

Gymnázium J. K. Tyla Hradec Králové

Rostislav Brož

Leden 2024



Vizualizace asteroidů a jejich světelných křivek

studentská odborná práce

Vedoucí práce: doc. Mgr. Josef Ďurech, Ph.D., Astronomický ústav UK

Prohlášení

Prohlašuji, že jsem tuto studentskou odbornou práci vypracoval/a samostatně pod dohledem vedoucího uvedeného na první straně. Všechny použité zdroje jsou uvedeny v seznamu zdrojů a informace z nich získané jsou v textu řádně označeny odkazem na zdroj. Souhlasím s tím, aby tištěná forma práce byla uchována na Gymnáziu J. K. Tyla a tam používána jako tištěný zdroj např. pro další studentské práce či pro prezentaci vzdělávání na GJKT.

V Hradci Králové dne

Podpis autora práce

Poděkování

Tímto bych rád poděkoval svému vedoucímu odborné práce, doc. Mgr. Josefu Ďurechovi, Ph.D. z Astronomického ústavu UK, za pomoc, cenné rady a přátelský přístup při psaní odborné práce.

Anotace

Motivací odborné práce je vytvoření programu pro vizualizaci asteroidů, který v komunitě astronomů chyběl. Produktem je program, který asteroidy zobrazuje šesti různými způsoby a počítá světelnou křivku. Pro přesný popis povrchu asteroidu používá program trojúhelníkovou síť, tři funkce rozptylu (Lambertovu, Lommelovu–Seeligerovu a Hapkeho) a počítá realistické stínění. V teoretické části práce se koncepty jako rozptyl světla a vlastnosti trojúhelníku vysvětlují, v praktické části jsou pak implementovány za pomoci programovacího jazyka Python a akcelerované grafické knihovny VisPy. Program díky tomu zvládne efektivně zobrazovat asteroidy se stovkami tisíc trojúhelníků, jako například asteroid (101955) Bennu.

Klíčová slova

Vizualizace, Python, asteroidy, planety, světelná křivka, knihovna VisPy, formát .obj, stínění, funkce rozptylu

Abstract

The motivation for this work is to create a program for visualization of asteroids, which is missing in the astronomical community. The product is a program that displays asteroids in six different ways and calculates the light curve. For an accurate description of the asteroid's surface, the program uses three scattering functions (Lambert's, Lommel–Seeliger's, and Hapke's) and calculates realistic shadowing. In the theoretical section of the work, concepts such as light scattering and triangle properties are explained. In the practical section, these concepts are then implemented using the Python programming language and the VisPy accelerated graphics library. These tools allow the program to handle hundreds of thousands of triangles and effectively display asteroids such as (101955) Bennu.

Keywords

Visualization, Python, asteroids, small bodies, light curve, VisPy library, .obj format, shadowing, scattering functions

Obsah

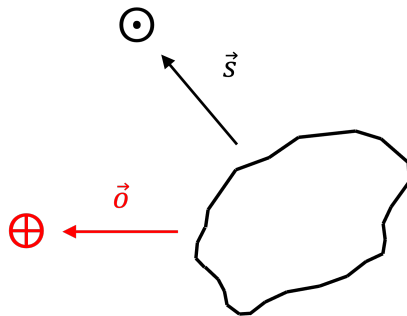
Úvod	6
1 Teoretické koncepty programu	7
1.1 Asteroidy	7
1.2 Triangulace tvaru	7
1.3 Vlastnosti trojúhelníku	8
1.4 Rozptyl světla	9
1.5 Světelná křivka	10
2 Technické řešení programu	12
2.1 Nástroje	12
2.2 Načítání souboru .obj	12
2.3 Třída Asteroid()	13
2.4 Výpočet fyzikálních vlastností	14
2.5 Výpočet světelné křivky	15
2.6 Vizualizace pomocí knihovny VisPy	16
2.6.1 Příklad trojúhelníku	16
2.6.2 Příklad asteroidu	18
2.7 Hlavní program	20
3 Produkt	22
3.1 Instalace	22
3.1.1 Instalace programovacího jazyka Python	22
3.1.2 Instalace knihovny NumPy	22
3.1.3 Instalace knihovny VisPy	23
3.1.4 Ostatní predispozice	23
3.1.5 Instalace samotné aplikace	23
3.2 Manuál	23
Závěr	26
Zdroje	27

Úvod

V naší sluneční soustavě obíhá kolem Slunce mnoho těles. Jedněmi z nich jsou asteroidy; kamenná tělesa podstatně menší než planety, avšak větší než rodinný automobil. Asteroidy (tzn. „slabé hvězdy“) na rozdíl od planet nebývají vidět na obloze pouhým okem. Až pohled dalekohledem dokáže odhalit jejich existenci a ty nejvýkonnější dokonce i jejich tvar. Při malém zvětšení se asteroidy jeví jen jako tečky, rozmazané chvěním zemské atmosféry. Na obloze je takových teček známo přes jeden milion [1].

Asteroidy nemají zcela pravidelný tvar, proto bývá obtížné vytvořit jejich přesné modely. Astronomové si za léta našli spoustu cest, jak se k přesnému tvaru dostat. Jedním ze způsobů je poslat sondu k asteroidu a podívat se zblízka [2]. Pro pozorování těch větších nemusíme opouštět Zemi, potřebujeme ovšem velmi velké dalekohledy (např. VLT, Keck) [3]. Obě tyto cesty jsou nesmírně nákladné. Nejčastěji proto tak astronomové zdálky sledují, jak se asteroidy otáčejí, a pozorují jejich kolísající jasnost [4].

Asteroidy nesvítí samy, světlo jen odráží. Zdrojem tohoto světla je Slunce. Jeho paprsky přicházejí z určitého směru, většinou jiného, než ze kterého se díváme (obr. 1). Jedna strana asteroidu je tak zcela temná, a jedna zalitá světlem. Když se asteroid otáčí, mění se plocha osvětlené části, kterou pozorujeme. Tento jev se projevuje jako změna jasnosti. Změna jasnosti v závislosti na čase se nazývá *světelná křivka* [5].



Obrázek 1: Ukázka vektorů \vec{s} (směru ke Slunci) a \vec{o} (směru k pozorovateli) vůči asteroidu.

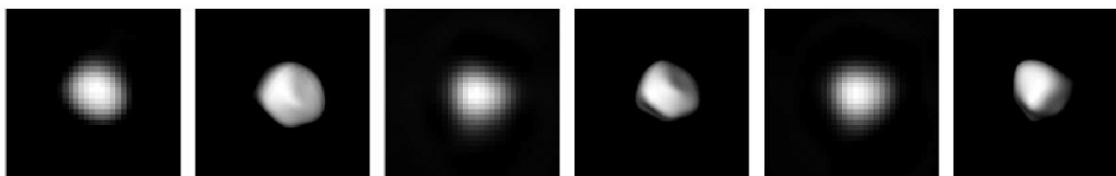
Ze světelných křivek byly odvozeny desetitisíce modelů asteroidů [4, 6]. Nástroje pro jejich vizualizaci sice existují, ale stávající programy jsou dosti jednoduché a pomalé. Cílem mé práce je vytvořit jednotný program, který asteroidy efektivně vizualizuje a kreslí jejich světelné křivky, za použití moderního programovacího jazyka a akcelerované grafické knihovny.

1 Teoretické koncepty programu

Teoretická část popisuje koncepty důležité pro hlubší pochopení programu. Všechny z nich jsou jeho nedílnou součástí a jeho fungování na nich závisí.

1.1 Asteroidy

Asteroidy jsou vesmírná tělesa, velká jednotky metrů až stovky kilometrů. Ty největší se od Země nachází typicky 1,5 astronomických jednotek daleko ($1 \text{ au} = 149,6 \text{ milionů kilometrů}$) a na obloze zabírají řádově jednu desetinu úhlové vteřiny ($100 \text{ km}/1,5 \text{ au} = 4,5 \cdot 10^{-7} \text{ rad} = 0,1''$). Pouhým okem tak není poznat že se jedná o asteroid. Jejich viditelnost ještě zhoršuje chvění vzduchu, které rozmazává obraz o zhruba jednu úhlovou vteřinu. Chvění vzduchu je jev způsobený zemskou atmosférou, ve které vrstvy vzduchu fungují jako malé čočky, lomící světelné paprsky.

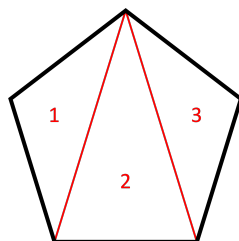


Obrázek 2: Obrázky asteroidu 354 Eleonora z adaptivní optiky a jeho modely. Převzato z [7].

