Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Newtonovská gravitácia s retardáciou

Bakalárska práca

2021
Matej Magát

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Newtonovská gravitácia s retardáciou

Bakalárska práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	RNDr. Peter Borovanský, PhD.
Konzultant:	RNDr. Eduard Masár, PhD.

Bratislava, 2021
Matej Magát

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Matej Magát
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Newtonovská gravitácia s retardáciou
Newtonian gravitation with retardation

Anotácia: Predkladaná téma má tri stránky: fyzikálnu, výpočtovú a programátorskú. Po fyzikálnej stránke ide o simuláciu v slabých gravitačných poliach, veľkosťou porovnateľných s tými, aké sa vyskytujú v našej Slnečnej sústave. Pretože vplyv retardácie na pohyb telies je v týchto podmienkach malý, výpočty musia byť veľmi presné. Vytvorený program by mal graficky zobrazovať odlišnosť dráh (prípadne aj rýchlostí) telies s retardáciou od dráh bez prítomnosti retardácie. Meniteľnými parametrami výpočtov by mali byť hmotnosti telies, začiatočné polohy a rýchlosti telies. Rôzne spôsoby pridania retardácie môžu byť v práci tiež uvažované. Práca má aj prezentačný aspekt, a cieľom je aj grafické a názorné ilustrovanie javov.

Cieľ: Cieľom práce je numericky vypočítať a graficky znázorniť riešenie úlohy dvoch telies v Newtonovej gravitačnej teórii s pridanou retardáciou.

Literatúra: Feynman, Leighton, Sands: Feynmanove prednášky z fyziky 1., Alfa, 1988
David Morin: Introduction to Classical Mechanics With Problems and Solutions, Cambridge University Press, 2007

Vedúci: RNDr. Peter Borovanský, PhD.
Konzultant: RNDr. Eduard Masár, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 20.09.2020

Dátum schválenia: 06.10.2020 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie:

Chcel by som sa takouto formou poďakovať môjmu školiteľovi RNDr. Petrovi Borovanskému, PhD. za rady a usmernenia počas mojej práce na tejto téme. Moja vďaka patrí aj môjmu školiteľovi RNDr. Eduardovi Masárovi, PhD. za odborné rady, odborné vedenie a veľkú trpezlivosť počas mojej práci.

Abstrakt

MAGÁT, Matej. *Newtonovská gravitácia s retardáciou* [Bakalárska práca]. Univerzita Komenského v Bratislave. Fakulta matematiky, fyziky a informatiky; Katedra aplikovanej informatiky. Školiteľ: RNDr. Peter Borovanský, PhD.; Konzultant: RNDr. Eduard Masár, PhD.; Komisia pre obhajoby: Aplikovaná informatika. Predseda : doc. RNDr. Damas Gruska PhD. stupeň kvalifikácie: Bakalár. Bratislava : FMFI UK, 2021. 50 strán (vrátane príloh).

Cieľom mojej bakalárskej práce bolo numericky vypočítať a graficky znázorniť riešenie úlohy dvoch telies v Newtonovej gravitačnej teórii s pridanou retardáciou a demonštrovať pritom nesprávnosť tohto fyzikálneho modelu. Počas plnenia jednotlivých míľnikov vznikli tri samostatné programy. Programy dokážu načítať vstupné údaje z konfiguračného súboru a podľa toho spustiť a zobraziť simuláciu, každý podľa svojho fyzikálneho modelu. Aplikácie zamerané na pohyby dvoch telies dokážu zobrazovať predchádzajúce výpočty. Aplikácie sú implementované pomocou jazyka Golang.

K bakalárskej práci sú elektronicky priložené výsledné aplikácie. Ostatné prílohy sú návody na ich použitie a príloha k zadaniu bakalárskej práce vypracovaná RNDr. Eduardom Masárom, PhD.

Kľúčové slová: Newtonova gravitačná teória s pridanou retardáciou, simulácia, fyzikálny model

Abstract

Newton's theory of gravitation with added retardation, simulation, physical model
MAGÁT, Matej. Newton's gravitation with retardation [Bachelor's thesis]. Comenius University in Bratislava. Faculty of Mathematics, Physics and Informatics; Department of Applied Informatics. Thesis supervisor: RNDr. Peter Borovanský, PhD.; Consultant: RNDr. Eduard Masár, PhD.; Thesis Defense Committee: Applied Informatics. Chairman: doc. RNDr. Damas Gruska, PhD. Academic degree: Bachelor. Bratislava : FMPI of the CU, 2021. 50 pages (including appendices).

The goal of my bachelor's thesis was to calculate the solution of the two-body problem in Newton's theory of gravitation with added retardation numerically and to represent it graphically, and at the same time to demonstrate the incorrectness of this physical model. Three independent programmes were created during the completion of the individual milestones. The programmes are able to load the input data from the configuration file and to run and show the simulation accordingly, each according to its physical model. The applications which are focused on the movements of the two objects are able to show previous calculations. The applications are implemented using the Golang language.

The resulting applications are attached to the bachelor's thesis electronically. The other appendices are the instructions on how to use them and an appendix to the bachelor thesis assignment, which was drawn up by RNDr. Eduard Masár, PhD.

Keywords: Newton's theory of gravitation with added retardation, simulation, physical model

Obsah

Slovník pojmov

- Za **únikovú rýchlosť** sa pokladá minimálna rýchlosť aby teleso uniklo z (dominantného) gravitačného pôsobenia iného, často hmotnejšieho telesa, jej veľkosť dá sa vypočítať vzorcom:

$$v_{escape} = \sqrt{\frac{2GM}{R}} \tag{1}$$

  kde G je gravitačná konštanta, M je hmotnosť a R je vzdialenosť

- **Horizont udalostí** je oblasť z ktorej nemôže uniknúť nič, dokonca ani svetlo - úniková rýchlosť je vyššia ako rýchlosť svetla

- **Čierna diera** je vesmírny objekt, ktorý vznikol veľkým stlačením hmoty do malého objemu, jej hustota je tak vysoká, že na svoje okolie pôsobí tak veľkou silou, že vytvára oblasť známu ako horizont udalostí, vzniká najčastejšie pri zániku ťažkej hviezdy o veľkosti vyššej ako 25 násobok hmotnosti nášho slnka

- **Super-masívnou čiernou dierou** sa nazýva čierna diera, keď presiahne hmotnosť jedného milióna hmotností nášho slnka

- **biely trpaslík a neutrónová hviezda** sú veľmi husté a ťažké vesmírne objekty, sú to pozostatky mŕtvych hviezd

- **Štruktúra v jazyku Golang** je obdoba C-éčkovského typu struct, hoci sa rovnako píše má vlastnosti podobajúce sa triedam v jazykoch Java alebo Python, okrem konštruktora

- **Goroutina v jazyku Golang** sa nazýva aj odľahčený thread, na rozdiel od klasických threadov, ktoré manažuje operačný systém sa viaceré goroutiny mapujú na jeden thread v operačnom systéme ich správu má na starosti zabudovaný runtime plánovač

- **Embeded structs** alebo aj zabudované štruktúry sú zvláštnou obdobou dedenia v objektovo orientovaných jazykoch programovacích jazykoch, navonok pri písaní kódu sa to javí ako dedenie, ale pri implementácii sa ukážuje, že to sú vnorené štruktúry, ktoré sú oddelené kompozičné triedy podobné triednym premenám v

jazyku java. Pri veľmi špecifických prípadoch je výhodné sa k nim správať ako k triednym premenám. Oproti triednym premenám v jazyku java však štruktúra, ktorá ich má zabudované môže volať všetky funkcie (navonok) bez toho aby programátor musel exciplitne zavolať najprv originálnu štruktúru na ktorej funkciu volá

- **Čistý kód** je označenie používajúce sa v informatike na označenie spôsobu písania kódu aby bol lepšie čitateľnejší a prehľadnejší viac v [19]

Úvod

Vo fyzike sa stále používajú vzorce na výpočet gravitačnej sily, ktoré sú známe viac ako tri storočia, a ktoré sformuloval velikán svojej doby Isaac Newton. Tieto vzorce používame na rôzne situácie, od výpočtu trajektórie padajúceho telesa až po výpočet pohybu planét, slnka, hviezd či galaxií. Od doby, keď ďalší velikán vo fyzike Albert Einstein definoval gravitáciu ako zakrivenie časopriestoru, máme oveľa presnejšie vzorce na výpočet tejto sily. Albert Einstein definoval aj maximálnu rýchlosť šírenia informácie (rýchlosť svetla), na ktorú sme neskôr nadviazali. Napriek tomu, že vzorce, ktoré objavil Albert Einstein, sú oveľa presnejšie, používajú sa iba pre prípady, kedy už zlyhávajú vzorce od Isaaca Newtona (pri silných gravitačných poliach čiernych dier, bielych trpaslíkov či neutrových hviezd). Fyzika Isaaca Newtona však stále používa nekonečnú rýchlosť šírenia gravitácie, a teda aj informácie. Táto skutočnosť niektorým fyzikom vnukla ideu (dnes všeobecne odmietanú, najmä pre porušovanie zákona zachovania energie) pridať do newtonovskej mechaniky retardáciu.

Vo svete fyziky (alebo všeobecnejšie vedy) sa však niekedy aj najpresvedčivejšie teórie ukázali ako nepravdivé a tie najmenej presvedčivé sa nakoniec ukázali ako pravdivé. Je preto potrebné, aby sa informatici púšťali aj do programovania simulácií menej známych, či dokonca prevažnou väčšinou odmietaných vedeckých hypotéz.

V dnešnej dobe sa už počítače natoľko zlepšili, že dokážu simulovať reálny svet takmer v reálnom čase. Počítače sa stále viac využívajú na simulovanie extrémnych javov, napriek tomu sa málo pozornosti venuje simuláciám, ktoré by sa venovali stimulovaniu kritického myslenia u žiakov alebo znovuvyvráteniu (názornému ukázaniu nesprávnosti) niektorých vedeckých hypotéz v oblasti fyziky.

V súčasnosti ponúka svet počítačov stále viac možností v oblasti hardvéru, ale najmä softvéru. V oblasti softvéru, ale aj programovacích jazykov, sa objavujú stále nové nástroje, ktoré ponúkajú nevídané možnosti v oblasti výpočtových problémov, ktoré treba preskúmať, využiť alebo aj vylepšiť.

Cieľom tejto bakalárskej práce je výpočtovo simulovať newtonovskú gravitáciu s pridaním retardácie (oneskorenia šírenia gravitácie) a graficky znázorniť riešenie úlohy dvoch telies. Pri tejto úlohe vznikne niekoľko menších funkčných verzií, ktoré môžu byť použité na vizuálne znázornenie problémov. Sú to problémy: problém jedného telesa (telesa obiehajúceho hmotný statický bod), problém dvoch telies bez pridania retardá-

cie a nakoniec problém dvoch telies s pridaním retardácie.

Táto bakalárska práca je usporiadaná do kapitol, ktoré sú usporiadané do logických celkov. Takmer každá kapitola má niekoľko podkapitol, ktoré riešia jednotlivé iteratívne verzie.

Prvá kapitola je zameraná na prehľad technológií, ktoré hrali pri vytváraní softvéru úlohu a sú buď použité vo finálnych verziách alebo sme sa s nimi pri vytváraní softvéru stretli. V tejto kapitole rozoberieme výhody použitých technológií.

Druhá kapitola obsahuje opis fyzikálnej problematiky. V tejto kapitole sa nachádzajú všetky potrebné vzorce a fyzikálne úvahy, ktoré sú v bakalárskej práci použité (uvažované a naprogramované) .

V tretej kapitole sú rozpísané riešenie problému pomocou daných technológií a výpočtové metódy pre daný problém.

Štvrtá kapitola sa zaoberá analýzou a testovaním jednotlivých postupov a metód, ako aj testovaním výstupov.

V záverečnej kapitole sú zhrnuté poznatky nadobudnuté z vývoja tejto aplikácie (a jej rozličných verzií) a pojednáva sa tu aj o ďalších možnostiach skúmania v tejto oblasti a možných rozšíreniach tejto aplikácie.

Kapitola 1

Prehľad technológií

V tejto kapitole sa budeme venovať technológiám, ktorým sme sa behom vytvárania softvéru venovali (s ktorými sme sa stretli). Podrobne sú však rozpísané iba tie technológie, ktoré nás zaujali alebo ktoré sme priamo použili vo výsledných verziách. Táto kapitola je rozdelená na dve podkapitoly. Prvá kapitola opisuje programátorský jazyk, ktorý bol vybratý, jeho výhody, ale aj nevýhody a nakoniec aj dôvody jeho výberu.

1.1 Programovací jazyk

1.1.1 Popis jazyka

Na tvorbu bakalárskej práce sme si vybrali programovací jazyk Go, taktiež zvaný aj Golang, ktorý je dostupný na [7]. Golang je relatívne nový jazyk, ktorý vnikol vo firme Google Inc. v roku 2007, ale jeho prvé uplatnenie nastalo až o dva roky neskôr, 10. novembra 2009, ako je uvedené v [8].

Prvá verzia jazyka Golang bola vytvorená skupinou tvorenou známymi osobnosťami ako Rob Pike, ktorý je predovšetkým známy prácou v Bellových laboratóriách, kde bol členom tímu Unix, podieľal sa na tvorbe Plan 9 z operačných systémov Bellových laboratórií a operačného systému Inferno, participoval na vytvorení programovacieho jazyka Limbo a je spoluautorom veľmi známeho a rozšíreného kódovania UTF-8 [9], Robert Griesemer, ktorý predtým pracoval na technológii JavaScript V8 spoločnosti Google, jazyku Sawzall, virtuálnom stroji Java HotSpot a systéme Strongtalk [10] a Ken Thompson, ktorý je skutočným velikánom v informatickej oblasti, najmä vďaka prácam v Bellových laboratóriách na mnohých projektoch, ako napríklad operačný systém Plan 9, spolupráca s Robertom Pikeom na kódovaní UTF-8, vďaka operačnému systému Inferno, veľmi známej a fundamentálnej utilite "grepä vývoju programovacieho jazyka B, ktorý bol predchodcom najrozšírenejšieho jazyka na svete (programovacieho jazyka C). Podieľal sa na vývoji Belle, šachového počítača. Ken Thompson tiež napísal

programy pre vytvorenie absolútneho hodnotenia (výhra, remíza alebo prehra hráča, ktorý je na ťahu) šachových postavení [11].

