UNIVERZITA PALACKÉHO V OLOMOUCI

PŘÍRODOVĚDECKÁ FAKULTA

KATEDRA OPTIKY

BAKALÁŘSKÁ PRÁCE

Modul 3D vizualizace optických soustav v C++

Vypracoval: Štěpán Venos
ve 3. ročníku

Studijní program: B1701 Fyzika

Studijní obor: Optika a optoelektronika

Forma studia: Prezenční

Vedoucí bakalářské práce: RNDr. Ivo Vyšín, CSc.

Olomouc 2015

Bibliografická identifikace

Jméno a příjmení autora:	Štěpán Venos
Název práce:	Modul 3D vizualizace optických soustav v C++
Typ práce:	Bakalářská
Pracoviště:	Katedra optiky
Vedoucí práce:	RNDr. Ivo Vyšín, CSc.
Rok obhajoby práce:	2015
Abstrakt:	Základním cílem práce je vytvoření modulu – soustavy zdrojových souborů v C++ pro 3D vizualizaci optických soustav, složených ze sférických, mírně asférických a rovinných optických ploch. Modul musí být začlenitelný do jiných programů napsaných v C++, zpracovávaných pro výpočty, související s návrhem optických soustav. Modul by měl kromě jiného umožňovat základní operace s 3D vizualizací, jako je např. změna měřítka, natáčení soustav, výběr detailů apod. Modul musí umožňovat znázornění průchodů vybraných paprsků optickými soustavami. Součástí teoretické části práce bude přehled známých algoritmů pro propočet chodu obecně mimoběžných paprsků uvedenými optickými plochami.
Klíčová slova:	C++, optická soustava, 3D vizualizace, asférická plocha, Federovy vztahy, sledování paprsků
Počet stran:	42
Počet příloh:	1
Jazyk:	Český

Bibliographic identification

Autor's first name and surname:	Štěpán Venos
Title:	The Module of 3D Visualization of the Optical Systems in C++
Type of thesis:	Bachelor
Department:	Department of Optics
Supervisor:	RNDr. Ivo Vyšín, CSc.
The year of presentation:	2015
Abstract:	The main goal of the thesis is to create a module – a system of source files in C++ for 3D visualization of the optical systems, consisting of spherical, moderately aspherical and plane optical surfaces. The module must be embeddable into other programs written in C++ which process calculations, and which are related to the design of optical systems. The module should allow among others the basic operations with a 3D visualization such as change of the scale, rotation of systems, selection of details, etc. The module must allow the representation of a trace of selected rays through the optical systems. In the theoretical part, there will be an overview of known algorithms for the calculation of a trace of generally skew rays for given optical surfaces.
Keywords:	C++, optical system, 3D visualization, aspheric surface, Feder's equations, raytracing
Number of pages:	42
Number of appendices:	1
Language:	Czech

Prohlašuji, že jsem předloženou bakalářskou práci vypracoval samostatně pod vedením RNDr. Ivo Vyšína, CSc., a výhradně s použitím citovaných zdrojů.

V Olomouci dne 1. srpna 2015

. . . . . .

Štěpán Venos

Děkuji panu RNDr. Ivo Vyšínovi, CSc. za odborné vedení při psaní kvalifikační práce, věnovaný čas a jeho trpělivost a děkuji za poskytnutí literatury.

Obsah

Úvod **7**

1 Přehled algoritmů průchodu paprsku centrovanou optickou soustavou **9**

1.1 Algoritmus průchodu paprsku centrovanou optickou soustavou v paraxiálním prostoru 9

1.2 Algoritmus průchodu obecně mimoběžného paprsku sférickými plochami v mimoparaxiálním prostoru 10

1.3 Algoritmus průchodu obecně mimoběžného paprsku asférickými plochami v mimoparaxiálním prostoru 15

2 OptVis, modul 3D vizualizace optických soustav **21**

2.1 Realizace modulu 21

2.1.1 TrackballControls 22

2.2 Integrace modulu do programu pro návrh optických soustav Opt 23

2.3 Vnitřní struktura modulu 24

2.3.1 Významné pomocné procedury a funkce v modulu 25

2.3.2 Významné třídy v modulu 26

2.4 Průvodce uživatele 30

2.5 Aplikace v praxi a budoucí vývoj modulu 33

Závěr **36**

Slovník pojmů **37**

Seznam použité literatury a zdrojů **40**

Příloha na CD-ROM **42**

Úvod

Optika je součástí každodenního života. Mnoho lidí využívá výhody brýlí a kontaktních čoček. Samotné lidské oko je komplikovanou optickou soustavou. Fotoaparáty, kamery, dalekohledy a mikroskopy využívají zákonů optiky, aby mohly sloužit svým účelům. Používá se optická telekomunikace. Je tedy důležité těmto systémům rozumět a umět je konstruovat.

Vlastní realizaci optického systému předchází návrh jeho optické soustavy, kdy v této souvislosti chápeme optickou soustavu jako soubor lámavých či odrazných nejčastěji rotačně symetrických sférických či asférických optických ploch, oddělujících různá prostředí. Optickým systémem pak můžeme nazvat vlastní realizaci optické soustavy ve formě čoček a zrcadel. Pokud použijeme známé rozdělení optiky jako vědního oboru podle použitých metod zkoumání, je známo, že základní návrhy optických soustav se provádějí s využitím metod geometrické optiky, jejímž předpokladem je šíření světla ve formě paprsků. Můžeme trasovat zvolený soubor paprsků, které se účastní zobrazení, optickou soustavou a na výstupu z ní např. vyhodnocovat, jak se svazky paprsků, vycházející z bodů zobrazovaného předmětu, liší od ideální homocentricity v obrazovém prostoru, která by znamenala, že bod předmětu se optickou soustavou opět zobrazí jako ideální bod. Můžeme sledovat geometrické vady neboli aberace řešené optické soustavy. V pokročilejším stadiu návrhu optických soustav a při konečném hodnocení výsledků návrhu se dále uplatňují metody vlnové optiky. Můžeme např. vyhodnocovat vlnové aberace optické soustavy, zkonstruovat pupilovou funkci a následně optickou funkci přenosu nebo bodovou rozptylovou funkci. Ale i tyto metody vlnové optiky jsou v praktických výpočtech založeny na propočtu chodu paprsků optickou soustavou. Proto je teoretická část předložené práce především věnována prezentaci algoritmů pro propočet chodu paprsků soustavou rotačně symetrických sférických a asférických optických ploch. Jsou prezentovány algoritmy, které se používají v paraxiálním prostoru a dále algoritmy pro propočet chodu obecně mimoběžných paprsků v prostoru mimoparaxiálním.

V průběhu návrhu optické soustavy je nutno neustále sledovat průchod souboru vybraných paprsků optickou soustavou (anglicky raytracing). Je nutno např. sledovat dopadové výšky paprsků na jednotlivé optické plochy soustavy, je nutno sledovat dopadové úhly, aby se např. vyloučila možnost blízkého totálního odrazu apod. Korigování vad optické soustavy se provádí změnou konstrukčních parametrů – poloměrů optických ploch, vzdáleností vr-

cholů optických ploch, optických skel a v případě sférických ploch také změnou sférických koeficientů. S výjimkou optických skel se změna všech ostatních parametrů projeví ve tvaru čoček nebo zrcadel. Proto je také velmi důležité mít možnost během procesu návrhu neustále sledovat aktuální tvar optických ploch a tak např. vyloučit, že se lámavé plochy čočky protnou v menší vzdálenosti od optické osy, než je uvažovaný účinný poloměr optické plochy nebo eliminovat případy, kdy okrajová tloušťka čočky bude natolik malá, že čočka bude prakticky nevyrobitelná. Ale na základě těchto sledování je možné eliminovat i opačný případ, kdy čočka bude zbytečně tlustá a tím i dražší (samozřejmě s výjimkou případů, kdy je větší osová tloušťka čočky výsledkem procesu optimalizace aberací), nebo když celkové rozměry optické soustavy již neodpovídají požadavkům zadání apod. Je možné použít 2D nebo 3D grafická znázornění. Je ale zřejmé, že 3D grafika dává více informací a to především v těch případech, kdy budou možné i manipulace s 3D grafikou, jako je zoomování, různé způsoby natáčení, různé způsoby projekcí atd. Samozřejmě programování 3D grafiky je daleko náročnější a zdaleka ne každý relativně jednodušší program pro optické výpočty těmito funkcemi 3D grafiky disponuje. Cílem práce je proto vytvoření modulu 3D grafiky optických soustav včetně znázornění průchodu zvolených paprsků optickou soustavou v programovacím jazyku C++, který je nejrozšířenější v oblasti programování optických výpočtů. Pro tento úkol je dále zvolena grafická knihovna OpenGL a další pomocné knihovny, které jsou popsány v kapitole 2. Tento modul by měl obsahovat uvedené pokročilé funkce, různé režimy vizualizace jako např. drátěný model, ortografickou projekci, různá barevná schémata, řezy atd., samozřejmě funkce výstupu na tiskárny a měl by být pomocí relativně jednoduchého programového rozhraní implementovatelný do jiných programů pro návrh optických soustav.

Kapitola 1

Přehled algoritmů průchodu paprsku centrovanou optickou soustavou

V této kapitole se zabývám výpočtem průchodu paprsku centrovanou optickou soustavou k sférických a asférických optických ploch. Soustava je určena indexy lomu optických prostředí $n_0, n_1, \ldots, n_k$ mezi optickými plochami, osovými mezerami mezi vrcholy optických ploch $d_0, d_1, \ldots, d_{k-1}$ a hodnot křivostí $C_0, C_1, \ldots, C_k$ sférických ploch. V případě asférických ploch uvádíme kromě křivostí další údaje viz podkapitola 1.3.

1.1 Algoritmus průchodu paprsku centrovanou optickou soustavou v paraxiálním prostoru

Centrovaná optická soustava je řada sférických, asférických a rovinných rotačně symetrických optických ploch oddělených optickými prostředími. Účelem optické soustavy je optické zobrazování. V případě fyzikálně dokonalého zobrazení se bod zobrazí jako bod, přímka se zobrazí v přímku a rovina v rovinu. Fyzikálně dokonalé zobrazení je uskutečněno pouze v paraxiálním prostoru, který je oblastí v blízkosti optické osy a pouze při malých průměrech optických ploch. Paraxiální paprsky svírají úhly s osou menší než 2°. Je-li předmět veliký, zobrazení není fyzikálně dokonalé a je zatíženo vadami – aberacemi. Pro dosažení kvalitního zobrazení je nutné vhodně čočky kombinovat – je potřeba propracovaný návrh, jak se uvádí v [1]. K tomuto účelu byly vytvořeny CAD programy, které využívají algoritmy pro sledování paprsků, řeší automatické korigování a poskytují podrobnou vizualizaci soustav.