Při pozorování asteroidů tedy musíme spoléhat na jiné přístroje než naše oči. Adaptivní optika dokáže korigovat chvění vzduchu a zvětšit vesmír před objektivem natolik, že jsou asteroidy rozpoznatelné (obr. 2). Pro pozorování těch malých ale ani adaptivní optika nestačí, a tak se astronomové musí spoléhat na jiné metody. Nejdetailnějšího popisu tvaru se dosáhne vysláním sondy přímo k asteroidu, která ho snímkuje z oběžné dráhy, a zaznamená tak i topografii povrchu.

1.2 Triangulace tvaru

Tvar asteroidů není možné zachytit dokonale přesně. Asteroidy jsou totiž tvarem nepravidelné, některé části mají oblé, jiné rovné, hladké, drsné — takový tvar je počítačem velmi těžké popsat. Svoji práci si usnadňujeme tím, že tvar popisujeme jen *zhruba*, a to rozdělením povrchu tělesa na menší a jednodušší části. Jednou částí může být trojúhelník, který je svým tvarem jednoduchý — lze pro něj například snadno vypočítat obsah.



Obrázek 3: Ukázka triangulace pětiúhelníku.

Procesu rozdělení povrchu na trojúhelníkové plošky se říká triangulace. S triangulací se můžeme setkat například při výpočtu obsahů složitějších útvarů. Ty se často dělí právě na soubor trojúhelníků, u kterých obsah spočítat umíme (obr. 3).

Trojrozměrný tvar asteroidu se v databázích ukládá jako soubor kartézských souřadnic vrcholů trojúhelníků a indexů, které určují, jaké tři vrcholy tvoří jeden trojúhelník. Jednou takovou databází je DAMIT [4]. Formát se nazývá .obj a je pro modelování tvarů asteroidů tím nejrozšířenějším.

1.3 Vlastnosti trojúhelníku

Trojúhelníky v prostoru mají unikátní soubor vlastností. Všechny pro nás důležité vlastnosti jsou shrnuté v následujícím seznamu [8]. Obvod je:

$$o = a + b + c, \quad (1)$$

kde a , b , c označují délky stran.

Délka strany:

$$a = |\vec{B} - \vec{C}|, \quad (2)$$

kde \vec{A} , \vec{B} , \vec{C} označují vektory z počátku souřadné soustavy \vec{O} do příslušného vrcholu. Ostatní strany je možné spočítat cyklickou záměnou.

Součet úhlů:

$$\alpha + \beta + \gamma = 180^\circ. \quad (3)$$

Obsah:

$$S = \frac{av_a}{2}, \quad (4)$$

kde v_a je výška příslušná straně a . Alternativně:

$$S = \frac{1}{2}|(\vec{B} - \vec{C}) \times (\vec{C} - \vec{A})|, \quad (5)$$

kde operace vektorového součinu (\times) je definována jako:

$$\vec{a} \times \vec{b} = (a_2b_3 - a_3b_2; a_3b_1 - a_1b_3; a_1b_2 - a_2b_1). \quad (6)$$

Těžiště:

$$\vec{T} = \frac{1}{3}(\vec{A} + \vec{B} + \vec{C}). \quad (7)$$

Sevřený úhel:

$$\cos \gamma = \frac{(\vec{B} - \vec{C}) \cdot (\vec{A} - \vec{C})}{|\vec{B} - \vec{C}||\vec{A} - \vec{C}|}, \quad (8)$$

kde operace skalárního součinu (\cdot) je definována následovně:

$$\vec{a} \cdot \vec{b} = a_1b_1 + a_2b_2 + a_3b_3. \quad (9)$$

Ostatní úhly je možné spočítat cyklickou záměnou.

Normála:

$$\vec{n} = \frac{(\vec{B} - \vec{C}) \times (\vec{C} - \vec{A})}{|(\vec{B} - \vec{C}) \times (\vec{C} - \vec{A})|}, \quad (10)$$

kde jmenovatel zajiřtuje normalizaci na delku 1.

První směrový kosinus:

$$\mu_i = \vec{s} \cdot \vec{n}, \quad (11)$$

kde \vec{s} je vektor mířící ke Slunci.

Druhý směrový kosinus:

$$\mu_e = \vec{o} \cdot \vec{n}, \quad (12)$$

kde \vec{o} je vektor mířící k pozorovateli. Promítnutá plocha:

$$S' = \mu_e S, \quad (13)$$

kde S' je pro pozorovatele viditelná plocha.

Posunutí o vektor \vec{O}' :

$$\vec{A}' = \vec{A} + \vec{O}', \quad (14)$$

$$\vec{B}' = \vec{B} + \vec{O}', \quad (15)$$

$$\vec{C}' = \vec{C} + \vec{O}'. \quad (16)$$

Posunutí trojúhelníku o vektor \vec{O}' spočívá v posunutí všech jeho vrcholů.

Otočení okolo osy z o úhel ϕ :

$$A'_1 = A_1 \cos \phi - A_2 \sin \phi, \quad (17)$$

$$A'_2 = A_1 \sin \phi + A_2 \cos \phi, \quad (18)$$

$$A'_3 = A_3. \quad (19)$$

Ostatní vrcholy obdobně. Ostatní otočení také obdobně.

1.4 Rozptyl světla

V síti trojúhelníků má každý svoji vlastní míru osvětlení. Osvětlení je způsobené tokem světelného záření od Slunce (označovaný Φ_\odot). Hodnota toku se udává ve W m^{-2} a ve vzdálenosti Země dosahuje hodnoty 1361 W m^{-2} . Pro odlišné vzdálenosti se počítá jako:

$$\Phi = \frac{P}{4\pi r^2}, \quad (20)$$

kde P označuje zářivý výkon Slunce ($3,827 \cdot 10^{26} \text{ W}$) a r vzdálenost od Slunce.

To, jak trojúhelník je (či není) osvětlený závisí na úhlu, pod jakým paprsky světla dopadají. Rovnici pro výpočet toku na trojúhelník musíme upravit tak, aby tento úhel brala v potaz [9]:

$$\Phi_i = \Phi \mu_i, \quad (21)$$

kde μ_i je první směrový kosinus. Je-li první směrový kosinus záporný, rovná se Φ_i nule (trojúhelník je odkloněný). U extrémních úhlů může dojít ke stínění, které je způsobené nerovnostmi na povrchu. Jedná-li se o konvexní povrch, ke stínění nedochází.

Povrch asteroidu rozptyluje světelné paprsky do různých směrů — je vidět i pod úhlem různým od úhlu dopadu (obr. 4). Tento rozptyl je způsoben mikroskopickými nerovnostmi, od kterých se paprsky odrážejí. Intenzita I rozptýleného záření se obecně počítá jako:

$$I = f \Phi_i, \quad (22)$$

kde f je funkce rozptylu, která popisuje souvislost mezi dopadajícími a rozptýlenými paprsky. Funkcí rozptylu existuje celá řada. Vzájemně se mezi sebou liší složitostí a přesností popisu rozptylu. Nejjednodušší je Lambertova funkce rozptylu:

$$f = f_L, \quad (23)$$

kde f_L je konstanta, která charakterizuje materiál povrchu. Další, komplexnější funkcí je Lommelova–Seeligerova funkce rozptylu:

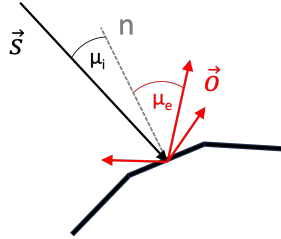
$$f = \frac{f_L}{\mu_i + \mu_e} \quad (24)$$

kde se konstanta dodatečně dělí součtem prvního a druhého směrového kosinu — tato funkce lépe popisuje preferovaný rozptyl paprsků do stran. Ještě přesněji by rozptyl popisovala Hapkeho funkce [10, 11].

Tok rozptýleného záření z trojúhelníku je potom:

$$\Phi_e = I\mu_e, \quad (25)$$

kde μ_e je druhý směrový kosinus.



Obrázek 4: Ilustrace rozptylu světla po dopadu na povrch asteroidu.