Programovací jazyk Golang je rýchlo sa rozvíjajúcim jazykom, od jeho vydania do napísania tejto bakalárskej práce vyšlo 15 verzií [8]. Syntax a stavba programovacieho jazyka Golang boli inšpirované jazykmi ako sú C, Modula, Pascal a Smalltalk.

Golang je navrhnutý tak, aby sa v ňom dali písať rozličné paradigmy, medzi známe paradigmy sa zaraďujú Object Oriented, aj keď mnohí kritici poukazujkú na to, že sa v Go nedá objektovo programovať (namiesto tried sa kód môže štruktúrovať pomocou package-ov) a imperatívne programovanie. Vďaka svojej natívnej podpore konkurencie najznámejšou a najvýraznejšou paradigmou programovacieho jazyka Golang je Concurrent programming [12].

Samotný programovací jazyk má jednu hlavnú a tri vedľajšie implementácie (kompilátory). Väčšina kompilátorov pre Golang podporuje "natívne"prepájanie (cez takzvané rozhranie cgo) a kompiláciu programovacích jazykov Golang a starého C.

Hlavná implementácia je pomenovaná ako gc, je to oficiálny kompilátor go-komunity, prekladá (kompiluje) kód napísaný v jazyku Golang priamo do strojového (binárneho) kódu, ktorý je už plne spustiteľný, tento kompilátor síce podporuje prepájanie programovacích jazykov Golang a starého C, ale musí byť previazaný s kompilátorom, ktorý skompiluje C-éčkovskú časť kódu a komunikácia medzi nimi je časovo náročná, preto sa odporúča iba v prípade nutnosti. Tento kompilátor kompiluje veľmi rýchlo (v našich prípadoch mu trvá skompilovať program s 3 750 riadkami 2 sekundy), binárny kód od tohto kompilátora obsahuje všetku binárnu informáciu vrátane debuggovacích nástrojov a celého runtime, preto je oproti iným binárnym kódom pomerne veľký.

Programovací jazyk Golang má aj vedľajšie implementácie (kompilátory), sú to napríklad gccgo [13] a tiny-go [14] .

gccgo je implementácia (kompilátor), ktorý je front-end kompilátora gcc, ktorý je kompilátor pre programovací jazyk C. Na rozdiel od oficiálneho kompilátora gc, ktorý kompiluje priamo do binárnej podoby, kompilátor gccgo najprv preloží kód písaný v programovacom jazyku Golang do programátorského jazyka C a až ten skompiluje do strojového jazyka (binárnej podoby). Ďalšia odlišnosť od oficiálneho kompilátora je v odlišnom prístupe k systémovým knižniciam, respektíve k linkovaniu na nich. Kým oficiálny kompilátor nepoužíva zdieľané a dynamicky linkované knižnice (shared, dynamic linked library), kompilátor gccgo ich používa, a tým dosahuje výrazne menší binárny kód, ale za cenu menšej prevoditeľnosti medzi jednotlivými počítačmi (musí mať všetky knižnice). Ďalšou výhodou je, že má lepšie zvládnuté previazanie jazykov C a Golang (menšie časové nároky). Najväčšou nevýhodou oproti štandardnému kompilátoru je, že má väčšie časové nároky na kompiláciu.

Ostatné kompilátory sme nepoužívali, preto len v krátkosti opíšeme jeden, ktorý sa nám zdal najzaujímavejší.

Tiny-go je kompilátor, ktorý si dáva za úlohu skompilovať kód napísaný v programátorskom jazyku Golang do menšej formy, vhodný pre mikročipy a mikro-počítače.

Golang poskytuje množstvo nástrojov, ktoré sa volajú príkazom „go [príkaz]“. Pri našej práci sme sa stretli s týmito: „get“, ktorý je fundamentálny nástroj jazyka, tento nástroj slúži najmä na získavanie knižníc tretích strán.
Nástroj gomobile [15] spolu s niektorými knižnicami sa v budúcnosti môže použiť na rozšírenie tohto programu na platformy ako Android, či IOS. Tento nástroj poskytuje dve voľby, voľbu natívneho previazania s NDK (build) alebo voľbu priviazania (bind) s programovacími jazykmi Java (Android) a Objective-C (IOS). [20] Ďalšie nástroje sú napríklad godoc, ktorý sa snaží byť ekvivalentom nástroja jazyka javadoc, nástroje cover na analýzu kódu a testov, „guru“ a „vet“.

1.1.2 Dôvody výberu

Programovací jazyk Golang sme si vybrali, pretože je to pomerne nový jazyk so zaujímavým pozadím, v ktorom sa pomerne jednoducho (a hlavne natívne) píšu konkurentné procesy. Golang poskytuje (v natívnej forme) takzvané goroutiny, ktoré v jazyku vystupujú v roli odľahčených threadov. V našej práci sme potrebovali mať veľa paralelných (súčasne bežiacich) procesov, a preto bol programovací jazyk Golang pre našu prácu ideálnym jazykom. Tento jazyk produkuje binárny kód, ktorý nepotrebuje žiadne virtuálne prostredia a v prípade klasického kompilátora (gc) ani systémové knižnice, navyše je prenositeľný na rôzne hardvérové platformy, ako sú napríklad architektúry arm32, arm64, i386, amd32, amd64 a mnohé ďalšie, ale hlavne Golang je prenositeľný na softvérové platformy; okrem operačného systému Windows sa jeho kód môže skompilovať aj na operačné systémy Linux, Darwin (na ktorý nadväzujú systémy macOS, iOS, watchOS, tvOS a iPadOS), aix, dragonfly, android a ďalšie. Tento jazyk sme si vybrali aj vďaka jeho rýchlosti pri behu programu, ktorá sa približuje k rýchlosti vykonávania programovacieho jazyka C, ľahkej pisateľnosti a čitateľnosti kódu a nakoniec aj veľkému výberu ľahko dostupných knižníc tretích strán, ktoré sa automaticky stiahnu so všetkými závislosťami.

1.2 Platforma

Hlavná platforma pre túto bakalársku prácu bol Windows 10, ktorý beží na architektúre amd, dôvodom bol fakt, že tento softvér bol a aj je určený aj pre laickú verejnosť, u ktorej je táto platforma veľmi rozšírená.

1.3 Prídavné programátorské knižnice

V tejto sekcii sme popísali niektoré knižnice tretích strán, s ktorými sme sa zoznámili, ale uviedli sme iba tie, nad ktorými sme uvažovali ešte pred prvým prototypom. Takmer všetky knižnice sme našli na stránke [3], ktorá zhrňuje všetky knižnice tretích strán, ktoré sú známe a registrované. V tejto sekcii sme neopísali žiadne zabudované knižnice v samotnom jazyku.

1.3.1 GUI

Fyne.io/fyne

Táto programátorská knižnica je veľmi vyspelá a je to knižnica, ktorú podporujú aj autori pôvodného jazyka. Ako je uvedené na ich hlavnej stránke [5], tak táto knižnica je multi-platformová. Knižnica používa kompilátory gc a C-éčkový gcc, ktorý musí byť 64-bitový. Túto knižnicu som sa rozhodol nepoužiť, kvôli náročnosti robenia animácií a sťaženej kompilácií na Windowsoch.

gioui/gio

Táto programátorská knižnica je veľmi vyspelá a je to knižnica. Ako je uvedené na ich hlavnej stránke[18], tak táto knižnica je multi-platformová. Knižnica navyše pre operačné systémy Windows, Linux a Mac na štandardných platformách používa iba kód napísaný v Golang. Knižnica sa dá spustiť aj na menej využívaných platformách, ako napríklad Linux pre mobilné zariadenia alebo FreeBSD, OpenBSD, hoci tam už využíva kompilátor gcc. Výhodou tejto knižnice je, že sa dá podľa[18] skompilovať (bez úprav samotného kódu) aj pre mobilné zariadenia s operačnými systémami Android a IOS/tvOS, to sa však nezaobíde bez externých nástrojov pre dané systémy. Hoci je táto vlastnosť ešte len v experimentálnej fáze, najväčšou zaujímavosťou tejto knižnice je, že jej autor ju vymyslel tak, že sa dá skompilovať do Webassembleru, ktorý sa stal akýmsi novým štandardom pre internetové aplikácie. Túto knižnicu sme sa rozhodli nepoužiť kvôli jej komplexnosti.

Gen2brain/raylib-go

Táto programátorská knižnica má výhodu, že využíva už vyladenú C knižnicu zvanú raylib, síce využíva rozhranie cgo, je veľmi rýchla a istú dobu bola hlavnou knižnicou pre prvý prototyp. Túto knižnicu sme však vymenili za jednoduchšiu.

Gonutz/prototype

Toto je knižnica, ktorá tvorí základ väčšiny programov, jej hlavnou výhodou je jednoduchý interface, ktorý sa naozaj riadi heslom „v jednoduchosti je krása“, napriek tomu je to veľmi výkonná knižnica a to aj oproti iným, ďalšou veľmi dôležitou výhodou je to, že je celá napísaná v programovacom jazyku Golang, a tým zabezpečuje aj ľahšiu kompiláciu oproti iným knižniciam. Bola optimalizovaná na procesorovú architektúru Amd na operačné systémy Windows, Linux a aj Mac. Knižnica má však aj nevýhody, jednou z najväčších je to, že kód je menej prenositeľný na iné platformy ako v prípade knižnice gioui/gio, nefunguje pri nej nástroj gomobile a táto knižnica je napísaná tak, že má malý potenciál, aby dosiahla preložiteľnosť na mobilné zariadenia s ich operačnými systémami (Android a IOS), nepôjde ani na vyššie uvedených operačných systémoch s inými procesorovými architektúrami, napríklad na Linuxe, bežiacom na procesorovej architektúre Arm.

Korok.io

Táto knižnica bola hlavným konkurentom používanej knižnice pre jej jednoduché rozhranie, napriek tomu táto knižnica nebola zatiaľ optimalizovaná na iné platformy ako OSX/Windows.

1.3.2 Logika

Každá väčšia (fyzikálna) simulácia potrebuje zapisovať priebežné výstupy, preto sme tu opisovali dve programátorské knižnice, ktoré dokážu pomerne efektívne zapisovať do súborov „xlsx“.

360EntSecGroup-Skylar/excelize

Excelize je knižnicou, ktorá je určená pre operačný systém Windows, jej primárnou úlohou je ukladanie dát do súborov „xlsx“, ale aj načítať dáta z už existujúcich súborov a pracovať s nimi. Táto knižnica má možnosť vytvárania grafov z načítaných dát, má však nevýhodu, a tou je pamäťová náročnosť pri uchovávaní počas behu a dlhý čas pri následnom ukladaní veľkého množstva dát (100 megabajtov).

Alternatívy

Ďalšími knižnicami, ktoré sme stretli, boli LibreOffice-GO a go-libreofficekit, pričom go-libreofficekit je knižnica primárne pre Linux a využíva programovacie jazyky C a Golang, oproti tomu LibreOffice-GO je určená pre Windows, Linux aj Mac, obsahuje oproti predošlej knižnici závislosť aj na C++.

Kapitola 2

Pohyb telies v gravitačnom poli

V tejto kapitole je rozpísaná teória k pohybu telies v gravitačnom poli, ktorá je nutná pre urobenie fyzikálne správnej simulácie. Sú tu popísané jednotlivé fyzikálne podproblémy, ktoré sú neoddeliteľnou súčasťou poznatkov potrebných na to, aby výsledná simulácia fungovala podľa fyzikálnych zákonov.

Na výpočet gravitačnej sily medzi telesami (v našom prípade medzi dvomi) sa používa vzorec:

$$G\frac{m_1 m_2}{r^2}$$

Tento vzorec bol známy od Isaaca Newtona a ponúka nám efektívny spôsob výpočtu gravitačného pôsobenia, v nasledujúcej kapitole sú rozpísané numerické metódy, ktoré sa odkazujú na tieto pod-problémy.

2.1 Keplerov problém

Keplerov problém znamená, že riešime pohyb jedného telesa v centrálnom poli, čo znamená, že v strede (bod 0,0 v karteziánskej sústave) simulovaného poľa sa nachádza veľmi hmotný bod, ktorý môže predstavovať Slnko, ktoré obsahuje až 99,86 % z celej slnečnej sústavy, tu sa rieši pohyb jednej planéty (napríklad Zeme), alebo pohyb jednej hviezdy okolo super-masívnej čiernej diery, napríklad pohyb hviezdy známej ako „S2" okolo super-masívnej čiernej diery v strede našej galaxie, známej ako „Sagittarius A*".

V simulácii bolo potrebné vypočítať závislosť rýchlosti v danom smere od času, čo sme mohli vyjadriť ako zrýchlenie v danom smere, toto sme mohli vyjadriť tromi rovnicami pre tri osi troj-dimenzionálneho priestoru, ale keďže simulácia bola v dvoj-dimenzionálnom priestore, tak sa rovnica pre os Z vynechala, teda rovnice vyzerali:

$$m(dv_x/dt) = -GMmx/r^3. \tag{2.1}$$

$$m(dv_y/dt) = -GMmy/r^3. \tag{2.2}$$

$$r = \sqrt{x^2 + y^2}. \tag{2.3}$$

Pri týchto rovniciach bolo potrebné si všimnúť, že pravé strany rovníc 2.1 a 2.2 mali záporné znamienka. Bol to dôsledok voľby smerovania vektora od bodu 0,0 ku planéte, pričom všetky súradnice telesa boli reprezentované ako x alebo y, pričom vzdialenosť od bodu 0,0 je tiež kladná, gravitačná sila v prírode však pôsobí smerom do stredu, teda k bodu 0,0, a preto sa tejto sile vo výpočtoch muselo pridať záporné znamienko, aby sa docielil správny smer pôsobiacej sily. Pre príklad nech sú počiatočné podmienky takéto (z knihy[1]):

$$\epsilon = 0.1$$

$$x(0) = 0.500 \qquad y(0) = \quad 0.000$$

$$v_x(0) = 0.000 \qquad v_y(0) = +1.630$$

Za predpokladu, že sme si určili gravitačnú konštantu G a aj hmotnosť telesa rovnú jednej, sme mohli napísať: $\mathbf{GM = 1}$. Teraz bolo možné vypočítať vzdialenosť od stredu a zrýchlenie pôsobiace v danej osi:

$$\mathrm{r}(0) = 0.500 \qquad 1/\mathrm{r}^3(0) = 8.000$$

$$a_x(0) = -4.000 \qquad a_y(0) = 0.000$$

Od tohto bodu sa dalo pokračovať niekoľkými spôsobmi, o tom bola napísaná ďalšia kapitola.