Průchod paprsku j-tou plochou soustavy v paraxiálním prostoru je popsán zobrazovací rovnicí

$$\frac{n'_j}{s'_j} - \frac{n_j}{s_j} = C_j \left(n'_j - n_j\right), \tag{1.1}$$

kde s_j a s'_j je vzdálenost předmětu, resp. obrazu od vrcholu j-té plochy soustavy.

Používám zde znaménkovou konvenci podle [1] – směr šíření světla zleva doprava je kladný. Leží-li předmětový, nebo obrazový bod vpravo od vrcholu lámavé plochy, je jeho vzdálenost kladná. Poloměr lámavé plochy je kladný, leží-li střed křivosti plochy vpravo od vrcholu plochy. Úhel sklonu paprsku orientujeme od optické osy k paprsku, přičemž směr hodinových ručiček je považován za kladný. Úhel dopadu a úhel lomu orientujeme ve směru od paprsku ke kolmici k rozhraní, přičemž opět směr hodinových ručiček považujeme za kladný.

Pro přechod na další plochu soustavy je pak nutno použít přechodového vztahu

$$s_{j+1} = s'_j - d_j. \tag{1.2}$$

Výpočtu průchodu paprsku soustavou v paraxiálním prostoru bude použito k dalším výpočtům, např. k výpočtu obrazové sečné vzdálenosti optické soustavy s'_k, kterou získáme postupným opakováním aplikace vztahů (1.1) a (1.2) na všechny plochy optické soustavy, a ohniskové vzdálenosti soustavy f', jejíž hodnota je dána vztahem

$$f' = \frac{s'_1 s'_2 \dots s'_k}{s_2 s_3 \dots s_k}. \tag{1.3}$$

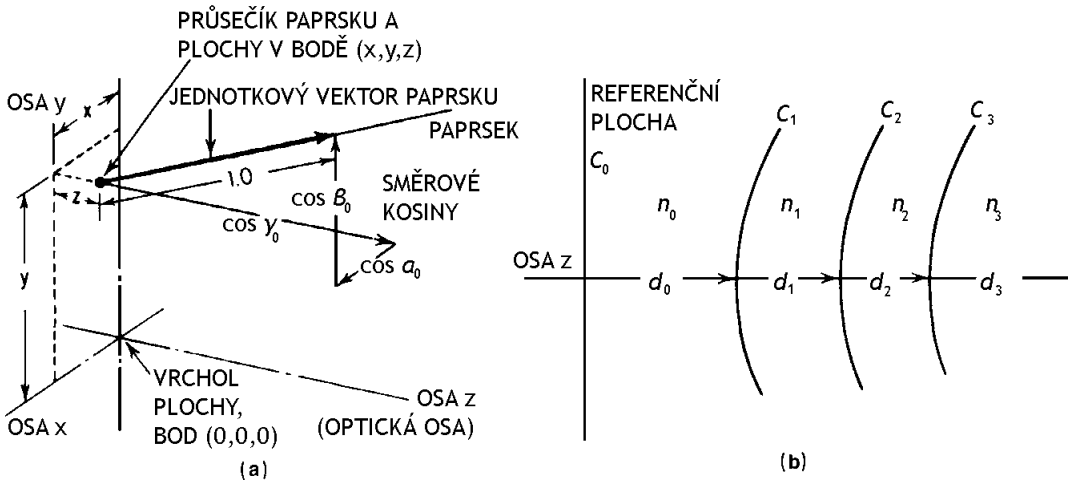
Všechny hodnoty s_j, s'_j v rovnici (1.3) jsou vypočítány pomocí vztahů (1.1) a (1.2) při dosazení $s_1 = \infty$. [2]

1.2 Algoritmus průchodu obecně mimoběžného paprsku sférickými plochami v mimoparaxiálním prostoru

V současné době, kdy se k návrhu a analýze optických soustav používají počítačové programy (OSLO, Zemax, CodeV), není tolik důležité znát techniku sledování paprsků. Nicméně tato znalost může být občas užitečná. Prezentuji zde rovnice, které jsou určeny pro vykonání na počítači (jsou ošetřeny proti zacyklení, pokud nelze propočítat paprsek, protože mine plochu, nebo protože na ploše nastane úplný odraz).

Rovnice uvedené níže jsou lehce upravené rovnice, které D. Feder prezentoval v [3]. Federovy algoritmy se skládají ze čtyř kroků:

- Zahájení výpočtu definováním paprsku. (Popř. kontrola přesnosti předchozího výpočtu.)



Obrázek 1.1: *Symboly použité v rovnicích (1.4) až (1.19). (a) Prostorové souřadnice (x_0, y_0, z_0) průsečíku plochy a paprsku a jeho směrové kosiny* ($\cos\alpha_0$, $\cos\beta_0$, $\cos\gamma_0$)*. (b) Zobrazuje systém značení dolních indexů. Převzato z [4].*

- Přenos na následující optickou plochu.
- Odraz, nebo lom. Odraznou plochu zadáme oproti ploše lámavé pomocí převráceného znaménka u osové mezery d_1 a indexu lomu n_1 prostředí za plochou, na které se odraz odehrává.
- Ukončení výpočtu předáním výstupních hodnot.

Obecně mimoběžný paprsek je definován směrovými kosiny $\cos\alpha$, $\cos\beta$, $\cos\gamma$ a souřadnicemi x, y, z jeho průsečíku P s plochou. V dalším textu připojuji k těmto veličinám dolní indexy. Počátek soustavy souřadnic je ve vrcholu O každé optické plochy. Obrázek 1.1 ilustruje tuto fyzikální situaci. Pokud jsou x a $\cos\alpha$ oba nulové, paprsek se nazývá meridiální (ležící v tzv. tangenciální rovině y–z). Směrové kosiny jsou projekce jednotkového vektoru podél paprsku na souřadné osy. (Pozn.: směrový kosinus vynásobený indexem lomu se nazývá optický směrový kosinus.)

Výpočet je zahájen určením hodnot pro x_0, y_0, z_0 a $\cos\alpha_0$, $\cos\beta_0$, $\cos\gamma_0$ vzhledem k referenční ploše. Referenční plochou je obecně nazvána plocha v optické soustavě před plochou, na které se řeší odraz, nebo lom paprsku. Může jí být rovina (což je obvyklá volba), nebo sféra. Vhodné volby pro umístění referenční plochy jsou v rovině předmětu, v tečné rovině k vrcholu zvolené optické plochy, nebo v rovině vstupní pupily.

Zahájení (na referenční ploše):

$$C_0\left(x_0^2+y_0^2+z_0^2\right)-2z_0=0 \tag{1.4}$$

$$\cos\alpha_0^2+\cos\beta_0^2+\cos\gamma_0^2=1.0, \tag{1.5}$$

kde C_0 označuje křivost referenční plochy.

Všimněte si, že rovnice (1.4) je rovnicí koule (to zaručuje, že počátek paprsku leží na referenční ploše) a že rovnice (1.5) zaručuje, že kvadrát jednotkového vektoru podél paprsku se rovná 1.0.

Přenos na plochu č.1 (nebo další plochu):

$$l_0=d_0\cos\gamma_0-(x_0\cos\alpha_0+y_0\cos\beta_0+z_0\cos\gamma_0) \tag{1.6}$$

$$M_{0z}=z_0+l_0\cos\gamma_0-d_0 \tag{1.7}$$

$$M_0^2=x_0^2+y_0^2+z_0^2-l_0^2+d_0^2-2d_0z_0 \tag{1.8}$$

$$\cos\theta_1=\sqrt{\cos\gamma_0^2-C_1^2M_0^2+2C_1M_{0z}} \tag{1.9}$$

$$L_0=l_0+\frac{C_1M_0^2-2M_{0z}}{\cos\gamma_0+\cos\theta_1} \tag{1.10}$$

$$z_1=z_0+L_0\cos\gamma_0-d_0 \tag{1.11}$$

$$y_1=y_0+L_0\cos\beta_0 \tag{1.12}$$

$$x_1=x_0+L_0\cos\alpha_0, \tag{1.13}$$

kde d_0 je osová mezera mezi referenční plochou a plochou č.1 a C_1 je křivost plochy č.1.

Směrové kosiny normály plochy č.1 v bodě průsečíku P_1 jsou

$$\cos\gamma_N = 1 - C_1 z_1$$
$$\cos\beta_N = -C_1 y_1$$
$$\cos\alpha_N = -C_1 x_1. \tag{1.14}$$

Odraz, nebo lom:

$$\cos\theta_1' = \sqrt{1 - \left(\frac{n_0}{n_1}\right)^2 (1 - \cos^2\theta_1)} \tag{1.15}$$

$$g_1 = \cos\theta_1' - \frac{n_0}{n_1}\cos\theta_1 \tag{1.16}$$

$$\cos\gamma_1 = \frac{n_0}{n_1}\cos\gamma_0 + g_1 (1 - C_1 z_1) \tag{1.17}$$

$$\cos\beta_1 = \frac{n_0}{n_1}\cos\beta_0 - g_1 C_1 y_1 \tag{1.18}$$

$$\cos\alpha_1 = \frac{n_0}{n_1}\cos\alpha_0 - g_1 C_1 x_1, \tag{1.19}$$

kde n_0, n_1 jsou indexy lomu viz obrázek 1.1.

Členy s indexem 0 odkazují na referenční plochu a následující optické prostředí. Členy s indexem 1 se vztahují k ploše č.1 a následujícímu optickému prostředí. Podrobný popis symbolů použitých v rovnicích je v tabulce 1.1.

Tabulka 1.1: *Významy symbolů v textu. Převzato z [4].*

x_0, y_0, z_0	Prostorové souřadnice průsečíku P_0 paprsku s referenční plochou.
x_1, y_1, z_1	Prostorové souřadnice průsečíku P_1 paprsku s plochou č.1.
$\vec{M}_0$	Vektor s počátkem ve vrcholu plochy č.1 končící na paprsku, ke kterému je kolmý. Úsek $l_0 = \overline{P_0 M_0}$.
M_{0z}	z-ová složka $\vec{M}_0$.
$\cos\theta_1$	Kosinus úhlu dopadu na plochu č.1.
$\cos\theta_1'$	Kosinus úhlu lomu na ploše č.1.
L_0	Vzdálenost podél paprsku od referenční plochy (x_0, y_0, z_0) k ploše č.1 (x_1, y_1, z_1). (L_j je vzdálenost od plochy j k $j+1$.)
$\cos\alpha_0$, $\cos\beta_0$, $\cos\gamma_0$	Směrové kosiny paprsku v prostředí mezi referenční plochou a plochou č.1 (před odrazem, nebo lomem).
$\cos\alpha_1$, $\cos\beta_1$, $\cos\gamma_1$	Směrové kosiny po odrazu, nebo lomu od plochy č.1.
$\cos\alpha_N$, $\cos\beta_N$, $\cos\gamma_N$	Směrové kosiny normály plochy č.1 v bodě P_1.
C_0	Křivost (reciproký poloměr) referenční plochy.
C_1	Křivost plochy č.1.
d_0	Délka osové mezery mezi referenční plochou a plochou č.1.
n_0	Index lomu prostředí mezi referenčním plochou a plochou č.1.
n_1	Index lomu prostředí za plochou č.1.

