všechny ostatní členy dopadá světlo pod stejným úhlem jako na tento referenční prvek.



Obrázek 18: Srovnání rovinné vlny ve staré (vlevo) a v nové verzi

4.3.5 Export do vektorové grafiky

Menu původní verze obsahovalo možnost exportu do vektorové grafiky. Tato položka však neměla žádnou obsluhu. Za použití knihovny **org.apache.batik.svggen** jsem funkci exportu oživil. Tento balík obsahuje třídu na základě **Graphics2D**, která má téměř všechny její funkce a navíc zvládá vykreslované objekty ukládat do formátu .svg.

4.4 Přidání nového členu

4.4.1 Úvod

Ukažme si, jaký přínos měly popisované změny, zejména zavedení abstraktních tříd a unifikace listenerů. Protože se v holografii používá nemalé množství různých optických členů, určitě bude žádoucí je přidat do aplikace a zvýšit tak její funkční rozsah. Příkladem takových prvků můžou být:

- zakřivená zrcadla (sférická, parabolická, off-axis parabolická)
- reálné (tlusté) čočky různých parametrů, asférické čočky
- optické hranoly
- difuzéry, difuzní odrazné plochy

Chceme-li do aplikace zařadit nový optický člen, budeme potřebovat třídu pro daný člen, jeho GUI a grafiku.

4.4.2 Třída pro nový člen

Vytvoříme třídu v balíku **Tvary** s příslušným názvem, ta bude dědit od supertřídy **OptickyClen**. Musíme přepsat všechny abstraktní metody, zatím nevíme, jak je definovat, necháme tedy jejich těla prázdná a vrátíme se k nim později. Zbývá nám vytvořit konstruktor. Pokud nepotřebujeme nějakou speciální vlastnost, bude mít stejné vstupní argumenty jako v rodiči. Použijeme konstruktor z rodiče a předáme mu parametry voláním **super.** V konstruktoru definujeme, jak se budou tvořit názvy nového optického členu. U stávajících členů se používá koncepce „název členu“ „číslo instance“ např. Zrcátko 1. Také přidáme *observery* pro pozice x a y. Po případě dodefinujeme speciality daného členu.

4.4.3 GUI přidaného členu

Základ třídy pro nový člen máme vytvořený. Teď uděláme třídu pro GUI. Připravíme si nejdříve místo pro ni. Ve složce GUI založíme novou podsložku (balík) s názvem členu. Do nového balíku přidáme třídu pro grafické uživatelské rozhraní s příslušným názvem. Opět dědíme abstraktní třídu, tentokrát **TvarGUI** obsahující metody a atributy tvořící grafické rozhraní. Důležité je hned v konstruktoru použít funkce **inicializaceJazyku**, **inicializacePrvku**, které zajistí funkčnost všech prvků. Podstatná je také abstraktní metoda **vytvorKomponenty**, ve které by se měl seskládat celý panel grafického uživatelského rozhraní. K tomuto účelu se využívají zmíněné metody z **TvarGUI**, ty ovšem lze rozšířit, je-li třeba.

4.4.4 Grafika nového prvku

Zbývá poslední součást, a tou je grafická třída. V ní definujeme, jak bude prvek vypadat na plátně a jak budou vypadat výstupní paprsky v závislosti na vstupních. Grafické třídy patří do balíku **Grafika** a dědí od stejnojmenné třídy. Musíme překrýt metody **vykresliRamecek** a **vypocitejVystupy.** V první jmenované definujeme vzhled členu a v druhé výpočet výstupních paprsků. Grafika již obsahuje metody pro vykreslení vstupních paprsků pocházejících od světel. Také se v ní nachází atribut **seznamVstupu** a metody, které s ním pracují, to můžeme využít v nové třídě.

4.4.5 Závěrečné úpravy

Nyní zbývá doplnit těla metod v třídě pro nový optický člen a také udělat v kontextovém menu položku pro jeho přidání. Celý tento postup není nejjednodušší a snadno se v něm něco vynechá nebo se udělá chyba, proto je dobré vždy nahlídnout do již fungujících členů, ne-li přímo převzít některé postupy z těchto tříd.

5 Popis implementace

Celá aplikace se skládá z velkého množství tříd a balíků. V této kapitole popíši, jak spolu jednotlivé části spolupracují a co která třída má na starosti atp. Původní verze byla i svou strukturou zaměřena pouze na dva prvky. Každý se nacházel ve svém balíku, obsahoval třídy, které jej tvořily a obsluhovaly a vše bylo striktně odděleno od ostatních částí. Zaměřil jsem na sjednocení stejných částí a vymazání duplicity za pomoci rozhraní a abstraktních tříd. To navíc pomůže při možném implementování nových členů. Zároveň jsem se snažil, co nejvíce zachovat logickou adresářovou strukturu původního programu.