2.1.1 Keplerove zákony:

Problém jedného telesa riešia aj Keplerove zákony, pričom najdôležitejší pre simuláciu je druhý Keplerov zákon, všetky zákony boli popísané tu[16] na strane 102. Pre overenie správnosti výpočtov bol dôležitý hlavne druhý Keplerov zákon, ktorý hovorí o tom, že plocha pozametaná sprievodičom za rovnaký čas je rovnaká na akýchkoľvek úsekoch.

2.2 Problém dvoch telies

Tento problém si vyžadoval aj analytické riešenie, popísané v nasledujúcej kapitole, zároveň sa tu používala už len jedna výpočtová metóda, a to leapfrog.

Vo všeobecnosti sa pri probléme pohybu viac ako jedného telesa používajú vzorce:

$$m_i \frac{d\,v_{i\,x}}{d\,t} = \sum_{j=1}^{N} -\frac{G\,m_i m_j (x_i - x_j)}{r_{ij}^3}. \tag{2.4}$$

$$m_i \frac{dv_{i\,y}}{d\,t} = \sum_{j=1}^{N} -\frac{G\,m_i m_j (y_i - y_j)}{r_{ij}^3}. \tag{2.5}$$

$$m_i \frac{dv_{i\,z}}{dt} = \sum_{j=1}^{N} -\frac{G\, m_i m_j (z_i - z_j)}{r_{ij}^3}. \tag{2.6}$$

V simulácii sa vyskytoval vzorec pre os Z, pretože telesá mali nulovú súradnicu a aj rýchlosť v tomto smere a teda bolo možné túto os úplne vynechať, v tomto prípade sa premenná N rovnala dvom (N = 2). V simulácii N nebolo väčšie, pretože donedávna analytické riešenie existovalo iba pre dve a pre tri telesá existuje iba vo výnimočných prípadoch, navyše aj v prípadoch, že existuje, tak by bolo veľmi náročné ho vypočítať, navyše toto už bolo mimo záber tejto bakalárskej práce.

V vzorcoch 2.4,2.5 a 2.6 sa vypočítala suma síl gravitačného pôsobenia z jednotlivých objektov na daný objekt.

Pre tento problém bolo potrebné vypočítať aj analytické riešenie, preto boli potrebné vzorce, ktoré vypočítali z počiatočných podmienok celú dráhu pomocou vzorcov pre výpočet eliptickej dráhy:

výpočet dráhy v ťažiskovej sústave:

$$r = \frac{p}{1 + e\cos\varphi}. \tag{2.7}$$

Kde vystupoval parameter p a excentricita e, ktorá označuje, ako veľmi je dráha pretiahnutá oproti kruhovej dráhe. Parameter a excentricita sa dajú získať ako:

$$p = \frac{L^2}{am} = \frac{L^2}{GMm^2}. \tag{2.8}$$

$$e^2 - 1 = \frac{2L^2E}{a^2m} = \frac{2L^2E}{G^2M^2m^3}. \tag{2.9}$$

Kde vystupovali premenné ako celková energia E, hmotnosti M,m a nakoniec moment hybnosti L.

z 2.9 sa dalo jednoduchou úpravou dostať:

$$e = \sqrt{1 - \frac{2L^2E}{G^2M^2m^3}}. \tag{2.10}$$

L - moment hybnosti sa dal získať pomocou vzorca:

$$L = mr^2\dot{\varphi}. \tag{2.11}$$

Kde $\dot{\varphi}$ bola derivácia φ.

Vzorec 2.7, ktorý počítal analytickú dráhu, bolo potrebné doplniť o uhol voči ose a taktiež ho rozlíšiť pre súradnicu x a y:

$$x = \frac{p * \cos(\psi + \varphi)}{1 + \epsilon\cos\varphi}. \tag{2.12}$$

$$y = \frac{p * \sin(\psi + \varphi)}{1 + \epsilon\cos\varphi}. \tag{2.13}$$

uhol psi sa musí vypočítať nasledovne:

$$\psi = \arctan\left(\frac{(v_{y_1} - v_{y_2}) * L - \frac{a*(x_1-x_2)}{r}}{-(v_{x_1} - v_{x_2}) * L - \frac{a*(y_1-y_2)}{r}}\right). \tag{2.14}$$

V arkustangense sa vyskytovali rozdiely v rýchlostiach oboch telies, rozdiely v polohách, celková vzdialenosť a konštanta a.

'L' - moment hybnosti, vzdialenosť r a konštanta a sa vypočítali ako:

$$r = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}. \tag{2.15}$$

$$a = Gm_1m_2. \tag{2.16}$$

Predtým, ako sa dal vyrátať L - moment hybnosti, bolo potrebné vyrátať niekoľko menších rovníc:

Redukovaná hmotnosť μ, ktorá sa vypočítala súčinom predeleným súčtom hmotností oboch zúčastnených telies:

$$\mu = \frac{m_1m_2}{m_1 + m_2}. \tag{2.17}$$

Rýchlosť ťažiska sústavy V, ktorá sa získala súčtom ich rýchlostí prenásobených ich hmotnosťami, ktorý sa ešte predelil súčtom ich hmotností. To isté aj pre rýchlosť v osi x, ako aj v osi y:

$$V_x = \frac{m_1 * v_{x_1} + m_2 * v_{x_2}}{m_1 + m_2}. \tag{2.18}$$

$$V_y = \frac{m_1 * v_{y_1} + m_2 * v_{y_2}}{m_1 + m_2}. \tag{2.19}$$

Prevod do ťažiskovej sústavy sa musel urobiť, pretože analytické riešenie potrebovalo, aby fyzikálny systém, ktorý rieši, mal nehybné ťažisko. Urobilo sa to odpočítaním rýchlosti ťažiska od rýchlostí planét:

$$v_{x_{12}} = v_{x_1} - V_x.\ v_{y_{12}} = v_{y_1} - V_y. \tag{2.20}$$

$$v_{x_{22}} = v_{x_2} - V_x.\ v_{y_{22}} = v_{y_2} - V_y. \tag{2.21}$$

A konečne s pomocou predchádzajúcich vzorcov sme mohli odvodiť moment hybnosti, ktorý potrebujeme na výpočet uhla psi φ, ale aj na výpočet excentricity e a parametra p z 2.7:

$$L = mu * ((v_{y_{12}} - v_{y_{22}}) * (x_1 - x_2) - (y_1 - y_2) * (v_{x_{12}} - v_{x_{22}})). \tag{2.22}$$

2.3 Problém dvoch telies - s pridanou retardáciou

Tento problém úzko nadväzoval na predchádzajúci problém, a teda zdedil všetky riešenia v probléme dvoch telies a pridáva retardáciu. Tu je stručne opísaná fyzikálna

stránka, ktorá bola potrebná na riešenie tohto problému, podrobnejšie znenie sa nachádza v prílohe B.

Ak sa gravitačná sila šíri konečnou rýchlosťou, tak je nevyhnutné, aby v Newtonovskom svete dochádzalo k oneskoreniu gravitačnej interakcie, toto sa nazýva retardácia. Pre numerické výpočty by bolo možné iteratívne prechádzať minulé polohy a hľadať čo najpresnejšiu aproximáciu pre vzťah:

$$(t - t')c = R(t') \tag{2.23}$$

Toto v realite by bolo veľmi ťažké dosiahnuť, navyše bolo by to náročné na strojové zdroje, preto bolo žiaduce až nutné sa obrátiť na analytickú aproximáciu s presnosťou $\frac{v}{c}$:

$$\mathbf{R}(t') \approx \mathbf{R} + R\frac{\mathbf{v}}{c} \tag{2.24}$$

$$R(t') \approx R + \frac{\mathbf{R} \cdot \mathbf{v}}{c} \tag{2.25}$$

Kde rýchlosť v je vektor. Vyššie spomenuté vzorce nám poslúžili, keď sme chceli vypočítať silu F vo vektorovom tvare, ktorou teleso 2 pôsobí na teleso 1 v mieste r v čase t, prvým spôsobom bol:

$$\mathbf{F}_{21} = -\frac{G\, m_1\, m_2}{R^3(t')}\mathbf{R}(t') \tag{2.26}$$

V tomto vzorci vystupovali premenné ako gravitačná konštanta G, hmotnosti oboch planét m_1, m_2, vypočítaný vektor $\mathbf{R}$(t') a R(t').

Druhým spôsobom pridania retardácie by bol:

$$\varphi(\mathbf{r}, t) = -\frac{G\, m_2}{R(t')} \tag{2.27}$$

$$\mathbf{F}_{21} = -m_1 \nabla\varphi(\mathbf{r}, t) = -\frac{G\, m_1 m_2}{R^2(t')} \frac{n'}{1 - \frac{n' \cdot v(t')}{c}} \tag{2.28}$$

Aj v tomto vzorci vystupujú premenné ako gravitačná konštanta G, hmotnosti oboch planét m_1, m_2 vypočítaný vektor $R(t')$, vypočítaná rýchlosť po retardácii $v(t')$ a n' bol skrátený zápis:

$$n' = n(t') = \frac{\mathbf{R}(t')}{R(t')}$$

Kapitola 3

Numerické metódy

Ako bolo spomenuté v predchádzajúcej kapitole, tak na výpočet obežnej dráhy sa používajú newtonovské vzorce, ktoré však platia pre kontinuálne prírodné javy, lenže aby sme vedeli urobiť simuláciu, tak sme potrebovali vyrátať určitý stav po našom inicializačnom stave, keď sme použili časový úsek, ktorý sa rovnal jednej jednotke času (obyčajne sekunde), tak náš výpočet bol veľmi nepresný a tu sme mali dve možnosti: buď použiť výpočtové metódy, ktoré veľmi spresňujú výpočty, alebo použiť analytické riešenie elíps, preto táto kapitola je rozdelená na dve podkapitoly. V prvej podkapitole sa pojednávalo o analytickom riešení tvaru dráh, zatiaľ čo v druhej sa pojednávalo o numerických metódach výpočtu konkrétnych polôh.

3.1 Samotné metódy

Všetky metódy riešili diferenciálne rovnice popísané v predchádzajúcej kapitole.

3.1.1 Euler

Táto metóda sa používala na miestach, kde sme potrebovali menej presné výpočty, bola to metóda prvého rádu, metóda využívala zmenšovanie časových úsekov na dosiahnutie väčšej presnosti. Výpočty sa realizovali spôsobom (pohyb) pripočítania rýchlosti k polohe a vynásobenie časovým úsekom (zlomkom, desatinným číslom) a až potom pripočítania zrýchlenia k rýchlosti, tak ako sa spomína v [1]. Vzťahy pre túto metódu sa dali vyjadriť ako:

$$a_{\mathrm{x}}(t) = -x/r^3(t) \tag{3.1}$$

$$v_{\mathrm{x}}(t+\epsilon) = v_{\mathrm{x}}(t) + \epsilon a_{\mathrm{x}}(t) \tag{3.2}$$

$$x(t+\epsilon) = x(t) + \epsilon v_{\mathrm{x}}(t) \tag{3.3}$$

$$a_{\mathrm{y}}(t) = -y/r^3(t) \tag{3.4}$$

$$v_{\mathrm{y}}(t+\epsilon) = v_{\mathrm{y}}(t) + \epsilon a_{\mathrm{y}}(t) \tag{3.5}$$

$$y(t+\epsilon) = y(t) + \epsilon v_{\mathrm{y}}(t) \tag{3.6}$$

Kde x, y boli súradnice telesa, epsilon bol krok výpočtu, a_x, a_y boli zrýchlenia v osiach x a y. Premenná t predstavovala čas a r bola vzdialenosť, ktorú sme vypočítali $\sqrt[2]{x^2+y^2}$. Ako bolo z rovníc vidieť, najprv sme pripočítali k starému x a y pôvodnú rýchlosť v daných osiach a až potom sme aktualizovali rýchlosti pomocou zrýchlení v daných osiach.

3.1.2 Euler-Cromer

Táto metóda bola podobná predchádzajúcej, bola trochu presnejšia, používala opačné poradie príkazov, a aj podľa [6] sme ju mohli charakterizovať rovnicami:

$$a_{\mathrm{x}}(t) = -x/r^3(t) \tag{3.7}$$

$$v_{\mathrm{x}}(t+\epsilon) = v_{\mathrm{x}}(t) + \epsilon a_{\mathrm{x}}(t) \tag{3.8}$$

$$x(t+\epsilon) = x(t) + \epsilon v_{\mathrm{x}}(t) \tag{3.9}$$

$$a_{\mathrm{y}}(t) = -y/r^3(t) \tag{3.10}$$

$$v_{\mathrm{y}}(t+\epsilon) = v_{\mathrm{y}}(t) + \epsilon a_{\mathrm{y}}(t) \tag{3.11}$$

$$y(t+\epsilon) = y(t) + \epsilon v_{\mathrm{y}}(t) \tag{3.12}$$

Kde opäť x, y boli súradnice telesa, epsilon bol krok výpočtu, a_x, a_y boli zrýchlenia v osiach x a y. Premenná t predstavovala čas a r bola vzdialenosť, ktorú sme vypočítali $\sqrt[2]{x^2+y^2}$. Z rovníc bolo vidieť, že sa najprv aktualizovala rýchlosť v osiach x, y a až potom sme aktualizovali polohu.

