TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Modelovaní astronomických jevů

Modeling of astronomical phenomena

Diplomová práce

Autor: **Bc. Martin Brenner**
Vedoucí práce: Ing. Petr Kretschmer

Liberec 2011

Poděkování

Děkuji Ing. Petru Kretschmerovi za vedení práce a za ochotu, se kterou mi poskytoval svoje cenné rady a připomínky k práci. Dále bych chtěl tímto poděkovat všem, kteří mě podporovali, zejména pak svým příbuzným.

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum

Podpis

Anotace

V první části textu jsou podle potřeby popsané důležité základní pojmy z oblasti nebeské mechaniky a teorie 3D prostoru, které jsou potřebné k pochopení a popisu problému. Hlavní část diplomové práce se pak zabývá návrhem aplikace v programovacím jazyce C/C++ zpracovávající počítačovou grafiku v reálném čase na příkladu modelování astronomických jevů jako zatmění Slunce nebo Měsíce. K implementaci je využito hardwarové akcelerace pomocí Direct3D a jazyka HLSL. Na konci jsou prezentovány výsledky a zhodnocení spolu s výstupy aplikace.

Annotation

In the first part of this text, there are described important basic definitions and terms of celestial mechanics and theory of 3D programming, which are needed to understand and describe the problem. Thesis primarily deals with design of 3D real time application using programming language C/C++ applied to and example of celestial mechanics, especially modeling of astronomical phenomena called solar eclipse or lunar eclipse. Direct3D and language HLSL are used to get hardware acceleration. At the end, the results and estimation with output of the application are presented.

Klíčová slova: pohyb planet, sluneční soustava, nebeská mechanika, zatmění slunce, teorie 3D prostoru, 3D grafika, 3D programování v reálném čase, DirectX, c++, elementy dráhy.

Key words: solar system, celestial mechanics, solar eclipse, 3D space theory, 3D graphic, real time 3D programming, DirectX, c++, orbital elements.

Obsah

1 Úvod ... **1**

1.1 Vymezení cílů ... *1*

1.2 Popis problému ... *1*

2 Teorie 3D prostoru ... **3**

2.1 Matice ... *3*

2.1.1 Vektor ... 4

2.1.2 Normálový vektor ... 5

2.1.3 Normovaný vektor ... 5

2.2 Násobení matic ... *5*

2.2.1 Násobení matice a vektoru ... 6

2.3 Homogenní souřadnice ... *6*

2.4 Geometrické transformace ... *7*

2.4.1 Posun ... 7

2.4.2 Rotace v 3D prostoru ... 8

2.4.3 Zvětšení ... 8

2.5 Lokální souřadnicový systém ... *9*

2.6 Souřadnicový systém pozorovatele ... *9*

2.7 Projekce do 2D roviny ... *10*

2.8 Skládaní transformací ... *11*

2.9 Světla a stínování ... *11*

2.10 Stíny pomocí techniky shadow volume ... *12*

3 Nebeská mechanika ... **14**

3.1 Sférické souřadnice ... *14*

3.1.1 Transformace sférických souřadnic na kartézské ... 15

3.1.2 Transformace kartézských souřadnic na sférické ... 15

3.2 Juliánské datum ... *15*

3.3 Pohyb planet ... *16*

3.3.1 Elementy dráhy ... 17

3.3.2 Postup pro výpočet přibližných poloh planet ... 18

3.3.3 Řešení Keplerovy rovnice ... 20

3.3.4 Teorie s vysokou přesností ... 21

3.4 Astronomické jevy ... 21

3.4.1 Zatmění Slunce ... 22

3.4.2 Zatmění Měsíce ... 23

4 Grafická rozhraní ... 25

4.1 OpenGL ... 25

4.2 DirectX ... 25

4.2.1 HLSL ... 26

4.3 Analýza vhodnosti datových typů ... 27

5 Návrh aplikace ... 29

5.1 Návrh tříd ... 29

5.2 Implementace ... 30

5.3 Inicializace Direct3D ... 31

5.4 Jádro aplikace ... 34

5.5 Implementace shadow volume ... 36

6 Popis a návod ... 40

7 Zhodnocení ... 46

8 Závěr ... 50

8.1 Navrhovaná vylepšení ... 50

Příloha A Elementy drah validní pro epochu J2000 ... 52

Literatura ... 53

Obsah CD ... 59

Seznam obrázků

Obrázek 2.1 Souřadnicové systémy ... 3

Obrázek 2.2 Normálový vektor ... 5

Obrázek 2.3 Lokální souřadnicový systém ... 9

Obrázek 2.4 Ovládání kamery ... 9

Obrázek 2.5 Umístění kamery ve scéně ... 10

Obrázek 2.6 Ilustrace problému perspektivní projekce ... 10

Obrázek 2.7 Difusní složka světla ... 11

Obrázek 2.8 Konstantní stínování ... 12

Obrázek 2.9 Přepočítání normálových vektorů 12
Obrázek 2.10 Konstantní a Gouraudovo stínování 12
Obrázek 2.11 Technika shadow volume 13
Obrázek 3.1 Sférické souřadnice 15
Obrázek 3.2 Vnější elementy dráhy 17
Obrázek 3.3 Zatmění Slunce (geometrická interpretace) 21
Obrázek 3.4 Zatmění Slunce 22
Obrázek 3.5 Zatmění Měsíce 24
Obrázek 4.1 Grafická pipeline OpenGL 25
Obrázek 4.2 Grafická pipeline DirectX 26
Obrázek 4.3 Podpora hardwarové akcelerace pomocí Direct3D 26
Obrázek 5.1 Návrh tříd 30
Obrázek 5.2 Hlavní smyčka aplikace 32
Obrázek 6.1 Výpis stavu na obrazovku 40
Obrázek 6.2 Ovládání času pomocí menu 41
Obrázek 6.3 Okno pro zadání data a času 41
Obrázek 6.4 Výběr centrálního tělesa 42
Obrázek 6.5 Pohled z oběžné dráhy Měsíce 42
Obrázek 6.6 Zvětšení měřítka modelu 43
Obrázek 6.7 Synchronní kamera 43
Obrázek 6.8 Viditelné hvězdy během úplného zatmění Slunce 44
Obrázek 6.9 Vypnout/zapnout jednotlivé pomocné křivky 44
Obrázek 6.10 Zobrazení objemu stínu vrženého Měsícem na Zemi 44
Obrázek 6.11 Zobrazení objemu stínu vrženého Měsícem na Zemi 45
Obrázek 7.1 Výpis heliocentrických souřadnic Země během zatmění Slunce 46
Obrázek 7.2 Výpis heliocentrických souřadnic Měsíce během zatmění Slunce 47
Obrázek 7.3 Porovnání výsledku dopadu stínu na povrch Země dne 29. 3. 2006 48
Obrázek 7.4 Průběh zatmění Slunce podle NASA dne 29. 3. 2006 48
Obrázek 7.5 Porovnání výsledku dopadu stínu na povrch Země dne 22. 6. 2001 49
Obrázek 7.6 Průběh zatmění Slunce podle NASA dne 22. 6. 2001 49

A - 1 Elementy dráhy pro epochu J2000 52

1 Úvod

S rozvojem levných a dostupných počítačů v posledních desetiletích proniklo počítačové modelování do většiny technických věd. V praxi bývá cílem pochopit pozorované jevy, napodobit chování zkoumaného systému, simulovat je na vlastním modelu a následně ovlivnit jeho chování požadovaným způsobem, např. za pomoci zjištěných jeho vnitřních stavů. Modelovaný systém případně lze formálně popsat a následně simulovat i v reálném čase, například na počítači, a dále ho pozorovat. Model je vždy pouze přiblížením skutečnosti: reálný systém může být velmi složitý, model mu nemusí odpovídat. Z těchto důvodů provádíme ověřovací experimenty, kterými doplňujeme a dále zpřesňujeme parametry modelu nebo i model samotný[23].

Model ale málokdy dokáže popsat dění dostatečně přesně. Výsledek jsme např. získali s použitím zjednodušujících předpokladů, nebo byl použit nevhodný model. Při modelování je proto zásadní znát omezení použitého modelu a nevyvozovat z modelování nepatřičné závěry[23].

1.1 Vymezení cílů

Cílem práce je navrhnout aplikaci, která bude v reálném čase graficky interpretovat zjednodušený 3D model Sluneční soustavy, pohyb vesmírných těles a simulovat zatmění Slunce nebo zatmění Měsíce. Snahou je geometricky ilustrovat stín dopadající na povrch tělesa při konjunkci dvou různých vesmírných těles a Slunce (v tomto případě Země, Měsíc a Slunce) .

Uživatelské vstupy by měly umožňovat základní operace s časovačem pro možnost zrychlení, zpomalení, zpětný chod nebo vložení nového okamžiku startu. Výstupem je pak aplikace simulující průběh zatmění Slunce nebo Měsíce.

1.2 Popis problému

Prvním krokem při počítačovém modelování bývá sestavení matematického modelu zkoumaného systému. Model může být získán buďto teoreticky ze základních fyzikálních vlastností systému, nebo empiricky z naměřených hodnot. Určování parametrů teoreticky vytvořeného modelu z empirických hodnot se nazývá identifikace systémů[23].

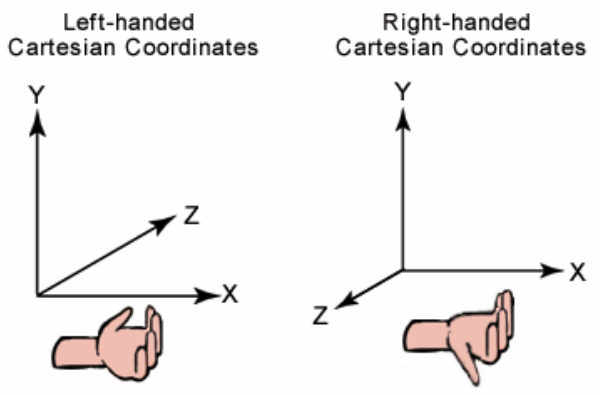
Protože je třeba při řešení problému konjunkce znát polohu vesmírných těles, matematický model daného problému vychází z dynamiky Sluneční soustavy. Touto problematikou se zabývá obor nebeská mechanika a jednou z možností je použít empirický model, kde se aproximuje pozice vesmírných těles v závislosti na čase a naměřených elementů jejich drah. Je nutné tedy zjistit, jaké možnosti k určení polohy vesmírných těles se nabízí a jaké přesnosti dosahují.

Vizualizace dat je výpočetně velice náročný proces, zejména 3D vizualizace v reálném čase se provádí pomocí grafických rozhranní, které využívají paralelní hardwarovou akceleraci. Z tohoto důvodu se k návrhu aplikací tohoto typu používá nejčastěji programovací jazyk Microsoft C/C++ s podporou jednoho z grafických API rozhranní. Nabízejí se hlavní dva konkurenti: DirectX od Microsoft nebo OpenGL od SGI.

Při vizualizaci 3D dat se využívá teorie 3D prostoru aplikovaná v programovaní, takzvané 3D programování. Jde o obor relativně mladý, ale rychle se rozvíjející. V dnešní době již existuje řada patentů, které popisují konkrétní implementace jednotlivých metod (např. stíny pomocí přídavného zásobníku na grafické kartě aj.).

2 Teorie 3D prostoru

Teorie 3D prostoru hraje důležitou roli jak v počítačové grafice, tak i v dynamice Sluneční soustavy. Základem je kartézský souřadnicový systém definovaný třemi ortonormálními vektory. V praxi pak rozlišujeme dva typy kartézských souřadnicových systémů: levotočivý a pravotočivý.



Obrázek 2.1 Souřadnicové systémy[11]

3D prostor se popisuje pomocí matematické disciplíny zvané geometrie a implementuje pomocí lineární algebry.

2.1 Matice

Matice je v matematice obdélníková tabulka čísel nebo nějakých matematických objektů tzv. prvků matice (též elementů matice). Obsahuje obecně ***m*** řádků a ***n*** sloupců. Hovoří se pak o matici typu $m \times n$. Matice se často využívají pro vyjádření obecné rotace vektorů, transformace vektorů od jedné báze k bázi jiné nebo k výpočtu soustav lineárních rovnic[1].

Pro označení matic se používají velká písmena tištěná tučně jako **A**. Prvky matice jsou označeny indexy udávajícími řádek a sloupec, v nichž se prvek nalézá. Prvek v $i-tém$ řádku a $j-tém$ sloupci matice **A** se obvykle značí a_{ij}. Obecně se matice **A** o rozměrech $m \times n$ zapisuje ve tvaru[1]:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & \dots & \dots & \dots \\ \dots & \dots & \dots & a_{n(m-1)} \\ a_{m1} & \dots & a_{m(n-1)} & a_{mn} \end{pmatrix}. \qquad (2.1)$$

V praxi se také používá tzv. transponovaná matice $\mathbf{A^T}$ k matici **A**. Tato matice se získá tak, že se prohodí sloupce s řádky matice **A**[1]:

$$\mathbf{A}^T = \begin{pmatrix} a_{11} & a_{21} & \ldots & a_{m1} \\ a_{12} & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & a_{m(n-1)} \\ a_{1n} & \ldots & a_{n(m-1)} & a_{mn} \end{pmatrix}. \tag{2.2}$$

Zavádí se i speciální případ matice $\mathbf{I_n}$, nazývanou maticí identity neboli jednotková matice. Tato matice má rozměr $n \times n$ a obecně se zapisuje ve tvaru[1]:

$$\mathbf{I}_n = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 \end{pmatrix}. \tag{2.3}$$

2.1.1 Vektor

Vektor je speciální případ matice a matematicky je definován jako uspořádaná $n-tice$ prvků (typicky čísel), označovaných jako složky vektoru. Obecněji lze vektor chápat jako abstraktní prvek vektorového prostoru. Na rozdíl od skaláru nese vektor kromě čísla i směr. Počet složek vektoru souvisí s dimenzí vektorového prostoru. Pro značení vektorů se používají malá tučně tištěná písmena jako **a**. Obecně lze vektor **a** zapsat[1]:

$$\mathbf{a} = \begin{pmatrix} a_1 & a_2 & \ldots & a_{n-1} & a_n \end{pmatrix}, \tag{2.4}$$

kde vektor **a** je nazýván řádkovým vektorem. Jinou možností zápisu je transponovaný vektor $\mathbf{a^T}$ neboli vektor sloupcový zapsán ve tvaru:

$$\mathbf{a}^{\mathbf{T}} = \begin{pmatrix} a_1 \\ a_2 \\ \ldots \\ a_{n-1} \\ a_n \end{pmatrix}, \tag{2.5}$$

kde $a_1, a_2, \ldots, a_{n-1}, a_n$ jsou složky vektoru **a** respektive vektoru $\mathbf{a^T}$.

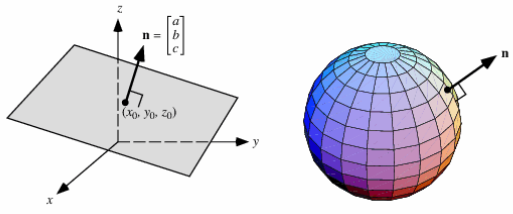
V kartézském souřadnicovém systému se potom používá značení pro vázaný polohový vektor bodu v prostoru:

$$\mathbf{r} = \begin{pmatrix} x & y & z \end{pmatrix}, \tag{2.6}$$

kde ***x*** se nazývá ***x-ovou*** složku vektoru **r**, ***y*** pak ***y-ovou*** složku vektoru **r** a ***z*** jako ***z-vou*** složku vektoru **r**.

2.1.2 Normálový vektor

Normálový vektor **n** v tří dimenzionálním prostoru je kolmice k obecné rovině. Pokud nějaká rovina existuje, nazývá se vektor svírající s touto rovinou úhel 90 stupňů normálovým vektorem k dané rovině. Na obr. 2.2 představuje normálový vektor šipka. Velikost normálového vektoru bývá zpravidla 1[1].



Obrázek 2.2 Normálový vektor[36]

2.1.3 Normovaný vektor

Normovaný vektor **e** má velikost 1. Tento vektor je také označován jako jednotkový vektor nebo směrový vektor. Normovaný vektor **e** k vektoru **r** se získá podělením všech složek vektoru **r** jeho velikostí[1]:

$$\mathbf{e} = \frac{\mathbf{r}}{|\mathbf{r}|}, \tag{2.7}$$

kde $|\mathbf{r}|$ značí velikost vektoru **r**, která se v tří rozměrném prostoru vypočítá:

$$|\mathbf{r}| = \sqrt{x^2 + y^2 + z^2}\,. \tag{2.8}$$

Vektor se tedy dá rozložit na absolutní a směrovou složku:

$$\mathbf{r} = |\mathbf{r}|\mathbf{e}\,. \tag{2.9}$$

2.2 Násobení matic

Pokud **A** je matice $m \times n$ a **B** je matice $n \times p$ (tedy pokud první matice má tolik sloupců, kolik má druhá matice řádků, jejich součin $\mathbf{A} \times \mathbf{B}$ je matice $m \times p$ zadaná[1]:

$$(AB)_{ij} = \sum_{r=1}^{n} a_{ir} b_{rj} = a_{i1} b_{1j} + a_{i2} b_{2j} + \ldots + a_{in} b_{nj} \,. \tag{2.10}$$

Jednou z velice důležitých vlastností násobení dvou matic je, že maticové násobení není komutativní:

$$\mathbf{AB} \neq \mathbf{BA} \,.$$

Pro lepší ilustraci je zde uveden příklad násobení dvou tří rozměrných matic $\mathbf{A} \times \mathbf{B} = \mathbf{C}$:

$$\mathbf{AB} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} = \mathbf{C}, \tag{2.11}$$

kde **C** je výsledná matice o rozměrech 3x3 a její prvky se získají následovně:

$$\mathbf{AB} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} & a_{11}b_{13} + a_{12}b_{23} + a_{13}b_{33} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} & a_{21}b_{13} + a_{22}b_{23} + a_{23}b_{33} \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} & a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32} & a_{31}b_{13} + a_{32}b_{23} + a_{33}b_{33} \end{pmatrix} = \mathbf{C} \cdot \tag{2.12}$$

2.2.1 Násobení matice a vektoru

V praxi se používají dva způsoby jak násobit matici a vektor. V prvním případě se násobí řádkový vektor **r** maticí $\mathbf{A}^{[1]}$:

$$\mathbf{r}' = \begin{pmatrix} x' & y' & z' \end{pmatrix} = \mathbf{rA} = \begin{pmatrix} x & y & z \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \tag{2.13}$$

pak se výsledný řádkový vektor **r´** získá následujícím postupem:

$$\mathbf{r}' = \begin{pmatrix} x' & y' & z' \end{pmatrix} = \begin{pmatrix} xa_{11} + ya_{21} + za_{31} & xa_{12} + ya_{22} + za_{32} & xa_{13} + ya_{23} + za_{33} \end{pmatrix}$$

V druhém případě se násobí transponovaná matice $\mathbf{A}^{\mathbf{T}}$ sloupcovým vektorem $\mathbf{r}^{\mathbf{T}}$, kde výsledkem je transponovaný vektor $\mathbf{r}'^{\mathbf{T}}$[1]:

$$\mathbf{r}'^{\mathbf{T}} = \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \mathbf{A}^{\mathbf{T}} \mathbf{r}^{\mathbf{T}} = \begin{pmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} xa_{11} + ya_{21} + za_{31} \\ xa_{12} + ya_{22} + za_{32} \\ xa_{13} + ya_{23} + za_{33} \end{pmatrix} \tag{2.14}$$

2.3 Homogenní souřadnice

Homogenní souřadnice se zavádějí speciálně pro počítačovou grafiku pro zjednodušení výpočtů, což zvyšuje výkon aplikace. Výhodou je snadná

implementace geometrických transformací tvořených násobením matic mezi sebou. Použitím homogenních souřadnic lze sestrojit translační matici nebo aplikovat matici perspektivní projekce. Homogenní souřadnice se převádějí z kartézských přidáním čtvrté složky ***w***[1]:

$$\mathbf{r}_h = (x,y,z,w), \tag{2.15}$$

kde obvykle homogenní složka $w = 1$.