5.1.1 AppletHologram

Vstupním bodem do aplikace je třída **AppletHologram**, která při spuštění nastaví a uloží všechny důležité parametry, jako je šířka a výška appletu, atd., a vytvoří instance tříd, důležitých pro celkový běh programu jako jsou **Jazyk, MeziPlatno** a **Stoly** (viz níže). Následně po načtení a instanciování všeho důležitého se vytvoří hlavní okno appletu daných rozměrů.

5.2 Data a struktury

5.2.1 Jazyk

O jazyk uživatelského rozhraní se stará třída **Jazyk**. Každý nápis v programu je uložen v češtině i angličtině. Tato třída je proto musí vždy korektně načíst podle zvoleného jazyka. Pro každý jazyk je definovaný textový soubor s koncovkou *.properties*. V těchto souborech se nachází univerzální klíč k danému popisku a za symbolem = překlad v daném jazyce, který se zobrazí v uživatelském rozhraní. **Jazyk** implementuje rozhraní **Observable**. To znamená, že instance této třídy můžou být sledovány tzv. **Observery**. Toto spojení zajišťuje, že se popisky přizpůsobí, při změně jazyka. V balíku **Jazyk** se kromě této třídy nachází všemožné komponenty – tlačítka, panely, checkboxy atp., přizpůsobené pro práci s jazykem.

5.2.2 Meziplatno

Dalším modulem načteným při vytváření appletu je **MeziPlatno**. Jak název napovídá, meziplátno stojí mezi plátnem, které skutečně vykresluje dané komponenty, a uživatelem, který zadává různé akce. **MeziPlatno** zajišťuje správné vykreslení hlavního okna a generování grafického uživatelského rozhraní. K tomu patří především vodorovné a svislé rozdělování obrazovky. Pro vykreslení celého okna se volají instance tříd **Platno** a následně **PlatnoGrafika**.

5.2.3 Platno

Platno zastupuje jednotlivé části obrazovky a jejich nastavení. Každé plátno může zobrazovat jiný stůl s jiným nastavením s odlišným přiblížením, polohou nebo například zobrazením videa na pozadí. K tomu **Platno** schraňuje odkazy tomu určených objektů. Jedním z nich je **Nahled**, který graficky zobrazuje polohu tohoto plátna na daném stolu v ostatních oknech. Dalším z těchto objektů je **Stul.**

5.2.4 Stul

Stul v programu nahrazuje reálný stůl, na kterém sestavujeme různé optické soustavy a zaznamenáváme hologramy. Aby se stoly od sebe daly odlišit, každý identifikuje jeho unikátní číslo a jméno. Struktura **Stul** obsahuje seznam optických členů, které se na stole nachází. Dále pak sadu metod pro přidání a mazání všech druhů součástek. Pro vytvoření optického členu je nutné vědět, jaké má mít takový člen výchozí parametry. K těmto účelům slouží třída **Konfigurace**. Jde vlastně o sbírku konstant pro všemožné účely.

5.2.5 Tvar a IVykreslitelny

Protože je nutné slučovat všechny optické komponenty do různých seznamů, pro takové účely se zde nachází abstraktní třída **Tvar**. Ta slučuje vlastnosti společné pro všechny prvky, které se vykreslují na stole. Mezi takové atributy patří např. pozice na stole, název, rozměry, seznam duplikátů atp. Při vytváření nového objektu se už tyto společné vlastnosti nemusí definovat znova. Každý tvar, který se má vykreslovat na stole, musí dědit tuto abstraktní třídu. Prvky seznamů se typují na třídu **Tvar**. Při vytváření seznamů objektů se využívá vlastnosti polymorfismu. K rozlišení, jaký člen se má vykreslit, dochází až během zpracovávání seznamu. Kvůli grafické stránce **Tvar** navíc implementuje rozhraní **IVykreslitelny**, při jehož implementaci je nutné překrýt metody **isVybrany, vykresli** a další. Tento předpis zajistí, že se všechny tvary budou umět vykreslit, označit atd.