3.1.3 Leapfrog

Túto metódu sme mohli podľa [1] vyjadriť vzorcami ako:

$$a_{\mathrm{x}}(t) = -x/r^3(t) \tag{3.13}$$

$$v_{\mathrm{x}}(t+\frac{\epsilon}{2}) = v_{\mathrm{x}}(t-\frac{\epsilon}{2}) + \epsilon a_{\mathrm{x}}(t) \tag{3.14}$$

$$x(t+\epsilon) = x(t) + \epsilon v_{\mathrm{x}}(t+\frac{\epsilon}{2}) \tag{3.15}$$

$$a_{\mathrm{y}}(t) = -y/r^3(t) \tag{3.16}$$

$$v_{\mathrm{y}}(t+\frac{\epsilon}{2}) = v_{\mathrm{y}}(t-\frac{\epsilon}{2}) + \epsilon a_{\mathrm{y}}(t) \tag{3.17}$$

$$y(t+\epsilon) = y(t) + \epsilon v_{\mathrm{y}}(t+\frac{\epsilon}{2}) \tag{3.18}$$

Kde ešte raz x, y boli súradnice telesa, epsilon bol krok výpočtu, a_x, a_y boli zrýchlenia v osiach x a y. Premenná t predstavovala čas a r bola vzdialenosť, ktorú sme vypočítali $\sqrt[2]{x^2 + y^2}$. Tentokrát sa tu však urobil jeden trik, namiesto toho, aby sa výpočet bral z celého kroku, sa bral do úvahy „medzikrok", čiže sme brali vždy do úvahy „spriemerovanú" silu, ktorá by pôsobila v jednom časovom intervale. To je dôvod, prečo bola táto metóda o rád presnejšia ako dve predošlé pri rovnako zvolenom časovom kroku ϵ a to hlavne pri silnom gravitačnom poli (bola to metóda druhého rádu), existujú aj presnejšie metódy, ale pre náš program sa táto výpočtová metóda ukazovala ako dostačujúca. Pre spoľahlivý numerický výpočet efektu retardácie sme si museli byť istí, že nami použitá numerická metóda bola dostatočne presná. Totiž v Keplerovom probléme sa vedelo, ako má dráha vyzerať, ale pri retardácii neexistujú analytické vyjadrenia dráhy.

Kapitola 4

Súčasný stav

Newtonove vzorce predpovedajúce dráhu telies v gravitačnom poli boli známe už niekoľko storočí, preto nebolo zvláštne, že v súčasnosti existovalo veľa simulácií či dokonca hier, ktoré simulovali správanie objektov, či už na vedecké, technické alebo aj zábavné účely. Aj platformy, na ktorých boli súčasné programy vytvorené, boli veľmi rozdielne; boli by sa našli na veľkých superpočítačoch, stolných počítačoch, notebookoch, tabletoch, ipadoch, mobiloch, alebo iných. Taktiež sa nachádzali na rôznych typoch softvéru, ako je Windows, Linux, Mac, IOS, Android či iných. Takmer všetky simulácie dostupné na internete sa opierali o newtonovské vzorce napriek tomu, že už celé storočie boli známe presnejšie vzorce, od doby, kedy Albert Einstein publikoval svoju teóriu relativity. Veľkou väčšinou to bolo preto, lebo newtonovské vzorce boli vo väčšine prípadov dostatočne presné, aby sa odchýlka od reality dala zanedbať, navyše newtonovské vzorce boli oproti vzorcom Alberta Einsteina výrazne jednoduchšie.

V súčasnosti boli niektoré práce v oblasti fyziky modifikáciami newtonovskej teórie gravitácie bez nutnosti sa úplne ponoriť do vzorcov Alberta Einsteina, jedna z fyzikálnych teórií, zaoberajúca sa takýmito modifikáciami, bola aj „MOND“ (Modified Newtonian dynamics), hoci tieto pokusy vysvetliť niektoré javy spojené s pôsobením gravitácie bez nutnosti exotických častíc alebo energií, ako boli napríklad temná hmota a temná energia, boli veľkou väčšinou vedeckých pracovníkov odmietané, ich zástancovia boli o nich pevne presvedčení.

Simuláciám málo uznávaných, hoci niekedy zaujímavých, teórií sa nevenovala takmer žiadna pozornosť a počítačové prostriedky, inak to nebolo ani pri gravitácii. Na internete bolo na tému „MOND“ veľmi málo počítačových simulácií. Tie, ktoré sa vo verejnej sfére nachádzali, boli dvoch typov.

Prvý typ zastupovali simulácie, ktoré boli spustené na superpočítačoch a boli dostupné vo forme videí, ich výhodou boli hlavne vysoká presnosť výpočtov a vysoká výpočtová kapacita, ktorou zvládli vypočítať oveľa zložitejšie N-body problémy ako obyčajne počítač. Druhým (menej častým) typom boli Javascriptové animácie, ktoré

slúžili len ako veľmi nepresná ukážka daného problému, ich výhodou bola ľahká dostupnosť, ale chýbala im presnosť a veľká výpočtová kapacita, čo bolo spôsobené aj tým, že tieto kódy mohli využívať iba jedno vlákno na výpočet. Skutočnosti nepomáhal ani fakt, že Javascript nebol jazykom, ktorý bol blízky strojovému kódu, ktorý by bol s čo najväčšou efektivitou využíval potenciál počítačov (Javascript bol interpreter).

V dobách, kedy Apollo lietalo na Mesiac a kedy považovali za zázrak, že počítač, ktorý mal 2048 „slov“ (2 kB) RAM pamäte, dokázal kontrolovať vesmírnu loď, bolo ešte viac pozoruhodné, že sa za relatívne krátky čas z neho vyvinuli moderné počítače. V súčasnosti každý, hoci aj menej výkonný, mobil je tisícnásobne výkonnejší, ako bol počítač na Apolle 11. Dnešné stolné počítače mohli každému používateľovi poskytnúť dostatočný výkon na to, aby si mohol každý používateľ spustiť, hoci nie takú presnú ako na superpočítačoch, ale stále veľmi presnú simuláciu najrôznejších fyzikálnych modelov. No i napriek súčasnej technike sa simulácie na tému „MOND“ pre osobné počítače takmer vôbec nevyskytovali.

Ľudská myseľ je vo svojej podstate veľmi zvedavá a bolo, je aj bude dobré, keď dostáva podnety na predstavivosť a ďalšie otázky, nad ktorými môže rozmýšľať. Ak mysli ponúkame otázky a témy z menej prebádaných oblastí a tém fyziky, má možnosť skúmať, pýta sa a rozmýšľa, čo v nej môže prebudiť záujem o vedu a techniku. Zvedavosť spolu s kreativitou a záujmom o vedu obyčajne vedú k pokrokom na poli vedeckého pokroku a technických inovácií.

Kapitola 5

Ciele a hypotézy

V tejto kapitole sa rozoberali dokopy 3 podproblémy (problém jedného telesa, problém dvoch telies a problém dvoch telies - s pridanou retardáciou), pričom každý podproblém nadväzoval na poznatky z predchádzajúcich problémov. Konečným podproblémom bol problém dvoch telies - s pridanou retardáciou.

5.1 Problém jedného telesa

Tento podproblém pojednával o pohybe telesa okolo centrálneho nehybného bodu, boli tu vyskúšané numerické metódy a základné princípy, ktoré bolo nutné dosiahnuť, aby sa mohlo pokračovať v riešení ďalších podproblémov.

5.1.1 hypotéza

Ako sa pojednávalo v [1], pre väčšiu presnosť by sme mali zmenšovať krok výpočtu, tento krok výpočtu, známy tiež ako epsilon, udával presnosť numerickej simulácie. Podľa [1] bola Leapfrog presnejšia metóda ako Euler a Euler-crommer, a zároveň bola dostatočne presná, aby dokázala dávať presné výsledky.

5.1.2 cieľ

Cieľom bolo overiť, či je možné nájsť s metódami (Leapfrog, Euler, Euler-crommer) dostatočne malé epsilon, a zároveň dokončiť numerický výpočet v rozumnom čase, alebo či by bolo treba použiť iné, zložitejšie výpočtové metódy. Neoddeliteľnou súčasťou cieľa pre tento podproblém bolo overenie, či dátový typ float64 (C-ovský typ double) má dostatočnú desatinnú presnosť, aby vyhovoval numerickému výpočtu, alebo či by bolo nutné použiť float128, ktorý by však kládol väčšie nároky na výpočtový čas.

Taktiež bolo cieľom tohto podproblému zistiť, či sa v numerickej simulácii dodržiavali Keplerove zákony, či bolo naozaj možné v jazyku Golang počítať fyzikálne procesy

s dostatočnou presnosťou.

5.2 Problém dvoch telies

V tejto časti sa pojednáva o vzájomnom pohybe dvoch telies bez vplyvu okolia. Neoddeliteľnou súčasťou tohto podproblému bolo využitie paralelných procesov na výpočet dráh.

5.2.1 hypotéza

Pri probléme dvoch telies metóda Leapfrog a dátový typ float64 boli dostatočne presné na to, aby sa s nimi dalo rýchlo vypočítať požadované výsledky, ktoré sa dali považovať za situáciu, ktorá by mohla vo vesmíre nastať. V porovnaní s problémom jedného telesa bol výpočet dvojnásobne náročnejší kvôli potrebe počítať dve telesá namiesto jedného, a aj prenos a ukladanie dát kládli dvojnásobok požiadaviek na beh programu, bola nutnosť implementovať efektívny spôsob ich manipulácie.

5.2.2 cieľ

Cieľom bolo vyriešiť problémy s dvojnásobnými požiadavkami zo strany množstva dát, ich prenosu a ukladania a overiť presnosť metódy Leapfrog a dátového typu float64 pri dvoch telesách. Súčasťou cieľa bolo vylepšenie ukladania dát a následná rekonštrukcia stavu programu pred uložením. Cieľom bolo čo najefektívnejšie využiť viacjadrový procesor a s tým spojené synchronizačné problémy, no hlavným cieľom bolo overiť správanie sa numerickej metódy v porovnaní s analytickým riešením rovníc na eliptickú dráhu. Analytické riešenie bolo neskôr možné využiť v probléme dvoch telies s pridanou retardáciou.

5.3 Problém dvoch telies - s pridanou retardáciou

Táto časť vychádza z Problému dvoch telies, ale táto časť pojednáva o tom, ako zapracovať retardáciu v newtonovskej fyzike do výpočtov, čo sa od toho očakávalo a čo boli priority.

5.3.1 hypotéza

Aj pri tomto probléme stačila metóda Leapfrog a dátový typ float64, ak sa osvedčili v probléme jedného telesa. Rýchlosť bola menej dosiahnuteľná ako v predchádzajúcom probléme. Predpokladalo sa, že v dôsledku retardácie gravitácie, ktorá bola spôsobená konečnou rýchlosťou svetla, bude vidieť zmena dráhy u oboch telies. Táto zmena bude

závisieť od vzdialenosti telies a od ich vzájomnej rýchlosti. Zmena sa týka veľkosti dráh, a to tak, že sa od seba neustále vzďaľujú - retardácia gravitácie telesám dodáva stále väčší moment hybnosti.

5.3.2 cieľ

Cieľom tejto časti bolo simulovať hypotetickú možnosť pridania retardácie do newtonovských vzorcov a vyskúšať rôzne nastavenia podľa rôznych parametrov, od vzájomnej rýchlosti polôh až po rôzne nastavenia rýchlostí šírenia gravitačnej interakcie. Hlavným cieľom bolo skúmať dôsledok retardácie gravitácie na kruhové a všeobecnejšie eliptické dráhy, ktoré boli bez nej uzavreté a stabilné. Ako aj v minulých častiach bolo potrebné docieliť optimálnu rýchlosť vykresľovania a optimálnu rýchlosť výpočtu.

Kapitola 6

Návrh experimentov

6.1 Keplerov problém

Pri Keplerovom probléme sa bude možné najviac opierať o voľne dostupné programy, kde bude potrebné zistiť, či program bude počítať pre rovnaké vstupy tie isté výstupy, či sú zvolené metódy dostatočne stabilné aj pri silných gravitačných poliach. Je potrebné vyskúšať rôzne dráhy a overiť, či dráha opísaná sprievodičom za rovnaký čas je vždy rovnaká (či platí 2. Keplerov zákon), na to nám poslúži výpis v samotnom programe. experimenty budú pozostávať z postupného zmenšovania ϵ pri rôznych konfiguráciách.

6.2 Problém dvoch telies

Aj pri probléme dvoch telies sa bude možné najviac opierať o voľne dostupné programy, kde bude potrebné zistiť, či program bude počítať pre rovnaké vstupy tie isté výstupy. Experimenty budú založené na porovnávaní výstupov z rôznych konfiguračných súborov a tiež postupnom znižovaní ϵ tak, aby sa čo najpresnejšie opisovala dráha vypočítaná analytickým spôsobom, a tak zistiť, ktoré ϵ je dostatočné malé na to, aby výsledok bol dostatočne presný, aby sa prakticky nelíšil od výpočtu dráhy pomocou analytických metód.

6.3 Problém dvoch telies - s pridanou retardáciou

Vo fyzikálnom modeli Isaaca Newtona sa musí počítať s okamžitou gravitačnou interakciou, čo znamená, že gravitácia dostáva v našom ponímaní nekonečnú rýchlosť jej šírenia. Vo vesmíre je rýchlosť, akou sa môže šíriť informácia, obmedzená na hodnotu 299 792 458 metrov za sekundu, medzi laikmi známou aj ako rýchlosť svetla. Dôsledkom teórie relativity od A. Einsteina je, že aj gravitačné pôsobenie sa šíri v súlade s

týmto rýchlostným obmedzením šírenia informácie, zároveň môžeme predpokladať, že sa gravitácia skutočne šíri rýchlosťou, ktorá je rovnaká alebo blízka maximálnej rýchlosti šírenia informácie a zároveň fyzici vedia, že jednoduchým pridaním retardácie z Newtonových vzorcov sa nedá vyrobiť ekvivalent teórie relativity, preto ak v simulácii nebudeme mať nastavenú rýchlosť šírenia gravitácie niekoľkonásobne väčšie ako rýchlosť svetla, či ju budeme dokonca „umelo" znižovať, síce by sme to mohli brať ako obyčajné zvýraznenie obyčajne zanedbateľného prírodného javu, ktorý nie je možné merať, alebo si čosi požičať od teórie relativity a môžeme to interpretovať ako relatívne a výrazné zväčšenie priestoru medzi nimi a aj relatívne zvýšenie ich rýchlostí bez nutnosti meniť ich reálne parametre, ale mýlili by sme sa. Takéto vzorce nebudú dávať presné výsledky, preto budú experimenty pozostávať zo skúmania, ako sa rýchlosť šírenia gravitačnej interakcie prejaví na tvare dráhy a správaní sa ostatných parametrov oproti prípadu, kedy sa počíta s okamžitou interakciou gravitácie. Pre aspoň jednu konfiguráciu bude potrebné vyskúšať, pre akú rýchlosť šírenia gravitačnej interakcie sa dráha takýchto výpočtov zhoduje s okamžitou interakciou, a pre ktorú už sa výpočty nezhodujú.