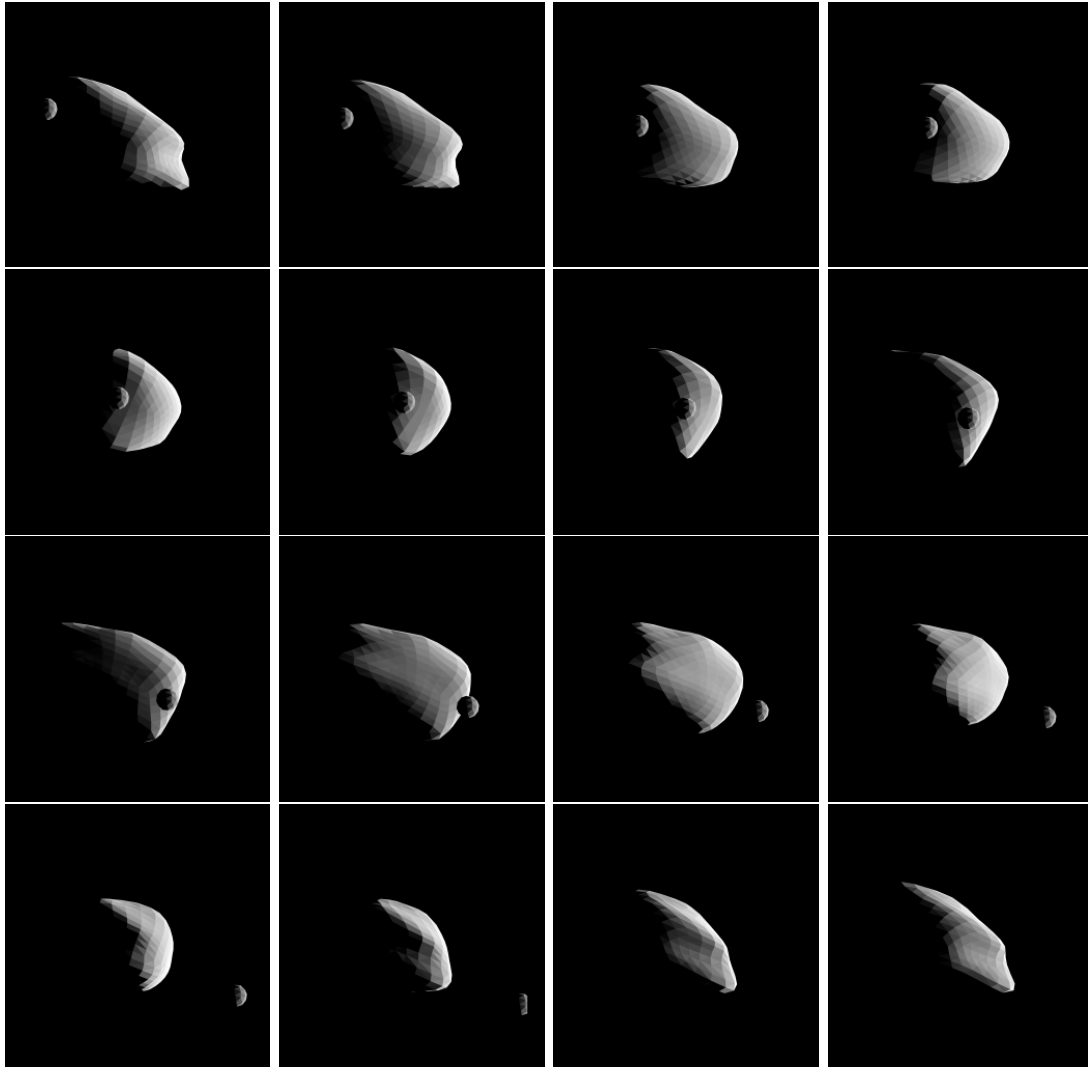
1.5 Světelná křivka

Když se po určitou dobu pozorují paprsky rozptýlené asteroidem, vznikne světelná křivka. Světelná křivka je tedy součtem toků přicházejících od jednotlivých trojúhelníků; matematicky vyjádřeno:

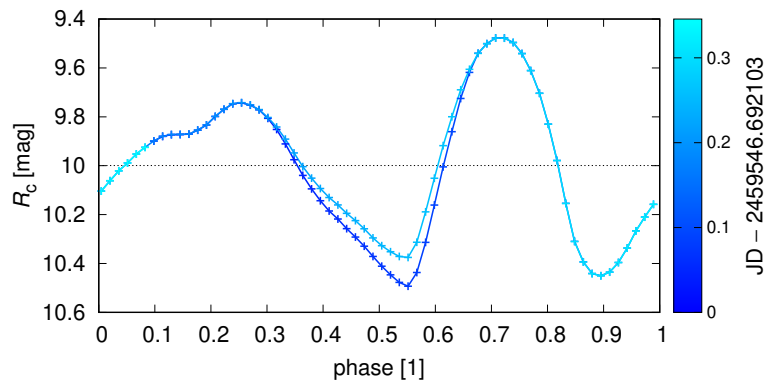
$$\Phi = \int \Phi_e(\vec{r})dS = \sum_j \Phi_e(j)dS_j, \quad (26)$$

kde trojúhelníky indexujeme písmenem j .

Otáčení asteroidu způsobuje na světelné křivce výkyvy (obr. 5, 6). Ty jsou znakem změn v odrazové ploše asteroidu. Společně s odrazovou plochou se mění i toky rozptýleného záření od jednotlivých trojúhelníků. Součet jejich hodnot je zaznamenaný v grafu světelné křivky na ose y ; na ose x je čas. Z grafu světelné křivky tak lze odvodit periodu otáčení asteroidu i jeho přibližný tvar.



Obrázek 5: Model asteroidu (22) Kalliope a jeho měsíce Linus. Převzato z [6].



Obrázek 6: Světelná křivka odpovídající obr. 5.

2 Technické řešení programu

Technické řešení popisuje zdrojový kód programu, použité nástroje a postup vývoje. Program je strukturovaný do čtyř souborů. Hlavního (`Main.py`), kde je uložena většina kódu aplikace, a tří vedlejších, podpůrných. `Load.py` se stará o načtení a zpracování `.obj` souboru, `Shadowing.py` počítá zákryt trojúhelníků a `Hapke.py` počítá složitou Hapkeho funkci rozptylu.

2.1 Nástroje

Celá aplikace je napsaná v programovacím jazyce Python3 [12]. Python je interpretovaný a objektově orientovaný jazyk, je v něm tedy lehké napsat přehledný a stručný program. Mimo jiného má vzhledem k jeho popularitě nejširší nabídku knihoven, na kterých lze aplikace podobné té naší stavět.

Aplikace je založena na knihovně VisPy [13], která se po vyzkoušení jiných jevila jako nejlepší. Jedná se o multiplatformní akcelerovanou knihovnu pro vizualizaci. Poskytuje přístup k funkcím výkonných grafických karet, a to pomocí připravených vysokoúrovňových objektů a metod.

2.2 Načítání souboru `.obj`

Jedním z prvních úkolů při vývoji je načítání souboru `.obj`. Ten slouží jako vstupní formát dat pro celou aplikaci. Soubor `.obj` obsahuje vrcholy a stěny 3D modelu. Řádky s vrcholy začínají písmenem `v` a skládají se ze tří dalších čísel; x , y , z souřadnic vrcholu. Ten samý soubor také obsahuje řádky začínající písmenem `f`. Následující čísla určují, jaké tři vrcholy tvoří jeden trojúhelník.

```
1 v 1.0000 0.0000 0.0000
2 v 0.0000 1.0000 0.0000
3 v 0.0000 0.0000 1.0000
4 f 1 2 3
```

Naším cílem je vytvořit jednoduchou funkci, která rozdělí data na listy samotných vrcholů a samotných stěn. Funkce bere jako argument cestu k souboru, zbaví data písmen na začátku řádků a rozliší řádky s vrcholy a stěnami. Využívá se systémové funkce `map()`, která zavolá určitou funkci pro každý prvek listu. Přitom je možné použít operátor `lambda`, který určitou proměnnou nahradí za výraz.

```
1 def load_obj(filename):
2
3     node = []
4     face = []
5
6     f = open(filename, "r")
7
8     for line in f.readlines():
9         if len(line) == 0:
10            continue
11        elif line[0] == '#':
12            continue
13
14        l = line.split()
15
16        if l[0] == 'v':
```

```

17         node.append(list(map(float, l[1:])))
18     elif l[0] == 'f':
19         face.append(list(map(lambda x: int(x) - 1, l[1:])))
20
21     f.close()
22
23     return node, face

```

2.3 Třída Asteroid()

Než definujeme třídu asteroidu, musíme importovat všechny knihovny, které bude program používat. Až do čtrnáctého řádku jde o knihovny volně dostupné na internetu. Další tři jsou mnou naprogramované knihovny. Knihovna Shadowing.py je kvůli rychlosti naprogramovaná v jazyce Fortran 90. Aby mohl Python Fortranovský soubor přečíst, používá knihovnu fmodpy.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import argparse
5  import fmodpy
6  import tkinter
7  import tkinter.filedialog
8  import numpy as np
9  import vispy
10 import vispy.app
11 import vispy.scene
12 import vispy.visuals
13 import vispy.io
14 import vispy.gloo
15
16 import Load
17 import Hapke
18
19 Shadowing = fmodpy.fimport("src/Shadowing/shadowing.f90")

```

Na začátku programu také definujeme funkce pro výpočet rozptylu světla. Lambertův anebo Lommelův–Seeligerův rozptyl je natolik jednoduchý, že se vejde na pár řádků. Funkce pro Hapkeho rozptyl [10, 11] má kolem sto padesáti řádků, je proto umístěna v separátním souboru.