Volba kladné hodnoty pro druhou odmocninu v rovnici (1.9) vybere ten průsečík paprsku s plochou, který je blíže vrcholu plochy. V případě, že argument pod odmocninou v rovnici (1.9) je záporný, paprsek míjí sférickou plochu. Pokud je argument pod odmocninou v rovnici (1.15) záporný, znamená to, že úhel dopadu je větší než mezní úhel, paprsek je tudíž vystaven úplnému odrazu a nemůže projít přes plochu.

Výpočet je zahájen vložením C_0, dvou souřadnic z (x_0, y_0, z_0) a dvou směrových kosinů z ($\cos\alpha_0$, $\cos\beta_0$, $\cos\gamma_0$) do rovnic (1.4) a (1.5) a řešíme třetí prostorovou souřadnici a třetí směrový kosinus. Potom se stanoví průsečík paprsku s plochou č.1 (x_1, y_1, z_1) pomocí rov-

nic (1.6) až (1.13). Směrové kosiny paprsku po odrazu či lomu na ploše č.1 ($\cos\alpha_1$, $\cos\beta_1$, $\cos\gamma_1$) získáme z rovnic (1.14) až (1.19). Tím je sledování paprsku přes plochu č.1 dokončeno. V tomto bodě můžeme použít rovnice (1.4) a (1.5) s indexy navýšenými o jedničku ke kontrole přesnosti výpočtu.

K přenosu na plochu č.2 zvýšíme o jedničku indexy v rovnicích (1.6) až (1.13) a určíme (x_2, y_2, z_2). Podobně směrové kosiny paprsku po odrazu či lomu na ploše č.2 ($\cos\alpha_2$, $\cos\beta_2$, $\cos\gamma_2$) nalezneme pomocí rovnic (1.14) až (1.19) zvýšením indexů o jedničku.

Tento postup se opakuje až do stanovení průsečíku paprsku s poslední plochou soustavy, kterou je obvykle obrazová rovina. Tím končí výpočet. Všimněte si, že každý paprsek, který protíná osu je meridiálním paprskem. Dále bez újmy na obecnosti předpokládáme, že předmětový bod leží v rovině y–z souřadného systému (protože předpokládáme systém s rotační symetrií). Proto každý obecně mimoběžný paprsek může začít s x_{j-1} rovno nule. Pokud to takto uděláme, je zřejmé, že obě poloviny optického systému, před a za rovinou y–z, jsou zrcadlové obrazy a že každý paprsek ($\cos\alpha_j$, $\cos\beta_j$, $\cos\gamma_j$) jdoucí přes (x_j, y_j, z_j) má zrcadlový obraz ($-\cos\alpha_j$, $\cos\beta_j$, $\cos\gamma_j$) procházející ($-x_j$, y_j, z_j) v druhé polovině systému. Z tohoto důvodu je nutné sledovat obecně mimoběžné paprsky pouze přes polovinu apertury optické soustavy. [4]

1.3 Algoritmus průchodu obecně mimoběžného paprsku asférickými plochami v mimoparaxiálním prostoru

Rovnici asférické plochy reprezentujeme ve výhodném tvaru pro účely výpočtu

$$z = f(x,y) = \frac{Cr^2}{1+\sqrt{1-(1+\kappa)C^2r^2}} + \\ +A_2r^2 + A_4r^4 + \ldots + A_jr^j, \tag{1.20}$$

kde z je vzdálenost bodu na ploše od roviny kolmé k optické ose procházející vrcholem plochy a r je vzdálenost bodu na ploše od optické osy. Platí tedy

$$r^2 = x^2 + y^2. \tag{1.21}$$

Prvním členem na pravé straně rovnice (1.20) je rovnice rotačně symetrické kvadratické plochy, jejíž oskulační sféra má křivost C (viz obrázek 1.2). Oskulační sféra se dotýká vrcholu této kvadratické plochy a v tomto bodě má společné první a druhé parciální derivace proměnných x a y. Následující deformační členy představují odchylky od této kvadratické plochy (kvadriky – algebraické plochy druhého stupně). Velikost koeficientů A_2, A_4 atd. volíme mnohem menší než jedna. Jelikož můžeme zahrnout libovolný počet deformačních členů, rovnice (1.20) je poměrně flexibilní a může představovat některé dosti extrémní asféry. Všimněte si, že je tato rovnice redundantní v deformačním členu druhého řádu $(A_2 r^2)$, který není nutný ke specifikování plochy, protože může být implicitně zahrnut v křivosti oskulační sféry C. Význam zařazení tohoto členu je, že v případě velké hodnoty C, paprsky, které by ve skutečnosti protínaly asférickou plochu, nemusí protínat oskulační sféru.

V rovnici (1.20) se ještě vyskytuje kónická konstanta κ. Hodnota kónické konstanty určuje typ kuželosečky a kvadriky získané rotací této kuželosečky okolo osy symetrie viz tabulka 1.2. Kromě ní existují ještě další dvě kónické konstanty pro definování kuželoseček, pro něž platí

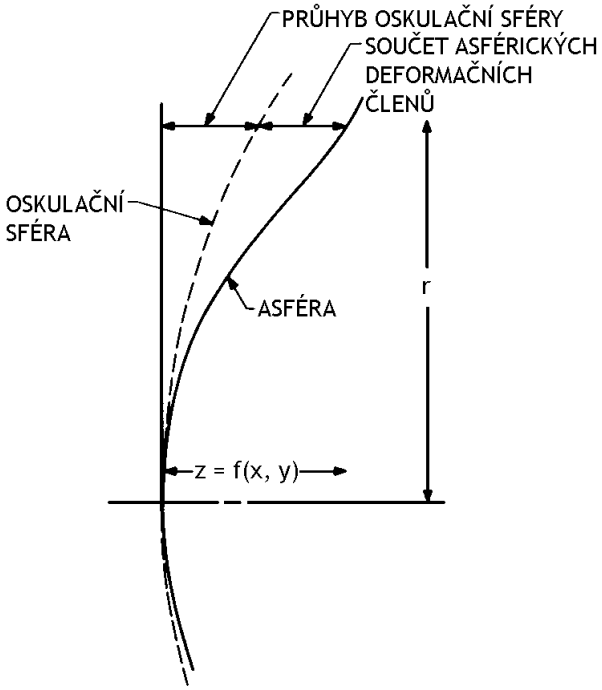
$$\kappa = p - 1 = -e^2, \tag{1.22}$$

kde e se nazývá excentricita. Asférické plochy, které jsou rotačními kvadrikami (paraboloid, elipsoid, hyperboloid), mají koeficienty A_i, $i > 4$, nulové.

Tabulka 1.2: *Kuželosečky a jejich kónické konstanty. Převzato z [4].*

Zploštělá elipsa	$p > 1$	$\kappa > 0$	$e^2 < 0$
Kružnice	$p = 1$	$\kappa = 0$	$e = 0$
Protáhlá elipsa	$1 > p > 0$	$0 > \kappa > -1$	$0 < e^2 < 1$
Parabola	$p = 0$	$\kappa = -1$	$e = 1$
Hyperbola	$p < 0$	$\kappa < -1$	$e^2 > 1$

Obtížnost propočtu paprsku přes asférické plochy spočívá ve stanovení průsečíku paprsku s asférou. Není možné ho určit přímo. Ve zde uvedené metodě provádíme řadu přiblížení, která pokračují, dokud není chyba zanedbatelná. V následujícím textu opět připojuji k těmto veličinám dolní indexy.



Obrázek 1.2: *Význam rovnice (1.20), která definuje asférickou plochu pomocí deformace od oskulační sféry. Souřadnice z bodu na ploše je součtem souřadnice z oskulační sféry a součtu všech deformačních členů. Převzato z [4].*

Prvním krokem je výpočet souřadnic průsečíku paprsku s oskulační sférou asférické plochy, což je obvykle slušná aproximace asférické plochy. To je provedeno pomocí rovnic (1.6) až (1.13) z podkapitoly 1.2. Průsečík P_1 má souřadnice x_1, y_1, z_1.

Následuje přiřazení hodnot do nových proměnných

$$\begin{aligned} x_{1,1} &= x_1 \\ y_{1,1} &= y_1 \\ z_{1,1} &= z_1. \end{aligned} \tag{1.23}$$

Potom do rovnice pro asféru (1.20) vložíme $r_{1,1}^2 = x_{1,1}^2 + y_{1,1}^2$ a nalezneme z-ovou souřadnici asféry ($\widetilde{z_{1,1}}$) odpovídající této vzdálenosti od osy

$$\widetilde{z_{1,1}} = f(x_{1,1}, y_{1,1}). \tag{1.24}$$

Následně se spočítá

$$c_{1,1} = \sqrt{1 - (1+\kappa_1) C_1^2 r_{1,1}^2} \tag{1.25}$$

$$b_{1,1} = -y_{1,1} \left[C_1 + c_{1,1}(2A_2 + 4A_4 r_{1,1}^2 + \ldots + jA_j r_{1,1}^{(j-2)} \right] \tag{1.26}$$

$$a_{1,1} = -x_{1,1} \left[C_1 + c_{1,1}(2A_2 + 4A_4 r_{1,1}^2 + \ldots + jA_j r_{1,1}^{(j-2)} \right] \tag{1.27}$$