Matice geometrických transformací se sestavují o rozměrech 4×4, aby umožnili aplikaci geometrických transformací na homogenní souřadnice[1]:

$$\mathbf{M} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & w \end{pmatrix} \quad \mathbf{M}^T = \begin{pmatrix} m_{11} & m_{21} & m_{31} & m_{41} \\ m_{12} & m_{22} & m_{32} & m_{42} \\ m_{13} & m_{23} & m_{33} & m_{43} \\ m_{14} & m_{24} & m_{34} & w \end{pmatrix} \tag{2.16}$$

Převod homogenních souřadnic na kartézské:

$$\begin{aligned} &\mathbf{r}_k = (x/w, y/w, z/w), \\ &w \neq 0. \end{aligned} \tag{2.17}$$

2.4 Geometrické transformace

Geometrické transformace jsou transformace používané v počítačové grafice při přípravě scény. Transformace se aplikují na bod, transformovat objekt tedy znamená transformovat všechny jeho body. Základními transformacemi jsou posun, rotace kolem pevných os, zvětšení nebo zkosení[1].

V astronomii se těchto transformací (zejména rotace) využívá k transformaci souřadnic z jedné roviny do druhé.

2.4.1 Posun

Posun neboli translace je transformace, která udává kterým směrem a o kolik se má bod posunout. Tato matice se značí např. $\mathbf{T}^{\mathbf{T}}(\mathbf{t_x},\mathbf{t_y},\mathbf{t_z})$, kde $\mathbf{t_x}$ je vzdálenost posunutí ve směru osy ***x***, $\mathbf{t_y}$ je vzdálenost posunutí ve směru osy ***y*** a $\mathbf{t_z}$ je vzdálenost posunutí ve směru osy ***z***. Transformační matice translace a její aplikace na bod[1] [19]:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \\ z + t_z \\ 1 \end{pmatrix} \tag{2.18}$$

2.4.2 Rotace v 3D prostoru

Rotace je transformace, která udává o kolik se vektor otočí kolem jedné z pevných os. Transformační matice rotace kolem osy ***x*** o úhel beta $\mathbf{R_x^T(\beta)}$, a její aplikace na bod[1] [19]:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta & 0 \\ 0 & \sin\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ \cos\beta y - \sin\beta z \\ \sin\beta y + \cos\beta z \\ 1 \end{pmatrix} \tag{2.19}$$

Transformační matice rotace kolem osy ***y*** o úhel beta $\mathbf{R_y^T(\beta)}$ a její aplikace na bod:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\beta x + \sin\beta z \\ y \\ -\sin\beta x + \cos\beta z \\ 1 \end{pmatrix} \tag{2.20}$$

Transformační matice rotace kolem osy ***z*** o úhel beta $\mathbf{R_z^T(\beta)}$ a její aplikace na bod:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\beta & -\sin\beta & 0 & 0 \\ \sin\beta & \cos\beta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\beta x - \sin\beta y \\ \sin\beta x + \cos\beta z \\ z \\ 1 \end{pmatrix} \tag{2.21}$$

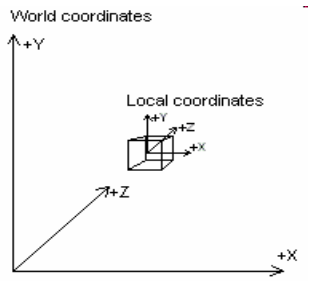
2.4.3 Zvětšení

Zvětšení je transformace změny měřítka objektu podle souřadnicových os. Tato matice se značí například $\mathbf{S^T(s_x,s_y,s_z)}$, kde $\mathbf{s_x}$ je velikost zvětšení vzhledem k ose ***x***, $\mathbf{s_y}$ je velikost zvětšení vzhledem k ose ***y*** a $\mathbf{s_z}$ je velikost zvětšení vzhledem k ose **z**. Transformační matice zvětšení a její aplikace na bod[1] [19]:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} xs_x \\ ys_y \\ zs_z \\ 1 \end{pmatrix} \tag{2.22}$$

2.5 Lokální souřadnicový systém

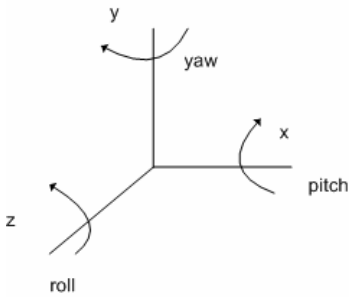
Každý objekt je umístěn v souřadnicovém systému nazývaném tzv. model space, kde jsou jednotlivé body relativně definované k počátku soustavy daného objektu. Pokud je objekt umístěn do 3D scény, je zapotřebí provést transformaci souřadnic do nového souřadnicového systému označován jako world space. Tato transformace se nazývá world transform a převádí souřadnice modelu z model space do world space. Tato transformace se skládá ze základních geometrických transformací (rotace, posun, změna měřítka aj.) [22].



Obrázek 2.3 Lokální souřadnicový systém[22]

2.6 Souřadnicový systém pozorovatele

V souřadnicovém prostoru pozorovatele je kamera umístěna v počátku kartézského souřadnicového systému tzv. camera space. Je tedy potřeba provést tzv. view transform, která umístí pozorovatele (kameru) do scény. Zpravidla se sestaví matice tzv. view matrix, jež určuje umístění, normálový vektor a orientaci kamery. Existuje několik způsobů jak tuto matici vytvořit. Stavba matice záleží na souřadnicovém systému a na uspořádaní os. Kromě změny pozice kamery vysvětluje další možnosti ovládání kamery obr. 2.4[21].

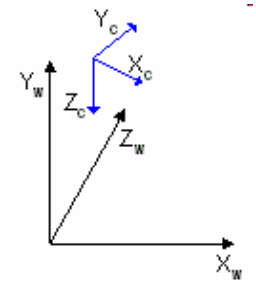


Obrázek 2.4 Ovládání kamery[16]

Výsledná matice se skládá z posunu a tří rotací kolem každé z os:

$$V = T \cdot R_z \cdot R_y \cdot R_x \tag{2.23}$$

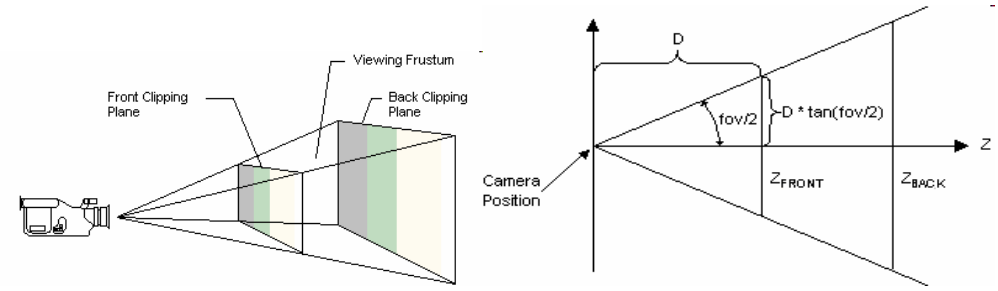
kde **V** je výsledná matice a **T** je matice translace, která posune kameru ve scéně. **Rx**, **Ry**, **Rz** jsou matice rotace. Matice **V** odpovídá souřadnicovému systému (uspořádání os) na obr. 2.5[21].



Obrázek 2.5 Umístění kamery ve scéně[21]

2.7 Projekce do 2D roviny

Jako poslední a nejsložitější zde uvedená transformace je projekce. Nejčastěji se využívá perspektivní projekce 3D scény do 2D roviny. Princip popisuje obr. 2.6[17].



Obrázek 2.6 Ilustrace problému perspektivní projekce[20]

Matice projekce transformuje prostor zájmu do krychlovitého tvaru, kde rovina v popředí zabírá menší plochu než rovina v pozadí. Matice pro perspektivní projekci[17]:

$$\begin{bmatrix} w & 0 & 0 & 0 \\ 0 & h & 0 & 0 \\ 0 & 0 & Q & 1 \\ 0 & 0 & -QZ_n & 0 \end{bmatrix} \tag{2.24}$$

Jednotlivé koeficienty matice pro perspektivní projekci se získají podle rovnic 2.25, kde **Zn** značí vzdálenost z-near (Z-front), **Zf** vzdálenost z-far (Z-back) a proměnná **fov** vertikální a horizontální frustrum podle obr. 2.6[17].

$$w = \cot\left(fov_w/2\right)$$
$$h = \cot\left(fov_h/2\right)$$
$$Q = \frac{Z_f}{Z_f - Z_n} \tag{2.25}$$

2.8 Skládaní transformací

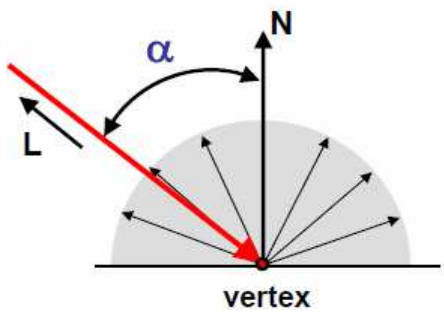
Skládání geometrických transformací se využívá např. při přípravě scény. Realizuje se pomocí násobení transformačních matic pro určení nové polohy bodu **p**[1]:

$$p' = p\mathbf{M}_{\text{view}}\mathbf{M}_{\text{world}}\mathbf{M}_{\text{projection}}, \tag{2.26}$$

kde $\mathbf{M}_{\text{view}}$ značí „view matrix“, $\mathbf{M}_{\text{world}}$, „world matrix“, $\mathbf{M}_{\text{projection}}$ „projection matrix“ a **p’** je nová pozice bodu.

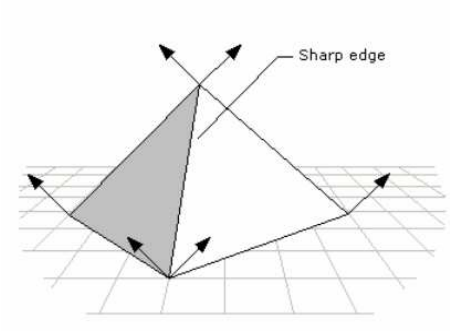
2.9 Světla a stínování

Světla tvoří v počítačové 3D grafice důležitou složku. Vhodně nastavená světla značně ovlivní vzhled 3D scény. Je-li ale potřeba vykreslovat scénu v reálném čase, je nutné osvětlovací model přizpůsobit možnostem hardwaru. Phongův osvětlovací model např. započítává kromě jiných hlavně ambientní, difusní a odrazovou složku. Nejdůležitější složkou je však složka difusní, která se spočítá pomocí úhlu svírajícího vektor směru světla a normálového vektoru plochy[25].



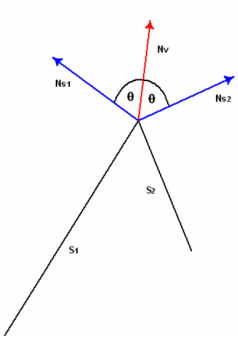
Obrázek 2.7 Difusní složka světla[25]

Výpočet difusní složky znamená zjistit intenzitu světla dopadající na plochu. Jednou z možností je použít konstantní stínování, kde se spočítají hodnoty v bodech pomocí skalárního součinu normálového vektoru a vektoru směru světla[25].



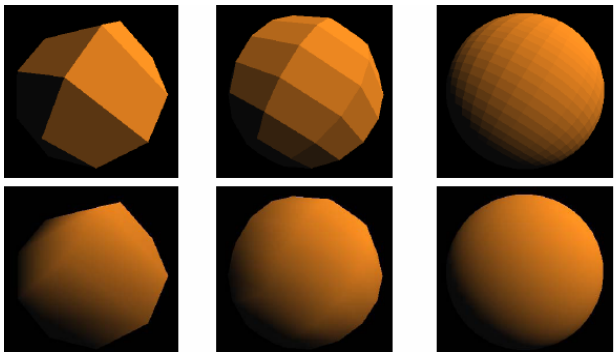
Obrázek 2.8 Konstantní stínování[14]

Gouraudovo stínování využívá interpolace barev ke zjemnění přechodu mezi ostrými hranami. K tomu je potřeba spočítat společný normálový vektor bodu následujícím způsobem vyjádřeném pomocí obr. 2.9[14].



Obrázek 2.9 Přepočítání normálových vektorů[14]

Rozdíl mezi stínováním je znázorněn na obr. 2.10 (lepší je normálové vektory přepočítat a připravit předem).

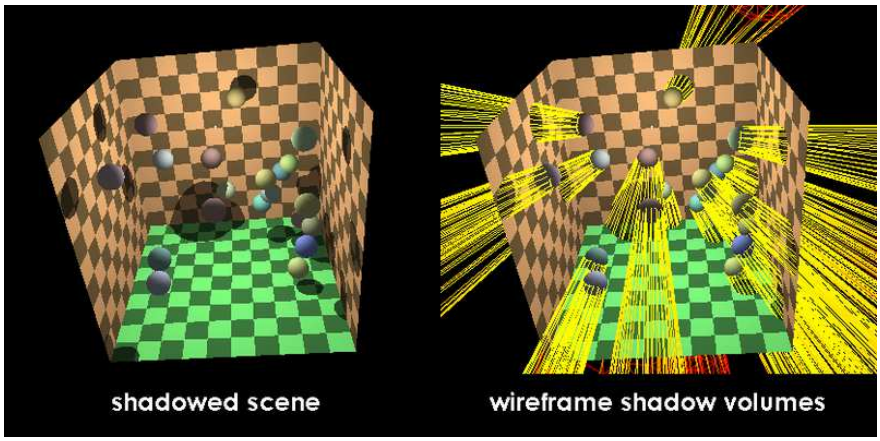


Obrázek 2.10 Konstantní a Gouraudovo stínování[8]

2.10 Stíny pomocí techniky shadow volume

Shadow volume (objem stínu) je vedle shadow mapping (mapování stínů) a planar shadow (rovinné stíny) technika využívaná v počítačové 3D grafice v reálném čase k přidání stínů do scény. Tato technika rozděluje virtuální svět na dvě části, ta co je ve stínu a ta co není. Hlavní výhodou techniky je její přesnost na pixel, avšak kvůli

obrysovým hranám nastává menší problém při samo-stínování objektu vrhajícího stín. Další nevýhodou je potřeba vytvořit geometrii stínu, která je náročná na výkon CPU, a tak je tato technika pomalejší než např. shadow mapping[40].



Obrázek 2.11 Technika shadow volume[40]

Základní kroky konstrukce objemu stínu jsou[40] :

1. najít všechny obrysové hrany objektu
2. posunout tyto hrany do požadované vzdálenosti směrem od světelného zdroje

Shadow volume je tedy další objekt ve scéně, který ovšem není vidět obr. 2.11.

Stencil buffer je přídavný buffer na grafické kartě, který pro každý pixel pracuje s hodnotou integer. Pomocí tohoto zásobníku se např. vytváří zrcadlení nebo stíny.

Pro implementaci shadow volume se používají běžně 3 variace: depth pass, depth fail a exklusive-or. Pro všechny tyto techniky je stejný základ[40] [9]:

1. Vykreslit scénu celou ve stínu.
2. Pro každé ze světelných zdrojů:
    1. Pomocí z-informace vytvořit masku v přídavném zásobníku tak, že vytvoříme díry, které určují, že tato část není ve stínu.
    2. Vykreslit scénu po druhé tak, jakoby by byla celá osvětlená a použít stencil buffer k odmaskování stínu smíšením těchto dvou obrázků.

Objekt, na který dopadá stín se tedy vykresluje více než jednou. Záleží na počtu světel použitých ve scéně. Toto je jeden z hlavních důvodů náročnosti techniky na výpočetní výkon. Při použití většího množství světel dochází k výraznému zpomalení vykreslování.

3 Nebeská mechanika

Snaha předpovědět polohu planet na obloze se traduje už do dávných časů. Postupný vývoj ve snaze pochopit astronomické jevy vyústil k různým teoretickým modelům. Podle pečlivých pozorování a geometrických výpočtů vznikl např. model Sluneční soustavy známý jako Ptolemaiův systém, který byl založen na geocentrickým systému[27].

Byla to však práce Tycho de Braha, Johannese Keplera a Isaaca Newtona, která zformalizovala moderní heliocentrický systém. Metody klasické nebeské mechaniky jsou odvozeny od Newtonova zákona všeobecné gravitace a jeho tří pohybových zákonů. Problém je, že na jednotlivá tělesa nepůsobí pouze gravitační síla Slunce, ale i síly ostatních těles. Tyto síly způsobují tzv. pertubace oběžné dráhy, jež se nedají přesně spočítat, protože se v čase mění. K aproximaci se využívá pokročilého stupně matematiky a implementace vyžaduje ohromný výkon počítačů[27].