5.2.6 OptickyClen

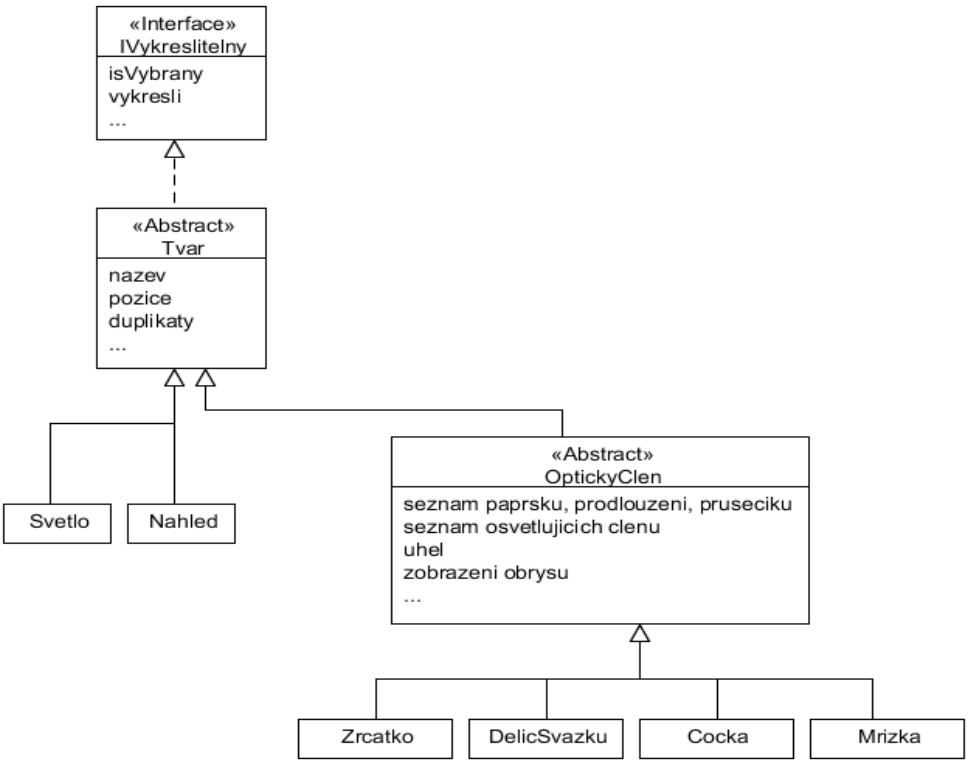
OptickyClen je užší specializací (potomkem) **Tvaru**. Stejně jako její rodič i tato třída seskupuje vlastnosti určitého souboru objektů. Podle názvu jde o optické členy. Ty potřebují na rozdíl od ostatních tvarů určit seznam objektů, kterými jsou osvětlovány, úhel natočení, počet segmentů, seznamy výstupních paprsků, průsečíků a prodloužení a také různé nastavení, jako např. jestli se u toho členu mají zobrazovat obrysy paprsků a jakou mají tyto paprsky barvu atd. Příkladem optických členů jsou dříve zmiňované difrakční mřížky, děliče svazku, čočky, zrcátka…

5.2.7 StatickeMetody

V různých částech aplikace se nachází metody, využívané všemi třídami, a bylo by neefektivní je psát na několik míst. Pro tyto účely je připraven soubor statických metod s názvem **StatickeMetody**. Mezi takové metody spadá: výpočet a různá práce s úhly, rotace bodu okolo zadaného středu, ořezávání čar, aby nepřesahovaly zobrazovanou oblast, zjištění průsečíku dvou čar a metody pro průchod grafu do hloubky a do šířky.

5.2.8 StrukturaZaznamu a Paprsek

V programu se nachází ještě velké množství struktur plnících menší úlohy a je zbytečné je zde rozepisovat všechny. Důležité je zmínit dvě z nich: **StrukturaZaznamu** a **Paprsek**. Strukturu záznamu si můžeme představit jako část vzoru interferenčních maxim a minim zaznamenaných v mřížce. Ukládá se do ní proto frekvence vzoru a další parametry. **Paprsek** je třída podle návrhového vzoru přepravka, pro přenos informace o daném paprsku na plátně a jeho vlnové délce.



Obrázek 19: Diagram tříd

5.3 Grafika

5.3.1 PlatnoGrafika

Vykreslování komponent zajišťují třídy v balíku **Grafika**. Veškeré grafické prostředky poskytují třídy **Graphics** a **Graphics2D** a balíky grafických prostředků **java.awt** a **java.swing**. Jednou z nejdůležitějších tříd v balíku **Grafika** je **PlatnoGrafika,** ta se stará o vykreslení daného plátna (stolu) a všeho, co na něm má být vykresleno. Plátno prostřednictvím **MeziPlatna** a svých navolených atributů získá seznam tvarů, které má vykreslit. Prochází tedy seznam a volá metodu **vykresli** všech vykreslovaných objektů. Ty už se odkazují na grafiku pro ten konkrétní člen. Protože vykreslování optických členů a jejich vstupů a výstupů nejsou jednoduché, jsou zde třídy **MrizkaGrafika**, **ZrcatkoGrafika** atd.

5.3.2 Grafika

Stejně jako u tvarů a optických členů schraňuje všechny společné vlastnosti abstraktní třída **Grafika,** aby se vyhnulo duplicitě kódu. Důležitá z pohledu výpočtu chování jednotlivých prvků je metoda **vypoctiVystupy**. Každá grafická třída ji tedy musí přepsat, vypočítávají se tak výstupní paprsky optických členů. Poněvadž se výstupní parsky vypočítávají na základě paprsků, které do prvku vstupují, musí být vypočítány výstupy všech prvků, které tento prvek osvětlují. Toto zajišťuje vykreslování členů tak, že nejdříve se vykreslí komponenty v okolí světel a pak jejich sousedi. Takový postup je vlastně procházení grafu do šířky od světelných zdrojů (viz 4.2.3).