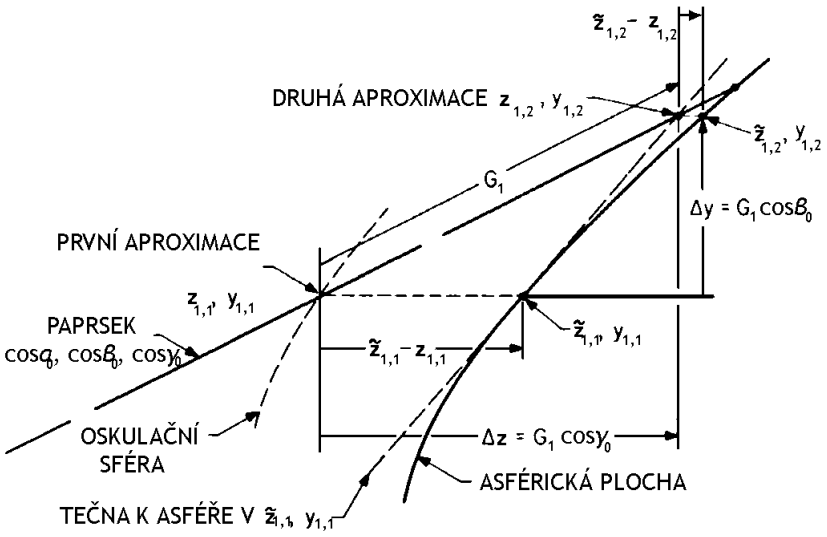
$$G_{1,1} = \frac{c_{1,1}(\widetilde{z_{1,1}} - z_{1,1})}{c_{1,1}\cos\gamma_0 + b_{1,1}\cos\beta_0 + a_{1,1}\cos\alpha_0}, \tag{1.28}$$

Další aproximace souřadnic průsečíku je dána rovnicemi

$$x_{1,2} = G_{1,1}\cos\alpha_0 + x_{1,1} \tag{1.29}$$

$$y_{1,2} = G_{1,1}\cos\beta_0 + y_{1,1} \tag{1.30}$$

$$z_{1,2} = G_{1,1}\cos\gamma_0 + z_{1,1}. \tag{1.31}$$



Obrázek 1.3: *Určení průsečíku paprsku s asférickou plochou. Průsečík je nalezen konvergentní řadou aproximací. Zde jsou zobrazeny vztahy spojené s hledáním první aproximace po průniku paprsku oskulační sférou. Převzato z [4].*

Postup je ilustrován na obrázku 1.3. Aproximace se nyní opakuje (rovnice (1.24) až (1.31)), dokud není chyba menší než zvolená přesnost výpočtu δ, tj. dokud (po n-tém opakování tohoto postupu)

$$\left|\widetilde{z_{1,n}} - z_{1,n}\right| < \delta. \tag{1.32}$$

Nyní aktualizujeme složky průsečíku P_1, který je nyní průsečíkem s asférou

$$\begin{aligned} x_1 &= x_{1,n} \\ y_1 &= y_{1,n} \\ z_1 &= z_{1,n}. \end{aligned} \tag{1.33}$$

Odraz, nebo lom na ploše se provede prostřednictvím následujících rovnic:

$$P_1^2 = c_{1,n}^2 + b_{1,n}^2 + a_{1,n}^2 \tag{1.34}$$

$$F_1 = c_{1,n}\cos\gamma_0 + b_{1,n}\cos\beta_0 + a_{1,n}\cos\alpha_0 \tag{1.35}$$

$$F_1' = \sqrt{P_1^2\left(1-\frac{n_0^2}{n_1^2}\right)+\frac{n_0^2}{n_1^2}F_1^2} \tag{1.36}$$

$$g_1 = \frac{1}{P_1^2}\left(F_1' - \frac{n_0}{n_1}F_1\right) \tag{1.37}$$

$$\cos\gamma_1 = \frac{n_0}{n_1}\cos\gamma_0 + g_1 c_{1,n} \tag{1.38}$$

$$\cos\beta_1 = \frac{n_0}{n_1}\cos\beta_0 + g_1 b_{1,n} \tag{1.39}$$

$$\cos\alpha_1 = \frac{n_0}{n_1}\cos\alpha_0 + g_1 a_{1,n}. \tag{1.40}$$

V rovnicích (1.25) až (1.28) veličiny $a_{1,i}$, $b_{1,i}$ a $c_{1,i}$ jsou směrové kosiny normály plochy krát P_1 a v rovnicích (1.26) a (1.27) složený člen v hranatých závorkách je aproximovaná křivost v bodech na asféře vzdálených $r_{1,i}$ od osy při i-té aproximaci. V rovnicích (1.35) a (1.36) $F_1 = P_1\cos\theta_1$ a $F_1' = P_1\cos\theta_1'$.

Tím končí propočet paprsku přes asféru. Výstupem jsou prostorové souřadnice průsečíku s asférickou plochou x_1, y_1 a z_1 a nové směrové kosiny paprsku $\cos\alpha_1$, $\cos\beta_1$ a $\cos\gamma_1$. [4]

Kapitola 2

OptVis, modul 3D vizualizace optických soustav

Tvorbu tohoto modulu jsem si zvolil po konzultaci s vedoucím práce z důvodu usnadnění zkoumání optických soustav při jejich návrhu. Realistická a graficky precizní vizualizace, kterou lze přepínat do různých režimů, je užitečným nástrojem při jakékoli technické konstrukci. Pomáhá např. odhalit návrh, ve kterém jdou čočky do břitu.

Režimy vizualizace v OptVis lze kombinovat. Jejich výčet spolu s ovládacím prvkem je v podkapitole 2.4. V podkapitole 2.5 některé režimy pak prezentuji na obrázcích. V celé této kapitole používám termíny z informatiky, které vysvětluji ve slovníku na konci textu bakalářské práce.

2.1 Realizace modulu

Modul jsem napsal v programovacím jazyce C++ (standard C++98). Kód jsem zkompiloval s GCC (sada kompilátorů ve verzi 4.7) v editoru Code::Blocks [5] ve verzi 13.12 pro platformu Microsoft Windows. Zdrojové soubory je možné zahrnout do jiného projektu libovolného vývojového prostředí určeného pro jazyk C++, je však nutné vhodně přenastavit kompilátor (především *linker* a nahradit hlavičkové soubory knihoven funkcí těmi, které zvolené prostředí podporuje). Při tvorbě OptVis jsem používal hlavně objektově orientované programování, které jsem na některých místech nahradil procedurálním stylem z důvodu úspory času při vývoji. Stav nastavení modulu se ukládá do globálních proměnných ve jmenném prostoru OptVis.

K vykreslování 3D počítačové grafiky jsem používal rozhraní pro programování aplikací pro grafický hardware OpenGL API ve verzi 2.0. OpenGL je stavový automat – výpočetní model, který si udržuje stav, dokud mu ho příkazy nezměníme. Komunikace probíhá podle protokolu klient–server. Zde jsem řešil problémy *aliasingu*, průhlednost, míchání barev za-

ložené na řazení objektů ve scéně, šířku čar atd.

Dále jsem využil tří momentálně nejrozšířenějších knihoven pro zjednodušení práce s OpenGL a knihovnu NanoVG pro 2D vektorovou grafiku v OpenGL pro vlastní jednoduché GUI. Všechny uvádím v tabulce 2.1.

Tabulka 2.1: *Popis knihoven pro práci s OpenGL.*

GLFW [6]	Multiplatformní knihovna pro vytváření oken s OpenGL *canvasem*.
GLEW [7]	Knihovna, která načítá rozšíření pro OpenGL.
GLM [8]	Matematické funkce pro účely 3D grafiky (počítání s vektory, maticemi, kvaterniony, geometrické transformace, efektivně naprogramované goniometrické funkce atd.). Příkazy této knihovny se volají s předponou `glm::`.
NanoVG [9]	2D vektorová grafika. Vykreslení textu, načtení obrázků, kreslení spline křivek atd. na OpenGL *canvas*.

2.1.1 TrackballControls

Třída `TrackballControls` je implementací rotace, přibližování a posouvání kamery v OpenGL pomocí myši. Po dlouhém hledání jsem našel tuto jedinou vhodnou implementaci. Základní myšlenkou je pohyb kamery po myšlené kouli se středem v bodě, který jsem nazval referenční (viz třída `ReferencePoint`). Výhodou je pohodlné ovládání. Je zde navíc zabráněno jevu *gimbal lock* známému z mechaniky gyroskopů použitím kvaternionů z knihovny GLM.

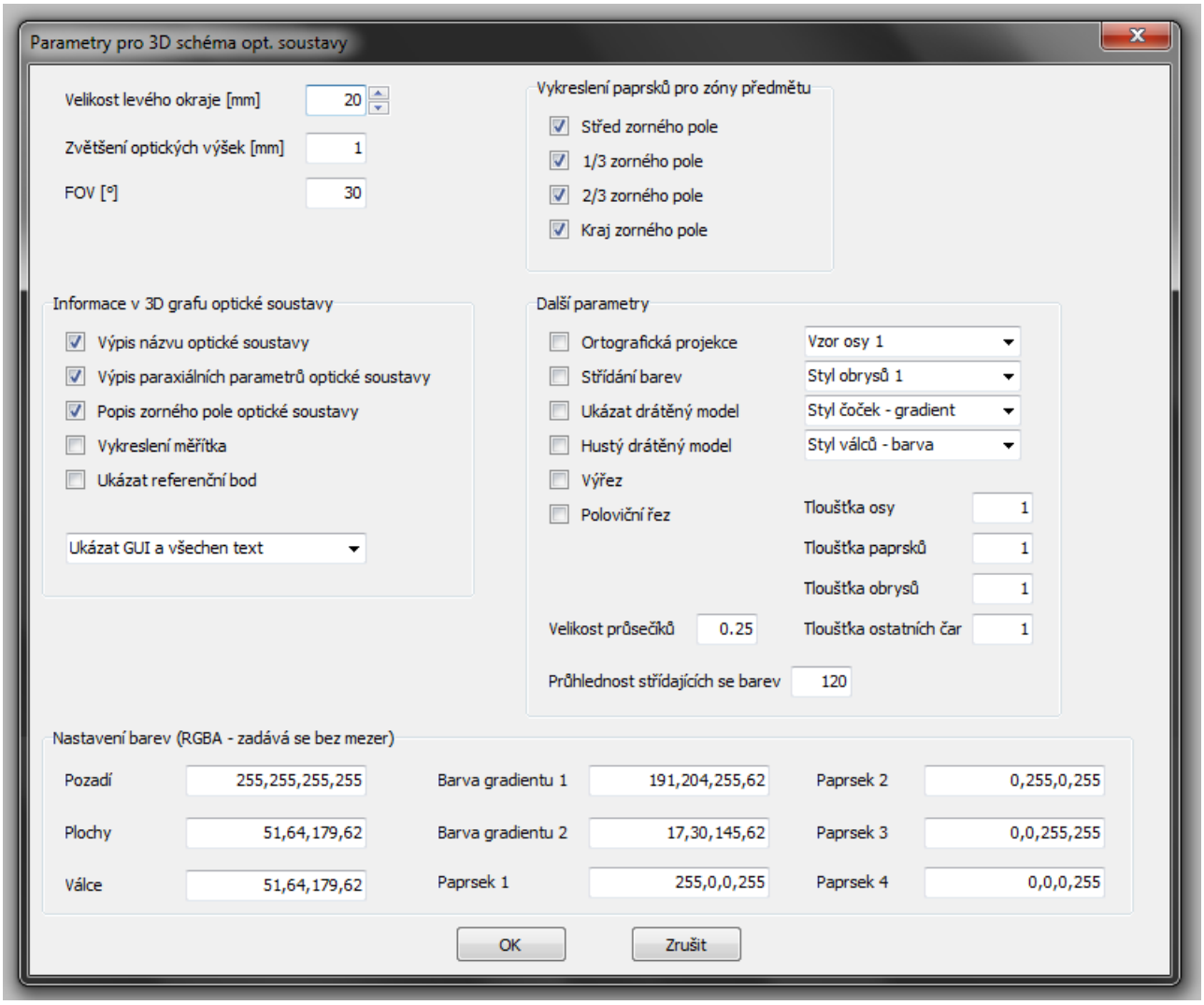
Na vývoji původním kódu jsem se osobně podílel. Původní kód je dostupný z [10]. V modulu OptVis je jeho upravená verze, do níž jsem navíc přidal ovládání pomocí klávesnice a možnost změny zorného pole (FOV) kamery.