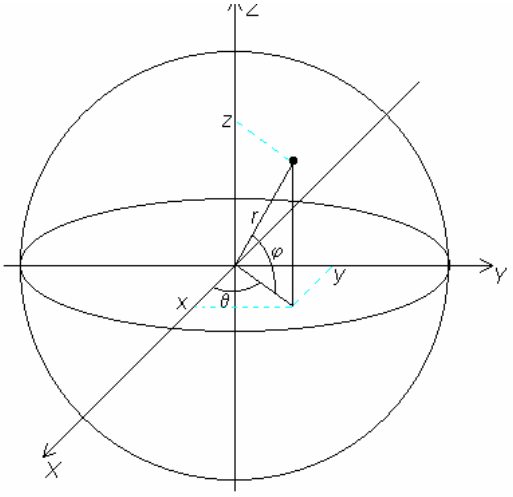
V roce 1781 Joeph Louis Lagrange poprvé nabídl řešení pomocí linearizační metody. Dnes se na základě těchto poznatků rozšířené v roce 1897 (Hill) modelují teorie dynamiky Sluneční soustavy s obdivuhodnou přesností[27].

Jelikož se ale problém n-těles řeší složitými numerickými metodami, které jsou velice náročné na výkon počítače, používají se často pro získání pozic těles v závislosti na čase výpočty, které využívají elementů drah získané z předešlých pozorování nebo teoriích zahrnující jednotlivé pertubace[29].

Pro orientaci v problematice je potřeba znát základní pojmy z nebeské mechaniky. V následující části textu jsou popsány především pojmy, které jsou často využívané k popisu chování vesmírných těles.

3.1 Sférické souřadnice

Astronomická matematika hojně využívá k popisu právě sférické souřadnice. Jedná se o soustavu souřadnic v prostoru, u které jedna souřadnice $\boldsymbol{r}$ udává vzdálenost bodu od počátku souřadnic, druhá souřadnice $\boldsymbol{\varphi}$ udává úhel odklonu průvodiče bodu od osy $\boldsymbol{x}$ a třetí souřadnice $\boldsymbol{\theta}$ úhel mezi průvodičem a osou $\boldsymbol{z}$[1].



Obrázek 3.1 Sférické souřadnice[26]

3.1.1 Transformace sférických souřadnic na kartézské

Tato transformace je základem k převodu mezi různými druhy astronomických souřadnic. Zpravidla se jedná o transformaci z jedné roviny do druhé (např. rovina ekliptiky a rovina oběžné dráhy vesmírného objektu).

$$x = r \sin\theta \cos\varphi,$$
$$y = r \sin\theta \sin\varphi, \qquad (3.1)$$
$$z = \cos\theta.$$

3.1.2 Transformace kartézských souřadnic na sférické

Tato transformace je také často používána v astronomických výpočtech. Je-li dán bod v prostoru, zjistí se úhly svírající s osami následovně:

$$r = \sqrt{x^2 + y^2 + z^2},$$
$$\varphi = arctg2(y, x), \qquad (3.2)$$
$$\theta = \arccos\left(\frac{z}{r}\right),$$

kde $arctg2(y, x)$ je zobecněním funkce arkus tangens.

3.2 Juliánské datum

Juliánské datum, též juliánský den, je spojitá časová škála, používaná v astronomii zejména pro sledování dlouhých a periodických časových úseků.

Je definována jako počet dní (tj. úseků dlouhých 86 400 sekund), které uplynuly od poledne světového času dne 1. ledna roku 4713 př. Kristem. Zapisuje se jako desetinné číslo, přičemž zlomková část čísla odpovídá příslušnému dílu dne. Mnohdy se v zápisu čísla vypouštějí mezery mezi číslicemi za miliony a tisíci[1].

Juliánské datum je používáno zejména pro určování specifických okamžiků v pohybu nebeských těles.

Juliánské datum **JD** se z občanského data, kde **R** značí rok, **M** měsíc a **D** den, vypočítá podle následujícího algoritmu[26]:

$$\text{Pro } M < 3: f = M + 12, g = R - 1, \quad \text{Pro } M \geq 3: f = M, g = R,$$

$$JD = D + \left[\frac{153f - 457}{5}\right] + 365g + \left[\frac{g}{4}\right] - \left[\frac{g}{100}\right] + \left[\frac{g}{400}\right] + 1721118.5, \tag{3.3}$$

kde podíl v hranaté závorce značí dělení bezezbytku, tedy výsledek daného podílu je číslo s useknutou desetinnou čárkou.

Občanské datum se z Juliánského data **JD**, kde **R** značí rok, **M** měsíc a **D** den, vypočítá podle následujícího algoritmu[26]:

$$\begin{aligned}
&z = [JD + 0.5],\\
&pro\ z < 2299161: \mathrm{b} = 0, c = \mathrm{z} + 1524,\\
&pro\ z \neq 2299161: b = [(z - 1867216.25) / 36524.25],\\
&c = z + b - [b / 4] + 1525,\\
&d = [(c - 122.1)/365.25],\\
&e = 365\, d + [d/4],\\
&f = [(c - e)/30.6001],\\
&D = [c - e + 0.5] - [30.6001\ f] + JD + 0.5 - z,\\
&M = f - 1 - 12\,[f/14],\\
&R = d - 4715 - [(7 + M) / 10].
\end{aligned} \tag{3.4}$$

3.3 Pohyb planet

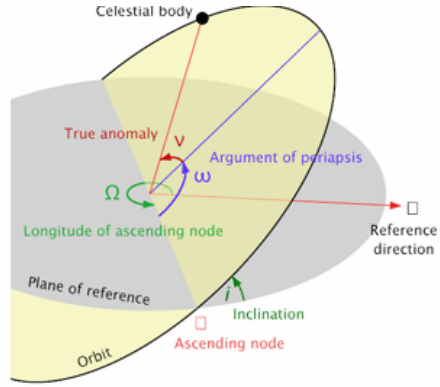
Pro přibližný výpočet poloh planet se využívá třech zákonů, které popsal Johannes Kepler v 17. století. Z prvního zákonu vyplívá, že planety se pohybují po rovinných křivkách podobné elipsám kolem stálého středu. Z druhého pak, že plošná rychlost je konstantní a třetí zákon říká, že planety blízko Slunce jej oběhnou dříve než ty vzdálenější.

Z těchto tří zákonů Johannes Kepler odvodil elementy dráhy tělesa a na svou dobu s obdivuhodnou přesností rovnici pro výpočet poloh planet z těchto elementů. Dodnes se tohoto postupu využívá v důležitých aplikacích, kde není nutná absolutní přesnost. Hojně jich využívají observatoře pro plánování rozvrhů, zaměřovaní dalekohledů, simulování jistých fenomén nebo jich bylo například využíváno i k navrhování kosmických letů[29].

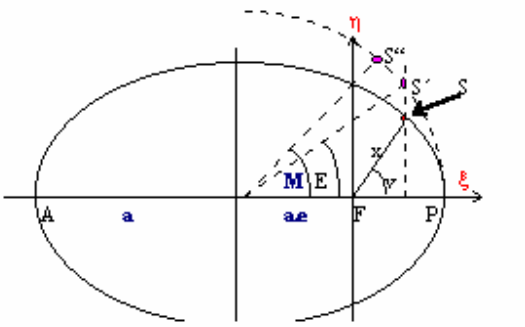
3.3.1 Elementy dráhy

Přibližná pozice devíti hlavních planet Sluneční soustavy může být spočtena užitím Keplerovy rovnice s odpovídajícími elementy a jejich přírůstky. Elementy jsou platné pouze pro jistý časový interval s nimi spjatý a mimo tento interval nejsou výsledky relevantní [29].

Základních elementů je celkem 6, ale v praxi se užívají i elementy odvozené. Rozlišují se vnější Keplerovy elementy obr. 3.2 a vnitřní Keplerovy elementy obr. 3.3[34].



Obrázek 3.2 Vnější elementy dráhy[38]



Obrázek 3.3 Vnitřní elementy dráhy[34]

Velká poloosa dráhy *a* viz obr. 3.2 je definována jako střední vzdálenost tělesa od hmotného středu soustavy vyjádřená v astronomických jednotkách AU[1] [3].

Excentricita neboli výstřednost dráhy udává typ oběžné dráhy a její tvar. Značí se ***e*** a její význam spočívá v určení, jak moc se křivka liší od kružnice. Pro kružnici je tedy $e = 0$, pro elipsu $0 < e < 1$, pro parabolu $e = 1$ a pro hyperbolu $e > 1$[1] [3].

Sklon dráhy viz obr. 3.2 je definován jako úhel, který svírá rovina dráhy se základní rovinou příslušné souřadné soustavy. Značí se ***i*** a vyjadřuje v úhlových stupních[1] [3].

Střední délka je definována jako úhel, který směřuje na pozici, kde by dané těleso bylo, kdyby oběžná dráha byla kružnice a inklinace by byla nulová. Značí se ***L*** a vyjadřuje se v úhlových stupních[1] [3].

Argument délky pericentra je jedním z alternativních elementů dráhy. Je definován jako lomený úhel, který měříme nejprve v základní rovině souřadné soustavy ve směru od kladné osy ***x*** ke kladné ose ***y*** (při pohledu shora, tj. od kladných hodnot souřadnice ***z***) až do vzestupného uzlu. Potom se měří v rovině oběžné dráhy ve směru pohybu obíhajícího tělesa od vzestupného uzlu k pericentru dráhy. Značí se ϖ a vyjadřuje se v úhlových stupních[1] [3].

Délka vzestupného uzlu viz obr. 3.2 je definována jako úhel, který svírá průvodič polohy tělesa nacházejícího se ve vzestupném uzlu své dráhy se směrem kladné osy ***x*** v základní rovině příslušné souřadné soustavy. Tento úhel se vždy měří v základní rovině souřadné soustavy ve směru od kladné osy ***x*** ke kladné ose ***y*** (při pohledu shora, tj. od kladných hodnot souřadnice ***z***). Značí se Ω a vyjadřuje se v úhlových stupních[1] [3].

Argument šířky pericentra viz obr. 3.2 je definován jako úhel, který svírá průvodič tělesa nacházejícího se ve vzestupném uzlu dráhy se směrem průvodiče téhož tělesa nacházejícího se v pericentru své dráhy. Značí se ω a vyjadřuje se ve stupních. Tento úhel se vždy měří v rovině oběžné dráhy ve směru pohybu obíhajícího tělesa od vzestupného uzlu k pericentra dráhy[1] [3].

Střední anomálie viz obr. 3.2 je definována jako úhel, který by opsal průvodič polohy tělesa pohybujícího se po kruhové dráze o poloměru ***r*** rovném velké poloose ***a*** skutečné dráhy od okamžiku průchodu pericentrem $\mathbf{T_0}$ do daného okamžiku ***T***. Střední anomálie se značí ***M*** a vyjadřuje se v úhlových stupních[1] [3].

3.3.2 Postup pro výpočet přibližných poloh planet

Následující algoritmus spolu s elementy dráhy pro čas J2000 je převzat z oficiálních stránek NASA. K řešení Keplerovy rovnice je využit algoritmus Newtonovy metody[29].

Označení důležitých elementů užitých v algoritmu:

$a_0, \dot{a}$: velká poloosa dráhy $[au, au/století]$,

$e_0, \dot{e}$: excentricita $[, / \textit{století}]$,

$I_0, \dot{I}$: sklon dráhy $[\textit{stupně}, \textit{stupně} / \textit{století}]$,

$L_0, \dot{L}$: střední délka $[\textit{stupně}, \textit{stupně} / \textit{století}]$,

$\varpi_0, \dot{\varpi}$: argument délky pericentra $[\textit{stupně}, \textit{stupně} / \textit{století}]$ $(\varpi = \omega + \Omega)$,

$\Omega_0, \dot{\Omega}$: délka vzestupného uzlu $[\textit{stupně}, \textit{stupně} / \textit{století}]$.

Nejdříve je třeba spočítat počet století uplynulých od data J2000.0 **T** podle rovnice (3.3). Dále je nutno spočítat hodnoty pro každý z šesti elementů:

$$a = a_0 + \dot{a} T$$

Nyní již lze určit argument šířky pericentra ω a střední anomálii **M**:

$$\omega = \varpi - \Omega \tag{3.5}$$

$$M = L - \varpi \tag{3.6}$$

Střední anomálie se přepočítá tak, aby pasovala do intervalu $\langle -180°, 180° \rangle$.

Výsledkem řešení Keplerovy rovnice je výstřední anomálie **E** viz. obr.3.3:

$$M = E - e^* \sin E, \tag{3.7}$$

Kde $e^* = 180 / \pi e = 57.29578 e$.

Na základě spočítané výstřední anomálie se určí heliocentrické souřadnice planet vzhledem k jejich orbitální rovině, kde kladná osa **x** směřuje od ohniska do pericentra:

$$x' = a(\cos E - e)$$

$$y' = a\sqrt{1 - e^2} \sin E \tag{3.8}$$

$$z' = 0$$

Je-li potřeba, lze dále souřadnice transformovat na souřadnice ekliptikální, $\mathbf{r_{ecl}}$, vztažené k rovině ekliptiky pro J2000, kde osa **x** směřuje k jarnímu bodu:

$$\mathbf{r}_{\text{ecl}} = \mathbf{M}\mathbf{r}' \equiv \mathbf{R_z}(-\Omega)\mathbf{R_x}(-I)\mathbf{R_z}(-\omega)\mathbf{r}', \tag{3.9}$$

což lze rozepsat jako:

$$x_{ecl} = (\cos\omega\cos\Omega - \sin\omega\sin\Omega\cos I)x' + (-\sin\omega\cos\Omega - \cos\omega\sin\Omega\cos I)y',$$

$$y_{ecl} = (\cos\omega\sin\Omega - \sin\omega\cos\Omega\cos I)x' + (-\sin\omega\sin\Omega + \cos\omega\cos\Omega\cos I)y', \tag{3.10}$$

$$z_{ecl} = (\sin\omega\sin I)x' + (\cos\omega\sin I)y' .$$

3.3.3 Řešení Keplerovy rovnice

Keplerova rovnice lze řešit například pomocí numerické metody nazvané iterace. Je-li známa střední anomálie ***M*** a výstřednost ***e****, obojí zadáno ve stupních, začne se postup následovně [29]:

$$E_0 = M + e^* \sin M \tag{3.11}$$

a pokračuje iterací následujících tří rovnic kde $n = 0,1,2,3...$, dokud není $|\Delta E| \le tol$, (***e**** je zadáno ve stupních, ***e*** je v radiánech):

$$\Delta M = M - (E_n - e^* \sin E_n),$$

$$\Delta E = \Delta M / (1 - e\cos E_n), \tag{3.12}$$

$$E_{n+1} = E_n + \Delta E .$$

Pro přibližnou formuli by mělo stačit nastavit hodnotu $tol = 10^{-6}$.

Pro uvedený algoritmus uvádí NASA chyby ve výpočtu rovníkových souřadnic rektascenze ***RA***, deklinace ***Dec*** a vzdálenost ***r*** viz tab. 3.1.

tabulka 3.1 Odchylky ve výpočtu[30]

	1800 -- 2050			3000 BC to 3000 AD		
	RA (arcsec)	**Dec.** (arcsec)	**r** (1000km)	**RA** (arcsec)	**Dec.** (arcsec)	**r** (1000km)
Merkur	15	1	1	20	15	1
Venuše	20	1	4	40	30	8
Země-Měsíc Bary centrum	20	8	6	40	15	15
Mars	40	2	25	100	40	30
Jupiter	400	10	600	600	100	1000
Saturn	600	25	1500	1000	100	4000
Uran	50	2	1000	2000	30	8000
Neptun	10	1	200	400	15	4000
Pluto	5	2	300	400	100	2500

3.3.4 Teorie s vysokou přesností

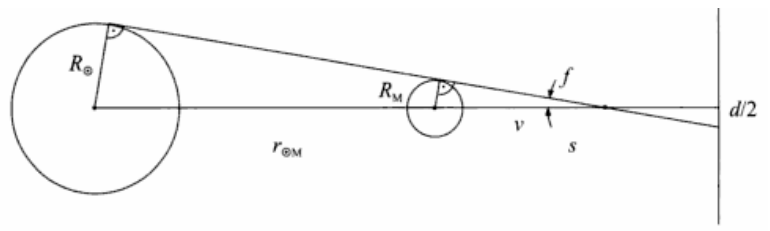
V dnešní době nabízejí počítače relativně vysoký výpočetní výkon, díky kterému lze realizovat složité teorie pro aproximaci různých perturbacích k získání maximální přesnosti v určování poloh vesmírných těles v naší Sluneční soustavě[27].

V roce 1982 dokončil Pierre Bretagnon první část své práce známé jako VSOP82. Tato teorie bylá následně vylepšena a označena jako VSOP87. Výsledky dosahují obdivuhodných přesností a garantují odchylku jedné úhlové sekundy v časovém rozmezí -2000 p.l. až 6000 n. l pro planety naší Sluneční soustavy[27] [32].

Mezi lety 1970 až 1990 vyvíjeli Jean Chapront a Michelle Chapront-Touzé teorii pro určení dráhy Měsíce známou jako ELP-2000 upravujících přes 20000 různých termů. Ani tato teorie však nezajistí přesnost polohy Měsíce na centimetr. NASA využívá tohoto modelu pro svoji numerickou integraci známou jako Jet Propulsion Laboratory Developement Ephemeris. Nejdříve byla zavedena pouze v modelu DE200, ale po vylepšení a upravení parametrů byla publikována v DE405. ELP-2000 je nyní velice oblíbená a je dostupná v různých verzích. Jednou z výhod je možnost ji upravit podle požadavku na přesnost výsledku[27] [32].