Kapitola 7

Implementácia

Implementácia sa delí na tri samostatné celky. Počas riešenia jednotlivých podproblémov, popísaných v kapitole Ciele a hypotézy, vznikli tri samostatné programy, ktoré riešia daný podproblém. Prvý program je najjednoduchší, každý ďalší program je komplexnejší. Napriek tomu, že na seba programy nadväzujú, v niektorých ohľadoch boli napísané úplne iným štýlom a nezdieľajú rovnaký kód. Pokiaľ prvý a druhý program nezdieľajú až na výpočtové časti skoro žiaden kód, tak tretí bol postavený na druhom s drobnými úpravami a s pridaním jednej fyzikálnej myšlienky zdieľa s druhým 95% kódu. Všetky tri samostatné celky využívajú knižnicu Gonutz/prototype, ktorá je popísaná v kapitole Prehľad technológií v sekcii GUI, a preto v jednotlivých celkoch je už táto informácia pokladaná za samozrejmosť.

7.1 Problém jedného telesa

Pri implementácii tejto cesty sme nevyužívali package (priečinky), ktoré sú vstavané v jazyku Golang, kód je organizovaný do jednotlivých súborov, ktoré sú v jednom hlavnom adresári. Z každého súboru môžeme zavolať hociktorú funkciu alebo globálnu premennú. Každú výpočtovú metódu sme implementovali ako samostatnú goroutinu. Celkovo v programe bežia tri výpočtové goroutiny, jedna, ktorá zbiera vypočítané dáta, a zároveň synchronizuje všetky tri výpočtové goroutiny, a ešte jedna goroutina, ktorá má na starosti vykreslenie nazbieraných dát.

Jednotlivé súbory sa v ďalšom probléme stali package-mi (priečinkami), ale v tejto fáze programovania to nebolo potrebné. Kvôli nutnosti overiť, či jednotlivé numerické metódy, popísané v predchádzajúcej kapitole, so želanou presnosťou vypočítajú správne dáta na jednotlivých vopred určených vstupoch, kód obsahuje knižnicu 360EntSecGroup-Skylar/excelize, ktorá zapisuje jednotlivé dáta do excelovských súborov. Táto knižnica bola dôležitá, aby boli dáta človekom ľahko čitateľné a interpretovateľné. Dáta sa nezapisujú priamo na disk, ale zapisujú sa do štruktúry, ktorú definuje

knižnica 360EntSecGroup-Skylar/excelize a po určitých intervaloch, ktoré si používateľ môže v konfiguračnom súbore nastaviť, sa vyrobí nová štruktúra a stará sa pomocou samostatnej goroutiny zapíše na disk.

V kóde sa pracuje s knižnicou Gonutz/prototype procedurálne, to znamená, že hlavná funkcia sa delí na viaceré podčasti, ktoré však nemajú spoločné objekty (class-triedy).

Na načítanie konfiguračného súboru, ktorý obsahuje vstupné nastavenia, využívame atypickú vlastnosť pre štandardný kompilovateľný jazyk, a to reflexiu. Reflexia je vykonávaná na štruktúre (objekte), ktorú sme nazvali Euler, tá má za úlohu načítať konfiguračný súbor a nastaviť počiatočné podmienky pre výpočtové goroutiny, gourutinám sa odovzdáva iba kópia načítanej štruktúry z konfiguračného súboru.

Hoci na začiatku bol hlavným problémom faktor rýchlosti, problémom pri implementácii tejto časti bola paradoxne aj privysoká rýchlosť animácie, ktorú človek nebol schopný vnímať, preto bolo nutné animáciu trochu spomaliť, aby boli ľudia schopní vnímať pohyb daného telesa okolo hmotného stredu.

Ďalším problémom pri implementácii bola skutočnosť, že výpočtová časť využíva jednotkovú sústavu, ktorú bolo treba škálovať, otočiť osi x, y a posunúť stred do stredu zobrazovacieho okna. Ďalšou výzvou bolo zobrazenie planéty v blízkosti centrálneho bodu tak, aby sa automaticky približoval obraz, keď bola planéta blízko stredu a vzďaľoval, keď planéta vybiehala zo zobrazovacieho okna.

Ďalšími výzvami, ktoré sme pri písaní tohto programu stretli, bolo vytvorenie okna pre zrýchlené počítanie bez animácie, možnosť voľby zmeniť niektoré vstupné parametre za behu programu v úvodnom okne a v neposlednom rade nepresnosti strojových desatinných čísel.

Pre nutnosť zrýchlenia bolo nutné vyriešiť obrodenia synchronizačných mechanizmov potrebných pri zobrazovaní animácie, ktoré spomaľovali samotný výpočet, preto vzniklo zobrazenie, v ktorom sa v polsekundových intervaloch aktualizoval aktuálny stav progresu výpočtu, a zároveň tak, aby samotný výpočet mohol bežať bez obmedzení; tu sa vyskytol problém v zvolenej grafickej knižnici (Gonutz/prototype), ktorá je napísaná tak, aby stále aktualizovala obraz, ktorý zakaždým prekreslila na čierne pozadie. Vyriešili sme to miernou modifikáciou samotnej knižnice tak, aby nezotierala starý obraz; vedľajším efektom tejto zmeny bola nutnosť manažovať zatieranie starého obrazu priamo vo vlastnom kóde.

Pre možnosť voľby, ktorá znamená zmeniť niektoré vstupné parametre počas behu programu v úvodnom okne, bolo potrebné, aby sme si vytvorili zoznam mien premenných, ktoré sa majú dať meniť, a opäť využili reflexiu, neskôr však sa ukázalo, že elegantnejšie riešenie poskytuje samotná reflexia a ďalšia vlastnosť jazyka Golang - zabudované štruktúry (embedded structs). Pomocou týchto vlastností sme rozdelili štruktúru Euler na mutable a immutable vlastnosti a spojili ich pod pôvodným menom.

Presnosť desatinných čísel v jazyku Golang typu float32 sa používa v bežných aplikáciách, avšak pre výpočty dráh sme museli siahnuť po dátovom type float64, ktorý je síce zabudovaný v samotnom jazyku, ale pre každé vykreslenie sa musí konvertovať hodnota do správneho typu. V knižnici Gonutz/prototype sa však používa dátový typ int, preto sme si okrem škálovania (popísané vyššie) museli pri každom kreslení byť istí, že dané číslo najprv prenásobíme na správnu veľkosť, a až potom sme mohli číslo skonvertovať na dátový typ int.

7.2 Problém dvoch telies

V tomto programe sme zabudovali jednotlivé súbory do priečinkov (package), kvôli vlastnosti jazyka Golang sme si nemohli dovoliť cyklické importovanie package-ov (Import cycles alebo cyclic dependency), preto sme v programe urobili dva package, ktoré obsahujú globálne premenné alebo globálne konštanty. Projekt sme rozdelili do 6 package-ov a 2 priečinkov, pričom package sú dôležité len pri písaní samotného kódu, ale priečinky nazvané „bin" a tiež „config" slúžia programu počas samotného behu. Program sme urobili tak, aby si priečinok „bin" vedel vyrobiť počas behu. Do priečinka „bin" program ukladá binárne súbory, ktoré popíšem nižšie. Priečinok „config" obsahuje konfiguračný súbor, ktorý je možné editovať, a tým meniť postupné parametre simulácie. Program využíva niektoré implementačné riešenia z predchádzajúceho problému. Riešeniami, ktoré sme využili z predchádzajúceho problému, sú škálovanie a využitie reflexie pre načítanie konfiguračných údajov.

Od začiatku implementácie sa objavoval problém v rýchlosti výpočtov, preto pri tomto programe pozostáva simulácia z vopred vypočítaných polôh (respektíve dráh). Pre zvýšenie rýchlosti a aj preto, aby sa výpočet dal znovu vyvolať po opätovnom spustení programu, sa vypočítané dráhy jednotlivých telies ukladajú do samostatných binárnych súborov namiesto jedného excelovského súboru. Každý binárny súbor obsahuje informácie o jednej orbite (obehu) pre jednu planétu.

Výhodou binárnych súborov oproti excelovským súborom z problému jedného telesa bola rýchlosť načítania / zapisovania dát, ktoré sme priamo skonvertovali do poľa float64 a naopak. Táto konverzia bola možná vďaka malému triku, ktorý sa týkal priamej konverzie z poľa bajtov. Vďaka tomuto triku, ktorý je vysvetlený nižšie, sme dramaticky zvýšili rýchlosť a jediné obmedzenia, ktoré ostávajú, sú rýchlosť načítania dát z disku do pamäte RAM (a naopak), najväčším obmedzujúcim faktorom však ostáva veľkosť pamäte RAM samotnej.

Konverzia polí je trik, označený v jazyku Golang ako „unsafe" operácia. Tento trik je nebezpečný v tom zmysle, že garbage collector, vstavaný v jazyku Golang, neregistruje skonvertovaný pointer na pole a tým pádom môže neplánovane a nekontrolovateľne

odstrániť (upratať) starý pointer spoločne s poľom, ktoré ešte chceme používať, a tým by spadol celý program (vlákno, ktorý by sa snažilo adresovať už neexistujúce pole). Dôvodom odstránenia poľa je, že už pôvodné pole nikde nepoužívam, a teda už naň neukazuje žiadny pointer, a preto z pohľadu garbage collector-a je nutné odstrániť - dealokovať nepoužívanú pamäť. Moje riešenie spočíva v pamätaní si starého pointra spolu s tým, ktorý používam v programe. Pamätanie si dvoch pointrov na to isté pole na dvoch rôznych miestach z hľadiska bezpečnosti-prehľadnosti a čitateľnosti kódu nie je dobré a celkovo vzďaľuje kód od ideálu čistého kódu, preto som to vyriešil štruktúrou, ktorá má v sebe oba pointre, teda aj pôvodný pointer, aj pointer na pole čísel.

Štruktúru „Euler" z problému jedného telesa sme museli znovu pozmeniť, a to tak, že sme pôvodnú štruktúru rozdelili na „features" a „planets" a znovu zabudovali do jednej pod pôvodným názvom „Euler". „Features" obsahuje väčšinu vlastností, ktoré obsahoval „Euler" z predchádzajúceho problému, ktoré ukladá do svojich premenných rôznych typov. „Planets" obsahuje všetky údaje o planétach a tiež sa tieto údaje načítajú z konfiguračného súboru. Pri vlastnostiach z „Features" sme pri implementácii využili aj takzvané „tag" označenia jednotlivých premenných, to mi umožnilo triediť premenné na tie, ktoré sa zobrazujú v úvodnom formulári a na tie, ktoré sa do úvodného formulára nemajú zobrazovať, štuktúru „Euler" sme však museli trochu oddialiť od ideálu čistého kódu, pretože bolo nutné, aby program nebol case-sensitive, preto vo vnútri mien premenných nie sú veľké písmená.

Keďže program mal vedieť počiatočný aj aktuálny stav výpočtov po celú dobu behu, tak sa tieto štruktúry museli odkladať do špeciálnych súborov obsahujúcich jsony týchto štruktúr. Tieto uložené údaje spolu s binárnymi súbormi umožňujú rýchle a jednoduché znovunačítanie akéhokoľvek minulého výpočtu. Práve popísané štruktúry slúžia aj na umožnenie zobrazenia analytickej dráhy v ktoromkoľvek okamihu animácie.

Počas implementácie programu sa ukázalo, že je potrebné spraviť prijateľnejšie grafické prostredie pre užívateľa, ako mal problém jedného telesa, ktoré malo relatívne jednoduché GUI. Keďže grafická knižnica, ktorú používam, nemá zabudované nástroje pre vytváranie tlačidiel textových polí, označených checkboxov, museli sme ich naimplementovať. Všetky grafické prvky sme kvôli prehľadnosti kódu združili pod jedným interface-som nazvaným „shape", ktorý navyše je celý implementovaný štruktúrou „Basic", ktorú všetky moje grafické prvky majú zabudovanú a od ktorej ostatné dedia dve hlavné metódy, ktorými sú „Paint" a „CarryEvent", ktoré vykresľujú a kontrolujú oblasť im prislúchajúcu. Metóda „CarryEvent" u väčšiny mojich grafických komponentov využíva metódu „GetFunc", ktorá vracia funkciu, ktorá sa za určitých podmienok aktivuje.

Počas behu programu sú implementované rôzne pohľady, pričom každý pohľad má vlastné štruktúry, ktoré sa inicializujú pri jeho prvom zavolaní. Všetky pohľady sú

implementované tak, aby sa používateľ vedel v nich orientovať a intuitívne bez čítania návodu ovládať program počas celého behu.

Jeden z problémov, ktorý sa vyskytol pri implementácii tohto problému, bol, ako program vypočítaval nové polohy telies (dráhu telies), preto sa pri každom cykle vykonávanie výpočtu rozdelí na dve samostatné goroutiny, ktoré samostatne vypočítajú novú polohu a zároveň ju zapíšu do adekvátnej štruktúry. Paralelizáciou pomocou samostatných goroutín sme pri výpočtoch docielili zrýchlenie o 80%.

7.3 Problém dvoch telies - s pridanou retardáciou

Pri implementácii tohto problému sme použili väčšinu zdrojového kódu z problému dvoch telies bez retardácie. Do štruktúry Euler sme pridali vlastnosti, ktoré sa využívajú práve pri retardácii. Týmito vlastnosťami sú rýchlosť šírenia gravitácie a metóda, ktorá sa má aplikovať.