Ovládání funkčnosti modulu je popsáno v podkapitole 2.4.

Zdrojové soubory (.cpp, .h), soubory projektu Code::Blocks spolu se spustitelnými soubory jsou k nalezení v příloze na CD-ROM.

2.2 Integrace modulu do programu pro návrh optických soustav Opt

Do programu Opt do menu nabídky „Grafika“ jsem přidal novou položku „3D schéma optické soustavy“, která otevře okno s formulářem, kde si uživatel zvolí parametry pro 3D schéma optické soustavy. Tyto volby jsou ekvivalentní těm popsaným v podkapitole Průvodce uživatele. Je možné je tedy měnit přímo za běhu v okně OptVis. Na obrázku 2.1 je ilustrace formulářového okna.



Obrázek 2.1: *Okno formuláře voleb parametrů 3D schématu optické soustavy.*

Stisknutím tlačítka „OK“ se provede příkaz `ShellExecute`, který spustí soubor OptVis.exe. Má-li uživatel běžící antivirový program, je možné, že zabrání spuštění a je nutné do antiviru přidat výjimku. `ShellExecute` přijímá šest parametrů:

- Reference na cílové okno. V našem případě, kdy mezi mezi hostitelským programem a modulem probíhá jednostranná komunikace, není odkaz potřeba a hodnotou je `NULL`.
- Specifikace akce, která má být provedena. Chceme okno otevřít.
- Úplná cesta k souboru.
- Řetězec znaků obsahující předávané parametry do souboru OptVis.exe. Přesné pořadí parametrů je detailně popsáno v komentáři u inicializační procedury modulu se jménem `init` ve zdrojovém souboru OptVis.cpp.
- Určení pracovní složky. Parametr `NULL` říká, že bude použita stávající složka.
- Specifikace způsobu zobrazení okna.

Příkaz vypadá takto: `ShellExecute(NULL, "open", "OptVis.exe", optVisParameters, NULL, SW_SHOW);`

2.3 Vnitřní struktura modulu

Součástí každého softwarového projektu středního a velkého rozsahu by měla být počáteční analýza. Je důležité si rozvrhnout zdroje – dobu trvání vývoje a stanovit si cíle, kterých má být dosaženo. V případě programování grafického uživatelského rozhraní nebo grafických aplikací je vhodné se držet následujících doporučení:

- Návrh kódu v UML (Unified Modeling Language), nebo alespoň dobře nadefinovaný vývojový diagram, který představuje mapu (schéma). Ta pomůže s orientací v kódu, plánováním programování a ušetří hodiny práce. Můj vývojový diagram, který shrnuje vnitřní logiku modulu je na obrázku 2.2.
- Projekt by měl mít strukturu Entity Component System (komponentový přístup). Každá funkce nebo procedura je jako černá skříňka, jejíž vnitřek není důležitý a nezávisí na ostatních komponentách. Důležitý je pouze vstup a výstup.

- Oddělení úloh – každá komponenta by měla mít jednu, dobře definovanou úlohu.
- Programovat podle návrhových vzorů.
- Dodržovat zvolenou konvenci při psaní kódu v celém projektu (např. [11]).
- V průběhu vývoje i na jeho konci je nezbytné provádět proces s názvem Unit testing – testování jednotlivých komponent, který pomáhá odhalovat nedokonalosti a systematické chyby.

Při návrhu softwaru jsem se inspiroval návrhovým vzorem Model-View-Controller (MVC). Vnitřní logika je tedy členěna do tří samostatných komponent. Model (model) zajišťuje převzetí dat od hostitelského programu a jejich zpracování. View (pohled) účelně prezentuje zpracovaná data a Controller (řadič) reaguje na události (vstupy z periferií počítače, časové spínače atd.) a provádí změny v modelu a pohledu. Do komponenty pohled jsem navíc přidal logiku, která vykresluje jednoduché grafické uživatelské rozhraní.

2.3.1 Významné pomocné procedury a funkce v modulu

`init` je výchozím bodem modulu. V těle procedury je funkčnost, která je znázorněná na obrázku 2.2. Přijímá celé číslo s počtem parametrů oddělených mezerou pro definování optické soustavy a nastavení modulu v textovém řetězci, který je druhým argumentem této procedury.

`callPrinter` volá funkci `displayPrintPropertySheet`, která otevře dialogové okno s volbou tiskárny a nastavením tisku.

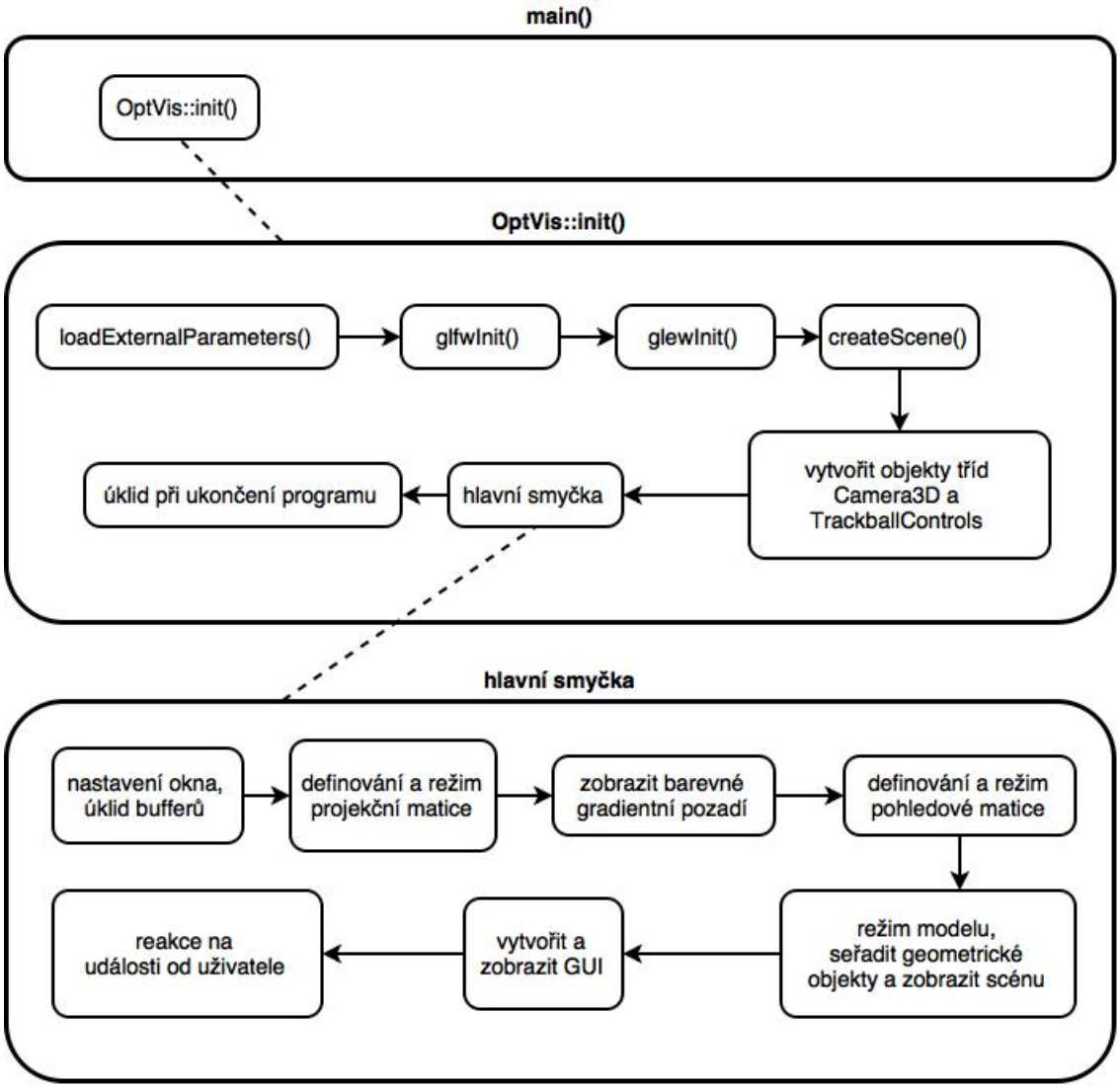
`createScene` vytvoří statickou scénu podle dat předaných modulu OptVis. Složí ji z objektů tříd odvozených od `Geometry`. Scénu je nutné překreslit v režimech s řezy.

`sortScene` řadí objekty na scéně v hlavní smyčce algoritmem `std::sort`. Pracuje s pomocí funkce `compare` s body, které jsem nazval `gravityCenter` (česky těžiště) geometrických útvarů definující polohu útvarů, a polohou kamery Camera3D. Procedura má svá omezení, která zmiňuji v podkapitole 2.5.

Zpětné volání `keyCallback` reaguje na klávesové vstupy. Tato funkčnost je zprostředkována knihovnou GLFW.

GUI procedury mého jednoduchého grafického uživatelského rozhraní: `createButtons` vytváří tlačítka, `showText` a `showScaleText` vykreslují text. Tyto procedury využívají knihovnu NanoVG pro kreslení 2D vektorové grafiky.

Ústřední částí interaktivního OpenGL programu je hlavní smyčka (anglicky main loop nebo render loop). Uvnitř ní se děje vykreslování scény v reálném čase. Je tedy zdrojově nejnáročnější částí.



Obrázek 2.2: *Vývojový diagram ukazuje logiku modulu OptVis.*

2.3.2 Významné třídy v modulu

V této části píši o nejvýznamnějších třídách, které jsem implementoval, včetně tříd `TrackballControls` a `Camera3D`, které jsem převzal z [10] a přizpůsobil.