3.4 Astronomické jevy

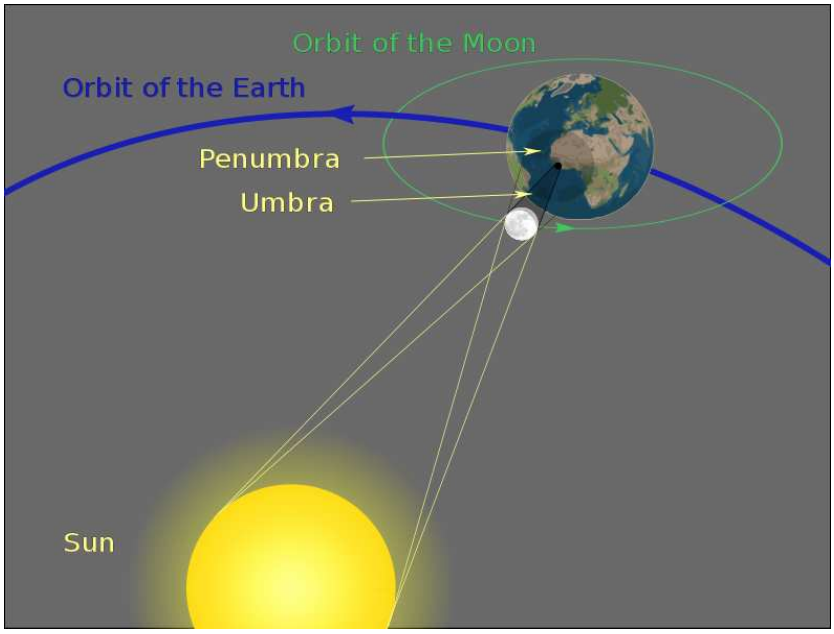
Mezi základní astronomické jevy se řadí např. pohyb planet po oběžné dráze jež vede ke střídání ročních období, rotace objektu kolem vlastní osy mající za následek střídání dne a noci nebo např. astronomické události, kdy dojde k zastínění jednoho vesmírného objektu druhým tzv. zatmění. K modelování zatmění Slunce nebo Měsíce je potřeba znát polohu Slunce, Země a Měsíce. Následně je možné geometricky interpretovat zatmění obr 3.3[27].



Obrázek 3.3 Zatmění Slunce (geometrická interpretace) [27]

3.4.1 Zatmění Slunce

Zatmění Slunce je astronomický jev, který nastane, když Měsíc vstoupí mezi Zemi a Slunce, takže jej částečně, nebo zcela zakryje. Taková situace se objevuje, jen pokud je Měsíc v novu a Slunce i Měsíc jsou při pohledu ze Země v jedné přímce. Na části Země, kde je zatmění pozorováno, dochází k výraznému setmění, ochlazení, kolem černého středu slunce je vidět výrazná záře sluneční koróny, objeví se hvězdy i některé planety a známé jsou také neobvyklé reakce zvířat. Tyto průvodní jevy v některých kulturách v minulosti vedly ke spojování události s náboženstvím a přisuzování mystických významů. V moderní době jsou však duchovní významy zatmění Slunce většinou odmítány v důsledku snadnosti pochopení jeho příčin[41] [42].



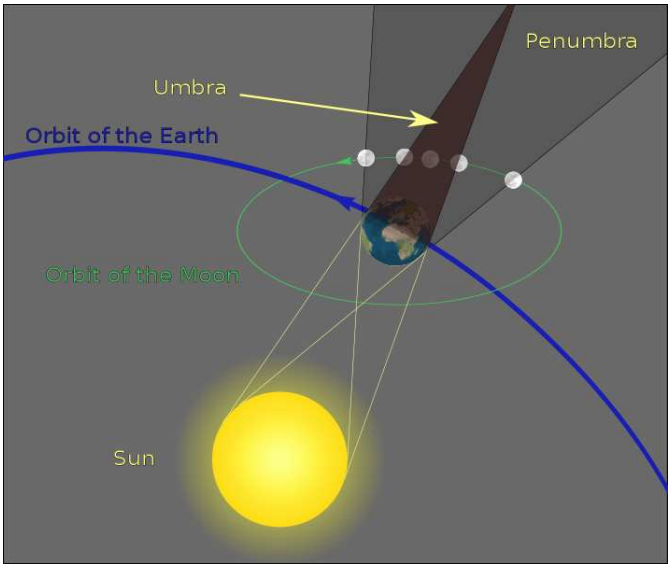
Obrázek 3.4 Zatmění Slunce[41]

Úplné zatmění Slunce je vzácný jev, který je možný díky tomu, že Slunce, které je čtyřistakrát větší než Měsíc je také čtyřistakrát dále od Země a navíc Měsíc i Země obíhají po eliptických drahách. Tyto dráhy všechna tři tělesa občas přivádějí do takového postavení, kdy je zakryt právě celý sluneční kotouč, takže je možno pozorovat korónu. Každý rok dojde ke dvěma až pěti případům, které jsou pozorovatelné z povrchu Země, pro jedno určité místo k tomu však dochází průměrně jen jednou za 360 let. Zatmění trvá jen několik minut (max. 7minut 31sekund). Oblast, podél které je úplné zatmění možno pozorovat, nebývá širší než 270 km. Zasažena je přibližně jen jedna setina zemského povrchu[41] [42].

- **Úplné zatmění** nastává, když Měsíc zcela zakryje Slunce. Velmi jasný sluneční disk je nahrazen černou plochou Měsíce a koróna, která má mnohem menší jas a za normálních okolností není vidět, je pozorovatelná. Úplné zatmění je pozorovatelné jen z oblasti Země, které se říká pás totality[41] [42].
- **Prstencové zatmění** je možné pozorovat, když Slunce a Měsíc jsou v jedné přímce, ale zdánlivá velikost Měsíce je menší než velikost Slunce. Z tohoto důvodu je ze Slunce vidět velmi jasný prstenec, nebo mezikruží okolo Měsíce[41] [42].
- **Částečné zatmění** se objevuje, když Slunce a Měsíc nejsou přesně v přímce, takže je Slunce zakryto jen z části. Toto zatmění je pozorovatelné z mnohem většího území a některá zatmění jsou pozorovatelná pouze jako částečná, protože oblast plného stínu leží mimo povrch Země[41] [42].
- **Kombinované (hybridní) zatmění** je vzácný stav, kdy z jednoho místa se zatmění jeví jako úplné a z jiného jako prstencové[41] [42].

3.4.2 Zatmění Měsíce

Zatmění Měsíce je astronomický jev, kdy měsíční kotouč je zastíněn planetou Zemí. Nastává při úplňku, pokud se Slunce, Země a Měsíc ocitnou v jedné přímce. Měsíc se ve stínu Země pohybuje rychlostí přibližně 1 km/s, a zatmění tak může trvat až 107 minut. Jde o běžněji pozorovatelný jev než zatmění Slunce, kdy dochází k zastínění části zemského povrchu Měsícem. Zatmění Měsíce nastává přibližně dvakrát až třikrát do roka[37] [43].



Obrázek 3.5 Zatmění Měsíce[37]

Zatmění Měsíce dělíme na tři druhy:

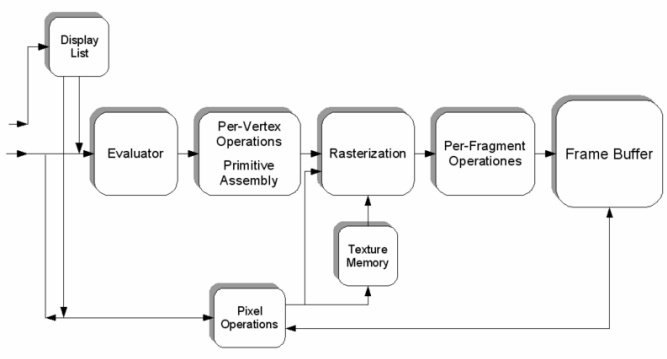
- **polostínové zatmění** – žádná část Měsíce není zcela zastíněna Zemí, hypotetičtí pozorovatelé na povrchu Měsíce by viděli částečné, ale nikoliv úplné zatmění Slunce[37] [43]
- **částečné zatmění** – část povrchu Měsíce je zcela zastíněna Zemí[37] [43]
- **úplné zatmění** – celý Měsíc je zcela zastíněn Zemí[37] [43]

4 Grafická rozhraní

4.1 OpenGL

OpenGL (Open Graphics Library) je soubor knihoven, které definují přenositelné API přes různé platformy a programovací jazyky. Toto API je spravováno skupinou Khronos Group a je vhodné pro návrh aplikací zpracovávajících 2D a 3D počítačovou grafiku. Rozhranní obsahuje přes 250 různých funkcí, pomocí kterých se z jednoduchých primitiv konstruují složité tří dimensionální scény. OpenGL vyvinulo Silicon Graphics Inc. (SGI) v roce 1992 a setkáváme se s ním často v aplikacích CAD nebo aplikacích zabývajících se simulováním virtuální reality, vizualizací vědeckých dat nebo např. leteckými simulátory. V OpenGL se také běžně programují počítačové hry[39] .

Základní funkcí OpenGL je zpracovávat jednotlivá primitiva jako jsou body, čáry nebo polygony a transformovat je na obrazové pixely. K tomu se využívá grafická pipeline nazývaná jako stavový automat OpenGL. Jednotlivé příkazy vkládají primitiva do grafické roury a mění stav, jakým způsobem budou tyto primitiva zpracovávána. OpenGL je nižší, procedurální API, kde je potřeba podrobně a přesně definovat jednotlivé kroky k úspěšnému zobrazení 3D scény. Design OpenGL vyžaduje dobrou znalost funkce grafické pipeline, ale dává prostor k vymýšlení nových vykreslovacích technik[39].



Obrázek 4.1 Grafická pipeline OpenGL[39]

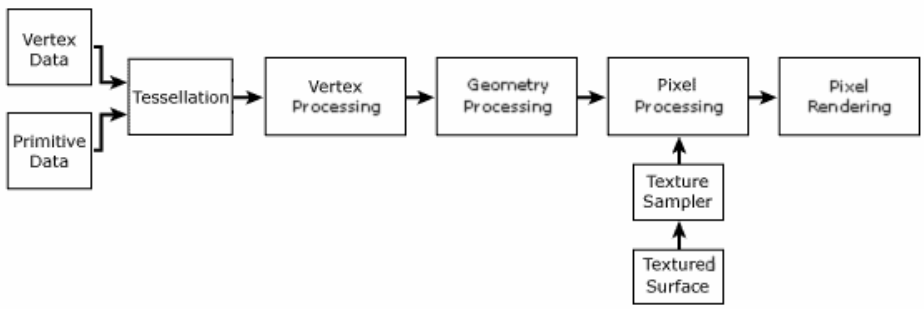
4.2 DirectX

Microsoft DirectX je kolekce knihoven definujících rozhraní (APIs) užívaných při návrhu aplikací zpracovávající multimedia, zejména počítačové hry nebo video.

Součástí kolekce jsou knihovny začínající slovem Direct jako např. Direct3D, DirectDraw, DirectMusic, DirectPlay, DirectSound aj. Název Direct*X* se užívá jako zkratka označující kolekci všech těchto rozhranní[13].

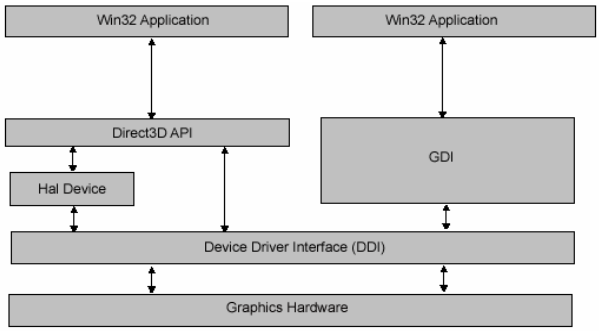
Direct3D (3D API) je využíváno k vývoji počítačových her pro platformu Microsoft Windows, Xbox, nebo Xbox 360. I přesto, že Direct3D je navrženo především k programování počítačových her, využívá se i v aplikacích typu CAD/CAM.

Jelikož je Direct3D nejpoužívanější komponenta z kolekce DirectX, často dochází k záměně významu slov Direct3D a DirectX.



Obrázek 4.2 Grafická pipeline DirectX[12]

Pokud jsou funkce podporované grafickou kartou, Direct3D využívá při zpracovávání grafické pipeline hardwarovou akceleraci obr. 4.3[13].



Obrázek 4.3 Podpora hardwarové akcelerace pomocí Direct3D[12]

4.2.1 HLSL

HLSL (High Level Shading Language) je jazyk využívaný společně s DirectX. Umožňuje přímý přístup k programování Direct3D grafické pipeline.

HLSL byl poprvé implementován v DirectX 8 ještě s možností využití kombinace s assembler instrukcemi. Od Direct3D 10 už je HLSL jediná možnost, jak generovat kód pro pixel a vertex shadery.

Na nejvyšší úrovni, data vstupují do grafické roury jako proud primitiv, který je následně zpracováván třemi úrovněmi shaderu. Všechny tyto úrovně jsou programovatelné pomocí HLSL.

Vertex shader se vyznačuje zpracováváním jednotlivých vertexů (per-vertex) a činnostmi jako jsou geometrické transformace, skinning, posunutí vertexů nebo výpočtů materiálů per-vertex. Rozklad vyšších primitiv na nižší se provádí ještě před vstupem do vertex shaderu. Výstupní hodnota musí být minimálně jedna, konkrétně nová pozice vertexu. Dalšími možnými výstupy jsou UV souřadnice textury, barva vertexu, hodnota osvětlení, faktor mlhy a jiné.

Geometry shader se vyznačuje zpracováváním jednotlivých primitiv (per-primitive) a metodami jako výběr materiálu, detekce obrysových hran nebo generováním nových primitiv. Tento shader je dostupný od verze DirectX 10 z důvodu zvýšení výkonu, a tím dosažení ještě věrohodnějších výsledků.

Pixel shader se vyznačuje zpracováváním jednotlivých obrazových bodů (per-pixel) jako je průhlednost, osvětlovací model aj. Hodnoty výstupu vertex shaderu vstupují jako vstupy do pixel shaderu. Výstupem pixel shaderu je pak zpravidla barva pixelu.

4.3 Analýza vhodnosti datových typů

Astronomická data přesahují přes několik řádů a vyžadují minimálně přesnost double. Naproti tomu běžné grafické adaptéry pracují z důvodu úspory výkonu s přesností single.

Ukázalo se, že hlavním problémem je nedostatečná mantisa, kdy v zásadě nefunguje např. sčítání velkého celého čísla a malého desetinného čísla. Přesnost je zaručena pouze na omezený počet cifer. Má-li mantisa 23b, přesnost čísla se dá předpokládat pouze na 7 cifer.

Inicializace Direct3D explicitně nastaví přesnost FPU na single a je tedy nutné při inicializaci nastavit FPU na přesnost double. Navíc funkce pro práci s maticemi a vektory v Direct3D nabízejí výpočty pouze s přesností single, takže dojde k přetypování při vstupu souřadnic do těchto funkcí. Direct3D 9 má v tomto ohledu oproti OpenGL, kde existují přetížené funkce pro různé typy, značnou nevýhodu.

Nicméně přetypování souřadnic nemá na výsledek nijak zásadní vliv, jelikož důležité výpočty se provádějí v matematickém modelu pomocí rozsahu double.

5 Návrh aplikace

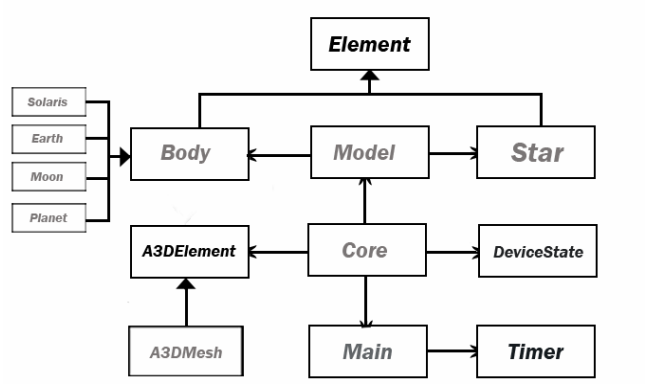
Moderní programovací techniky OOP umožňují analytický návrh aplikace tak, aby se mohl úkol např. rozdělit mezi více programátorů, kteří pracují na jednotlivých částech nezávisle. Další výhodou provedení analýzy před samotnou implementací je možnost návrh přizpůsobit očekávaným změnám, které mohou nastat v budoucnosti bez závažnějších problémů. Kromě jiného analýza pomáhá jako část dokumentace pro zjednodušení orientace v softwaru. Jelikož C/C++ není ryze jazyk OOP, ale spíše nižší procedurální jazyk, návrh tříd slouží hlavně k rozčlenění kódu.

Úlohu lze rozdělit na dvě části. Jedna část počítá reálnou pozici těles a ta druhá má za úkol výsledek graficky interpretovat. Model obsahuje elementy naší Sluneční soustavy např. Slunce, Zemi a Měsíc. Model se dá popsat měřítkem, časovým okamžikem a stavem jednotlivých elementů jako např. poloha v prostoru nebo rotace kolem osy. Pro každý prvek v modelu se určuje pozice v prostoru různě, ale v zásadě si jsou podobné. Podle stavu je třeba vykonstruovat 3D scénu a zobrazit na monitor.

5.1 Návrh tříd

Návrh je koncipován tak, aby usnadňoval rozšíření nebo znovupoužití jednotlivých částí programu. Snahou bylo oddělit výpočetní část modelu od vykreslování. Je toho dosaženo tak, že každému elementu v matematickém modelu přiřadíme jeden nebo více 3D elementů na grafickém adaptéru.

Základem matematického modelu je vektor v tří dimenzionálním souřadnicovém systému zaobalený ve třídě `Element`, ze které jsou odvozeny další konkrétnější objekty. Součástí tohoto modelu jsou vesmírná tělesa naší Sluneční soustavy (Slunce, Měsíc, planety) a hvězdy. Třída `Body` implementuje virtuální metodu `update` pro výpočet polohy v prostoru v závislosti na čase. Potomek třídy `Body` pak může jednoduše přetížit tuto metodu a implementovat vlastní algoritmus pro určení stavu (pozice tělesa). Třída `Star` reprezentuje stálice určené směrovým vektorem a jasností.



Obrázek 5.1 Návrh tříd

Všechny vesmírné objekty jsou sdruženy ve třídě `Model`, která řídí stav modelu a pomocí metody `update` přiřazuje pro určitý čas heliocentrické nebo geocentrické souřadnice pro všechny prvky v modelu. Třída `Body` si udržuje odkaz na centrální těleso a třída `Model` na těleso zájmu pozorovatele.

Třída `DeviceState` zaobaluje inicializaci Direct3D a udržuje stav související s grafickým adaptérem. Metoda `update` se dotazuje grafického adapteru pomocí metody Direct3D `TestCooperativeLevel`, zda-li je připraven k vykreslování.