5.4 Grafické uživatelské rozhraní a listenery

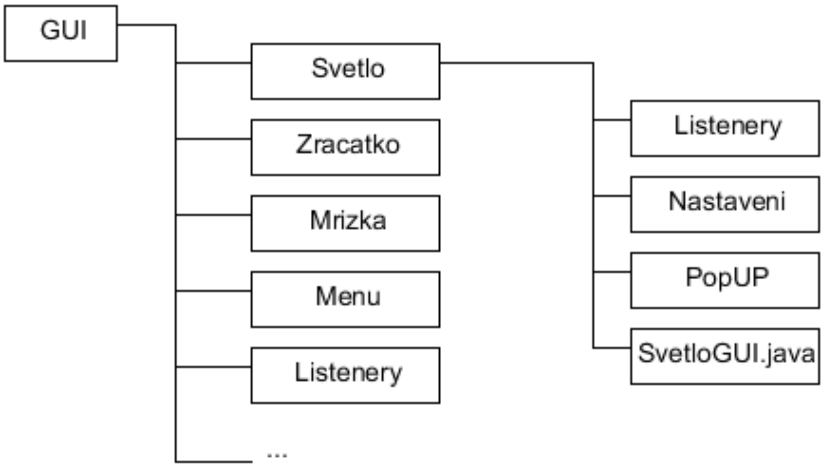
5.4.1 TvarGUI

Každou součástku, přidanou na stůl, musí být možné nějakým způsobem upravovat a zacházet s ní. K tomu slouží grafické uživatelské rozhraní. Každý člen ovšem má jiné atributy, a tak potřebuje i jiné rozhraní. Ale stejně jako **Tvar** a **OptickyClen** sdružuje společné vlastnosti, stejně tak existuje abstraktní třída **TvarGUI** jako základ pro grafické rozhraní všech částí. V **TvarGUI** se nachází funkce pro vytvoření základních prvků, panelů aj. Při definování GUI pro jednotlivé prvky se tyto části umisťují na obrazovku a je-li třeba, přidá se např. specifické tlačítko pro daný člen.

5.4.2 Adresářová struktura

V adresářové struktuře appletu se nachází složka **GUI**, ta obsahuje složky pro každý prvek, který potřebuje nebo nějakým způsobem zasahuje do grafického uživatelského rozhraní.

Na obrázku 20 lze vidět uspořádání adresáře pro světelné zdroje. Struktura je pro všechny komponenty podobná. Složka **GUI – Listenery** obsahuje posluchače společné pro více prvků, jako např. obsluha pro výběr osvětlujících prvků u optických členů, nastavení pozice tvarů, atp. Ve složce **Svetlo** se nachází další složka **Listenery,** ta obsahuje obslužné třídy speciálně pro **Svetlo**, které se nikde jinde nepoužijí. V **Nastaveni** je dialog nastavení spolu se vším, co je pro jeho běh třeba. Pro vyskakovací nabídku a její obsluhu je vytvořena složka **PopUP.**