`TrackballControls`

Třída ovládání *trackballem* přijímá parametry: ukazatel na objekt `Camera3D` a vektor `glm::vec4` obsahující rozměry okna. Reaguje na události vyvolané klávesnicí a myší pomocí procedur s názvy `mouseButtonCallback`, `mouseMoveCallback`, `mouseScrollCallback` a `keyCallback`, které zprostředkovává knihovna GLFW. Dále posílá zpracovaná data o změně polohy ve scéně kameře.

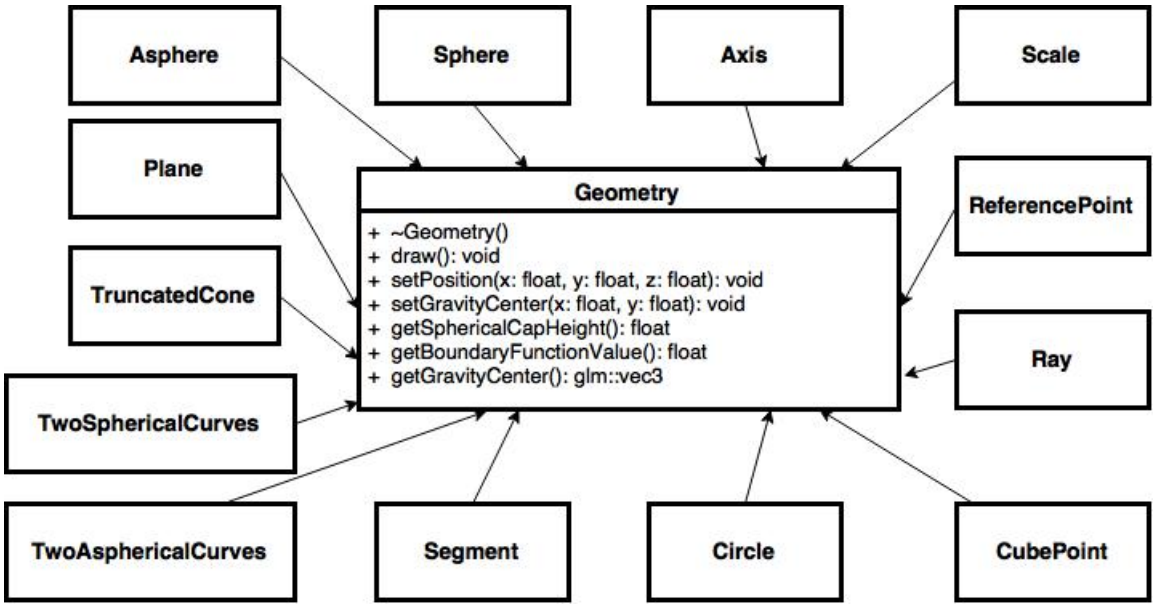
`Camera3D`

Třída vytvoří svůj objekt přijetím parametru umístění kamery ve scéně. Jejím úkolem je aktualizovat pohledovou matici v hlavní smyčce.

`Geometry`

Virtuální třída, která zobecňuje jednotlivé geometrické útvary uvedené dále. Tento vztah je k vidění na obrázku 2.3. Na jednoduchém diagramu jsou také zmíněny návratové hodnoty metod. Uvádím zde stručný popis chování metod třídy `Geometry`:

- Destruktor `~Geometry` ruší objekty této třídy.
- `draw` vykreslí objekt (v hlavní smyčce).
- `setPosition` nastaví umístění objektu ve scéně.
- `setGravityCenter` nastaví umístění těžiště v rovině x–y externě. (Šlo by udělat elegantnějším způsobem.)
- `getSphericalCapHeight` vypočítá a vrací výšku kulového vrchlíku lámavé plochy.
- `getBoundaryFunctionValue` vrací krajní funkční hodnotu funkce definující asféru a křivky na asféře.
- `getGravityCenter` vypočítá a vrací těžiště.



Obrázek 2.3: *Geometrické třídy.*

`Sphere`

Třída sférické plochy. Přijímá následujících pět parametrů: poloměr, optickou výšku, rozlišení v souřadnici šířky a délky a celé číslo, které určí barvu plochy v režimu střídajících se barev optických ploch. Všechna čísla určující rozlišení jsou datového typu `int`.

`Asphere`

Asférická plocha se zadává pomocí čtrnácti parametrů: poloměru, optické výšky, kónické konstanty (viz tabulka 1.2), osmi koeficientů se sudým indexem `alpha2` až `alpha16`, rozlišení v souřadnici šířky a délky a celého čísla, které určí barvu plochy v režimu střídajících se barev optických ploch.

`Plane`

Rovinná plocha se zadává pomocí tří parametrů: optické výšky, „obvodového rozlišení“ a celého čísla, které určí barvu plochy v režimu střídajících se barev optických ploch.

`TruncatedCone`

Části komolého kužele tvoří plášť, který opisuje čočky a přijímají šest parametrů: optické výšky č.1 a č.2, výšku kužele, rozlišení v souřadnici výšky kužele a „obvodové rozlišení“ a číslo od 1 do 4, které určí, ve kterém kvadrantu roviny kolmé na optickou osu se má část

komolého kužele objevit. Komolý kužel jsem rozřezal kvůli realizaci vizualizace čoček s výřezem a kvůli problémům s průhledností (viz podkapitola 2.5).

`TwoSphericalCurves`

Dvě křivky navzájem otočené o 90° ležící na sférické ploše. Zadáme je poloměrem nosné sférické plochy, optickou výškou a rozlišením v souřadnici šířky. Tvoří část kontur ke zvýraznění čoček.

`TwoAsphericalCurves`

Dvě křivky navzájem otočené o 90° ležící na asférické ploše. Zadáme prvních dvanáct parametrů z `Asphere`. Tvoří část kontur ke zvýraznění čoček.

`Segment`

Třída úsečka. Prvních šest parametrů jsou souřadnice počátečního a koncového bodu. Dále uvedeme pravdivostní hodnotu, jestli leží na komolém kuželi a celé číslo od 1 do 4 určující kvadrant. Neleží-li na komolém kuželi, zadáme libovolné celé číslo. Tento pravdivostní test je použit při tvorbě vizualizace v proceduře `createScene`. Tvoří část kontur ke zvýraznění čoček.

`Circle`

Třída kružnice přijímá poloměr a „obvodové rozlišení“. Tvoří část kontur ke zvýraznění čoček.

`Axis`

Třída optické osy. Velikost vzoru reaguje na vzdálenost kamery od osy.

`Scale`

Třída měřítko. Měřítko je zde ve formě milimetry ohodnocené mřížky. Velikost číslic reaguje na vzdálenost kamery od mřížky.

`ReferencePoint`

Třída referenčního bodu, který pomáhá s orientací ve scéně. Přijímá souřadnice pozice `glm::vec3`.

`Ray`

Třída paprsek. Přijímá dva argumenty – posloupnost poloh průsečíků s optickými plochami `vector<glm::vec3>` a vektor barvy RGBA `glm::vec4`.

`CubePoint`

Zvýrazňuje průsečíky paprsků s plochami přidáním bodu do scény tvaru kostky.

`Annulus`

Třída mezikruží. Ve scéně představuje clonu. Dvěma argumenty jsou vnější a vnitřní poloměr.

2.4 Průvodce uživatele

Okno modulu se ovládá klávesnicí a myší. Tabulka 2.2 obsahuje výčet efektů po stisku kláves a tabulka 2.3 seznamuje s pohybem po scéně.

Tabulka 2.2: *Popis ovládání modulu klávesnicí.*

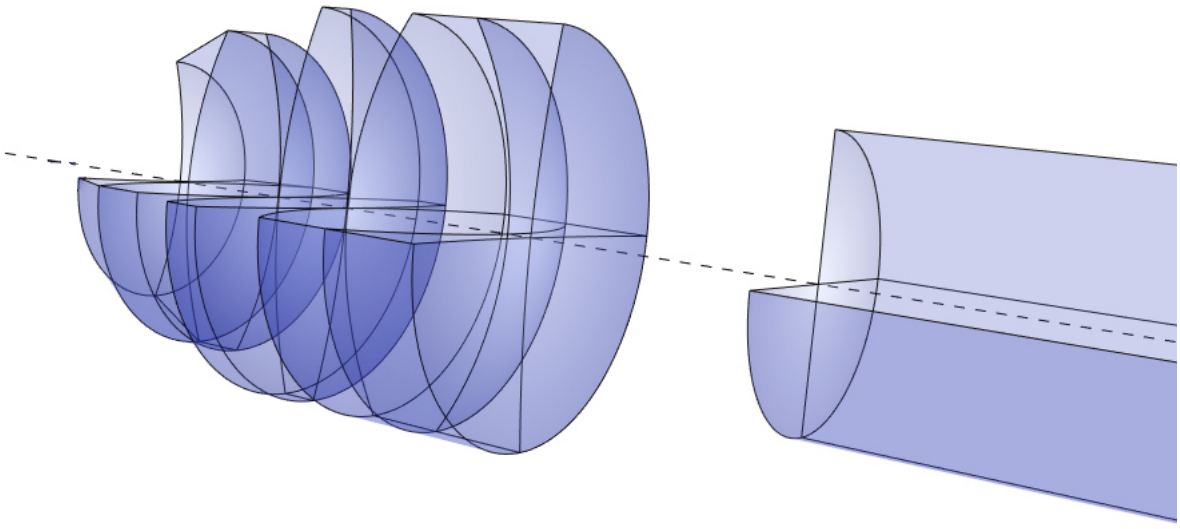
Ovládací prvek:	Funkce:
A	Vzory optické osy – 1–4
R	Ukázání / skrytí referenčního bodu, okolo kterého rotuje kamera, s popisem prostorových os
M	Ukázat / skrýt měřítko (mřížku)
D	Ukázat / skrýt drátěný model
H	Hustý / řídký drátěný model
K	Styly kontur – 1–4
T	Tloušťky všech čar – 1–3
O	Ortografická / perspektivní projekce
P	Ukázat / skrýt průchod paprsků s jejich průsečíky s optickými plochami
B	Barevná pozadí – barevné / šedé / bílé
S	Styly optických ploch – gradient / barva / šedý / prázdný
X	Styly plášťů komolého kužele opisujících čočky – barva / šedý / prázdný
V	Vizualizace čoček s / bez výřezu
U	Vizualizovat / skrýt řez tangenciální rovinou
G	Skrýt GUI a ponechat ohodnocení měřítka / skrýt GUI a veškerý text / ukázat GUI a všechen text
C	Zapnout / vypnout střídání barev optických ploch
Tlačítko „Tisk“	Volání tiskárny pro tisk plátna (OpenGL *canvas*)
Tlačítko „OK“, ESC	Ukončení modulu