Ke každému `Elementu` v modelu je potřeba přiřadit způsob vykreslení na obrazovku. Třída `A3DElement` reprezentuje interface implementovaný např. třídou `A3DMesh` s metodou `render`, která slouží k vykreslení drátěného modelu pomocí techniky shadow volume.

Třída `Main` vlastní referenci na časovač a zaobaluje hlavní smyčku aplikace. Tato třída také inicializuje hlavní okno a jednotlivé ovládací prvky.

5.2 Implementace

Rozhodování, jaké prostředky jsou nejvhodnější pro implementaci se odvíjí od smyslu aplikace. V zásadě se při 3D vizualizaci využívá ze dvou možných grafických API (DirectX nebo OpenGL), které umožňují v závislosti na grafickém adaptéru hardwarovou akceleraci.

Jelikož jsou tyto aplikace náročné na výpočetní výkon, je dobrou volbou zvolit programovací jazyk C nebo C++. Zdrojový kód se dá zkompilovat jak pod Windows, tak i pod Linux. Jedná se o velice robustní programovací jazyk od firmy Microsoft, který se nachází na rozhranní vyšších a nižších programovacích jazyků. DirectX je navíc přímo navrženo pro tento jazyk, a v zásadě se programuje pouze v C/C++.

Na druhou stranu OpenGL je možné programovat ve všech možných programovacích jazycích nejčastěji však C/C++, Java, Python, C# aj.

Vzhledem k předchozím zkušenostem autora s DirectX je aplikace vyvíjena právě pomocí DirectX 9.c SDK 2004, a proto byl vybrán programovací jazyk C/C++. Jednou z výhod DirectX oproti OpenGL je vyšší podpora Direct3D běžnými grafickými kartami. Nevýhodou je pak zejména omezení aplikace pouze na platformu Windows a nutnost znalosti C/C++.

5.3 Inicializace Direct3D

Typ `LPDIRECT3D9` je ukazatel zpřístupňující rozhraní Direct3D, které implementuje hlavní metodu `CreateDevice` s parametrem vyžadující krom jiného typ `LPDIRECT3DDEVICE9`, což je třída implementující metody pro 3D programování. Deklarační část aplikace:

```
#include <d3d9.h>

HWND              hWnd              = NULL; //the main window
LPDIRECT3D9       g_pD3D            = NULL; //the pointer to Direct3D interface
LPDIRECT3DDEVICE9 g_pd3dDevice      = NULL; //the pointer to device class

int CALLBACK WinMain(HINSTANCE hInst, HINSTANCE, LPSTR, INT );
LRESULT CALLBACK MsgProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam);
HRESULT initDevice(HWND hWnd);
void render();
```

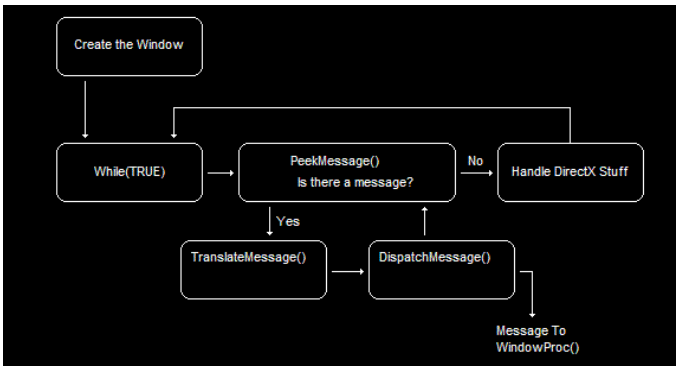
Při inicializaci Direct3D je třeba vytvořit hlavní okno aplikace `HWND`.

```
hWnd = CreateWindow(          "Klient",
                              "Solar Systém Simulator",
                              WS_OVERLAPPEDWINDOW|WS_CLIPCHILDREN,
                              0,
                              0,
                              1920,
                              1080,
                              NULL,
                              NULL,
                              hInst,
                              NULL );
```

K tomu je nutné zaregistrovat třídu `WNDCLASSEX`, jejímž hlavním parametrem je funkce typu `CALLBACK` zpracovávající zprávy (události) Windows. V deklarační části aplikace je deklarována jako např. prototyp `MsgProc`.

```
WNDCLASSEX newWc = {  sizeof(WNDCLASSEX),CS_PARENTDC, MsgProc, 0L,
                              0L,
                              GetModuleHandle(NULL),
                              NULL,
                              NULL,
                              (HBRUSH)COLOR_ACTIVECAPTION,
                              NULL,
                              "Klient",
                              NULL };
```

Součástí vstupního bodu `WinMain` je inicializace a registrace potřebných výše zmíněných objektů a hlavní smyčka aplikace zpracovávající zprávy Windows pomocí funkce `MsgProc`. Inicializace Direct3D se volá funkcí `initDevice` ještě před zavedením hlavní smyčky.



Obrázek 5.2 Hlavní smyčka aplikace[6]

Pokud není ve frontě žádná zpráva Windows ke zpracování, voláním funkce `render` se provede kód zpracovávající vykreslování do zásobníků na grafické kartě.

```
while( uMsg.message != WM_QUIT )
        {
                if( PeekMessage( &uMsg, NULL, 0, 0, PM_REMOVE ) )
                {
                        TranslateMessage( &uMsg );
                        DispatchMessage( &uMsg );
                }
         else
                    render();
        }
```

Vstupní bod aplikace `WinMain`, kde zeleně vyznačené komentáře zastupují místo pro výše zmíněné ukázky kódu:

```
int CALLBACK WinMain( HINSTANCE hInst, HINSTANCE, LPSTR, INT ) {

        WNDCLASSEX winClass;
        MSG        uMsg;
        ZeroMemory( &uMsg, sizeof(uMsg) );

        //create WNDCLASSEX here

        RegisterClassEx( &newWc );

        //create HWND here

        ShowWindow( hWnd, SW_SHOWDEFAULT );
        UpdateWindow( hWnd );

        initDevice( hWnd );

        //create main loop here

        UnregisterClass( "Klient", newWc.hInstance );

        return 0;
}
```

Události Windows obstarává funkce `MsgProc`, např. na zprávu `WM_DESTROY` aplikace ukončí svoji činnost. V parametrech `wParam` a `lParam` jsou mnohdy uloženy podrobnější vlastnosti jednotlivých zpráv.

```
LRESULT CALLBACK MsgProc( HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam ){
        switch( msg ) {
                case WM_DESTROY:
                        PostQuitMessage( 0 );
                return 0;
        };

        return DefWindowProc( hWnd, msg, wParam, lParam );
}
```

K inicializaci rozhraní `LPDIRECT3D9` je nutné nejdříve zavolat konstruktor `Direct3DCreate9`, kde se jako parametr udává verze knihovny a poté metodu `CreateDevice`, která vyžaduje kromě různých flagů také parametry typu `D3DDISPLAYMODE` a `D3DPRESENT_PARAMETERS`.

```
HRESULT initDevice(HWND hWnd){

        D3DDISPLAYMODE              d3ddm;
        D3DPRESENT_PARAMETERS       d3dpp;

        if( NULL == ( g_pD3D = Direct3DCreate9( D3D_SDK_VERSION ) ) )
                return E_FAIL;

        g_pD3D->GetAdapterDisplayMode( D3DADAPTER_DEFAULT, &d3ddm );

        ZeroMemory( &d3dpp, sizeof(d3dpp) );
        d3dpp.Windowed               = true;
        d3dpp.SwapEffect             = D3DSWAPEFFECT_COPY;
        d3dpp.BackBufferFormat       = d3ddm.Format;
        d3dpp.BackBufferWidth        = 1920;
        d3dpp.BackBufferHeight       = 1080;
        d3dpp.EnableAutoDepthStencil = TRUE;
        d3dpp.AutoDepthStencilFormat = D3DFMT_D24S8;
        d3dpp.Flags                  = 0;
        d3dpp.PresentationInterval   = D3DPRESENT_INTERVAL_ONE;
        d3dpp.BackBufferCount        = 1;

        if( FAILED( g_pD3D->CreateDevice(    D3DADAPTER_DEFAULT,
                                             D3DDEVTYPE_HAL,
                                             hWnd,
                                             D3DCREATE_HARDWARE_VERTEXPROCESSING|
                                             D3DCREATE_FPU_PRESERVE,
                                             &d3dpp,
                                             &g_pd3dDevice ) ) ){
                return E_FAIL;
        }
```

Funkce `render` je poslední částí aplikace, která inicializuje a v tomto případě pouze zobrazí čistý zásobník v hlavním okně aplikace. Nejdříve se vyplní zásobník jednou barvou pomocí metody `Clear` a následně se vykreslí nový snímek mezi voláním metody `BeginScene` a `EndScene`. Výsledek se poté presentuje na monitor pomocí metody `Present`.

```
void render(){
    g_pd3dDevice->Clear(0, 0, D3DCLEAR_TARGET, D3DCOLOR_XRGB(0,40,100), 1.0f, 0);

    g_pd3dDevice ->BeginScene();
       //drawing code here
    g_pd3dDevice ->EndScene();
    g_pd3dDevice ->Present(NULL, NULL, NULL, NULL);
}
```

5.4 Jádro aplikace

Návrh předpokládá znovu použitelnost některých prvků. Třída `Model` nevyžaduje žádných speciálních přídavných knihoven. Jejím hlavním účelem je pro zadaný čas určit polohu vesmírných těles. Třída `TDXState` zaobaluje interface ke grafickému adaptéru a kontroluje a spravuje jeho stav. Konstruktor této třídy v zásadě po úspěšné inicializaci Direct3D rozhranní vyvolá událost `onDeviceCreated`. Třída implementuje metodu `updateDevice`, která po dotazu na stav grafického adaptéru vyvolá jednu ze tří událostí: `onDeviceLost`, `onDeviceReset` nebo `onDeviceDisplay`. Uživatel třídy `TDXState` tedy musí deklarovat prototypy funkcí typu `CALLBACK` v deklarační části jádra a sám si zvolit co nastane v konkrétní událost.

Následující ukázka kódu je z deklarační části aplikace. Ukázka má ilustrovat kostru jádra, na kterou jsou následně nabalovány konkrétní prostředky, zdroje a vykreslovací techniky. Funkce typu `CALLBACK` reagují na jednotlivé události, takže si programátor může sám zvolit co nastane.

```
#include "TMain.h"
#include "TModel.h"
#include "TDXState.h"
#include "Abstract3DElement.h"

TModel*               model;
TMain*                application;
TDXState*             deviceState;
Abstract3DElement*    body[11];

int             CALLBACK WinMain( HINSTANCE hInst, HINSTANCE, LPSTR, INT);
LRESULT         CALLBACK MsgProc(HWND hWnd, UINT msg, WPARAM wParam,
                                 LPARAM lParam);
void            CALLBACK onTimer(double time);

void            CALLBACK onDeviceLost();
void            CALLBACK onDeviceReset();
void            CALLBACK onDeviceDisplay(LPDIRECT3DDEVICE9 g_pd3dDevice,
                                         double time);
void            CALLBACK onDeviceCreated(LPDIRECT3DDEVICE9 g_pd3dDevice);
void            CALLBACK onTimeChanged(int year,int month,int day,int hour,
                                         int minute, int second);
void            CALLBACK onModelInit(Star *star[]);
```

Vstupní bod aplikace se skládá z inicializace a z přiřazení funkcí k jednotlivým událostem. Následně se spustí hlavní smyčka aplikace a pokud je někdy ukončena, dojde k uvolnění paměti.

```
int CALLBACK WinMain( HINSTANCE hInst, HINSTANCE, LPSTR, INT ){

	model=new TModel(onModelInit);

	application=new TMain(hInst);
	application->createHandle(MsgProc);
	application->setOnTimer(onTimer);
	application->setOnUpdate(onUpdate);
	application->setOnTimeChanged(onTimeChanged);
	deviceState=new TDXState();
	deviceState->setOnDeviceCreated(onDeviceCreated);
	deviceState->initDevice(application->getHwnd());
	deviceState->setOnDisplay(onDeviceDisplay);
	deviceState->setOnReset(onDeviceReset);
	deviceState->setOnLost(onDeviceLost);

	application->mainLoop();

	delete application;
	delete model;
	delete deviceState;
	return 0;
}
```

Inicializace 3D geometrie (tedy alokace paměti na grafickém adaptéru) pro scénu je možná hned po vyvolání události `onDeviceCreated`.

```
void CALLBACK onDeviceCreated(LPDIRECT3DDEVICE9 g_pd3dDevice){
	init3DGeometry();
}
```

Při vyvolání události `onTimer` z hlavní smyčky aplikace jsou všechny prvky řízené časem přepočítány a připraveny k zobrazení. Nejdříve se přepočítá model v závislosti na čase, pak stav podle změn způsobené uživatelem a nakonec se připraví 3D objekty k vykreslení v závislosti na stavu modelu a stavu uživatelských změn. Proběhne-li vše v pořádku, je třeba se dotázat grafického adaptéru, zda-li je připraven a dojde-li k vyvolání události `onDeviceDisplay`, vykreslí se podle daného stavu 3D scéna na monitor.

```
void CALLBACK onTimer(double time){
	model->update(time*model->getSimulationSpeed());
	updateCamera(time);
	update3DElements(time);
	deviceState->updateDevice(time);
}
```

Událost `onDeviceDisplay` nastane, je-li grafický adaptér připraven k vykreslování a nenastala-li žádná chyba. Je tedy možné zapisovat do zásobníků a zobrazit výsledek na monitor.

```
void CALLBACK onDeviceDisplay(LPDIRECT3DDEVICE9 g_pd3dDevice,double time){

        g_pd3dDevice->BeginScene();

                render();

        g_pd3dDevice->EndScene();

        g_pd3dDevice->Present(NULL,NULL,NULL,NULL);
}
```

Další deklarovanou událostí je např. `onDeviceLost`. Tato událost indikuje nejčastěji přepnutí do režimu full screen za běhu programu. Pokud tato událost nastane, je třeba uvolnit všechny prostředky na grafickém adaptéru a pokusit se o restart, kdy úspěšný restart vyvolá událost `onDeviceReset`. Tato událost nastane, je-li úspěšně dokončen restart grafického adaptéru, a je tedy možné znovu zpřístupnit jeho prostředky.

5.5 Implementace shadow volume

K implementaci shadow volume v reálném čase je nejlepší využít hardwarovou akceleraci pro modelování drátěného modelu stínu a přídavný zásobník pro vytvoření masky stínů. Od verze DirectX 9 slouží k programování grafického adaptéru jazyk HLSL, kde se deklarují a implementují jednotlivé techniky popisující postup vertex a pixel shaderu pří vykreslování scény. Jazyk umožňuje pracovat s globálními proměnnými připravenými pomocí CPU (preprocessing) a předanými do paměti na grafickém adaptéru. To značně sníží počet operací prováděných GPU. Příklad deklarace globálních proměnných v jazyce HLSL:

```
float4   g_vAmbient;                // Ambient light color
float3   g_vLightView;              // View space light position
float4   g_vLightColor;             // Light color
float4   g_vShadowColor;            // Shadow volume color (for visualization)
float4   g_vMatColor;               // Color of the material
float4x4 g_mWorldView;              // World * View matrix
float4x4 g_mProj;                   // Projection matrix
float4x4 g_mWorldViewProjection;    // World * View * Projection matrix
texture  g_txScene;                 // texture for scene rendering
float    g_fFarClip;                // Z of far clip plane
float4x4 g_mView;                   // View matrix
float4x4 g_mWorld;                  // Object world matrix
float    g_mSunR;                   // Sun diameter
float4   g_mSunPosition;
```

V jazyce HLSL se nejdříve zapisují těla jednotlivých procedur popisujících jednotlivé pixel a vertex shadery a až na konci se deklarují techniky způsobem, že se určí, které procedury mají danou techniku vykonat a s jakým nastavením grafického adaptéru. Příklad kódu popisující práci vertex shaderu techniky `ShowShadowVolume` pro polostín[18]:

```
void VertShadowVolume(float4 vPos : POSITION,
                      float3 vNormal : NORMAL,
                      out float4 oPos : POSITION){

       float4 a=mul(float4(0.0,0.0,0.0,1.0), g_mWorld);
       float4 b=mul(vPos, g_mWorld);
       float3 d=normalize(a-b)*g_mSunR;
       float3 c=g_mSunPosition+d;
       float3 g_vLightView=mul(float4(c.x,c.y,c.z,1.0),g_mView);
       // Compute view space normal
       float3 N = mul( vNormal, (float3x3)g_mWorldView );
       // Obtain view space position
       float4 PosView = mul( vPos, g_mWorldView );
       // Light-to-vertex vector in view space
       float3 LightVecView = PosView - g_vLightView;
       // Perform reverse vertex extrusion
       if( dot( N, -LightVecView ) < 0.0f ){
              if( PosView.z > g_vLightView.z )
                     PosView.xyz += LightVecView*8800/LightVecView.z;
              else
                     PosView = float4( LightVecView, 0.0f );
       //Transform the position from view space to homogeneous  projection
              space oPos =  mul( PosView, g_mProj );
       }
       else
              oPos = mul( vPos, g_mWorldViewProjection );
}
```

Jelikož se tvoří geometrie pro polostín a plný stín různě, je třeba vytvořit obdobnou techniku i pro plný stín. Obě techniky však v případě potřeby využívají

stejného pixel shaderu pro zobrazení objemu. Pomocí pixel shaderu se určuje výsledná barva pixelu např. výpočet osvětlovacího modelu. Pro techniku `ShowShadowVolume` se však jedná pouze o průhlednost:

```
float4 PixShadowVolume() : COLOR0{

       return float4( g_vShadowColor.xyz, 0.1f );
}
```

Jako poslední je třeba deklarovat techniku na konci kódu a přiřadit jednotlivé vertex a pixel shadery. V deklaraci se též nastavují jednotlivé stavy grafického adaptéru.

```
technique ShowShadowVolume{

      pass P0{

               VertexShader = compile vs_1_1 VertShadowVolume();

               PixelShader  = compile ps_1_1 PixShadowVolume();

               CullMode = Ccw;

               AlphaBlendEnable = true;

               SrcBlend = SrcAlpha;

               DestBlend = InvSrcAlpha;

        // Disable writing to depth buffer

               ZWriteEnable = false;

               ZFunc = Less;

        // Setup stencil states

               StencilEnable = true;

               StencilRef = 1;

               StencilMask = 0xFFFFFFFF;

               StencilWriteMask = 0xFFFFFFFF;

               StencilFunc = Always;

               StencilZFail = Decr;

               StencilPass = Keep;

        }

        pass P1{

               VertexShader = compile vs_1_1 VertShadowVolume();

               PixelShader  = compile ps_1_1 PixShadowVolume();

               CullMode = Cw;

               StencilZFail = Incr;

        }
}
```