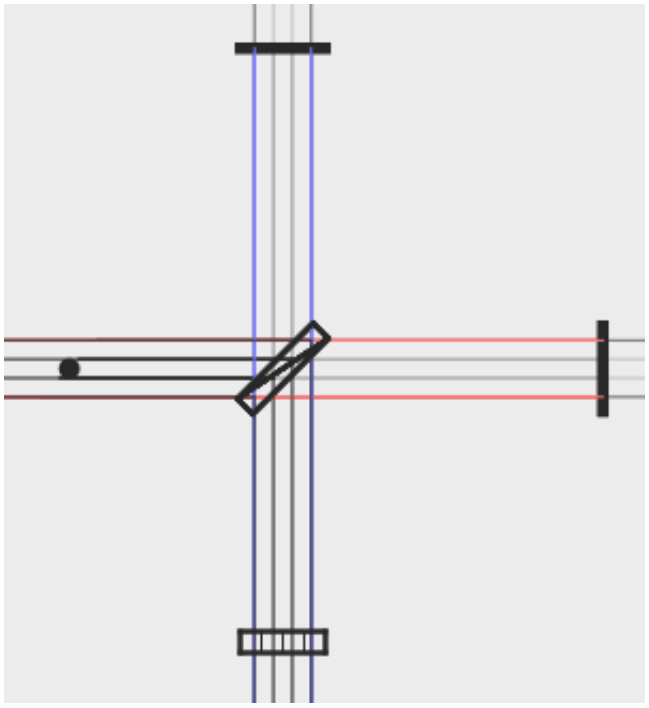


Obrázek 20: Ukázka adresářové struktury

6 Ověření funkčnosti nové verze

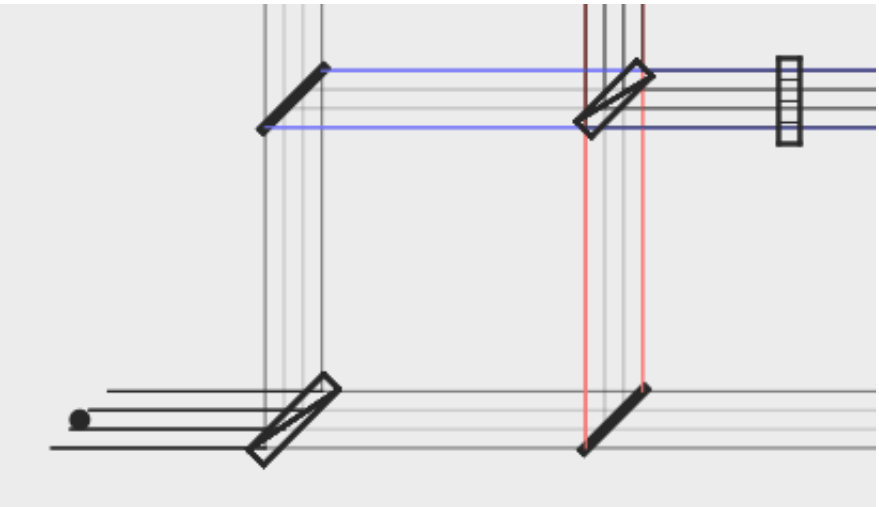
Na začátku jsme formulovali požadavky na novou verzi. Ukázali jsme několik schémat, které bychom měli být schopni simulovat.

Prvním z nich byl Michelsonův interferometr. Jak je vidět na obrázku 21, povedlo se ho v programu nasimulovat. Zároveň je ale vidět několik nehezkých záležitostí nové verze. Paprsky, pocházející z optických členů, se po dopadu na jiný člen „nezastaví“, ale prochází jím skrz. Dalším závažnějším námětem je již diskutovaný vznik cyklů. Vzhledem k tomu, že nejsou cykly dovoleny, není možné, aby svazky paprsků po průchodu děličem a odražením od zrcátek opět interagovaly s děličem uprostřed. Při simulaci jsem tento nedostatek řešil přidáním totožného děliče na stejné místo. Tento druhý dělič zpracovává paprsky po odrazech od zrcátek.



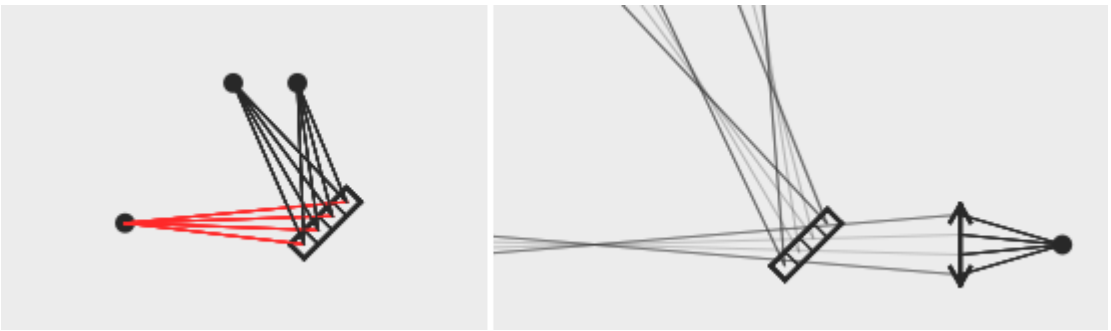
Obrázek 21: Simulace Michelsonova interferometru

Simulování Mach-Zehnderova interferometru bylo bezproblémové a velice rychlé (obrázek 22).



Obrázek 22: Simulace Mach-Zehnderova interferometru

Pro vyzkoušení záznamu a rekonstrukce off-axis hologramu mi posloužila i možnost vytvořit několik stolů. Nejprve jsem vytvořil hologram dvou bodových zdrojů (obrázek 23 vlevo). Následně jsem mřížku s vytvořeným záznamem duplikoval na druhý stůl společně s referenčním světlem. Přidal jsem spojnou čočku a nový zdroj světla. Soustavu světla a spojné čočky jsem nastavil tak, aby se paprsky protínaly na místě původního referenčního světla. Vytvořil jsem tak paprsky opačně orientované než byly původní a mohl jsem se pustit do rekonstrukce. Mřížku se záznamem jsem přepnul do režimu projekce a osvítil ji přes čočku. Následně jsem si nechal zobrazit minus první difrakční maximum a výsledkem byly dva svazky paprsků, sbíhajících se na místech původních bodů (obrázek 23 vpravo)



Obrázek 23: Simulace záznamu (vlevo) off-axis hologramu a jeho rekonstrukce pomocí spojné čočky

7 Závěr

Cílem této práce bylo opravit a zlepšit výukový holografický simulátor. Ten obsahoval několik zásadních chyb, které znemožňovaly jeho používání. Další postup se zaměřoval na přidání nových optických členů, nezbytných k simulování různých optických soustav používaných v holografii.

Nejdřív byl čtenář seznámen s původní verzí simulátoru. Na postupech při zaznamenávání a rekonstrukci hologramu bylo ukázáno, jaké hlavní chyby program vykazuje.