```

20 def f_lambert(f_L, mu_i, mu_e, alpha):
21     return f_L
22
23 def f_lommel(f_L, mu_i, mu_e, alpha):
24     if mu_i + mu_e > 0.0:
25         return f_L / (mu_i + mu_e)
26     else:
27         return 0.0
28
29 def f_hapke(f_L, mu_i, mu_e, alpha):
30     return Hapke.f_hapke(f_L, mu_i, mu_e, alpha)

```

Třída Asteroid začíná konstruktorem, který bere jako parametry terminálové argumenty a cestu k souboru. Následně jsou definované proměnné instance třídy, čili objektu (self). Program načítá data asteroidu knihovnou Load, konkrétně její funkcí load_obj().

```

31 class Asteroid(object):
32     def __init__(self, args=None, filename=None):
33         self.args = args
34         self.filename = filename
35
36         self.vertices, self.faces = Load.load_obj(self.filename)
37         self.size = np.max(self.vertices) - np.min(self.vertices)
38         self.vertices *= 1.9 / self.size
39         self.f_func = f_lambert
40
41         self.get_geometry()
42         self.get_cosines()
43         self.get_fluxes()
44         self.plot()

```

2.4 Výpočet fyzikálních vlastností

Aby byl model asteroidu realistický, musí být založený na reálných výpočtech. Příslušná metoda `get_geometry()` počítá těžiště a normály trojúhelníků.

```

45     def get_geometry(self):
46         self.centers = []
47         self.normal_s = []

```

V každém cyklu metoda uloží vrcholy trojúhelníku do proměnných A, B a C. Spočítaná těžiště a normály se poté ukládají do předem definovaných listů.

```

48         for face in self.faces:
49             A = self.vertices[face[0]]
50             B = self.vertices[face[1]]
51             C = self.vertices[face[2]]
52
53             T = 1/3 * (A + B + C)
54             a = B - C
55             b = C - A
56             n = np.cross(a, b)
57             n /= np.sqrt(np.dot(n, n))
58
59             self.centers.append(T)
60             self.normal_s.append(n)
61
62         self.centers = np.array(self.centers)
63         self.normal_s = np.array(self.normal_s)

```

Metoda `get_cosines()` počítá směrové kosiny trojúhelníků a jejich stínění. Jako argumenty bere vektory \vec{s} (směr, ze kterého svítí Slunce) a \vec{o} (směr, ze kterého se dívá pozorovatel).

```

64     def get_cosines(self, s=(1, 0, 0), o=(0, 0, 1)):
65         self.s = np.array(s)
66         self.o = np.array(o)
67         self.alpha = np.arccos(np.dot(s, o))
68
69         mu_i = []
70         mu_e = []

```

První a druhý směrový kosinus každého trojúhelníku se ukládá ve `for` cyklu do listů, respektive numpyovských polí. Z polí jsou pak vyřazeny záporné hodnoty směrových kosinů;

jedná se totiž o trojúhelníky, které se nacházejí na odvrácené straně asteroidu, nebo nesměřují ke Slunci/pozorovateli.

```

71     for normal in self.normal_s:
72         mu_i.append(np.dot(s, normal))
73         mu_e.append(np.dot(o, normal))
74
75     self.mu_i = np.array(mu_i)
76     self.mu_e = np.array(mu_e)
77
78     self.mu_i = np.where(self.mu_i > 0.0, self.mu_i, 0.0)
79     self.mu_e = np.where(self.mu_e > 0.0, self.mu_e, 0.0)
80
81     self.nu_i = np.zeros((len(self.faces)))
82     self.nu_e = np.zeros((len(self.faces)))
83
84     Shadowing.shadowing_module.non(self.mu_i, self.mu_e, self.nu_i, self.
85     nu_e)
86     Shadowing.shadowing_module.nu(self.faces + 1, self.vertices, self.
87     normals, self.centers, self.s, self.nu_i)
88     Shadowing.shadowing_module.nu(self.faces + 1, self.vertices, self.
89     normals, self.centers, self.o, self.nu_e)

```

Pro výpočet příchozích (ϕ_i) a rozptýlených (ϕ_e) světelných toků používá program metodu `get_fluxes()`. Příchozí tok na trojúhelník se spočítá jako násobek toku od slunce (ϕ_s), prvního směrového kosinu a veličiny nu_i , která se v případě, že je trojúhelník zastíněný jiným trojúhelníkem, rovná nule, jinak jedné. Před výpočtem rozptýleného (odraženého) toku metoda nejprve provede for cyklus, kde do listu uloží hodnoty dle vybrané funkce rozptylu.

```

87     def get_fluxes(self):
88         phi_s = 1361. # W/m^2
89         self.phi_i = phi_s * self.mu_i * self.nu_i
90
91         f = []
92         A_w = 0.23
93         self.f_L = A_w/(4.0*np.pi)
94         for i in range(len(self.mu_e)):
95             f.append(self.f_func(self.f_L, self.mu_i[i], self.mu_e[i], self.
96             alpha))
97         self.f = np.array(f)
98
99         self.l = self.f * self.phi_i
100        self.phi_e = self.l * self.mu_e * self.nu_e
101
102        self.total = np.sum(self.phi_e)

```

Na konci metoda provede sumu rozptýlených toků. Ta se použije k výpočtu světelné křivky.

2.5 Výpočet světelné křivky

Ještě než program přejde k vizualizaci, spočítá světelnou křivku. Jako argument bere metoda `light_curve()` počet kroků, které na jedno otočení kolem osy udělá. U každého úhlu spočítá metoda celkový rozptýlený tok a uloží ho do listu `total`.

```

102     def light_curve(self, n=100):
103         s = self.s

```

```

104     x, y, z = s
105     total = []
106
107     for i in range(n+1):
108         gamma = 0 + 2.0*np.pi * i/n
109         print("gamma = ", gamma/np.pi*180.0, " deg")
110
111         x_ = x * np.cos(gamma) + y * np.sin(gamma)
112         y_ = -x * np.sin(gamma) + y * np.cos(gamma)
113         z_ = z
114
115         s_ = np.array([x_, y_, z_])
116
117         self.get_cosines(s=s_)
118         self.get_fluxes()
119         total.append((gamma, self.total))

```

List je pak převeden na numpyovské pole a uložen do textového souboru s názvem `light_curve.txt`. Aby se světelná křivka dala dobře vykreslit na obrazovku, musejí se její hodnoty normalizovat. Normalizované hodnoty se pak předají třídě `Line()` knihovny `VisPy`, která je vykreslí na obrazovku v podobě světelné křivky.

```

120     total = np.array(total)
121     np.savetxt("light_curve.txt", total)
122
123     total[:, 0] = (total[:, 0] - np.min(total[:, 0])) / (np.max(total[:, 0]) - np.min(total[:, 0]))
124     total[:, 1] = -(total[:, 1] - np.min(total[:, 1])) / (np.max(total[:, 1]) - np.min(total[:, 1]))
125
126     light_curve = vispy.scene.visuals.Line(pos=(total*200) + (40, 560),
127     color='white', parent=self.canvas.scene)

```

2.6 Vizualizace pomocí knihovny VisPy

2.6.1 Příklad trojúhelníku

Pro lepší pochopení toho, jak knihovna `VisPy` funguje, načteme nejdříve samotný trojúhelník a spočítáme jeho normálu. Abychom mohli knihovnu používat, musíme ji nejdříve do programu importovat. K práci se nám také bude hodit knihovna `NumPy` [14].

```

1 import numpy as np
2 import vispy.app
3 import vispy.scene

```

Začneme definováním funkce `main()`, která je volaná po spuštění programu. V ní zavedeme proměnou `canvas`, která reprezentuje okno aplikace, a nastavíme jeho velikost.