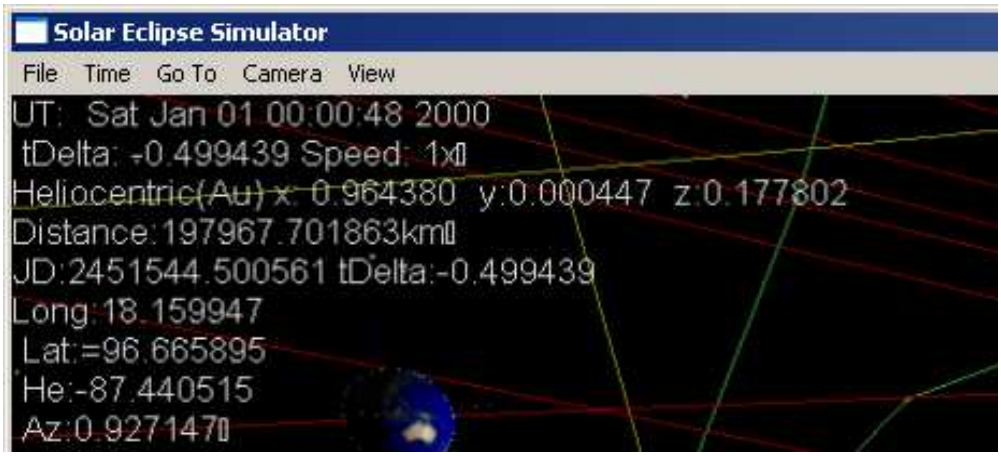
V jádru aplikace se inicializuje soubor obsahující jednotlivé techniky. V momentě, kdy je grafický adaptér připraven kreslit, je pak možné určit, jaká data se mají vykreslit kterou technikou s ohledem na stav globálních proměnných grafického adaptéru. Pomocí ukazatele `ID3DXEffect` (v následujícím kódu `g_pEffect`) lze předávat hodnoty grafickému adaptéru voláním metod `SetFloat`, `SetVector`, `SetMatrix`, `SetVector` aj. Konkrétní vykreslovací technika se nastaví voláním metody `SetTechnique`. `SetVertexDeclaration` specifikuje, jaký formát dat má grafický adaptér očekávat a kde v paměti se nachází. Metoda `render` objektu `Abstract3DElement` buď vykreslí drátěný model pomocí volání metody `DrawSubset` pro všechny části drátěného modelu, nebo může např. kreslit body, čáry nebo polygony pomocí volání metody `DrawPrimitive` pro jednotlivé „vertex buffery“.

```
void CALLBACK onDeviceDisplay(LPDIRECT3DDEVICE9 g_pd3dDevice,double time){

	g_pd3dDevice->BeginScene();

	g_pEffect->SetFloat("g_mSunR", rSun);
	g_pEffect->SetVector("g_mSunPosition", &center);
	g_pEffect->SetMatrix("g_mView", &matView);
	g_pEffect->SetVector("g_vLightView", &center);
	g_pd3dDevice->SetVertexDeclaration(  my_vertex_declaration ) ;
	g_pEffect->SetTechnique( "RenderAmbient" );
	for(UINT i=0;i<9;i++){
		mWorld=body[i]->getMatWorld();
		body[i]->render(g_pEffect,&(mWorld*matView*matProj),&(mWorld*matView),0);
	}

	g_pd3dDevice->Clear(0, 0, D3DCLEAR_STENCIL, D3DCOLOR_ARGB(0, 170,170,170),
			    1.0f, 0);
	g_pd3dDevice->SetVertexDeclaration(shadow_vertex_declaration);

	g_pEffect->SetTechnique("ShowShadowVolume");
	g_pEffect->SetVector("g_vShadowColor", &D3DXVECTOR4(255, 255, 255, 1.0f));
	for(UINT i=0;i<2;i++){
		mWorld=body[i]->getMatWorld();
		g_pEffect->SetMatrix( "g_mWorld", &mWorld ) ;
		body[i]->render(g_pEffect,&(mWorld*matView*matProj),&(mWorld*matView),1);
	}

	g_pd3dDevice->SetVertexDeclaration(  my_vertex_declaration ) ;
	g_pEffect->SetTechnique("Contrast");
	for(UINT i=0;i<2;i++){
		mWorld=body[i]->getMatWorld();
		body[i]->render(g_pEffect,&(mWorld*matView*matProj),&(mWorld*matView),0);
	}

	g_pd3dDevice->EndScene();

	g_pd3dDevice->Present(NULL,NULL,NULL,NULL);
}
```

6 Popis a návod

Aplikace je navržena pro operační systém Windows s podporou DirectX verze 9.c SDK 2004. K implementaci bylo využito programování pixel shaderu verze 2.0 a vertex shaderu verze 1.1 a je tedy nutné spouštět aplikaci na počítači s grafickou kartou podporující tyto verze.

Po spuštění aplikace se uživatel objeví v prostoru Sluneční soustavy asi 1AU od Slunce a kamerou natočenou směrem na sever. Souřadnice kamery se vztahují k lokálnímu systému místu pozorování, které lze uživatelsky měnit pomocí výběru *Go To* v horním menu aplikace. Po spuštění je explicitně nastaveno Slunce jako výchozí místo pozorování. Informace o stavu se vypisují hned pod hlavním menu v levém horním rohu hlavního okna aplikace.



Obrázek 6.1 Výpis stavu na obrazovku

UT je zkratka pro univerzální čas a udává, v jakém datu a časovém okamžiku se model nachází.

TDelta značí počet dní uplynulých od epochy J2000, tedy 1.1.2000 12:00, kde číslo za desetinnou čárkou značí část dne.

Souřadnice x,y,z jsou heliocentrické souřadnice centrálního objektu vztažené k rovině ekliptiky značené v jednotkách AU.

Distance je vzdálenost od středu vybraného tělesa k pozici pozorovatele značená v kilometrech.

JD je zkratka pro Juliánské datum.

Lattitude udává zeměpisnou šířku průniku průvodiče popisující místo pozorovatele s povrchem tělesa.

Longitude udává zeměpisnou délku průniku průvodiče popisující místo pozorovatele s povrchem tělesa.

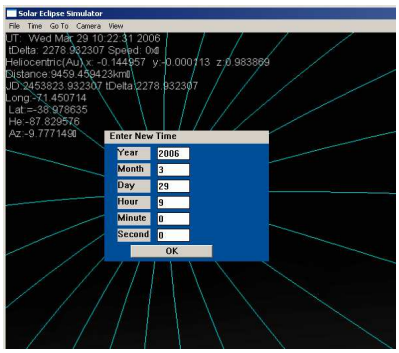
Height (výška nad obzorem) je úhel, který svírá směr pozorování s normálovým vektorem k rovině místa pozorování.

Azimut značí vertikální směr pohledu pozorovatele (sever, jih, východ nebo západ).

Během simulace je možné ovládat tok času. Je-li potřeba simulaci zrychlit, zpomalit, zastavit nebo dokonce pustit pozpátku, děje se tak pomocí hlavního menu v položce *Time* .

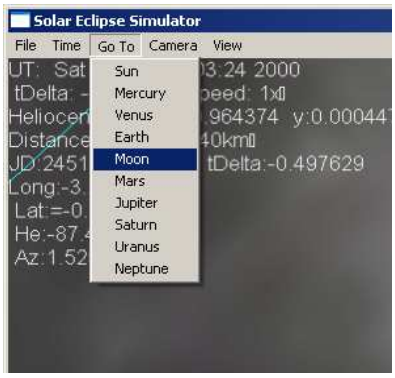
Obrázek 6.2 Ovládání času pomocí menu

K základním prvkům menu patří možnost zvolit časový okamžik, ze kterého se následně odvíjí simulace.



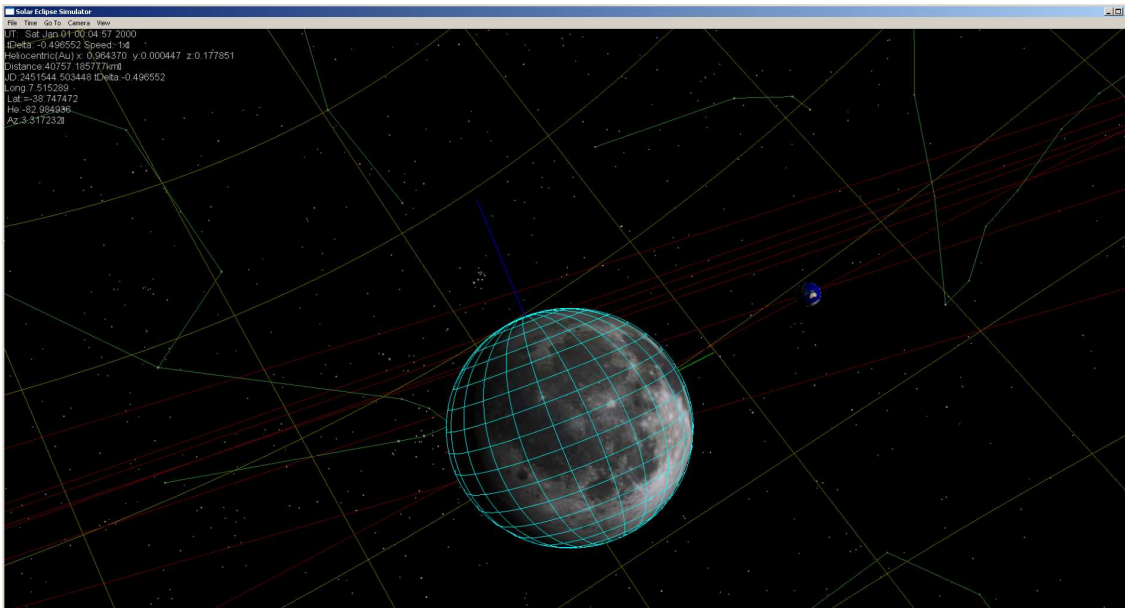
Obrázek 6.3 Okno pro zadání data a času

Jak bylo zmíněno výše, uživatel může volit mezi vesmírnými tělesy a zvolit tak nový počátek souřadnicového systému, což má za následek zdánlivý pohyb ostatních těles kolem zvoleného centrálního objektu.



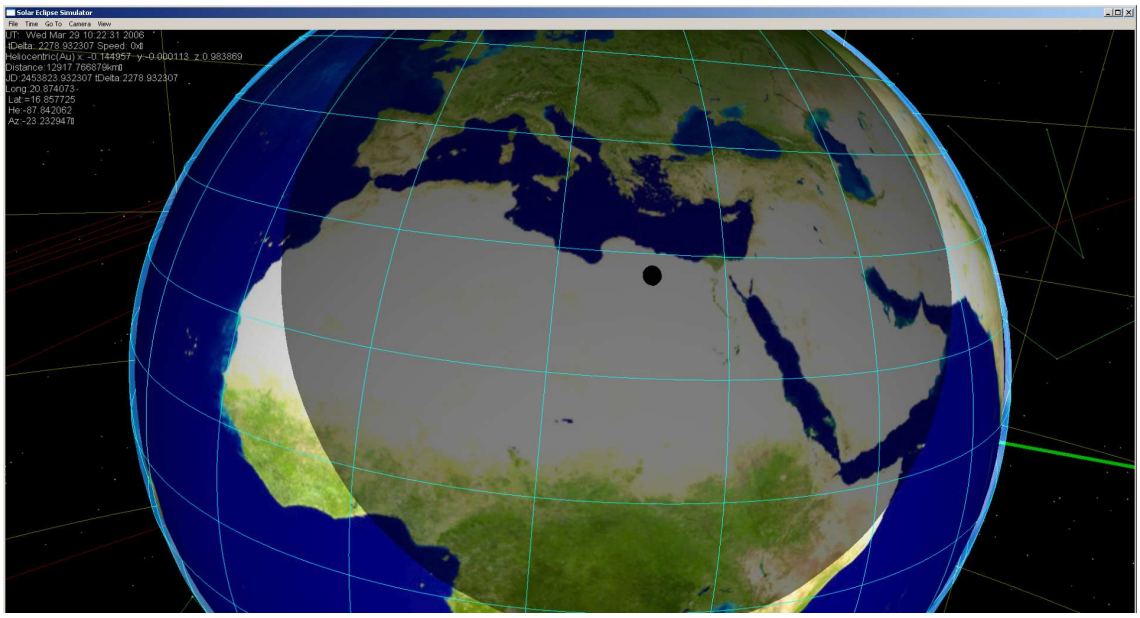
Obrázek 6.4 Výběr centrálního tělesa

Např. pro pozorování zatmění z Měsíce se jednoduše zvolí položka *Go To* a vybere se Moon. Podle potřeby lze nastavit směr kamery.



Obrázek 6.5 Pohled z oběžné dráhy Měsíce

Kamera se ovládá dvojím způsobem. Pomocí šipek na klávesnici se rotuje kolem tělesa, a tím se mění zeměpisná šířka a délka vzhledem k tomuto tělesu. Po stisku pravého tlačítka myši je možné ovládat kameru vztaženou k rovině pozorovatele. Pomocí kolečka pak lze měnit měřítko modelu obr. 6.6.

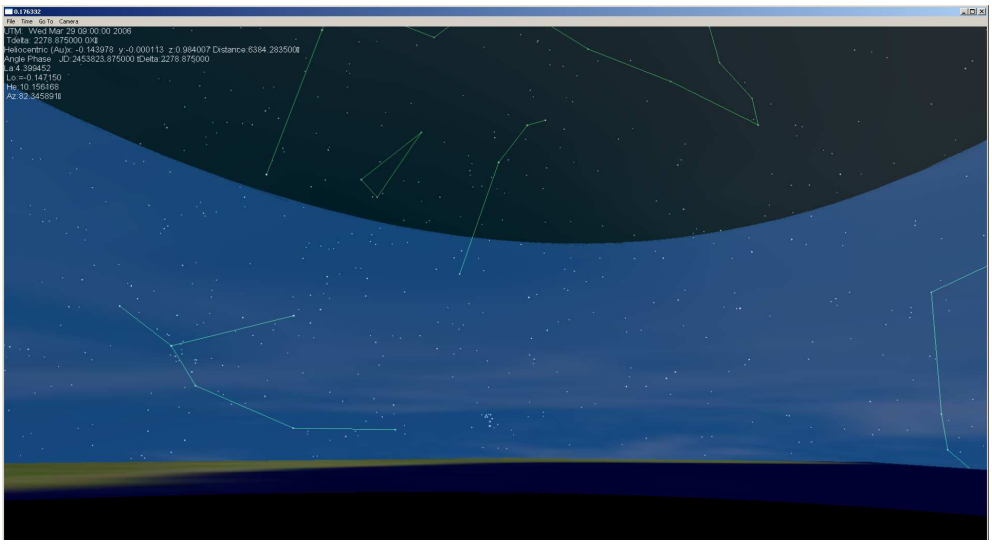


Obrázek 6.6 Zvětšení měřítka modelu

Další zajímavou vlastnosti aplikace je volba mezi synchronní a volnou kamerou, kdy synchronní kamera rotuje stejně rychle, jako se objekt otáčí kolem vlastní osy.

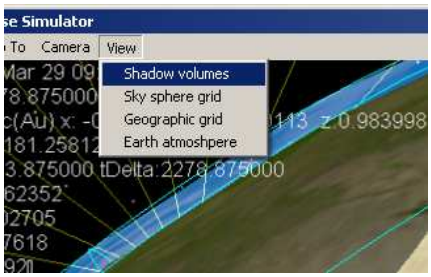
Obrázek 6.7 Synchronní kamera

Změna měřítka modelu na maximum zapříčiní, že se kamera ocitne na povrchu tělesa, a se synchronní kamerou tak dochází k simulaci zdánlivého pohybu hvězd nebo vesmírných těles. Při úplném zatmění Slunce dojde k setmění, a tak lze spatřit některé hvězdy, které by normálně viditelné nebyly. Umístění hvězd v aplikaci odpovídá skutečnosti a během zatmění Slunce lze pozorovat v jakých souhvězdích se objekty nachází obr. 6.8.



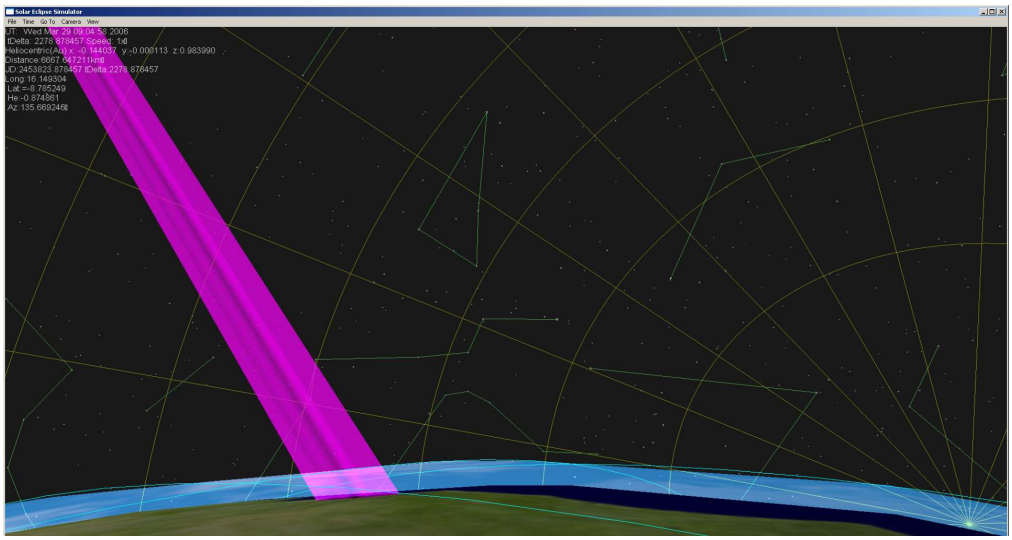
Obrázek 6.8 Viditelné hvězdy během úplného zatmění Slunce

Pro lepší orientaci v prostoru pomáhají pomocné křivky jako jsou např. mřížky nebo elipsy oběžných drah. Pomocné křivky lze zapínat/vypínat pomocí položky *View*. Jednou z možností je zapnout zobrazení objemu stínu.