Tabulka 2.3: *Popis ovládání pohybu ve scéně.*

Ovládací prvek:	Pohyb ve scéně:
Levé tlačítko myši	Rotace kamery okolo referenčního bodu
Kolečko myši	Změna FOV (zoomování), nebo pohyb kamery od/k referenčnímu bodu
Mezerník	Přepínač účelu kolečka myši: změna FOV (zoomování) / pohyb kamery od/k referenčnímu bodu
Šipka nahoru	Jít dopředu
Šipka dolů	Jít dozadu
Pravé tlač. myši, šipka doprava	Jít doprava
Pravé tlač. myši, šipka doleva	Jít doleva
Pravé tlač. myši, PAGE UP	Jít nahoru
Pravé tlač. myši, PAGE DOWN	Jít dolů
HOME	Posun měřítka
END	Posun měřítka v opačném směru
DELETE	Rotace měřítka o 90° okolo různých os

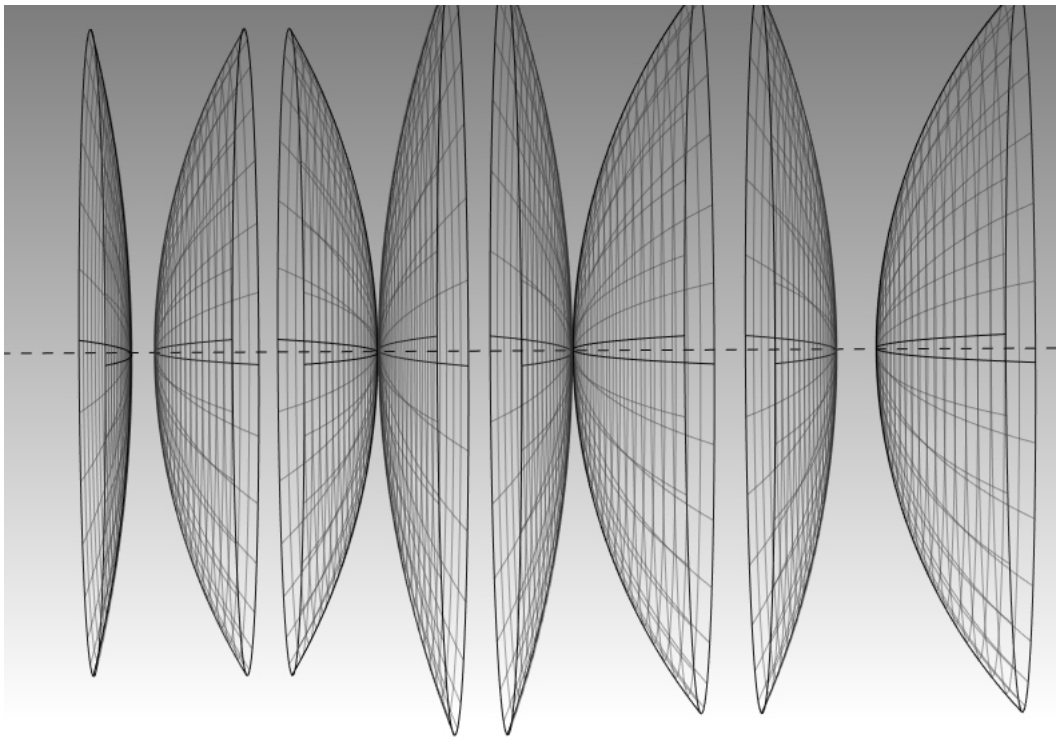
2.5 Aplikace v praxi a budoucí vývoj modulu

Modul OptVis se bude moci využívat jako součást kteréhokoli hostitelského optického programu, který poskytne dostatečná data o optické soustavě. Hotová je integrace do programu Opt I. Vyšína. Na obrázcích 2.4 – 2.7 prezentuji ukázky několika optických soustav.



Obrázek 2.4: *Optická soustava v režimu vizualizace čoček s výřezem.*

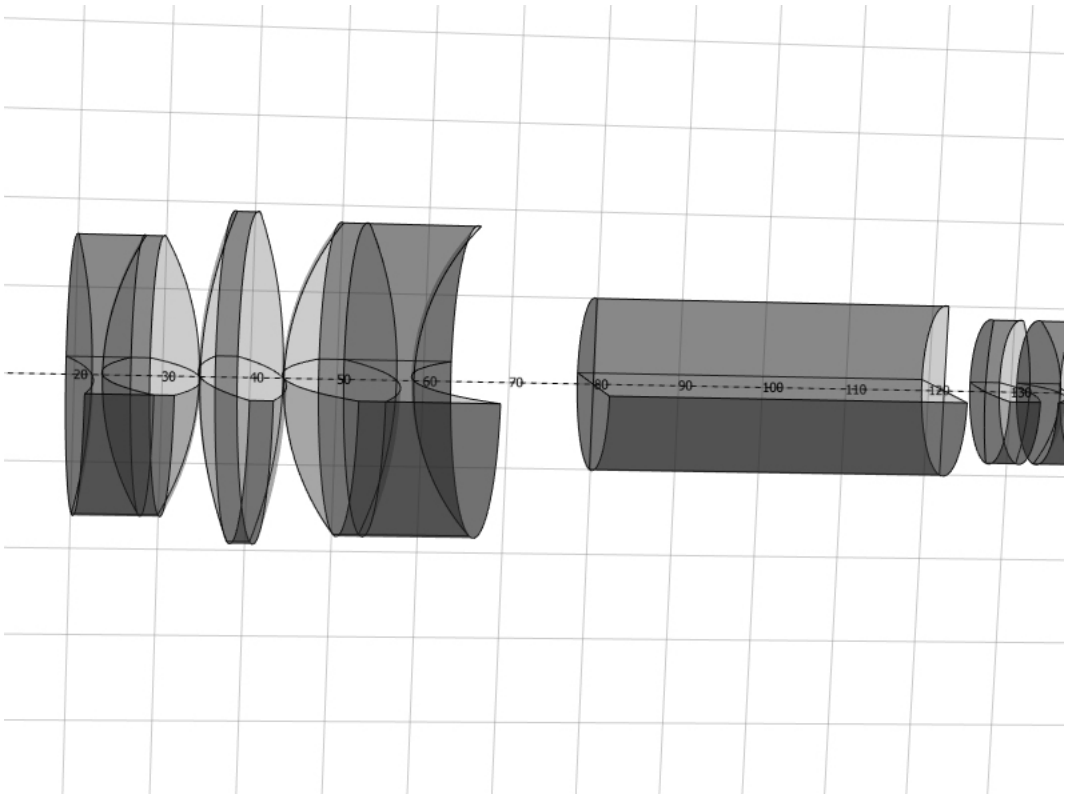
Při vykreslování soustav jsem narazil na jediný problém, který je velmi těžko řešitelný. Jde o drobné grafické artefakty špatně zobrazených průhledných barev, které jsou výrazné pouze v režimu střídajících se barev optických ploch, nebo když mají optické plochy a pláště komolých kuželů opisujících čočky velmi odlišné barvy. Jsou způsobeny zapnutím průhlednosti v OpenGL scéně. Tradiční metoda míchání kanálu alfa odpovědného za průhlednost je závislá na pořadí polygonů odeslaných do OpenGL. Polygony geometrických objektů se



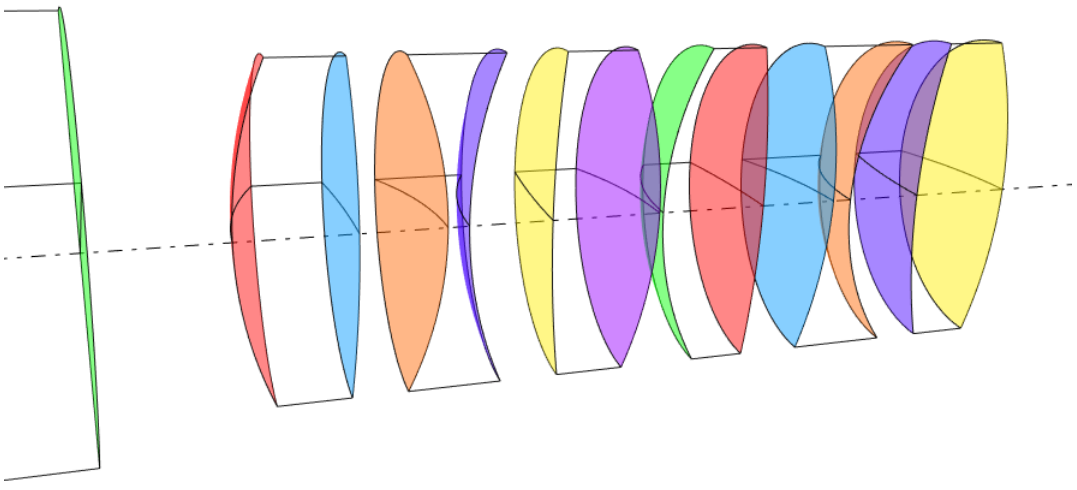
Obrázek 2.5: *Optická soustava v režimu drátěného modelu.*

občas v `sortScene` setřídí v jiném pořadí než „dále od kamery – blíže ke kameře“ a při různých natočeních soustavy je možné vidět barvu polygonu, která by měla být skrytá za polygonem blíž ke kameře. Mohl bych sice setřídit každý polygon ve scéně, ale to by extrémně zatížilo CPU. V současné době však probíhá rapidní vývoj ve vědecké vizualizaci. Existují nové techniky vystupující pod názvem *order independent transparency*, které dokáží ve velké míře redukovat výskyt grafických artefaktů způsobených průhledností, a u kterých není nutné seřazovat polygony ve scéně. Každá má svá pro a proti a s budoucími GPU, která budou implementovat flexibilnější hardwarovou architekturu, budou grafické aplikace realisticky vizualizovat průhledné plochy.

Při psaní softwaru jsem se maximálně snažil držet tzv. modulárního stylu, který klade důraz na dělení funkčnosti softwaru do nezávislých komponent, což umožňuje snadnou budoucí rozšířitelnost (např. o další optické prvky a grafické funkce) a změny v kódu.