```

4 def main():
5     global view
6
7     canvas = vispy.scene.SceneCanvas(keys='interactive')
8     canvas.size = 1920, 1080
9
10    view = canvas.central_widget.add_view()

```

Vrcholy trojúhelníku `ABC` jsou definovány třírozměrnými souřadnicemi. Ty jsou zároveň obaleny funkcí `np.array()`, která převádí datový typ `list` na pole pro rychlé výpočty.

```

11 A = np.array((1, 0, 0))
12 B = np.array((0, 1, 0))
13 C = np.array((0, 0, 1))

```

Pro zobrazení normály trojúhelníku potřebujeme znát jeho těžiště. Těžiště se nachází v průsečíku těžnic. Ty jsou spojnicí vrcholu a středu protější strany. Střed strany a se spočítá jako absolutní hodnota rozdílu vektorů \vec{B} a \vec{C} . Ostatní středy se spočítají obdobně, jak lze vidět níže.

```

14 # Stredy stran
15 a = np.array(abs(B - C))
16 b = np.array(abs(C - A))
17 c = np.array(abs(A - B))
18
19 # Teziste a normala
20 T = 1.0/3.0 * (A + B + C)
21 n = np.cross(a, b)
22 n = n / np.sqrt(np.dot(n, n))

```

Všechny dosavadní kroky se odehrávaly pouze v paměti programu. Definované věci musíme ještě vykreslit na obrazovku. Vykreslování jednotlivých objektů je vidět v následujícím úryvku kódu:

```

23 # Trojuhelnik
24 vertices = np.array([A, B, C])
25 faces = np.array(((0, 1, 2)))
26 mesh = vispy.scene.visuals.Mesh(vertices=vertices, faces=faces,
27 vertex_colors=vertices)
28 mesh.transform = vispy.scene.transforms.MatrixTransform()
29 view.add(mesh)
30
31 # Vrcholy
32 markers = vispy.scene.visuals.Markers(pos=vertices, face_color='gray')
33 view.add(markers)
34
35 # Normala a popis
36 normal = vispy.scene.visuals.Line(pos=np.array([T, n]), color='white')
37 text = vispy.scene.visuals.Text(text="n", pos=T + (0.3, 0.3, 0.45), color=
38 'white', font_size=40)
39 view.add(normal)
40 view.add(text)
41
42 # Teznice
43 median1 = vispy.scene.visuals.Line(pos=np.array([A, a/2]), color='white')
44 median2 = vispy.scene.visuals.Line(pos=np.array([B, b/2]), color='white')
45 median3 = vispy.scene.visuals.Line(pos=np.array([C, c/2]), color='white')
46 view.add(median1)
47 view.add(median2)
48 view.add(median3)
49
50 # Nazvy vrcholu
51 for tmp, node in zip(("A", "B", "C"), vertices):
52     text = vispy.scene.visuals.Text(text=tmp, pos=node + 0.03, color='
53 white', font_size=40)
54     view.add(text)

```

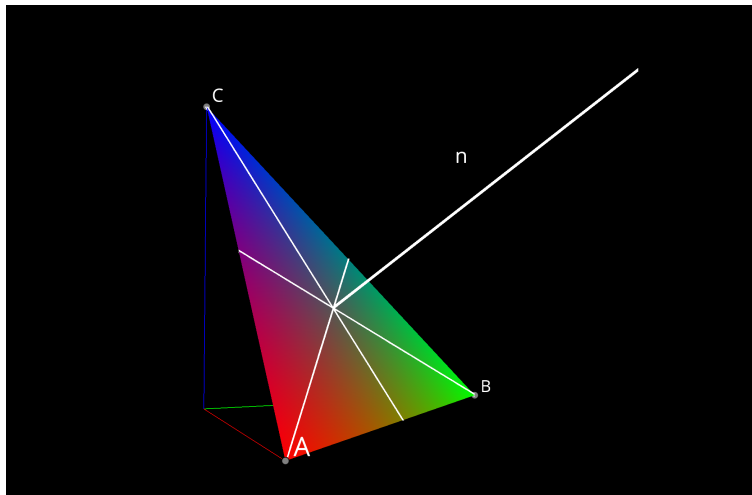
Nakonec se zvolí poloha kamery, přidají souřadnicové osy x , y , z , zobrazí připravený canvas a pomocí `vispy.app.run()` se předá řízení smyčce knihovny.

```

52 view.camera = vispy.scene.cameras.turntable.TurntableCamera(fov=30,
53 elevation=0.0, azimuth=0.0)
54 axis = vispy.scene.visuals.XYZAxis(parent=view.scene)
55
56 canvas.show()
57 vispy.app.run()
58
59 if __name__ == "__main__":
60     main()

```

Produktem popsaného programu je trojúhelník na obrázku 7. Je možné jej interaktivně ovládat.



Obrázek 7: Ukázka trojúhelníku zobrazeného knihovnou VisPy.

2.6.2 Příklad asteroidu

Knihovna VisPy zobrazuje model celého asteroidu obdobně jako trojúhelník samotný. Na začátku metody `plot()` definujeme okno programu, jeho velikost a obraz v něm.

```

114 def plot(self):
115     self.canvas = vispy.scene.SceneCanvas(keys='interactive')
116     self.canvas.size = 1920, 1080
117     self.view = self.canvas.central_widget.add_view()

```

Knihovna VisPy má pro síť spojených bodů zavedenou třídu `Mesh()`, která bere jako argumenty listy vrcholů, stěn a barev plošek. Nastavení parametru `color=` aplikuje vybranou barvu na všechny plošky.

```

118     mesh = vispy.scene.visuals.Mesh(self.vertices, self.faces, color='gray')
119     mesh.transform = vispy.scene.transforms.MatrixTransform()
120     self.view.add(mesh)

```

Metoda poté přejde k vizualizaci už spočítaných normál. Ty jsou vykreslené třídou `Line()`, které se předají souřadnice začátků a konců normál a informace o tom, který začátek a konec patří k sobě. List s těmito informacemi neumí knihovna VisPy vytvořit sama, je tak vytvořený

jednoduchým for cyklem. Viditelnost normál je jako výchozí nastavená na False; v základním zobrazení nejsou vidět.

```

121     pos = np.array([self.centers, self.centers + 0.1 * self.normals])
122     connect = []
123     n = len(self.centers)
124     for i in range(n):
125         connect.append(np.array([i, n + i]))
126
127     normals = vispy.scene.visuals.Line(pos=pos, connect=np.array(connect),
128         color='white', parent=self.view.scene)
129     normals.visible = False

```

V kódu jsou používány dva různé způsoby přidávání objektů na obrazovku, metoda self.view.add() a argument parent=self.view.scene. V tom, jak se objekty zobrazí, rozdíl není. Způsob self.view.add() se používá, když se jedná o objekt, který se dále upravuje (např. objekt normals, kterému se upravuje viditelnost). Aby byl program přehledný, vykreslují se na obrazovku vysvětlivky v podobě textu a významných vektorů.

```

129     s = vispy.scene.visuals.Line(pos=np.array([(0, 0.02, 0), self.s + (0,
130         0.02, 0)]), color='yellow', parent=self.view.scene)
131     o = vispy.scene.visuals.Line(pos=np.array([(0, 0.02, 0), self.o + (0,
132         0.02, 0)]), color='magenta', parent=self.view.scene)
133     vispy.scene.visuals.Text("'1' to show phi_i", anchor_x='left', pos
134         =(20, 20), font_size=10, color='white', parent=self.canvas.scene)
135     vispy.scene.visuals.Text("'2' to show phi_e", anchor_x='left', pos
136         =(20, 40), font_size=10, color='white', parent=self.canvas.scene)

```

Knihovna VisPy používá třídu ShadingFilter() pro vykreslování asteroidu Phongovou metodou (standardní metoda na vykreslování a nasvícení 3D objektu, která nepočítá stínění) a třídu WireframeFilter pro dodatečné vykreslování sítě.

```

182     shading_filter = vispy.visuals.filters.ShadingFilter(\
183         shading='smooth',
184         shininess=self.args.shininess, \
185         ambient_coefficient = 0.0, \
186         diffuse_coefficient = 1.0, \
187         specular_coefficient = 0.0, \
188         ambient_light = 'white', \
189         diffuse_light = 'white', \
190         specular_light = 'white', \
191     )
192     mesh.attach(shading_filter)
193
194     wireframe_filter = vispy.visuals.filters.WireframeFilter(\
195         width=self.args.wireframe_width, \
196         color='green', \
197         wireframe_only = False, \
198         faces_only = True, \
199         enabled = False, \
200     )
201     mesh.attach(wireframe_filter)

```

Funkce plot_fluxes() slouží jako zestručnění kódu, který by se jinak u ovládání klávesami mnohokrát opakoval. Funkce bere jako argument veličinu (phi) a vykresluje hrubý model asteroidu s viditelnými hranicemi sítě.