V části Nový simulátor jsou popsány změny a nově implementované věci. První vyřešenou věcí bylo špatné vykreslování a počítání výstupních paprsků mřížky. Dále se podařilo přidat nové optické členy – rovinné zrcátko, tenkou čočku a dělič svazku. Společně s jejich vkládáním se upravila struktura aplikace pro snadnější přidávání další prvků. K novým prvkům bylo nutné připravit grafické uživatelské rozhraní a jejich grafickou stránku, jako je vykreslování prvku na plátně a také výpočet jejich výstupních paprsků. Zavedení nových prvků podobných vlastností a jejich podpůrných tříd vedlo k implementaci abstraktních tříd sjednocujících jejich společné atributy a metody. Tak došlo ke zjednodušení celé infrastruktury a unifikaci obslužných tříd.

Spolu s rozšířením aplikace přišly i chyby a problémy s tím spojené. Je téměř nemožné je odstranit všechny bez dlouhodobého testování. Některé stávající řešení by bylo vhodné vylepšit v rámci efektivity a zlepšení ovládání pro uživatele.

Zadání obsahovalo další stěžejní bod – převést výpočty a zobrazení do 3D. Se zadavatelem jsme se dohodli, že by toto nebylo z důvodu rozsáhlosti celé aplikace a náročnosti ostatních bodů realizovatelné. Přerod aplikace do třetího rozměru bude nejspíš záležitostí celkového předělání simulátoru.

Simulátor je nyní lépe rozšiřitelný a obohacený o nové prvky, práce na něm bude nadále pokračovat. Dokonce byl program již úspěšně otestován v praxi při návrhu soustavy pro záznam holografického difuzéru na ČVUT v Praze.

Literatura

[1] RENDL, Kamil. *Vizualizace principu hologramu*. Plzeň, 2012. Bakalářská práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd. Vedoucí práce Petr Lobaz.

[2] BENTON, Stephen; BOVE, V. Michael. *Holographic imaging*. Hoboken, N.J.: Wiley-Interscience, c2008, xxiii, 261 p. ISBN 04-700-6806-X

[3] MALÝ, Petr. *Optika*. Vyd. 2., přeprac. Praha: Karolinum, 2013, 368 s. ISBN 978-802-4622-460

[4] Interactive Science Simulations [online]. [cit. 2014-05-01]. Dostupné z: http://phet.colorado.edu/en/simulation/wave-interference

[5] DEPARTMENT OF PHYSICS AND ASTRONOMY: UNIVERSITY OF ROCHESTER. Inerference and Diffraction Simulation [online]. [cit. 2014-05-01]. Dostupné z: http://physics.bu.edu/~duffy/java/Opticsa1.html

[6] BOSTON UNIVERSITY. Optics Simulation [online]. [cit. 2014-05-01]. Dostupné z: http://physics.bu.edu/~duffy/java/Opticsa1.html

[7] THE UNIVERSITY OF QUEENSLAND. Tim's Michelson Applet [online]. [cit. 2014-05-01]. Dostupné z: http://www.physics.uq.edu.au/people/mcintyre/applets/michelson/michelson.html

[8] ORACLE. What Applets Can and Cannot Do [online]. [cit. 2014-05-01]. Dostupné z: http://docs.oracle.com/javase/tutorial/deployment/applet/security.html