Oproti minulým problémom však využívame „middleware", ktoré sú veľmi dôležité pri počiatočnom výbere výpočtovej metódy, zvolenej používateľom pri spustení programu. Tieto „middleware" znamenajú, že pracujeme s funkciami ako s premennými, čo značne zlepšuje čitateľnosť a prehľadnosť kódu, a tak dostáva kód bližšie k ideálom čistého kódu. Najväčším problémom u tohto problému bola rýchlosť výpočtov, ktorú sa mi podarilo vyriešiť malými úpravami pôvodného kódu.

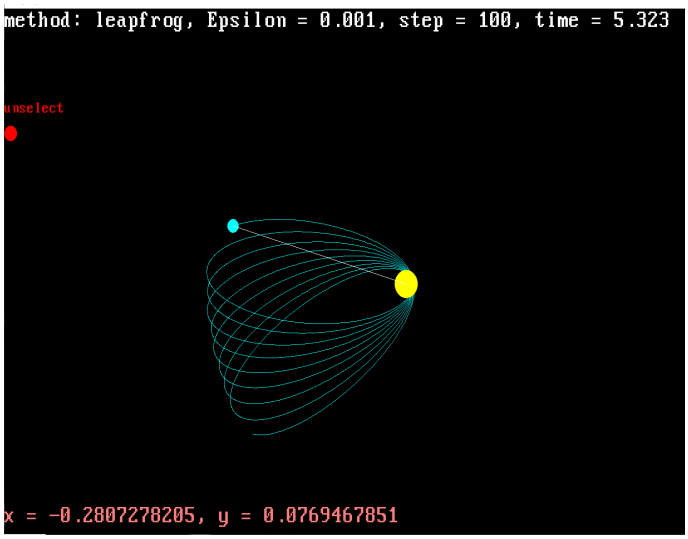
Nakoniec sme sa rozhodli, že iteratívne riešenie nenaimplementujeme, pretože by to bolo veľmi náročné nielen na rýchlosť, ale aj na operačnú pamäť a celkovú zložitosť pri správe pamäte.

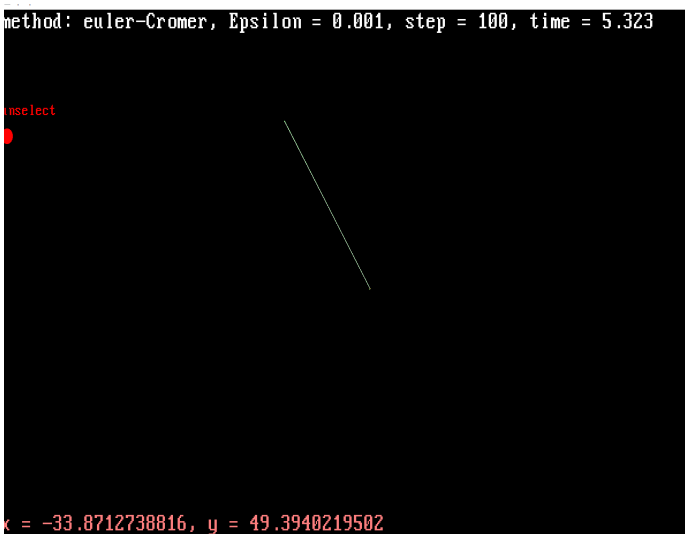
Kapitola 8

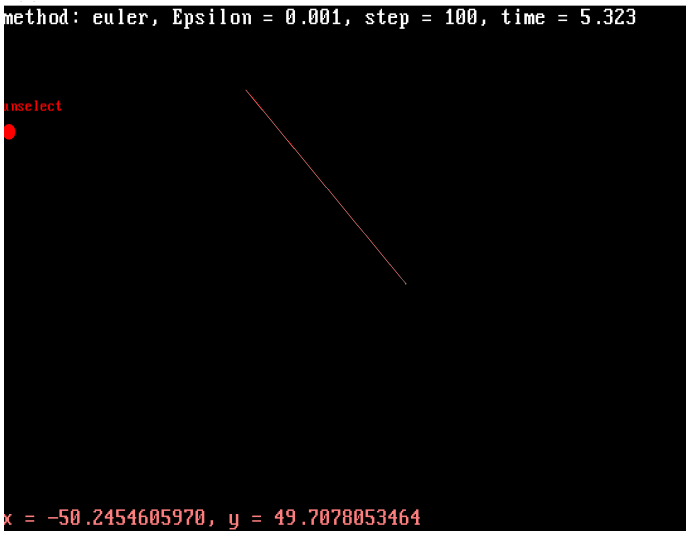
Experimenty a výsledky

8.1 Keplerov problém

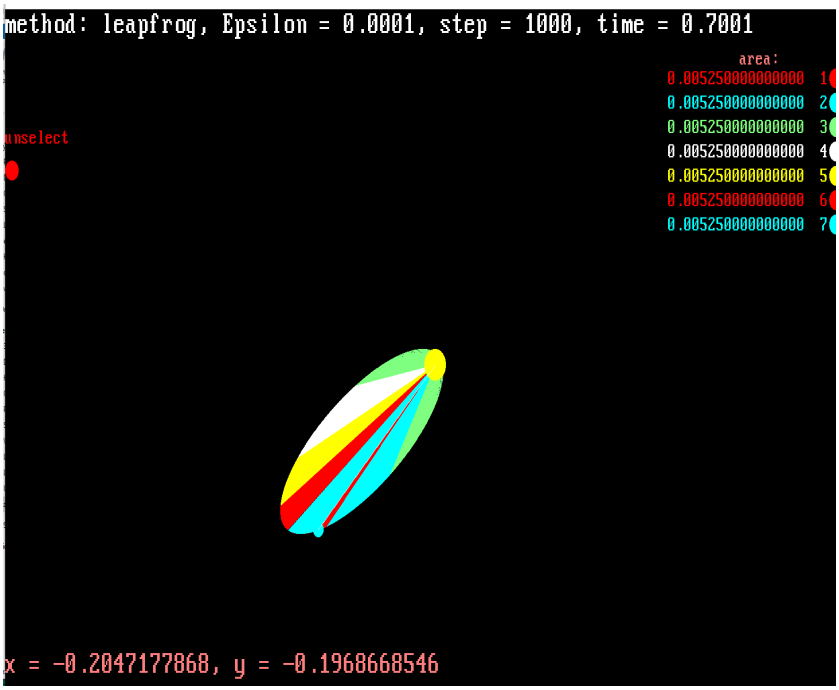
Pri riešení tohto problému sme urobili program, ktorý funguje s rýchlosťou vhodnou aj na priame predvedenie, teda je vhodný na okamžitú interakciu s používateľom. Nasledujúce obrázky budú ukážkami z kódu na tom istom konfiguračnom súbore:







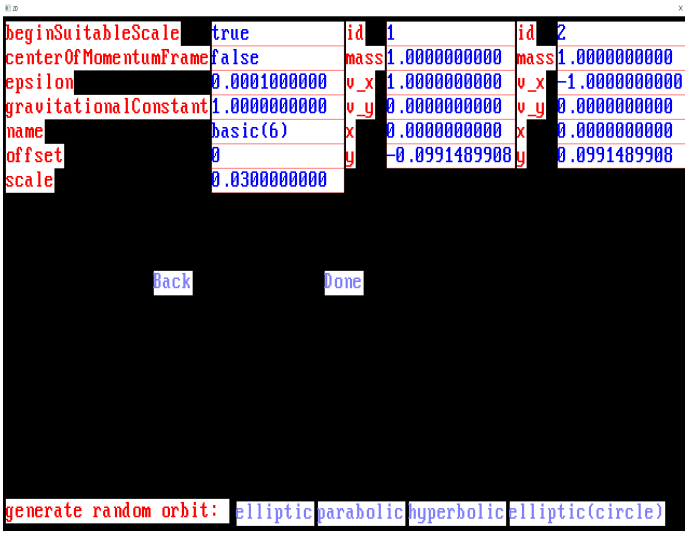
Ako je vidieť na obrázkoch, „leapfrog", ktorá je na prvom obrázku, bola najpresnejšia metóda podľa očakávania, hoci pri $\epsilon = 0,001$ metóda „leapfrog" vykazuje veľkú precesiu, stále sa udržala na stabilnej orbite okolo hmotného bodu, na rozdiel od klasickej metódy „Euler", ktorá pri nepresnosti a silnom gravitačnom poli nabrala vysokú únikovú rýchlosť. Podobne aj metóda „Euler-cromer" nabrala únikovú rýchlosť, ale ako je vidieť zo zobrazenia jej pozície, jej nepresnosť bola menšia oproti metóde „Euler".



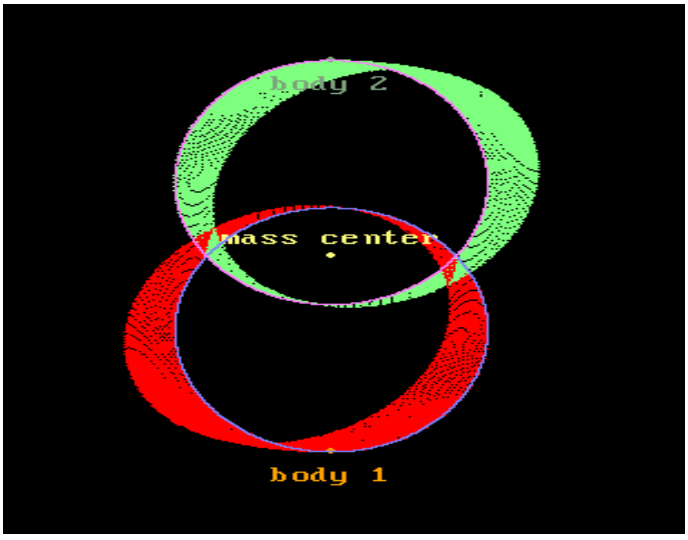
Tento obrázok ukazuje ďalší výsledok programu, ktorý ukazuje, že druhý Keplerov zákon pri výpočtoch platil. Podľa výpisu je možné vidieť, že plocha opísaná sprievodcom za rovnaký čas bola v každom okamihu rovnaká. Navyše metóda „leapfrog" pri tejto konfigurácii dokázala vykresliť presnú dráhu pri $\epsilon = 0,0001$.

8.2 Problém dvoch telies

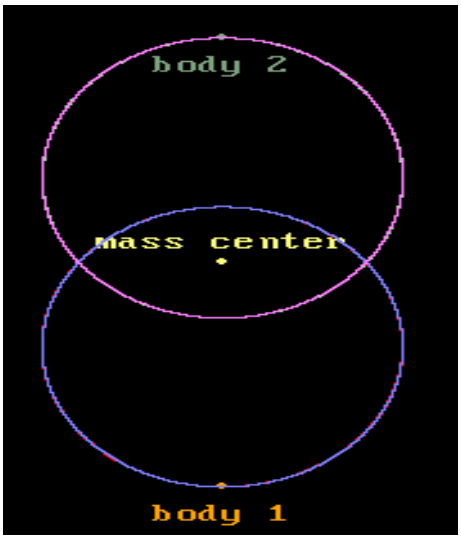
Oproti minulému problému sme tomuto programu urobili priateľskejšie užívateľské prostredie. Hlavnou výhodou je vstupný formulár, ktorý sa objavuje po spustení programu. Tento program navyše už používa astronomické jednotky, čiže je uspôsobený na to, aby počítal s reálnymi pomermi našej slnečnej sústavy, tie sú vysvetlené v prílohe B k zadaniu bakalárskej práce, vypracovanej RNDr. Eduardom Masárom, PhD.



Ako je vidieť na tomto obrázku, tak formulár obsahuje všetky atribúty nastavenia vstupných podmienok. Navyše formulár ponúka možnosť automaticky vygenerovať dráhy podľa želania.

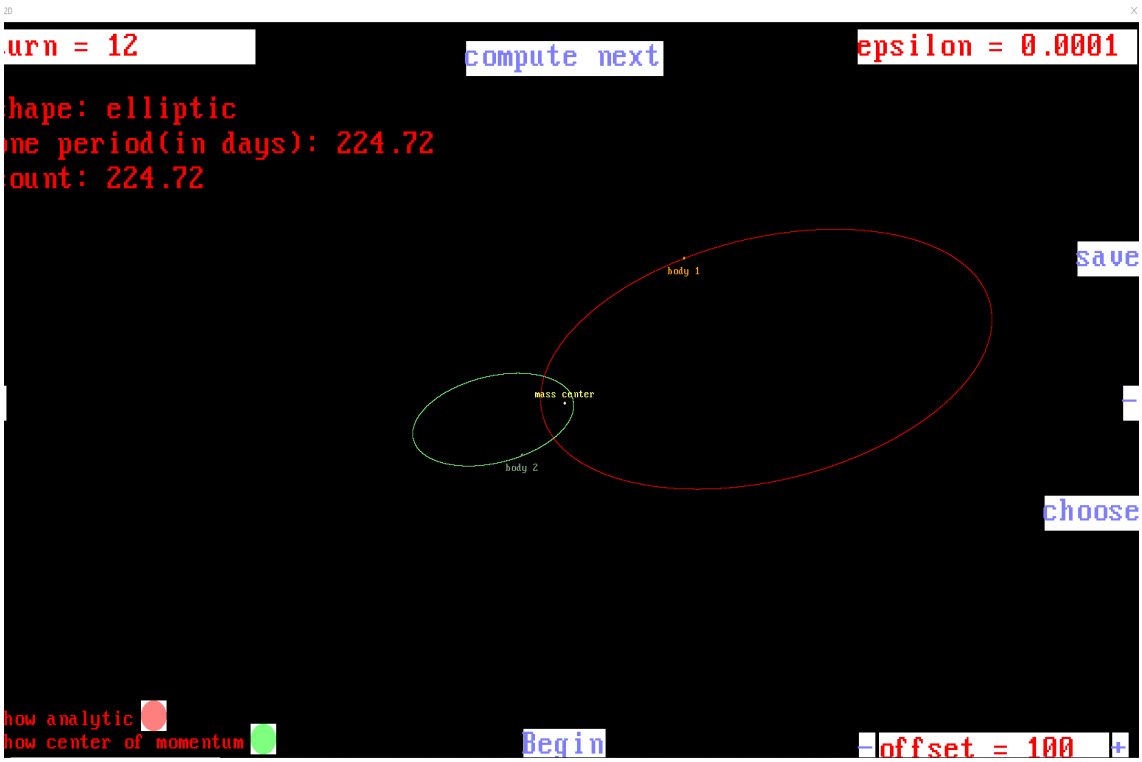


Aj v tomto probléme sa ukázalo, že pri väčšine konfiguračných súborov (dráh), v ktorých sa objekty nedostávajú príliš blízko, je $\epsilon = 0,001$ dostatočné, aby sa objekty udržali na stabilných obežných dráhach, ako je vidieť na obrázku, ale na to, aby sa eliminoval jav známy ako precesia, musíme ϵ znížiť ešte o jeden alebo dva rády. Potom dostaneme pomerne presnú dráhu, zodpovedajúcu analytickému riešeniu, ako ukazuje nasledujúci obrázok:



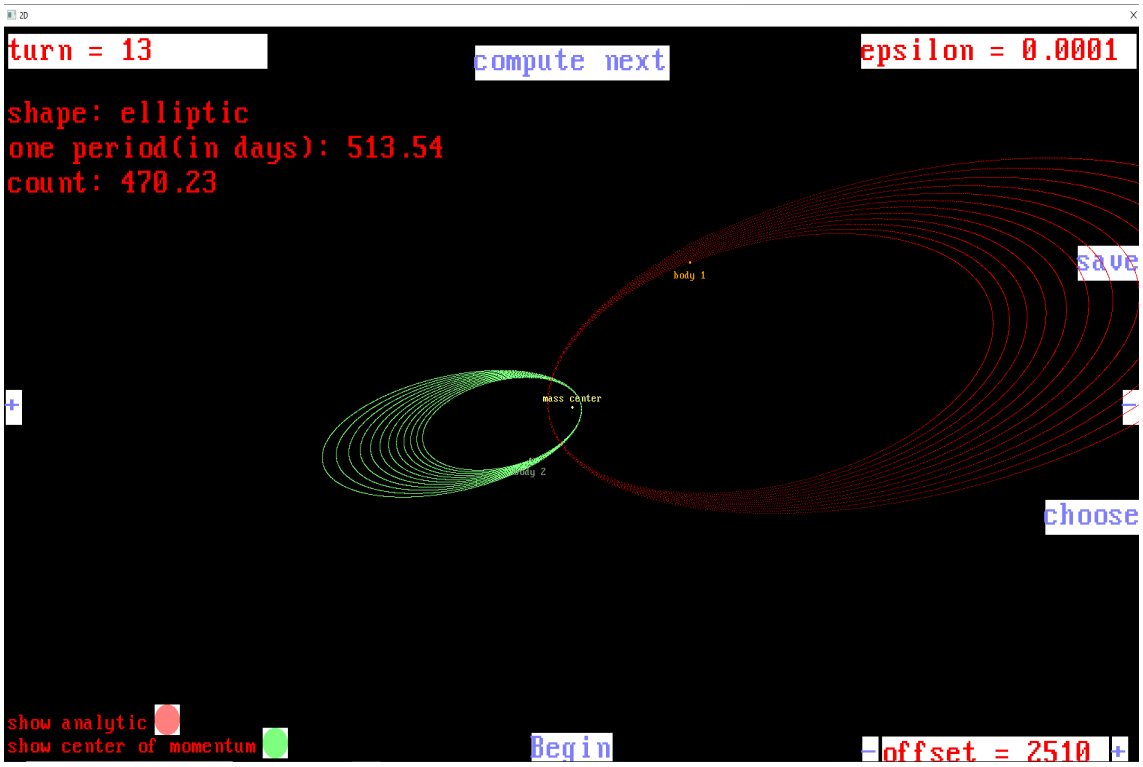
8.3 Problém dvoch telies - s pridanou retardáciou

Tu sa nám podarilo docieliť požadovanú presnosť podobne, ako pri predchádzajúcom probléme. Aj tento program používa astronomické jednotky, ako to bolo pri probléme dvoch telies bez retardácie. Pri experimentovaní s rôznymi rýchlosťami šírenia gravitačnej interakcie sa ukázalo (podľa očakávania), že rýchlosť šírenia gravitačnej interakcie, a teda, ako v kapitole o návrhoch experimentov bolo vysvetlené, aj vzdialenosť a vzájomná rýchlosť, hrajú v tomto modeli vesmíru veľmi dôležitú úlohu pri správaní dvoch telies.

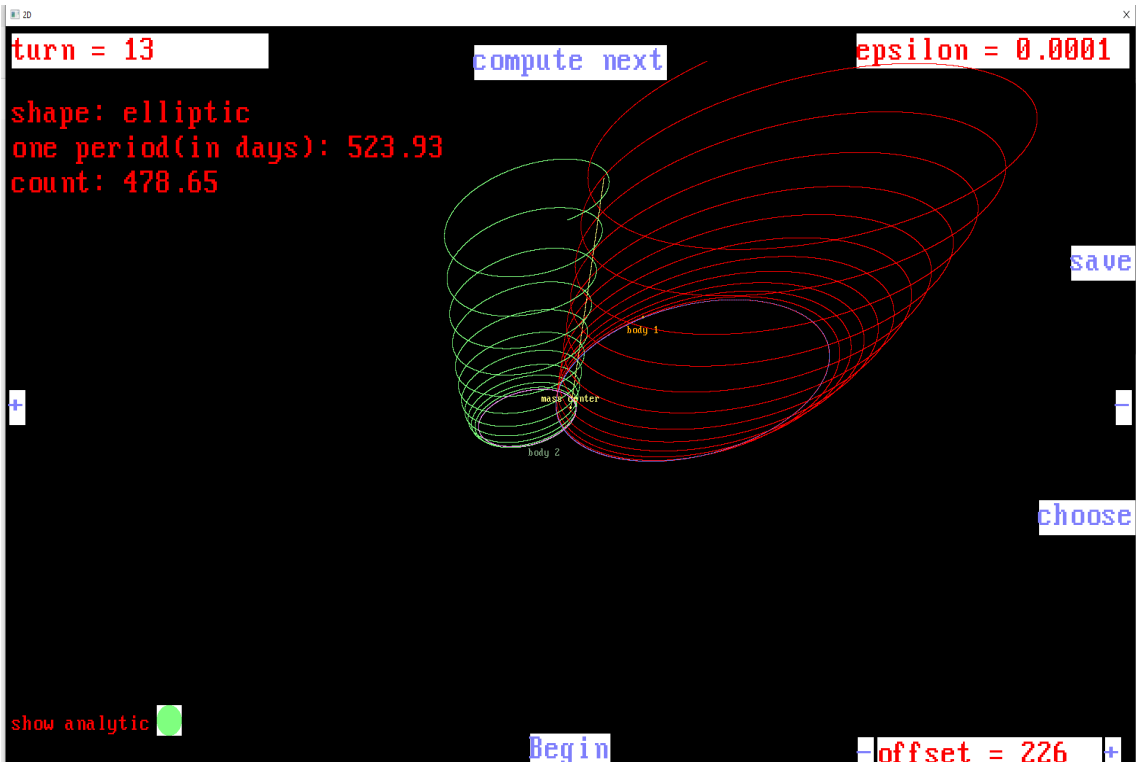


Na tomto obrázku bola rýchlosť šírenia gravitačnej interakcie nastavená na vysokú

hodnotu, konkrétne na stonásobok rýchlosti svetla. Ako je vidieť, telesám sa nepridala dodatočná energia, ktorú by sme dokázali na obrázku vidieť. Ťažisko binárnej sústavy bolo na tomto obrázku pevne zviazané so stredom (s bodom 0,0).



V tomto prípade bola rýchlosť šírenia gravitačnej interakcie nastavená na veľmi malú hodnotu (konkrétne na 17,3144 AU/deň, teda na 1/10 rýchlosti svetla), čo by sme mohli vnímať ako zveličenie alebo, ako je napísané v kapitole o návrhoch experimentov, ako relatívne zrýchlenie objektov a rozšírenie priestoru okolo nich aj medzi nimi, to spôsobí, že oneskorenie interakcie sa výrazne zväčší a energia, ktorú tento efekt dodá, mnohonásobne vzrastie. Na obrázku nie sú vidieť všetky pohyby, pretože v tomto prípade program odčítava dynamicky a pretransformuje systém do ťažiskovej sústavy iba na zobrazenie, to znamená, že nech sa ťažisko pohne akýmkoľvek smerom, tak program ho vždy umiestni do rovnakého bodu (0,0) na obrazovke. V skutočnosti by pozorovateľ, ktorý by bol takto zviazaný s ťažiskom, pociťoval zrýchlenie. Na nasledujúcom obrázku sa ťažisková sústava nastaví len na začiatku výpočtu a nebude sa upravovať dodatočne.



Ako môžeme vidieť na obrázku, keď bola na začiatku nastavená ťažisková sústava, oneskorenie gravitácie spôsobilo pohyb ťažiska sústavy dvoch telies. Toto môžeme vyhlásiť za úspech, pretože sme ukázali, že pridanie retardácie do Newtonovej fyziky prináša so sebou porušovanie mnohých fyzikálnych aspektov. Telesá v porovnaní so štandardnou newtonovskou gravitáciou získavajú väčšiu rýchlosť a uzavreté orbity sa menia na neuzavreté s väčšími vzdialenosťami medzi telesami. Táto simulácia teda ukázala, že pridanie retardácie do klasickej newtonovskej teórie gravitácie porušuje zákony zachovania energie, hybnosti aj momentu hybnosti, a teda je fyzikálne neprijateľné.

Záver

Témou tejto bakalárskej práce bola simulácia newtonovskej fyziky, do ktorej sa započítal efekt konečnej rýchlosti šírenia svetla, ktorý spôsobí oneskorenie gravitačnej interakcie, čiže takzvanú retardáciu. Celé úsilie sa rozdelilo do troch problémov. Každý problém dal vzniknúť samostatnému programu, pričom každý mal za úlohu overiť princípy predtým, než sa pokročilo na ďalší program. Najprv bolo potrebné, aby bolo vidieť, čo sa deje s výpočtami, ktoré program vykonáva pomocou numerických metód a klasických vzorcov. Bol predpoklad, že numerická metóda „leapfrog" by mala byť dostatočne presná, aby bola oporou nielen v raných fázach tvorby prvého programu, ale aj pri ďalších programoch. Tiež bolo potrebné, aby sa overilo, či v programovacom jazyku Golang je možné rýchlo a s dostatočnou presnosťou vykonávať fyzikálne výpočty. Nakoniec sa potvrdilo nielen, že sa v programovacom jazyku Golang dajú písať fyzikálne výpočty, ale aj, že metóda „leapfrog" je dostatočne presná a mohla byť základom aj pre problém dvoch telies bez retardácie, aj pre problém dvoch telies s retardáciou. Celý čas bola problémom rýchlosť vykonávania, spracovávania a aj zobrazovania výpočtov.

Hlavný cieľ sa nám podarilo naplniť, tretí program dôveryhodne simuluje (zobrazuje) efekt pridania retardácie a zo simulácie je vidno, že pri takýchto modeloch sveta sa objavujú javy, ktoré nemajú v prírode reálny základ, a preto môžeme vyhlásiť tento model za nesprávny. Jednotlivé programy sú pripravené demonštrovať používateľom rôzne fyzikálne modely fungovania binárnych systémov, či už by išlo o zjednodušenú verziu - model, ktorý sa označuje ako problém jedného telesa, alebo model známy ako problém dvoch telies a jeho upravenú verziu s pridanou retardáciou. Zároveň sa podarilo ukázať, že priame pridanie efektu retardácie nepovedie k správnym výsledkom, a teda ukázať, že tento model nezodpovedá fyzikálnej realite sveta okolo nás.

Hlavným prínosom tejto bakalárskej práce je priblíženie menej známeho fyzikálneho modelu verejnosti a jeho sprístupnenie na štandardné platformy, ktoré sú rozšírené medzi verejnosťou, ktorej je interaktívnou formou ukázaná nepravdivosť myšlienky pridávania retardácie do Newtonových vzorcov (tak, aby to stále bola nerelativistická teória).

Z hľadiska fyziky by sa 3. program mohol v budúcnosti rozšíriť o vzorce s väčšou presnosťou, ako používame v momentálnej verzii programu. Rovnako zaujímavou možnosťou by bolo iteratívne hľadať v minulých pozíciách podľa vzorca 23 z kapitoly 2, to by však nieslo veľmi veľkú výpočtovú náročnosť.

Z hľadiska informatiky tiež existuje niekoľko možností, ako v 3. programe pokračovať. Prvou zaujímavou možnosťou je výmena grafickej knižnice „Gonutz/prototype" za „gioui/gio", ktorá by umožnila rozšíriť program na viac platforiem, napríklad mobilných zariadení, okrem Webassembleru, kde by nastalo výrazné spomalenie výpočtov kvôli tomu, že štandard Webassembleru využíva iba jedno jadro. Druhou, nie menej zaujímavou, možnosťou by bola úprava vykresľovania, kde by sa dalo ušetriť na tom, že by vykreslilo nový obraz, iba keď by to bolo potrebné. Tým by sa dala ušetriť zátaž na grafickú kartu, ale aj na procesor počítača.