```

202     def plot_fluxes(phi):
203         shading_filter.shading = None
204         wireframe_filter.enabled = True
205         wireframe_filter.wireframe_only = False
206         wireframe_filter.faces_only = False
207         normals.visible = False
208         color = np.array([0.5, 0.5, 0.5]) / np.percentile(phi, 99)
209         face_colors = []
210         for face in phi:
211             face_colors.append(face * color)
212         mesh.set_data(self.vertices, self.faces, face_colors=face_colors)
213         mesh.update()

```

Wrapper `@self.canvas.events.key_press.connect` a funkce `on_key_press()` zajišťují fungování klávesového ovládání programu. Zde jsou na ukázkou uvedeny jen dvě klávesy.

```

214     @self.canvas.events.key_press.connect
215     def on_key_press(event):
216
217         if event.key in ['q', '0']:
218             vspy.app.quit()
219
220         elif event.key == '1':
221             plot_fluxes(phi=self.phi_i)
222             wireframe_filter.enabled = False
223         ...
224
225     self.canvas.show()

```

2.7 Hlavní program

Hlavní program, neboli funkce `Main()`, nejdříve definuje terminálové argumenty a uloží je do proměnné `args`.

```

298 def main():
299     parser = argparse.ArgumentParser()
300     parser.add_argument('--shininess', default=100)
301     parser.add_argument('--wireframe-width', default=1)
302     args, _ = parser.parse_known_args()

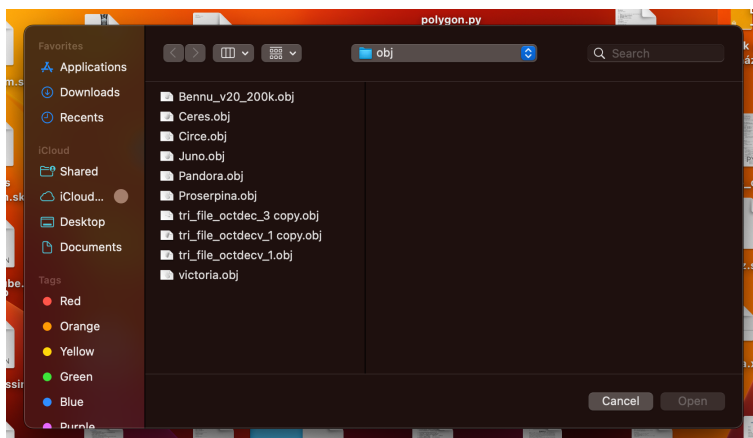
```

Program se po spuštění ptá uživatele, jaký asteroid chce zobrazit (obr. 8). K tomu používá knihovnu `Tkinter` a její metodu `.askopenfilename()`. Při výběru se může stát, že uživatel vybere špatný typ souboru. Program umí pracovat pouze se soubory `.obj`, ostatní jsou nepřijatelné. Kvůli tomu je výběr ošetřený funkcí `check filetype()`, která vrací hodnotu `True`, když se jedná o soubor `.obj`, a `False`, když má soubor jakoukoli jinou koncovku.

```

303     root = tkinter.Tk()
304     root.withdraw()
305     filename = tkinter.filedialog.askopenfilename()
306
307     def check_filetype():
308         for index, character in enumerate(filename):
309             if character == ".":
310                 if filename[index+1:] == "obj":
311                     return True
312                 else:
313                     return False

```



Obrázek 8: Ukázka výběru souboru za pomoci knihovny Tkinter.

Také se může stát, že uživatel nevybere žádný soubor. Program by pak měl skončit příslušným chybovým hlášením. Špatně vybraný i nevybraný soubor řeší dvojce podmínek.

```

314     if filename == '':
315         print("NO FILE WAS SELECTED")
316         vi_spy.app.quit()
317     elif check_filetype() == False:
318         print("NEEDS TO BE AN .OBJ FILE")
319         vi_spy.app.quit()

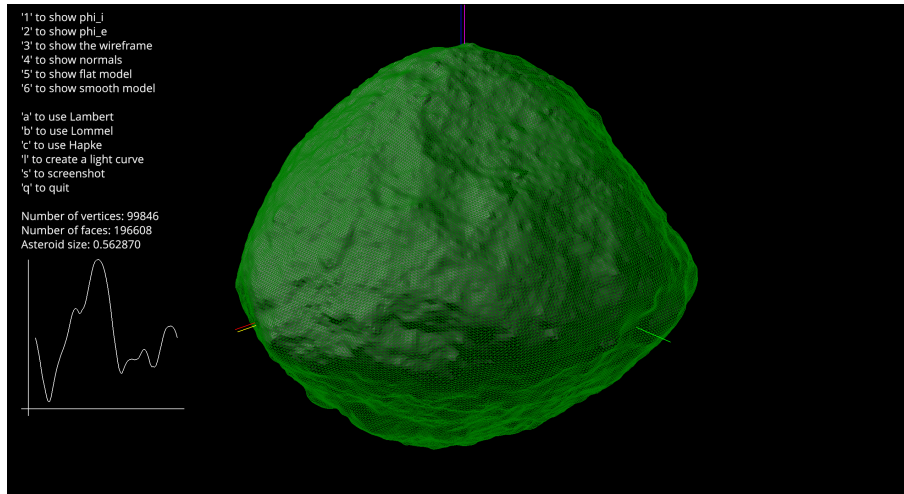
```

Nakonec funkce `main()` zavolá konstruktor třídy `Asteroid()`, s informací o terminálových argumentech a cestě k souboru, a předá řízení smyčce knihovny `VisPy`.

```

320     asteroid = Asteroid(args=args, filename=filename)
321     vi_spy.app.run()
322
323 if __name__ == "__main__":
324     main()

```



Obrázek 9: Ukázka výstupu programu, model asteroidu (101955) Bennu s Hapkeho funkcí rozptylu, vytvořený ze snímků pořízených sondou OSIRIS-REx, a jeho světelná křivka [15, 16].

3 Produkt

3.1 Instalace

Aby uživatel mohl aplikaci spustit, musí jeho počítač splňovat určité predispozice. Těmi jsou, mimo aplikace samotné, Python 3 [12], knihovna NumPy [14] a nejnovější verze knihovny VisPy [13].

3.1.1 Instalace programovacího jazyka Python

Nejnovější verze programovacího jazyka Python se nachází na domovské stránce, v menu Downloads (<https://www.python.org/downloads/>). Python je multiplatformní interpretovaný jazyk, čili funguje na více operačních systémech. Instalaci na MacOS provede uživatel stáhnutý soubor .pkg. U ostatních systémů funguje instalace obdobně. Úspěšnost instalace pak můžeme ověřit příkazem `python3 --version`, který by měl vrátit aktuální verzi nainstalovaného Pythonu.

3.1.2 Instalace knihovny NumPy

Další velice důležitou predispozicí pro fungování programu je knihovna NumPy. Tu program používá pro komplexní matematické operace, využívající takzvaná *numpyovská pole*, která zrychlují operace, zejména for cykly. Knihovnu NumPy, stejně jako další zmíněné knihovny, lze instalovat přes terminálový nástroj zvaný `pip`. Příkaz pro instalaci knihovny NumPy je uvedený níže.

```
1 pip install numpy
```

Informace o tom, jak nainstalovat terminálový nástroj `pip`, jsou dostupné na následující adrese, <https://pip.pypa.io/en/stable/installation/>.

3.1.3 Instalace knihovny VisPy

Knihovnu VisPy je lze nainstalovat příkazem uvedeným níže. Dokumentace knihovny VisPy a detailnější popis její instalace je dostupný na stránce [13].

```
1 pip install vispy
```

3.1.4 Ostatní predispozice

K dalším důležitým predispozicím patří knihovna Tk. Ta v programu umožňuje výběr souboru pro vizualizaci. Aby se mohl program ke knihovně Tk dostat, potřebuje modul zvaný tkinter. Ten je součástí pipu a lze jej nainstalovat příkazem níže.

```
1 pip install tk
```

Poslední knihovnou, která nesmí před spuštěním programu chybět, je PyOpenGL.