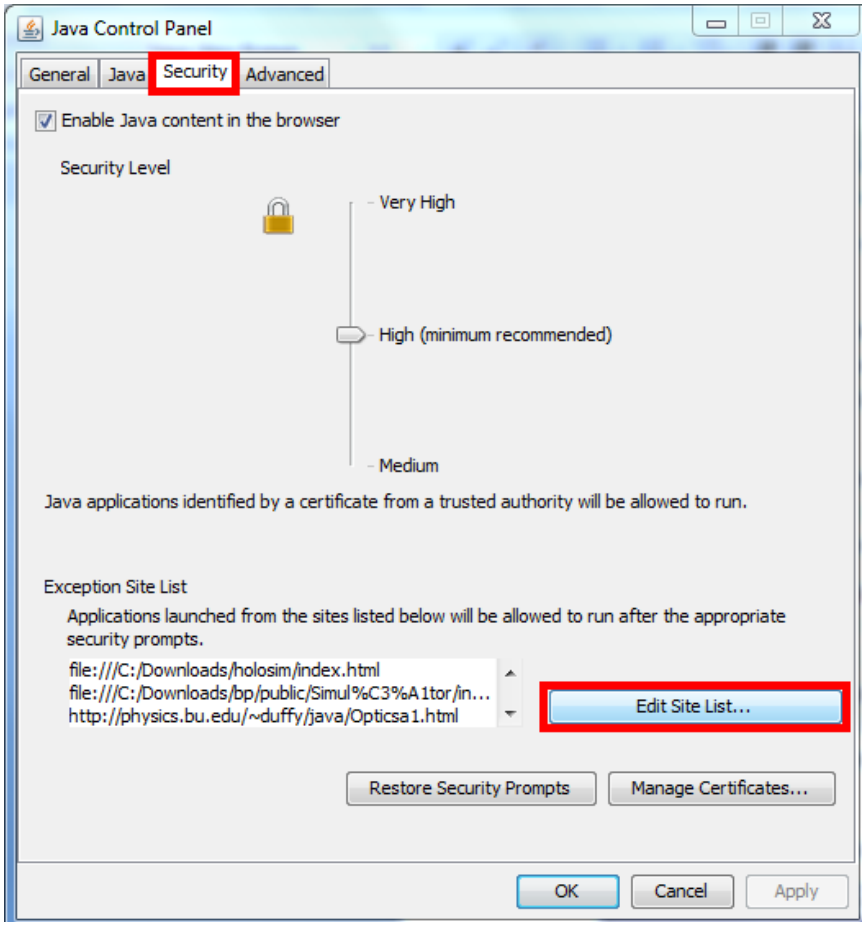
Uživatelská dokumentace

Spuštění simulátoru

Aplikaci lze spustit v jakémkoliv internetovém prohlížeči (Firefox, Internet Explorer aj.) s nainstalovanou podporou Javy. Nemá-li prohlížeč nainstalovánu podporu Javy, obrátíme se na uživatelskou dokumentaci webového prohlížeče a pomocí ní doinstalujeme podporu.

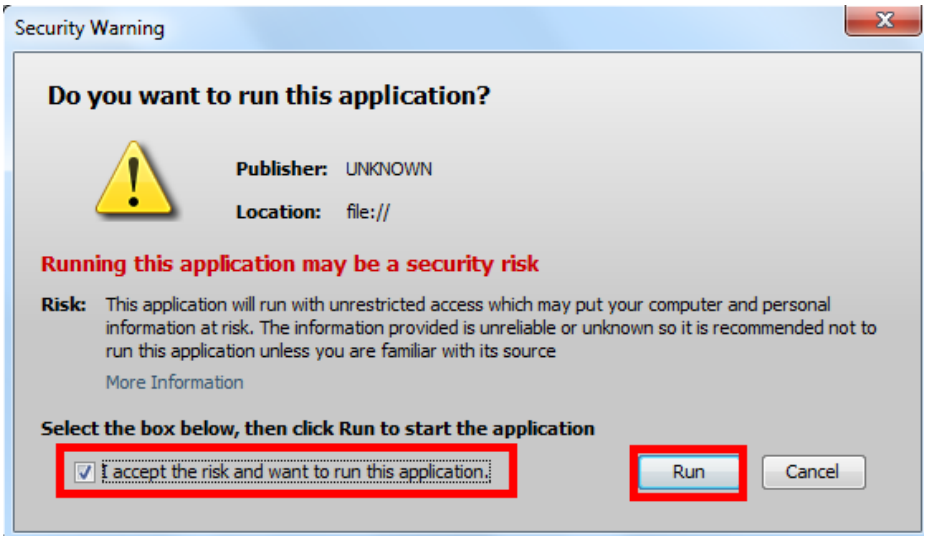
Pro spuštění aplikace prozatím slouží pouze zkušební index.html (obsah přiloženého CD). Poklepáním na něj by se měl otevřít příslušný prohlížeč, není-li tomu tak, lze zadat v internetovém prohlížeči cestu k souboru na disku.

Jelikož se jedná o Java applet, provází jeho spuštění několik bezpečnostních opatření (více viz [8]). Dokonce by se mohlo stát, že aplikace bude zcela zablokována. Taková situace se dá řešit pomocí lokálního nastavení Javy. Nastavení najdeme nejspíše pod názvem *Configure Java* např. přes nabídku Start (Windows 7). Přepneme na záložku *Security – Edit Site List* (viz obrázek 1). Zde pomocí tlačítka přidat (*Add*) vložíme adresu, ze které applet spouštíme.



Obrázek 1: Nastavení bezpečnostních opatření Javy

Při nastavení výjimky pro zobrazovaný applet můžeme být ještě vyzvání bezpečnostním varováním při spouštění. Stačí přijmout případná rizika a aplikaci spustit (obrázek 2).