Obrázek 2.6: *Optická soustava s měřítkem.*



Obrázek 2.7: *Optická soustava v režimu střídajících se barev optických ploch.*

Závěr

Práce je rozdělena na dvě části. V první části věnované teorii uvádím čtenáře do paraxiálního prostoru optických soustav a představuji postup počítání průchodu světelného paprsku tímto prostorem. Uvádím důležité fyzikální veličiny a jak se počítají. Následuje text obsahují Federovy algoritmy pro sledování obecně mimoběžného paprsku přes sférické a asférické optické plochy v mimoparaxiálním prostoru. Asférické plochy jsou reprezentovány rovnicí vhodnou pro účely výpočtu, která obsahuje kónickou konstantu, jejíž použití je vysvětleno v tabulce 1.2. Rovnice jsou doplněny ilustracemi.

Popis praktické části seznamuje čtenáře s technologiemi použitými při vývoji modulu 3D vizualizace optických soustav OptVis. Následuje popis integrace do hostitelského programu Opt. Tato integrace mi pomohla modul zdokonalit. Modul je začlenitelný do kteréhokoli optického programu, který poskytne potřebná data. Popis datového přenosu je v komentáři u procedury `init` ve zdrojovém souboru OptVis.cpp. Dále vysvětluji postup při návrhu softwaru a doporučení při programování. Modul je napsán zčásti procedurálním stylem a zčásti objektově orientovaným stylem. Dále uvádím výčet důležitých procedur a tříd, které tvoří základ modulu OptVis. Průvodce uživatele seznamuje s ovládáním.

Během více než ročního vývoje jsem se seznámil s oblastmi softwarového návrhu a objektového programování v C++ (naučil jsem se dodržovat doporučené programovací konvence používané vývojářskou komunitou C++ a používat programátorské postupy jako např. návrhové vzory). Dále jsem se seznámil s pokročilejšími tématy programování 3D grafiky (*pipelining*, míchání barev s alfa kanálem, rotace s kvaterniony).

Mým přínosem je napsání softwaru 3D interaktivní vědecké vizualizace, jehož kód obsahuje přes 3000 řádků, s kvalitním grafickým výstupem, rozmanitými režimy vizualizace a schopností začlenit se do kteréhokoli C++ projektu. Je tedy nástrojem použitelným jak v komerční praxi, tak ve výzkumu.

Vývoj modulu tímto nekončí. Je snadno rozšířitelný a v budoucnu pravděpodobně přibudou další optické prvky a režimy vizualizace a odstraním drobné grafické artefakty spojené s průhledností.

Slovník pojmů

aliasing – V teorii zpracování signálu a souvisejících disciplínách je aliasing efekt, který způsobuje, že se různé signály při vzorkování stávají nerozeznatelnými. Také se to vztahuje ke zkreslení nebo chybě, která vznikne, když se opravený signál ze vzorků liší od původního spojitého signálu. Aliasing může nastat mj. v prostorově vzorkovaných signálech (prostorový aliasing). Může se např. projevovat jako zubaté zobrazení čar a okrajů polygonů. Zmenšení aliasu se říká antialiasing.

buffer (vyrovnávací paměť) – Výstup vykreslovacího řetězce se uloží do framebufferu (paměť snímku). Hlavní část framebufferu tvoří několik samostatných bufferů: barvový buffer (color buffer), paměť hloubky (Z-buffer), paměť šablony (stencil buffer) a akumulační buffer (accumulation buffer). Podobně jako jsou v bitmapách uloženy jednotlivé pixely, jsou ve framebufferu uloženy takzvané fragmenty, které představují průřez všemi buffery.

CAD, computer aided design – počítačem podporované projektování. Oblast IT, která zastřešuje širokou činnost navrhování.

canvas, scene graph (plátno) – je kontejner, který pojímá různé grafické prvky (křivky, tvary, text atd.).

CPU, central processing unit – centrální procesorová jednotka v počítači.

design pattern (návrhový vzor) – představuje v softwarovém inženýrství obecné řešení problému, které se využívá při návrhu počítačových programů. Návrhový vzor není knihovnou nebo částí zdrojového kódu, která by se dala přímo vložit do našeho programu. Jedná se o popis řešení problému nebo šablonu, která může být použita v různých situacích. Objektově orientované návrhové vzory typicky ukazují vztahy a interakce mezi třídami a objekty, aniž by určovaly implementaci konkrétní třídy. Algoritmy nejsou považovány za návrhové vzory, protože řeší konkrétní problémy a nikoliv problémy návrhu.

flowchart (vývojový diagram) – typ schématu, které slouží ke grafickému znázornění jednotlivých kroků algoritmu, pracovního postupu nebo nějakého procesu. Vývojový diagram obsahuje obrazce různého tvaru, navzájem propojené pomocí šipek. Obrazce reprezentují jednotlivé kroky, šipky tok řízení.

FOV, field of view (zorné pole) – je část prostoru, kterou je oko nebo optický přístroj schopen zachytit. Jeho číselné vyjádření se nazývá zorný úhel.

GCC, the GNU Compiler Collection – je sada překladačů vytvořená v rámci projektu GNU pro různé programovací jazyky. Stažení je možné z [12].

gimbal lock – je ztráta jednoho stupně volnosti v Kardanově závěsu, která nastane, když se osy dvou ze tří prstenců uspořádají do paralelní konfigurace.

GPU, graphic processing unit – grafický procesor v počítači.

graphical pipelining (grafické zřetězené zpracování, grafický vykreslovací řetězec) – Základní myšlenkou pipeliningu je rozdělení zpracování jedné instrukce mezi různé části procesoru a tím i dosažení možnosti zpracovávat více instrukcí najednou (paralelismus). Speciálně v GPU je to posloupnost kroků vedoucí k vytvoření 2D rastrové (bitmapové) reprezentace 3D scény.

GUI (grafické uživatelské rozhraní) – je typem rozhraní, které dovoluje uživatelům interagovat s elektronickými zařízeními pomocí grafických ikon.

kvaternion – číselná soustava, která rozšiřuje komplexní čísla. Vlastností kvaternionů je nekomutativita násobení.

linker (sestavovací program) – počítačový program, který jeden nebo více objektových souborů vygenerovaných překladačem spojí do jediného spustitelného souboru, souboru knihovny, nebo jiného objektového souboru (tzv. linkování).

OOP (objektově orientované programování) – programovací styl založený na použití následující koncepce: objekty, abstrakce, zapouzdření, skládání, delegování, dědičnost a polymorfismus.

order independent transparency (průhlednost nezávislá na pořadí) – je třída technik, které renderují průhlednost ve 3D scéně, která nepotřebuje seřazovat geometrii pro skládání alfa kanálů.

ortografická projekce – je azimutální ekvidistantní mapové zobrazení.

procedura, funkce, metoda – pojmenované posloupnosti příkazů. Procedura je obvykle datového typu `void`. Funkce obvykle vrací hodnotu zvoleného datového typu. Pojem metoda se vyskytuje u objektově orientovaného programování.

trackball, virtual trackball – emuluje pohyb kamery po virtuální kouli. 2D poloha kurzoru myši je zobrazena na virtuální kouli ve 3D. To dovoluje 3D manipulaci s parametry kamery.

transformace v OpenGL – se provádí pomocí matic 4x4. Obecná transformace pro zobrazení na obrazovku je složena ze tří matic: modelu, pohledu a projekční matice (MVP, Model View Projection). Projekční matice definuje obvykle perspektivní, nebo ortografickou

projekci. Matice pohledu je tvořena parametry kamery - umístění kamery, bod ve směru pohledu a natočení kamery (obvykle směr nahoru). Matice modelu v sobě zahrnuje translaci, rotaci, nebo změnu měřítka.

UML, Unified Modeling Language – je v softwarovém inženýrství grafický jazyk pro vizualizaci, specifikaci, navrhování a dokumentaci programových systémů.

Obsah slovníku je zpracován podle [13] a [14].

Seznam použité literatury a zdrojů

[1] VYŠÍN, Ivo, ŘÍHA, Jan. *Paprsková a vlnová optika: studijní modul*. 1. vyd. Olomouc: Univerzita Palackého v Olomouci, 2012, 123 s. ISBN 978-80-244-3334-9.

[2] HUDEČEK, Tomáš. *Návrh optické soustavy kompaktního provedení triedru 8 x 20*. Olomouc, 1997. Diplomová práce. Univerzita Palackého v Olomouci.

[3] FEDER, D. Optical Calculations with Automatic Computing Machinery. *Journal of the Optical Society of America*. 1951, **41**(9): 630-636. DOI: 10.1364/JOSA.41.000630. ISSN 0030-3941. Dostupné z: https://www.osapublishing.org/josa/abstract.cfm?uri=josa-41-9-630

[4] SMITH, Warren J. *Modern optical engineering: the design of optical systems*. 4. vyd. New York: McGraw Hill, 2008, 754 s. ISBN 978-0-07-147687-4.

[5] Code::Blocks. [online]. [cit. 2015-08-01]. Dostupné z: http://www.codeblocks.org

[6] GLFW – An OpenGL library. [online]. [cit. 2015-08-01]. Dostupné z: http://www.glfw.org

[7] The OpenGL Extension Wrangler Library. [online]. [cit. 2015-08-01]. Dostupné z: http://glew.sourceforge.net

[8] OpenGL Mathematics. [online]. [cit. 2015-08-01]. Dostupné z: http://glm.g-truc.net

[9] NanoVG. [online]. [cit. 2015-08-01]. Dostupné z: https://github.com/memononen/nanovg

[10] TrackballControls. [online]. [cit. 2015-08-01]. Dostupné z: https://github.com/sasmaster/TrackballControls

[11] C++ Programming Style Guidelines. [online]. [cit. 2015-08-01]. Dostupné z: http://geosoft.no/development/cppstyle.html

[12] GCC, the GNU Compiler Collection. [online]. [cit. 2015-08-01].
Dostupné z: https://gcc.gnu.org

[13] *Wikipedia.* [online]. [cit. 2015-08-01].
Dostupné z: https://en.wikipedia.org

[14] *Wikipedie.* [online]. [cit. 2015-08-01].
Dostupné z: https://cs.wikipedia.org

Příloha na CD-ROM

Obsah CD-ROM

/Code Blocks projekt/

- buttonPicture.png
- icon.ico
- main.cpp
- nanovg.c
- OptVis.cbp
- OptVis.cpp
- OptVis.depend
- OptVis.h
- OptVis.rc
- Readme.txt
- soustava.dat
- tahoma.ttf
- tahomabd.ttf
- TrackballControls.cpp
- TrackballControls.h

/Spustitelná ukázka OptVis/

- buttonPicture.png
- glfw3.dll
- opengl32.dll
- OptVis.exe
- soustava.dat
- tahoma.ttf
- tahomabd.ttf

/Samostatné zdrojové soubory/

- nanovg.c
- OptVis.cpp
- OptVis.h
- Readme.txt
- TrackballControls.cpp
- TrackballControls.h

/Kvalifikační práce/

- Bp_Stepan_Venos.pdf