```
1 pip install pyopengl
2 pip install pyopengl tk
```

3.1.5 Instalace samotné aplikace

Nejaktuálnější verze zdrojového kódu je dostupná na stránce <https://github.com/scraptechguy/tvet>. Archiv .zip stačí rozbalit do adresáře.

3.2 Manuál

Uživatel spustí program kliknutím na soubor Main.py. Po spuštění, jak je vidět na obrázku 8, se aplikace zeptá na soubor .obj, který má zobrazit. Při vybrání žádného, nebo jiného souboru, než je .obj, program skončí s příslušnou chybovou hláškou.

Poté, co uživatel vybere platný soubor, se otevře hlavní okno programu. V něm uživatel uvidí zahlazený model asteroidu, nasvícený z jednoho směru (obr. 10). U tohoto zobrazení není počítané vzájemné stínění trojúhelníků.

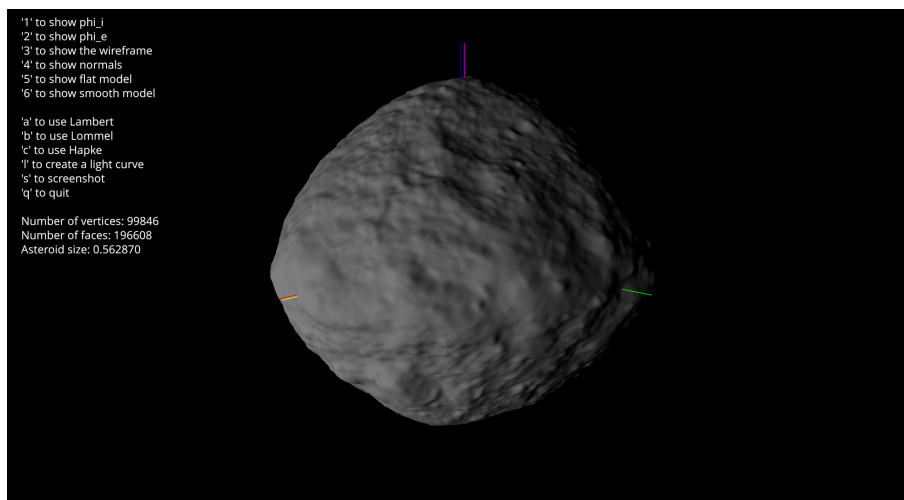
Kromě základního modelu má uživatel přístup k pěti dalším, jejichž seznam s popisky je uveden níže. Přepínat modely uživatel může příslušnými klávesami (číslicemi '1' až '6').

```
1 '1' - model, který ukazuje phi_i
2 '2' - model, který ukazuje phi_e
3 '3' - model, který ukazuje pouze sit asteroidu
4 '4' - model, který ukazuje normaly trojuhelni ku
5 '5' - hruby model asteroidu
6 '6' - hladky model asteroidu
```

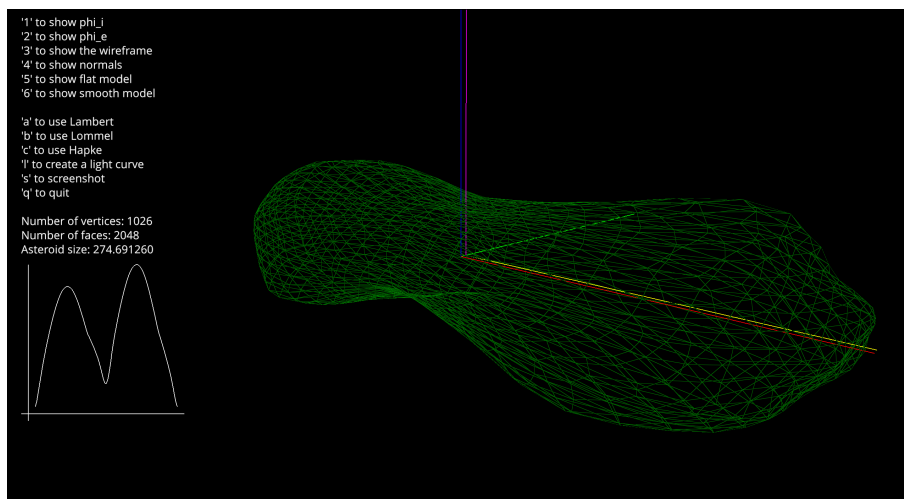
U druhého modelu se dá písmeny 'a', 'b' a 'c' volit funkce rozptylu. Další písmena pak umožňují dodatečné ovládání programu, jak je vidět v seznamu níže.