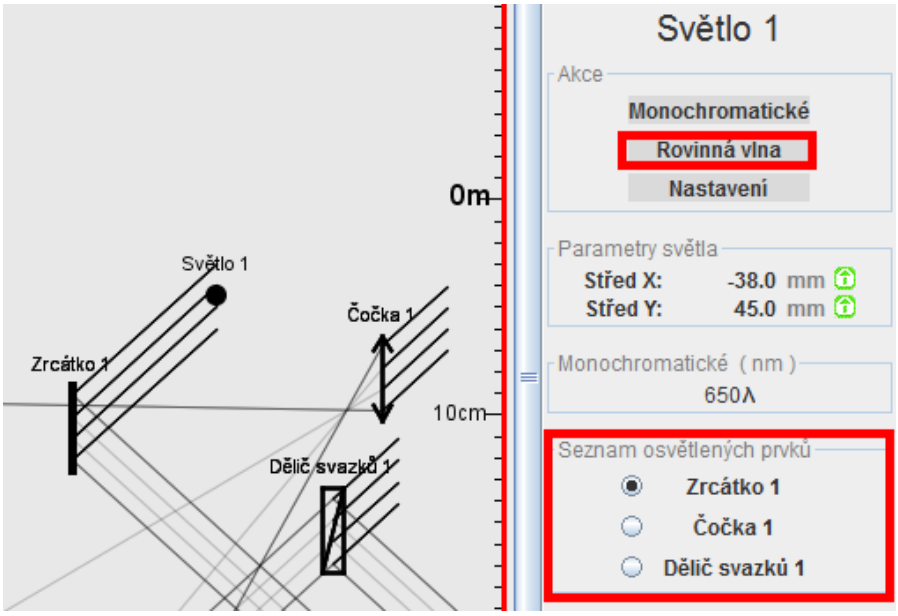


Obrázek 2: Spuštění appletu

Návod pro ovládání původní verze simulátoru viz [1] a v kapitole 2 tohoto textu. Nová verze se obsluhuje stejně jako předchozí, až na pár specifických rozličností (viz níž).

Nastavení světla

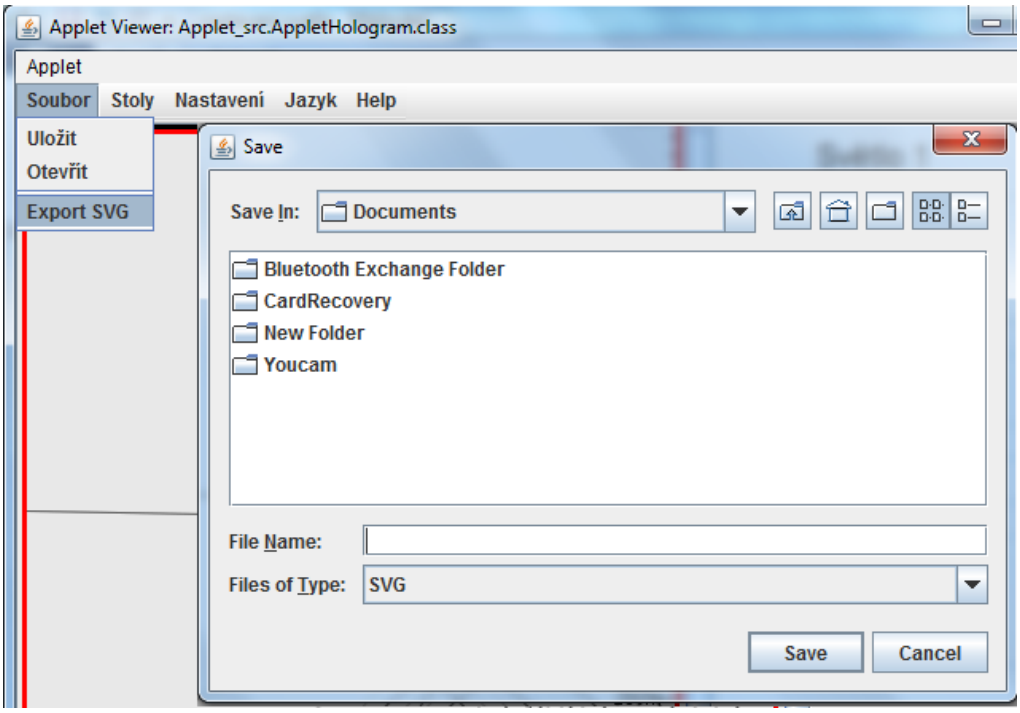
Vlastnosti světla nastaveného jako „*rovinná vlna*“ obohatil seznam s výběrem „referenčního“ členu – *seznam osvětlených prvků*. Úhel, pod kterým dopadají paprsky na všechny osvětlené optické členy, se určuje podle referenčního členu a je pro všechny ostatní stejný (viz obrázek 3).



Obrázek 3: Světlo s vybraným referenčním prvek „Zrcátko 1“

Export do SVG

Položka hlavního menu *Soubor – Export SVG* vytvoří náhled na obrazovku ve vektorové grafice a je možné si tento soubor s koncovkou .svg uložit. K tomu slouží průvodce pro ukládání souboru (viz obrázek 4). Nicméně v aktuální verzi vykazuje export problémové chování s českými názvy obsahujícími diakritiku.



Obrázek 4: Ukládání obrazovky do vektorové grafiky

Čočka

Se všemi nově přidanými prvky se zachází podobně jako s *Mřížkou* v původní verzi. (viz [1] Uživatelská dokumentace – Mřížka). Čočka má ale i své vlastní nastavení. Jde o rozlišení, zda čočka je rozptylka či spojka. Dále pak nastavení jejího ohniska.

Přepnutí mezi spojnou/rozptylnou čočkou lze provést ve vlastnostech čočky (po kliknutí na nastavovanou čočku) a kliknutím na tlačítko *Spojka/Rozptylka* v boxu *Akce*. Ohnisko čočky se určuje v části *Parametry čočky – Ohnisko*. Ohnisko rozptylné čočky se udává v záporných číslech (viz obrázek 5).

Obrázek 5: Vlastnosti čočky