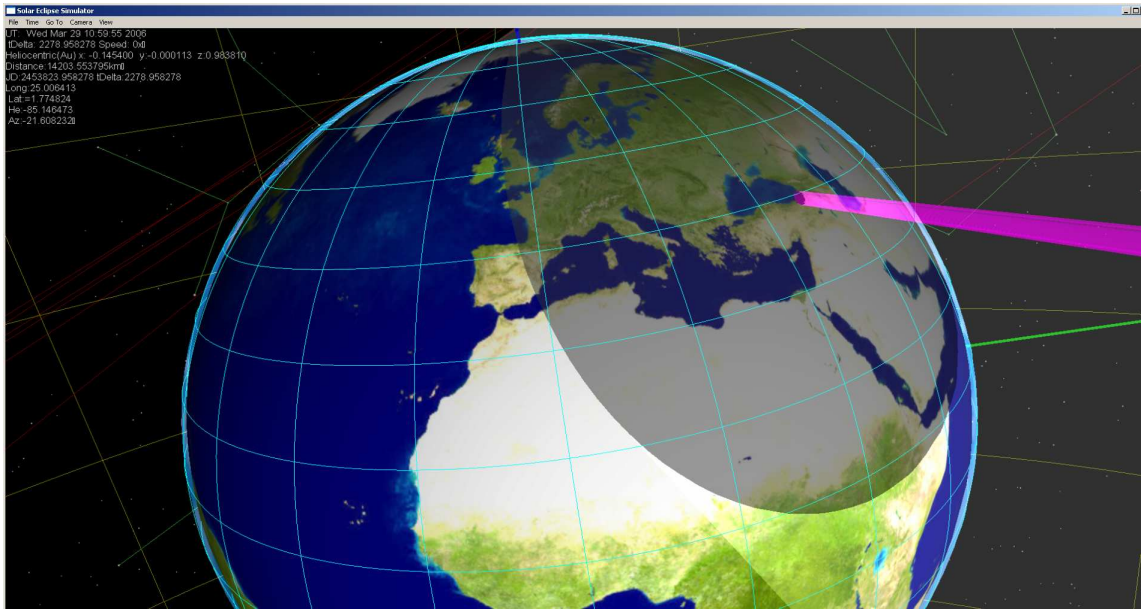


Obrázek 6.9 Vypnout/zapnout jednotlivé pomocné křivky

Výsledek výstupu aplikace zobrazující objem stínu při zatmění Slunce je viditelný na obr. 6.10 a 6.11.



Obrázek 6.10 Zobrazení objemu stínu vrženého Měsícem na Zemi



Obrázek 6.11 Zobrazení objemu stínu vrženého Měsícem na Zemi

7 Zhodnocení

Během implementace bylo testováno několik modelů, které se lišily pouze ve způsobu určování polohy planet. Zpočátku se zdálo, že základní empirický model bude pro modelování zatmění Slunce dostačující, ale při kontrole bylo zjištěno významné odchylky. Zejména odchylky polohy Země i Měsíce způsobovaly značné problémy a nedostatky. Proto bylo nutné vyzkoušet i jiné, přesnější metody. I přesto, že nejpřesnějších výsledků lze dosáhnout hlavně pomocí numerických integrací, byly z důvodu náročnosti těchto metod na výpočetní výkon nejdříve testovány teorie známé jako VSOP87 a ELP-2000 82b. Po jejich otestování bylo rozhodnuto, že přesnost výpočtů je dostačující a jelikož se na slabších počítačích projevilo výrazné zpomalení chodu aplikace, k testování numerických metod nebyl důvod.

Souřadnice se nejlépe kontrolují pomocí webového rozhranní známého jako HORIZONS dostupné na stránkách NASA - Jet Propulsion Laboratory. Uživatel zadá místo, objekt a čas pozorování, nastaví výstup, který např. zobrazí souřadnice Země vzhledem k Slunci ve formě zápisu: Juliánské datum, občanské datum, souřadnice x,y,z.

```
*******************************************************************************
$$SOE
2453823.500000000 = A.D. 2006-Mar-29 00:00:00.0000 (CT)
 X =-9.848123950489781E-01 Y =-1.375738075265339E-01 Z =-1.126348522358095E-04
2453823.541666667 = A.D. 2006-Mar-29 01:00:00.0000 (CT)
 X =-9.847231738322557E-01 Y =-1.382862709269494E-01 Z =-1.126531601325542E-04
2453823.583333333 = A.D. 2006-Mar-29 02:00:00.0000 (CT)
 X =-9.846334390744683E-01 Y =-1.389986613779873E-01 Z =-1.126714778379834E-04
2453823.625000000 = A.D. 2006-Mar-29 03:00:00.0000 (CT)
 X =-9.845431908432705E-01 Y =-1.397109784786250E-01 Z =-1.126898020232440E-04
2453823.666666667 = A.D. 2006-Mar-29 04:00:00.0000 (CT)
 X =-9.844524292070219E-01 Y =-1.404232218279647E-01 Z =-1.127081293619790E-04
2453823.708333333 = A.D. 2006-Mar-29 05:00:00.0000 (CT)
 X =-9.843611542347873E-01 Y =-1.411353910252400E-01 Z =-1.127264565311063E-04
2453823.750000000 = A.D. 2006-Mar-29 06:00:00.0000 (CT)
 X =-9.842693659963326E-01 Y =-1.418474856698234E-01 Z =-1.127447802113288E-04
2453823.791666667 = A.D. 2006-Mar-29 07:00:00.0000 (CT)
 X =-9.841770645621256E-01 Y =-1.425595053612329E-01 Z =-1.127630970878504E-04
```

Obrázek 7.1 Výpis heliocentrických souřadnic Země během zatmění Slunce[2]

Při porovnávání různých metod nevyužívajících numerického integrování s výsledky NASA, bylo nejlepších výsledků dosaženo pomocí teorie VSOP87 pro planety naší Sluneční soustavy a ELP-2000 82b pro Měsíc. Algoritmus pro přibližné pozice hlavních planet ze stránek NASA nedosahoval dostatečné přesnosti. Odchylka začátku zatmění se pohybovala řádově v hodinách, což se projevilo na přesnosti dopadu stínu na povrch. Z počátku se zdálo, že chyba je někde v návrhu, ale po otestování přesnějších metod se ukázalo, že vše pracuje správně.

tabulka 7.1 Heliocentrické souřadnice Země podle VSOP87

JD	X	Y	Z
2453823.500000000	-0.984820	-0.137568	-0.000112
2453823.541666667	-0.984730	-0.138281	-0.000113
2453823.583333333	-0.984641	-0.138993	-0.000113
2453823.625000000	-0.984551	-0.139706	-0.000113
2453823.666666667	-0.984460	-0.140418	-0.000113
2453823.708333333	-0.984368	-0.141130	-0.000113
2453823.75000000	-0.984277	-0.141842	-0.000113
2453823.791666667	0.984184	-0.142554	-0.000113

Stejným způsobem se dají kontrolovat souřadnice i jiných vesmírných těles jako např. Měsíce obíhajícího kolem Země.

```
*******************************************************************************
$$SOE
2453823.500000000 = A.D. 2006-Mar-29 00:00:00.0000 (CT)
 X =-9.824101432964516E-01 Y =-1.374863890754359E-01 Z =-1.211535647018240E-04
2453823.541666667 = A.D. 2006-Mar-29 01:00:00.0000 (CT)
 X =-9.823216481049210E-01 Y =-1.381726216923839E-01 Z =-1.187439716067557E-04
2453823.583333333 = A.D. 2006-Mar-29 02:00:00.0000 (CT)
 X =-9.822329068534530E-01 Y =-1.388587936662554E-01 Z =-1.163337005733671E-04
2453823.625000000 = A.D. 2006-Mar-29 03:00:00.0000 (CT)
 X =-9.821439193643724E-01 Y =-1.395449075457192E-01 Z =-1.139230226953899E-04
2453823.666666667 = A.D. 2006-Mar-29 04:00:00.0000 (CT)
 X =-9.820546854240698E-01 Y =-1.402309658745800E-01 Z =-1.115122088568931E-04
2453823.708333333 = A.D. 2006-Mar-29 05:00:00.0000 (CT)
 X =-9.819652047831187E-01 Y =-1.409169711912309E-01 Z =-1.091015296809475E-04
2453823.750000000 = A.D. 2006-Mar-29 06:00:00.0000 (CT)
 X =-9.818754771563993E-01 Y =-1.416029260281093E-01 Z =-1.066912554782592E-04
2453823.791666667 = A.D. 2006-Mar-29 07:00:00.0000 (CT)
 X =-9.81785502232371E-01 Y =-1.422888329111552E-01 Z =-1.042816561961140E-04
```

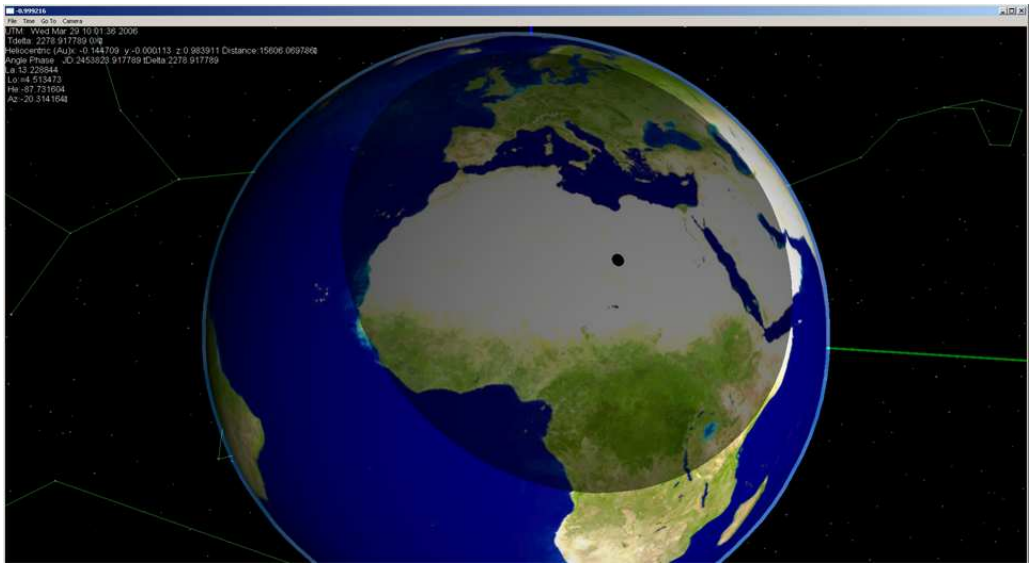
Obrázek 7.2 Výpis heliocentrických souřadnic Měsíce během zatmění Slunce[2]

Opět byla provedena kontrola souřadnic pro stejné časové okamžiky.

tabulka 7.2 Heliocentrické souřadnice Měsíce podle ELP-2000 82b

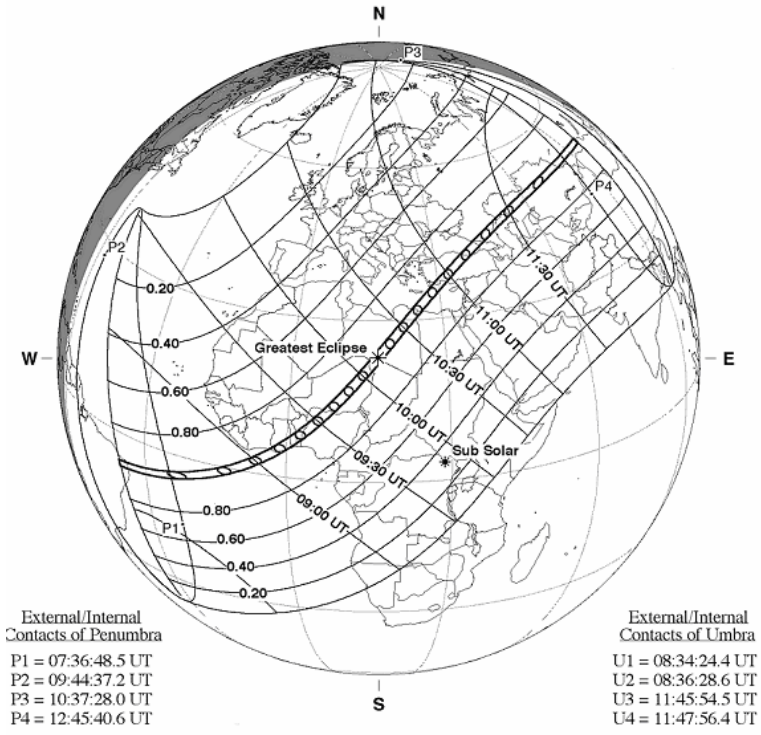
JD	X	Y	Z
2453823.500000000	-0.982417	-0.137481	-0.000121
2453823.541666667	-0.982329	-0.138176	-0.000119
2453823.583333333	-0.982240	-0.138853	-0.000116
2453823.625000000	-0.982151	-0.139540	-0.000114
2453823.666666667	-0.982062	-0.140226	-0.000111
2453823.708333333	0.981973	-0.140912	-0.000109
2453823.750000000	0.981883	-0.141598	-0.000107
2453823.791666667	0.981793	-0.142284	-0.000104

Pří kontrole dopadu stínu na povrch Země bylo odhadnuto odchýlení se od NASA v řádech pár minut. Jednou z příčin je vynechání některých termů v teorii ELP-2000 82b, kde v dokumentaci je uvedena odchylka v deklinaci až 800 úhlových sekund[33].



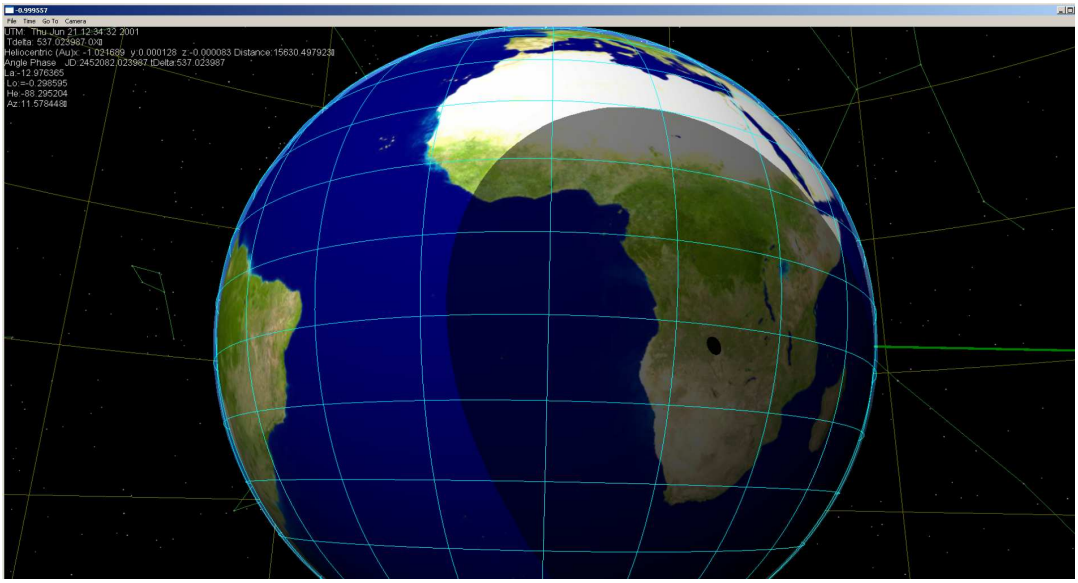
Obrázek 7.3 Porovnání výsledku dopadu stínu na povrch Země dne 29. 3. 2006

Při porovnávání výstupu programu NASA daná odchylka odpovídala. Na obrázku je průběh stínu na zemském povrchu s časovými údaji, kdy se polostín (P1-P4) a plný stín (U1-U2) poprvé a naposledy dotkne povrchu Země.



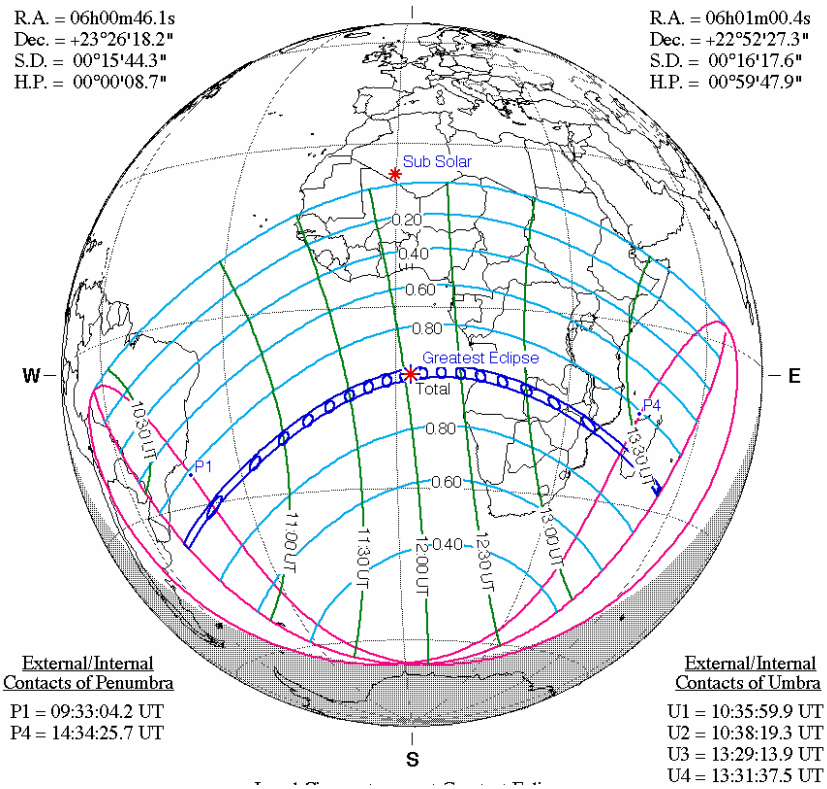
Obrázek 7.4 Průběh zatmění Slunce podle NASA dne 29. 3. 2006[4]

Pro jistotu bylo porovnáváno více různých případů zatmění Slunce. Na základě dopadu stínu na povrch Země lze posoudit správnost návrhu, jelikož je nutné správně simulovat pohyb těles a jejich rotace kolem os.