```
1 'a' - pouzije Lambertuv rozptyl
2 'b' - pouzije Lommeluv-Seeligeruv rozptyl
3 'c' - pouzije Hapkeho rozptyl
4 'l' - vypocita svetelnou krivku
5 's' - udel a snimek obrazovky
6 'q' - opusti aplikaci
```

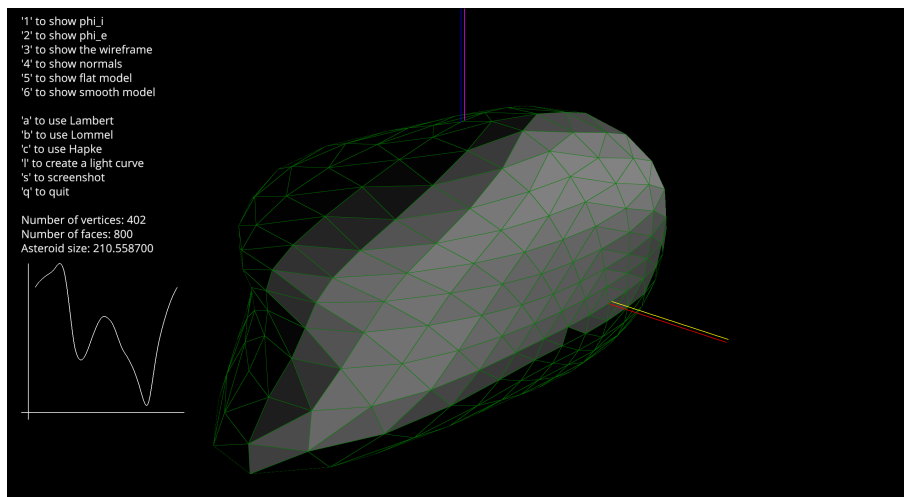
Nad světelnou křivkou také program vypisuje počet vrcholů asteroidu, počet jeho stěn a velikost modelu.



Obrázek 10: Model asteroidu (101955) Bennu, vytvořený ze snímků pořízených sondou OSIRIS-REx [15, 16]. Zobrazení klávesou '6'.



Obrázek 11: Síť modelu asteroidu (216) Kleopatra [17, 18]. Zobrazení klávesou '3'.



Obrázek 12: Model asteroidu (22) Kalliope s Hapkeho funkcí rozptylu [19, 6]. Zobrazení klávesami 'c' + '2'.

Závěr

Cílem práce bylo vytvořit jednotný program pro vizualizaci asteroidů a jejich světelných křivek, který v komunitě astronomů chyběl. Cíl práce se podařil, program je funkční a dostupný veřejnosti. Program umí zpracovat soubor `.Obj` a zobrazit jej šesti různými způsoby a spočítat světelnou křivku. Povrch asteroidů umí popsat třemi různými funkcemi rozptylu, mezi kterými může uživatel přepínat klávesami.

Celý program je psaný objektově, je rozdělený na třídu a její metody. Pro jiné vývojáře je tak velmi jednoduché program pochopit, dál na něm stavět a zlepšovat ho. Smysluplným rozšířením by bylo například napojení na databázi DAMIT [4], ve které se nachází tisíce modelů asteroidů, které by program mohl zobrazovat. Dále by bylo možné přidat načítání reálných efemerid z databáze JPL Horizons [20].

Zdroje

- [1] Minor Planet Center. <https://www.minorplanetcenter.net/iau/mpc.html>, 2023.
- [2] A. F. Cheng, A. S. Rivkin, P. Michel, J. Atchison, O. Barnouin, L. Benner, N. L. Chabot, C. Ernst, E. G. Fahnestock, M. Kueppers, P. Pravec, E. Rainey, D. C. Richardson, A. M. Stickle, and C. Thomas. AIDA DART asteroid deflection test: Planetary defense and science objectives. *Planet. Space Sci.*, 157:104–115, August 2018.
- [3] P. Vernazza, M. Ferrais, L. Jorda, J. Hanuš, B. Carry, M. Marsset, M. Brož, R. Fetick, M. Viikinkoski, F. Marchis, F. Vachier, A. Drouard, T. Fusco, M. Birlan, E. Podlowska-Gaca, N. Rambaux, M. Neveu, P. Bartzczak, G. Dudziński, E. Jehin, P. Beck, J. Berthier, J. Castillo-Rogez, F. Cipriani, F. Colas, C. Dumas, J. Ďurech, J. Grice, M. Kaasalainen, A. Kryszczyńska, P. Lamy, H. Le Coroller, A. Marciniak, T. Michalowski, P. Michel, T. Santana-Ros, P. Tanga, A. Vigan, O. Witasse, B. Yang, P. Antonini, M. Audejean, P. Aurard, R. Behrend, Z. Benkhaldoun, J. M. Bosch, A. Chapman, L. Dalmon, S. Fauvaud, Hiroko Hamanowa, Hiromi Hamanowa, J. His, A. Jones, D. H. Kim, M. J. Kim, J. Krajewski, O. Labrevoir, A. Leroy, F. Livet, D. Molina, R. Montaignut, J. Oey, N. Payre, V. Reddy, P. Sabin, A. G. Sanchez, and L. Socha. VLT/SPHERE imaging survey of the largest main-belt asteroids: Final results and synthesis. *Astron. Astrophys.*, 654:A56, October 2021.
- [4] J. Ďurech and J. Hanuš. Reconstruction of asteroid spin states from Gaia DR3 photometry. *Astron. Astrophys.*, 675:A24, July 2023.
- [5] B. D. Warner, A. W. Harris, and P. Pravec. The asteroid lightcurve database. *Icarus*, 202(1):134–146, July 2009.
- [6] M. Brož, J. Ďurech, M. Ferrais, H. J. Lee, M. J. Kim, D. G. Roh, H. S. Yim, E. Jehin, A. Burdanov, J. de Wit, P. Fatka, J. Hanuš, and B. Carry. 2021 occultations and transits of Linus orbiting (22) Kalliope. I. Polygonal and cliptracing algorithms. *Astron. Astrophys.*, 676:A60, August 2023.
- [7] M. Viikinkoski, J. Hanuš, M. Kaasalainen, F. Marchis, and J. Ďurech. Adaptive optics and lightcurve data of asteroids: twenty shape models and information content analysis. *Astron. Astrophys.*, 607:A117, November 2017.
- [8] K. Rektorys. *Přehled užité matematiky 1*. Prometheus, 2009.
- [9] M. Brož and M. Šolc. *Fyzika sluneční soustavy*. Matfyzpress, 2013.
- [10] B. Hapke. Bidirectional reflectance spectroscopy 3. Correction for macroscopic roughness. *Icarus*, 59(1):41–59, July 1984.
- [11] S. Spjuth. *Disk-resolved photometry of small bodies*. PhD thesis, Technical University of Braunschweig, Germany, January 2009.
- [12] Python. <https://python.org/>, 2023.
- [13] VisPy. <https://vispy.org/>, 2023.
- [14] NumPy. <https://numpy.org/>, 2023.

- [15] D. S. Lauretta, C. W. Hergenrother, S. R. Chesley, J. M. Leonard, J. Y. Pelgrift, C. D. Adam, M. Al Asad, P. G. Antreasian, R. L. Ballouz, K. J. Becker, C. A. Bennett, B. J. Bos, W. F. Bottke, M. Brozović, H. Campins, H. C. Connolly, M. G. Daly, A. B. Davis, J. de León, D. N. DellaGiustina, C. Y. Drouet d’Aubigny, J. P. Dworkin, J. P. Emery, D. Farnocchia, D. P. Glavin, D. R. Golish, C. M. Hartzell, R. A. Jacobson, E. R. Jawin, P. Jenniskens, J. N. Kidd, E. J. Lessac-Chenen, J. Y. Li, G. Libourel, J. Licandro, A. J. Liounis, C. K. Maleszewski, C. Manzoni, B. May, L. K. McCarthy, J. W. McMahan, P. Michel, J. L. Molaro, M. C. Moreau, D. S. Nelson, W. M. Owen, B. Rizk, H. L. Roper, B. Rozitis, E. M. Sahr, D. J. Scheeres, J. A. Seabrook, S. H. Selznick, Y. Takahashi, F. Thuillet, P. Tricarico, D. Vokrouhlický, and C. W. V. Wolner. Episodes of particle ejection from the surface of the active asteroid (101955) Bennu. *Science*, 366(6470):eaay3544, December 2019.
- [16] D. S. Lauretta, C. D. Adam, A. J. Allen, R. L. Ballouz, O. S. Barnouin, K. J. Becker, T. Becker, C. A. Bennett, E. B. Bierhaus, B. J. Bos, R. D. Burns, H. Campins, Y. Cho, P. R. Christensen, E. C. A. Church, B. E. Clark, H. C. Connolly, M. G. Daly, D. N. DellaGiustina, C. Y. Drouet d’Aubigny, J. P. Emery, H. L. Enos, S. Freund Kasper, J. B. Garvin, K. Getzandanner, D. R. Golish, V. E. Hamilton, C. W. Hergenrother, H. H. Kaplan, L. P. Keller, E. J. Lessac-Chenen, A. J. Liounis, H. Ma, L. K. McCarthy, B. D. Miller, M. C. Moreau, T. Morota, D. S. Nelson, J. O. Nollau, R. Olds, M. Pajola, J. Y. Pelgrift, A. T. Polit, M. A. Ravine, D. C. Reuter, B. Rizk, B. Rozitis, A. J. Ryan, E. M. Sahr, N. Sakatani, J. A. Seabrook, S. H. Selznick, M. A. Skeen, A. A. Simon, S. Sugita, K. J. Walsh, M. M. Westermann, C. W. V. Wolner, and K. Yumoto. Spacecraft sample collection and subsurface excavation of asteroid (101955) Bennu. *Science*, 377(6603):285–291, July 2022.
- [17] F. Marchis, L. Jorda, P. Vernazza, M. Brož, J. Hanuš, M. Ferrais, F. Vachier, N. Rambaux, M. Marsset, M. Viikinkoski, E. Jehin, S. Benseguane, E. Podlowska-Gaca, B. Carry, A. Drouard, S. Fauvaud, M. Birlan, J. Berthier, P. Bartczak, C. Dumas, G. Dudziński, J. Ďurech, J. Castillo-Rogez, F. Cipriani, F. Colas, R. Fetick, T. Fusco, J. Grice, A. Kryszczyńska, P. Lamy, A. Marciniak, T. Michalowski, P. Michel, M. Pajuelo, T. Santana-Ros, P. Tanga, A. Vigan, O. Witasse, and B. Yang. (216) Kleopatra, a low density critically rotating M-type asteroid. *Astron. Astrophys.*, 653:A57, September 2021.
- [18] M. Brož, F. Marchis, L. Jorda, J. Hanuš, P. Vernazza, M. Ferrais, F. Vachier, N. Rambaux, M. Marsset, M. Viikinkoski, E. Jehin, S. Benseguane, E. Podlowska-Gaca, B. Carry, A. Drouard, S. Fauvaud, M. Birlan, J. Berthier, P. Bartczak, C. Dumas, G. Dudziński, J. Ďurech, J. Castillo-Rogez, F. Cipriani, F. Colas, R. Fetick, T. Fusco, J. Grice, A. Kryszczyńska, P. Lamy, A. Marciniak, T. Michalowski, P. Michel, M. Pajuelo, T. Santana-Ros, P. Tanga, A. Vigan, D. Vokrouhlický, O. Witasse, and B. Yang. An advanced multipole model for (216) Kleopatra triple system. *Astron. Astrophys.*, 653:A56, September 2021.
- [19] M. Ferrais, L. Jorda, P. Vernazza, B. Carry, M. Brož, N. Rambaux, J. Hanuš, G. Dudziński, P. Bartczak, F. Vachier, E. Aristidi, P. Beck, F. Marchis, M. Marsset, M. Viikinkoski, R. Fetick, A. Drouard, T. Fusco, M. Birlan, E. Podlowska-Gaca, T. H. Burbine, M. D. Dyar, P. Bendjoya, Z. Benkhaldoun, J. Berthier, J. Castillo-

Rogez, F. Cipriani, F. Colas, C. Dumas, J. Ďurech, S. Fauvaud, J. Grice, E. Jehin, M. Kaasalainen, A. Kryszczyńska, P. Lamy, H. Le Coroller, A. Marciniak, T. Michalowski, P. Michel, J. L. Prieur, V. Reddy, J. P. Rivet, T. Santana-Ros, M. Scardia, P. Tanga, A. Vigan, O. Witasse, and B. Yang. M-type (22) Kalliope: A tiny Mercury. *Astron. Astrophys.*, 662:A71, June 2022.

[20] JPL Horizons. <https://ssd.jpl.nasa.gov/horizons/>, 2024.