Obrázek 7.5 Porovnání výsledku dopadu stínu na povrch Země dne 22. 6. 2001

I v tomto případě průběh zatmění Slunce přibližně odpovídal výsledkům NASA. Odchylka se opět pohybovala v předpokládaném rozmezí několika minut. Z toho lze usoudit, že model pracuje správně.



Obrázek 7.6 Průběh zatmění Slunce podle NASA dne 22. 6. 2001[5]

8 Závěr

Přesnost dopadu stínu na povrch země závisí především na přesnosti výpočtu polohy Měsíce a Země. K tomu je důležité použít jednu z metod, které jsou sice daleko přesnější, ale mnohem více náročné na výkon počítače. Velice oblíbené se staly metody VSOP87 a ELP-2000, protože dosahují přijatelných přesností a jsou volně dostupné. Tyto metody jsou použity i v této aplikaci a jsou dostupné z [31] a [33]. Výpočty jsou však již náročnější, a tak na pomalejších počítačích nastávají problémy s nedostatkem výkonu CPU a GPU. Porovnáním výsledků s výsledky NASA potvrdilo funkčnost aplikace a dosažení předem stanovených cílů.

Zpracování stínů v počítačové 3D grafice patří mezi pokročilé techniky náročné na výkon GPU i CPU a pro vykreslování v reálném čase se pro tento problém nabízejí pouze dvě známe techniky. Technika shadow volume však lépe vystihuje podstatu problému zatmění a po přizpůsobení vlastností světelného zdroje v modelu je i přes větší náročnost vhodnější.

Výsledkem práce je aplikace navržená v programovacím jazyce C/C++ s podporou DirectX 9.c SDK, která pro zadaný čas vypočítá polohu vesmírných těles v modelu s možností simulovat pohyb planet ve Sluneční soustavě od tohoto data. Návrh aplikace je snadno přizpůsobitelný dalším změnám nebo vylepšením. Jednou z nich je přidat více vesmírných objektů. Hlavním cílem však bylo zaměřit se na problém zatmění Slunce nebo Měsíce.

Aplikace zobrazuje sféry interpretující vesmírná tělesa potažené texturami a simuluje pohyb odpovídající trajektoriím vesmírných těles. Dojde-li k astronomickému jevu zvaném zatmění, pomocí geometrické interpretace aplikace vykreslí stín dopadající na povrch s možností zobrazení objemu stínu. Pomocí hardwarové akcelerace bylo dosaženo vykreslování v reálném čase, a tak se podařilo splnit požadavky kladené na aplikaci. Pro lepší navigaci bylo nutné navíc přidat různé prvky k zjednodušení orientace v prostoru.

8.1 Navrhovaná vylepšení

Jednou z možností jak dosáhnout lepších výsledků na starších PC je optimalizovat pomocí zřetězení a použít tzv. dvojího zápisu. Jakmile musí GPU čekat na složité výpočty matematického modelu, značně se to projeví na plynulosti simulace.

Algoritmy modelu VSOP87 a ELP-2000 82b nabízejí možnost částečně paralelizovat úlohu, a na více procesorovém počítači by to znamenalo jisté zlepšení v optimalizaci.

Technika shadow volume je vskutku náročná a i přes hardwarovou akceleraci jednoznačně konzumuje značnou část GPU. Možností jak dosáhnout přijatelných výsledků je celá řada, avšak i přes značná zjednodušení znamenalo 9 objektů spolu se dvěmi světelnými zdroji významnou zátěž. Moderní grafické karty nabízejí nový geometry shader určený právě pro tento případ zpracování stínu, což ale neřeší optimalizaci pro starší počítače.

Nyní model nabízí měnit vzdálenost od centrálního objektu pomocí změny měřítka. Implementace modelu pro světelné efekty a vytvořit např. korónu by společně s funkcí dalekohledu tvořilo velice atraktivní prvek aplikace. Avšak Implementace grafických technik pomocí hardwarové akcelerace je určena pro experty v 3D programování a bez použití 3D engine je implementace v rozsáhlejších aplikacích velice náročná.

Příloha A Elementy drah validní pro epochu J2000

	a AU, AU/Cy	***e*** Rad, rad/cy	***I*** Deg, deg/Cy	***L*** deg, deg/Cy	long.peri. deg, deg/Cy	long. node. deg, deg/Cy
Mercury	0.38709927	0.20563593	7.00497902	252.25032350	77.45779628	48.33076593
	0.00000037	0.00001906	-0.00594749	149472.67411175	0.16047689	-0.12534081
Venus	0.72333566	0.00677672	3.39467605	181.97909950	131.60246718	76.67984255
	0.00000390	-0.00004107	-0.00078890	58517.81538729	0.00268329	-0.27769418
EM Bary	1.00000261	0.01671123	-0.00001531	100.46457166	102.93768193	0.0
	0.00000562	-0.00004392	-0.01294668	35999.37244981	0.32327364	0.0
Mars	1.52371034	0.09339410	1.84969142	-4.55343205	-23.94362959	49.55953891
	0.00001847	0.00007882	-0.00813131	19140.30268499	0.44441088	-0.29257343
Jupiter	5.20288700	0.04838624	1.30439695	34.39644051	14.72847983	100.47390909
	-0.00011607	-0.00013253	-0.00183714	3034.74612775	0.21252668	0.20469106
Saturn	9.53667594	0.05386179	2.48599187	49.95424423	92.59887831	113.66242448
	-0.00125060	-0.00050991	0.00193609	1222.49362201	-0.41897216	-0.28867794
Uranus	19.18916464	0.04725744	0.77263783	313.23810451	170.95427630	74.01692503
	-0.00196176	-0.00004397	-0.00242939	428.48202785	0.40805281	0.04240589
Neptune	30.06992276	0.00859048	1.77004347	-55.12002969	44.96476227	131.78422574
	0.00026291	0.00005105	0.00035372	218.45945325	-0.32241464	-0.00508664
Pluto	39.48211675	0.24882730	17.14001206	238.92903833	224.06891629	110.30393684
	-0.00031596	0.00005170	0.00004818	145.20780515	-0.04062942	-0.01183482

A - 1 Elementy dráhy pro epochu J2000[29]

Literatura

[1] BRENNER, Martin. 2009. *Interaktivní mapa hvězdné oblohy jako dynamické webové stránky*. Liberec: TUL.

[2] CALTECH/JPL. 2006. *Horizons web-interface* [online]. Solar System Dynamic Group, [vid. 2011-4-15]. Dostupné z: http://ssd.jpl.nasa.gov/horizons.cgi.

[3] EGLER, Robert A. 2006. *A Brief explanation of orbital elements*. Department of Physics editováno 2006-3-13, [vid. 2011-4-20]. Dostupné z: http://www.physics.ncsu.edu/courses/astron/orbits.html.

[4] ESPENAK, F. 2005. *NASA eclipse web site* [online]. NASA's GSFC, [vid. 2011-4-15]. Dostupné z: http://eclipse.gsfc.nasa.gov/OH/image1/SE2006Mar29-Fig2.GIF.

[5] ESPENAK, F. 2005. *NASA eclipse web site* [online]. NASA's GSFC, [vid. 2011-4-15]. Dostupné z: http://eclipse.gsfc.nasa.gov/SEplot/SEplot2001/SE2001Jun21T.GIF.

[6] DIRECTXTUTORIAL.COM. 2006. *Lesson 4: The Real-Time Message Loop* [online]. [vid. 2011-4-15]. Dostupné z: http://www.directxtutorial.com/Tutorial9/A-Win32/dx9A4.aspx.

[7] EMERY, James D. 2010. *Scientific calculating, programming, and writing* [online]. [Vid. 2011-4-20]. Dostupné z: http://stem2.org/je/sciprog.pdf.

[8] HUDEC, Bohuslav, Jaroslav SLOUP a Petr FELKEL. 2007. *Barva a světlo v počítačové grafice* [online]. Praha: ČVUT, [vid. 2011-4-20]. Dostupné z: http://www.google.cz/url?sa=t&source=web&cd=1&ved=0CBgQFjAA&url=http%3A%2F%2Fedux2.felk.cvut.cz%2Fmodules%2Fedux%2Fget_file_from_dms.php%3Ffunction%3Dview%26FileID%3D914%26source%3Dp&rct=j&q=Barva%20a%20sv%C4%9Btlo%20v%20po%C4%8D%C3%ADta%C4%8Dov%C3%A9%20grafice&ei=ZhPVTbaFKsKr-gbaoO38Cw&usg=AFQjCNG6_LMzttv8I-UPUaOY8j5APDevrA&sig2=-BXEgkBiE2gX6bSEiERz9A&cad=rja.

[9] LEE, Ruen-Rone a Shao-Chen PENG. 2007. *Metod and apparatus for generating shadow by using an adaptive shadow volume algorithm* [online]. Patent. USA US 2007/0165029. [Vid. 2011-4-20]. Dostupné z: http://www.google.com/patents?id=S6qAAAAAEBAJ.

[10] LUNA, FRANK D. 2006. *Introduction to 3D game programming with DirectX 9.0c: a shader approach.* Plano: Wordware Publishing, Inc. ISBN-13: 978-1-59822-016-2. ISBN-10:1-59822-016-0 (pbk.).

[11] MICROSOFT. 2011. *Coordinate systems (Direct3D 9)* [online]. Microsoft editováno 2011-2-8, [vid. 2011-4-20]. Dostupné z: http://msdn.microsoft.com/en-us/library/bb204853(v=VS.85).aspx.

[12] MICROSOFT. 2011. *Direct3D architecture (Direct3D 9)* [online]. Microsoft, editováno 2011-2-8 [vid. 2011-4-20]. Dostupné z: http://msdn.microsoft.com/en-us/library/bb219679(v=VS.85).aspx.

[13] MICROSOFT. 2011. *Direct3D graphics* [online]. Microsoft, editováno 2011-2-8 [vid. 2011-4-20]. Dostupné z: http://msdn.microsoft.com/en-us/library/bb219837(v=vs.85).aspx.

[14] MICROSOFT. 2011. *Face and vertex normal vectors (Direct3D 9)* [online]. Microsoft editováno 2011-2-8, [vid. 2011-4-20]. Dostupné z: http://msdn.microsoft.com/en-us/library/bb173380(v=VS.85).aspx.

[15] MICROSOFT. 2011. *Lights and materials (Direct3D 9)* [online]. Microsoft editováno 2011-2-8, [vid. 2011-4-20]. Dostupné z: http://msdn.microsoft.com/en-us/library/bb174694(v=VS.85).aspx.

[16] MICROSOFT. 2011. *Matrix.RotateYawPitchRoll(Single,Single,Single) method (Microsoft.DirectX)* [online]. Microsoft, [vid. 2011-4-20]. Dostupné z: http://msdn.microsoft.com/en-us/library/ms918183.aspx.

[17] MICROSOFT. 2011. *Projection transform (Direct3D 9)* [online]. Microsoft editováno 2011-8-2, [vid. 2011-4-20]. Dostupné z: http://msdn.microsoft.com/en-us/library/bb147302(v=VS.85).aspx.

[18] MICROSOFT. 2011. *ShadowVolume sample* [online]. Microsoft, [vid. 2011-4-20]. Dostupné z: http://msdn.microsoft.com/en-us/library/ee418792(v=VS.85).aspx.

[19] MICROSOFT. 2011. *Transforms (Direct3D 9)* [online]. Microsoft editováno 2011-2-8, [vid. 2011-4-20]. Dostupné z: http://msdn.microsoft.com/en-us/library/bb206269(v=VS.85).aspx.

[20] MICROSOFT. 2011. *Viewports and clipping (Direct3D 9)* [online]. Microsoft editováno 2011-2-8, [vid. 2011-4-20]. Dostupné z: http://msdn.microsoft.com/en-us/library/bb206341(v=VS.85).aspx.

[21] MICROSOFT. 2011. *View transform (Direct3D 9)* [online]. Microsoft editováno 2011-2-8, [vid. 2011-4-20]. Dostupné z: http://msdn.microsoft.com/en-us/library/bb206342(v=VS.85).aspx.

[22] MICROSOFT. 2011. *World transform (Direct3D 9)* [online]. Microsoft editováno 2011-2-8, [vid. 2011-4-20]. Dostupné z: http://msdn.microsoft.com/en-us/library/bb206365(v=VS.85).aspx.

[23] MIŠÁK, Petr. 2006. *Simulace výroby firmy Bohemian Jewelery, s.r.o.*. Pardubice: Univerzita Pardubice. Dostupné z: http://dspace.upce.cz/bitstream/10195/21073/1/D16323.pdf.

[24] MURRAY, C. D. a S. F. DERMOTT. 1999. *Solar system dynamics.* Cambridge: Cambridge University Press. ISBN 0 521 57295 9 (hc.). ISBN 0-521-57597-4 (pbk.).

[25] NĚMEC, Martin. *Přednáška 7 – Osvětlení, stínování*. Ostrava: VSB. Dostupné z: http://www.google.cz/url?sa=t&source=web&cd=7&sqi=2&ved=0CFsQFjAG&url=http%3A%2F%2Fbarborka.vsb.cz%2Fnemec%2Fzpg%2Fprednasky%2Fprednask

a07.pps&rct=j&q=phong%C5%AFv%20osv%C4%9Btlovac%C3%AD%20model&ei=VJnVTY_jIsnIswbIhP2TBw&usg=AFQjCNGqj_S0k5r9Dlrl5uESFSI6HyM-Eg&sig2=DtRDYyB_Y6LjHM5RC8KnOg&cad=rja.

[26] [SAJRI]. 2001. *Nebeská mechanika* [online]. [Vid. 2011-4-15]. Dostupné z: http://nebmech.astronomy.cz/VYPOCTY/vypocty.htm#ObcJul.

[27] SEIDELMANN, Kenneth P., et al. 2011. *Explanatory supplement to the astronomical almanac*. Washington: First Paperback Impression by University Science Books. ISBN 1-891389-45-9.

[28] SCHLYTER, Paul. [Ca2000]. *How to compute planetary positions* [online]. Stockholm, [vid. 2011-4-20]. Dostupné z: http://www.stjarnhimlen.se/comp/ppcomp.html.

[29] STANDISH, E. M. [Ca2000]. *Keplerian elements for approximate positions of the major planets* [online]. [Vid. 2011-4-20]. Dostupné z: http://ssd.jpl.nasa.gov/txt/aprx_pos_planets.pdf.

[30] STANDISH, E. M. [Ca2000]. *Keplerian elements for approximate positions of the major planets* [online]. [Vid. 2011-4-20]. Dostupné z: http://ssd.jpl.nasa.gov/?planet_pos.

[31] TANNER, Jay. 2011. *NeoProgrammincs on-line multi-language VSOP87 functions code generator* [online]. NeoProgrammincs. [vid. 2011-4-15]. Dostupné z: http://www.neoprogrammics.com/vsop87_theory/.

[32] TANNER, Jay. 2011. *The VSOP87 theory* [online]. NeoProgrammincs. [vid. 2011-4-15]. Dostupné z: http://www.neoprogrammics.com/vsop87_theory/.

[33] TSYBA, Serhii. 2010. *Implementation of the ELP theory version ELP 2000-82B in C* [online]. Finsko. [Vid. 2011-4-15]. Dostupné z: https://github.com/sertsy/elp2000-82b.

[34] VRATISLAV, Filler. 1998. *Semestrální práce z předmětu Astrodynamika 11* [online]. NASA's GSFC, [vid. 2011-4-15]. Dostupné z: http://klobouk.fsv.cvut.cz/~vrf/prace/ad11/ad11.html.

[35] WALNUM, Clayton. 2004. *Programujeme grafiku v Microsoft Direct3D.* Brno: Computer Press. ISBN 80-251-0136-3 (brož.): 389,- Kč.

[36] WEISSTEIN, Eric W., et al. 2011. *Normal vector* [online]. MathWorld-A Wolfram Web Resource, [vid. 2011-4-20]. Dostupné z: http://mathworld.wolfram.com/NormalVector.html.

[37] WIKIPEDIA(PROJEKT). 2011. *Lunar eclipse – Wikipedia* [online]. Wikipedia, editováno 2011-5-17 [vid. 2011-4-15]. Dostupné z: http://en.wikipedia.org/wiki/Solar_eclipse.

[38] WIKIPEDIA(PROJEKT). 2011. *Orbital elements – Wikipedia* [online]. Wikipedia, editováno 2011-3-13 [vid. 2011-4-15]. Dostupné z: http://en.wikipedia.org/wiki/Orbital_elements.

[39] WIKIPEDIA(PROJEKT). 2011. *OpenGL – Wikipedia* [online]. Wikipedia, editováno 2011-5-17 [vid. 2011-4-15]. Dostupné z: http://en.wikipedia.org/wiki/OpenGL.

[40] WIKIPEDIA(PROJEKT). 2011. *Shadow volume – Wikipedia* [online]. Wikipedia, editováno 2011-4-3 [vid. 2011-4-15]. Dostupné z: http://en.wikipedia.org/wiki/Shadow_volume .

[41] WIKIPEDIA(PROJEKT). 2011. *Solar eclipse – Wikipedia* [online]. Wikipedia, editováno 2011-5-9 [vid. 2011-4-15]. Dostupné z: http://en.wikipedia.org/wiki/Solar_eclipse.

[42] WIKIPEDIE(PROJEKT). 2011. *Zatmění slunce – Wikipedie* [online]. Wikipedia, editováno 2011-3-12 [vid. 2011-4-15]. Dostupné z: http://cs.wikipedia.org/wiki/Zatmění_Slunce.

[43] WIKIPEDIE(PROJEKT). 2011. *Zatmění měsíce – Wikipedie* [online]. Wikipedia, editováno 2011-3-12 [vid. 2011-4-15]. Dostupné z: http://cs.wikipedia.org/wiki/Zatmění_Měsíce .

Obsah CD

Přiložené CD obsahuje následující adresáře:
Data – obsahuje soubory potřebné ke spuštění aplikace.
Zdroje – Obsahuje zdrojové kódy aplikace.
Text – Tento text ve formátech DOC a PDF.

CD neobsahuje volně dostupnou redistribuci DirectX 9.c, kterou je nutné mít nainstalovanou na počítači, kde má být program spuštěn.