Literatúra

[1] Feynman, Leighton, Sands: Feynmanove prednášky z fyziky 1., Alfa, 1988

[2] David Morin: Introduction to Classical Mechanics With Problems andSolutions, Cambridge University Press, 2007

[3] Newton's law of gravity, [online] Dostupné na internete: <https://www.britannica.com/science/gravity-physics/Interaction-between-celestial-bodies>, 7.12.2020

[4] Awesome Go, [online] Dostupné na internete: <https://awesome-go.com/>, 13.10.2020

[5] fyne.io/fyne, [online] Dostupné na internete: <https://github.com/fyne-io/fyne>, 10.1.2021

[6] Numerical Solution of Nonlinear Vibration by Euler-Cromer Method,June 2019IOP Conference Series Materials, Science and Engineering 546:032029, DOI: 10.1088/1757-899X/546/3/032029

[7] The Go Programming Language, [online] Dostupné na internete: <https://golang.org/>, 13.1.2021

[8] Release History - The Go Programming Language, [online] Dostupné na internete: <https://golang.org/doc/devel/release.html>, 13.1.2021

[9] Rob Pike Biography | Pantheon, [online] Dostupné na internete: <https://pantheon.world/profile/person/Rob_Pike/>, 13.1.2021

[10] Robert Griesemer: Software engineer (born: 1964) | Biography, Facts, Career, Wiki, Life, [online] Dostupné na internete: <https://peoplepill.com/people/robert-griesemer/>, 13.1.2021

[11] RKenneth Lane Thompson | American computer scientist | Britannica, [online] Dostupné na internete: <https://www.britannica.com/biography/Kenneth-Lane-Thompson>, 13.1.20211

[12] Why Golang may be a good choice for your project - CodiLime, [online] Dostupné na internete: <https://codilime.com/why-golang-may-be-a-good-choice-for-your-project/>, 13.1.2021

[13] Setting up and using gccgo - The Go Programming Language, [online] Dostupné na internete: <https://golang.org/doc/install/gccgo>, 13.1.2021

[14] Home :: TinyGo - Go on Microcontrollers and WASM, [online] Dostupné na internete: <https://tinygo.org/, 13.1.2021

[15] golang/mobile: [mirror] Go on Mobile, [online] Dostupné na internete: <https://github.com/golang/mobile>, 13.1.2021

[16] Shankar R. - Fundamentals of Physics Mechanics, Relativity, and Thermodynamics (2014)

[17] Podolský J. Teoretická mechanika v klasické formulaci Studijní text k přednášce NOFY003 „Teoretická mechanika"Ústav teoretické fyziky Matematicko-fyzikální fakulta Univerzita Karlova Praha březen 2020

[18] gioui/gio, [online] Dostupné na internete: <https://gioui.org/>, 10.1.2021

[19] Čistý kód Návrhové vzory, refaktorování, testování a další techniky agilního programování Robert C. Martin

[20] golang/tools: [mirror] Go Tools, [online] Dostupné na internete: <https://github.com/golang/tools>, 13.1.2021

Príloha A: Obsah elektronickej prílohy

V elektronickej prílohe priloženej k práci sa nachádzajú zdrojové kódy všetkých troch programov. Zdrojové kódy sú zverejnené aj na stránke `https://github.com/MatejMagat305/kosmos`.

Obsah prílohy je rozdelený na 3 adresáre, ktoré majú v sebe zdrojové kódy jednotlivých programov. Adresáre sú pomenované podľa fyzikálnych problémov, ktoré programy riešia. Okrem zdrojových kódov sú v priečinkoch už skompilované binárne súbory pre operačný systém Windows. Pre problém dvoch telies bez retardácie a problém dvoch telies s retardáciou sú skompilované binárne súbory pre operačný systém Linux.

Príloha B: Upresnenie zadania

Táto príloha pozostáva z priloženého (pôvodne samostatného) dokumentu, ktorý vypracoval RNDr. Eduard Masár, PhD. .

Retardácia v newtonovskom probléme dvoch telies (upresnenie zadania bak.práce)

Eduard Masár

Pohybové rovnice s okamžitým pôsobením na diaľku

V newtonovskej gravitácii pôsobia telesá na seba gravitačnou silou, ktorá sa šíri nekonečnou rýchlosťou. Inak povedané, veľkosť gravitačnej sily medzi dvoma pohybujúcimi sa telesami závisí v každom okamihu len od ich aktuálnej vzdialenosti, bez ohľadu na ich relatívnu rýchlosť.

Newtonov gravitačný zákon hovorí, že dva hmotné body s polohovými vektormi $\boldsymbol{r}_1$ a $\boldsymbol{r}_2$ a s hmotnosťami m_1 a m_2 sa navzájom priťahujú rovnako veľkou a opačne orientovanou gravitačnou silou s veľkosťou $\frac{Gm_1m_2}{|\boldsymbol{r}_1-\boldsymbol{r}_2|^2}$. Po dosadení Newtonovho gravitačného zákona do Newtonových pohybových rovníc pre telesá 1 a 2 dostávame sústavu dvoch diferenciálnych rovníc, opisujúcich časovú závislosť polohových vektorov $\boldsymbol{r}_1(t)$ a $\boldsymbol{r}_2(t)$

$$m_1\ddot{\boldsymbol{r}}_1 = -\frac{Gm_1m_2}{|\boldsymbol{r}_1-\boldsymbol{r}_2|^3}(\boldsymbol{r}_1-\boldsymbol{r}_2) \tag{1}$$

$$m_2\ddot{\boldsymbol{r}}_2 = -\frac{Gm_2m_1}{|\boldsymbol{r}_2-\boldsymbol{r}_1|^3}(\boldsymbol{r}_2-\boldsymbol{r}_1), \tag{2}$$

kde bodka znamená deriváciu podľa času. V týchto rovniciach sme už museli gravitačnú silu zapísať vo vektorovom tvare, aby sme mohli vyjadriť nielen jej veľkosť, ale aj smer. Rovnica (1) opisuje priťahovanie telesa 1 k telesu 2, rovnica (2) opisuje priťahovanie telesa 2 k telesu 1.

Retardovaný čas

Zdržanie gravitačnej interakcie v dôsledku konečnej rýchlosti jej šírenia sa nazýva *retardácia*. Ak sa gravitačná sila šíri konečnou rýchlosťou, rozdiel oproti nekonečnej rýchlosti jej šírenia sa prejaví, keď sa objekty voči sebe pohybujú. To je aj prípad problému dvoch telies. Na príklade rovnice (1) si ukážeme zmeny, ktoré nastanú v prítomnosti retardácie.

Rovnica (1) opisuje gravitačné pôsobenie telesa 2 na teleso 1. Gravitačná interakcia sa šíri rýchlosťou c od telesa 2 telesu 1. Vektor $\boldsymbol{r}_1$ je poloha telesa 1 v aktuálnom čase t. Vektor $\boldsymbol{r}_2$ je poloha telesa 2 v minulom čase $t' < t$, ktorý sa nazýva *retardovaný čas*. Podmienka na retardovaný čas t' je, že gravitačná interakcia prekoná za čas $t-t'$ vzdialenosť medzi minulou polohou telesa 2 v čase t' a aktuálnou polohou telesa 1 v čase t (rovnica (5) nižšie).

Nech bod P na obrázku Fig.1 je poloha telesa 1 v čase t. Konštantný vektor $\boldsymbol{r}$ spája počiatok súradnej sústavy O s bodom P. Polohu druhého telesa v aktuálnom čase t a v

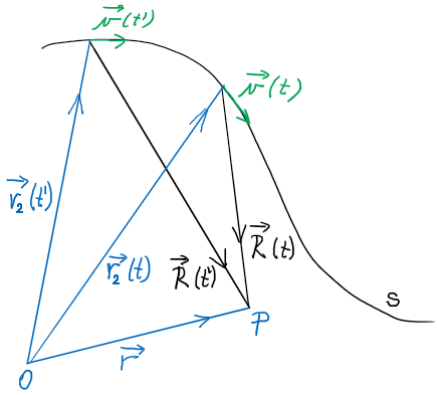


Fig. 1: Relatívne polohy $\boldsymbol{R}(t')$ a $\boldsymbol{R}(t)$ v retardovanom čase t' a v aktuálnom čase t

retardovanom čase t' ukazujú vektory $\boldsymbol{r}_2(t)$ a $\boldsymbol{r}_2(t')$. Pre vektory $\boldsymbol{R}(t)$ a $\boldsymbol{R}(t')$, spájajúce teleso 2 v čase t a v čase t' s bodom P platí

$$\boldsymbol{R}(t) = \boldsymbol{r} - \boldsymbol{r}_2(t) \tag{3}$$
$$\boldsymbol{R}(t') = \boldsymbol{r} - \boldsymbol{r}_2(t'), \tag{4}$$

a teda vzdialenosti medzi telesom 2 a bodom P v čase t a v čase t' sú $R(t) = |\boldsymbol{r} - \boldsymbol{r}_2(t)|$ a $R(t') = \left|\boldsymbol{r} - \boldsymbol{r}_2(t')\right|$.

Teleso 2 sa pohybuje po dráhe s, pričom poloha a rýchlosť telesa 2 v čase t je daná vektormi $\boldsymbol{r}_2(t)$ a $\boldsymbol{v}(t) = \frac{d\boldsymbol{r}_2(t)}{dt} = -\frac{d\boldsymbol{R}(t)}{dt}$. Zaujíma nás sila, ktorou v čase t teleso 2 pôsobí na teleso 1. Je zrejmé, že táto sila nemôže závisieť od relatívnej polohy $\boldsymbol{R}(t)$, prípadne rýchlosti $\boldsymbol{v}(t)$ telesa 2 voči telesu 1 v čase t, pretože gravitačná interakcia potrebuje istý čas, aby prekonala vzdialenosť od bodu $\boldsymbol{r}_2(t)$ do bodu P. Sila, ktorou teleso 2 pôsobí na teleso 1, je daná polohou $\boldsymbol{R}(t')$ a prípadne aj rýchlosťou $\boldsymbol{v}(t')$ telesa 2 v retardovanom čase t' pre ktorý platí

$$\left(t - t'\right) c = R(t'). \tag{5}$$

Numerické riešenie rovnice (5)

Retardovaný čas t' a vzdialenosť $R(t') = \left|\boldsymbol{R}(t')\right|$ vo všeobecnosti (t.j. pre ľubovoľný pohyb telesa 2) nevieme vypočítať ako presné riešenie rovnice (5). Ak však sústavu pohybových rovníc riešime numericky a pamätáme si vypočítané polohy telesa 2 v minulom čase, vieme si poradiť. Numericky vieme $\boldsymbol{R}(t')$ a $R(t')$ nájsť s presnosťou zodpovedajúcou presnosti vypočítaných polôh tak, že postupujeme od času t späť do minulosti a hľadáme polohu $\boldsymbol{r}_2(t')$ telesa 2 v takom čase $t' < t$, v ktorom sa vzdialenosť $R(t') = \left|\boldsymbol{r} - \boldsymbol{r}_2(t')\right|$ čo najpresnejšie rovná číslu $\left(t - t'\right) c$.

Nevýhodou numerického riešenia rovnice (5) takouto metódou je spomalenie výpočtov, lebo ho treba robiť v každom výpočtovom kroku (v ktorom sa posunieme z času t do času $t + \triangle t$). Ideálne by bolo poznať analytické riešenie pre $\boldsymbol{R}(t')$, resp. $R(t')$ vyjadrené prostredníctvom aktuálnych hodnôt $\boldsymbol{R}(t)$, resp. $R(t)$. Toto vieme vo všeobecnosti urobiť len s približnou presnosťou[1].

Polohy a rýchlosti v retardovanom čase s presnosťou do $\frac{v}{c}$

Ak chceme približne vyjadriť polohu a rýchlosť telesa 2 v retardovanom čase t' prostredníctvom jeho polohy, rýchlosti a zrýchlenia v čase t, môžeme to urobiť rozkladom funkcií $\boldsymbol{R}(t') = \boldsymbol{R}\left(t - \frac{R(t')}{c}\right)$, $R(t') = R\left(t - \frac{R(t')}{c}\right)$, $\boldsymbol{v}(t') = \boldsymbol{v}\left(t - \frac{R(t')}{c}\right)$ do radu v čase t. S presnosťou do $\frac{v}{c}$ tak dostaneme

$$\boldsymbol{R}(t') \approx \boldsymbol{R} + R\frac{\boldsymbol{v}}{c} \tag{6}$$
$$R(t') \approx R + \frac{\boldsymbol{R}\cdot\boldsymbol{v}}{c} \tag{7}$$
$$\boldsymbol{v}(t') \approx \boldsymbol{v} - \frac{\boldsymbol{a}}{c}R, \tag{8}$$

kde sme použili označenie

$$\boldsymbol{R} = \boldsymbol{R}(t), \quad R = R(t), \quad \boldsymbol{v} = \boldsymbol{v}(t) = \frac{d\boldsymbol{r}_2(t)}{dt}, \quad \boldsymbol{a} = \boldsymbol{a}(t) = \frac{d\boldsymbol{v}(t)}{dt}. \tag{9}$$

[1] $\boldsymbol{R}(t')$, resp. $R(t')$ je možné vyjadriť *presne* prostredníctvom aktuálnych hodnôt $\boldsymbol{R}(t)$ resp. $R(t)$ len v prípade konštantnej rýchlosti $v = const$, čo v našom prípade úlohy dvoch telies nepripadá do úvahy.

Vzťahy (6)-(8) sú aproximácie smerom do minulosti, keď z veličín v aktuálnom čase t približne vypočítame veličiny v čase $t' < t$. S malým časovým krokom môžu byť tieto priblíženia uspokojivo presné pre naše potreby. Výhodou je, že si nemusíme pamätať minulé polohy (prípadne aj rýchlosti) telesa 2. Ak potrebujeme presnejší výpočet, potom treba veličiny v retardovaných časoch hľadať numerickým riešením rovnice (5) prehľadávaním starých polôh telesa 2 ako bolo opísané vyššie.

Retardácia v newtonovskej gravitácii

Je viacej spôsobov, ako môžeme pridať retardáciu do Newtonovho gravitačného zákona. Obmedzíme sa na dva najjednoduchšie, ktorými zostávame v nerelativistickej oblasti rozsahov rýchlostí telies.

Spôsob 1

Označme $\boldsymbol{F}_{21}(\boldsymbol{r},t)$ silu, ktorou teleso 2 pôsobí na teleso 1 v mieste $\boldsymbol{r}$ v čase t tak ako na Fig.1. Newton samotný by zrejme retardáciu do gravitačnej sily $\boldsymbol{F}_{21}(\boldsymbol{r},t)$ pridal takto

$$\boldsymbol{F}_{21}(\boldsymbol{r},t) = -\frac{Gm_1m_2}{R^3(t')}\boldsymbol{R}(t'), \tag{10}$$

kde $R(t') = \left|\boldsymbol{R}(t')\right|$ a vektor $\boldsymbol{R}(t')$ je definovaný vzťahom (4). V probléme dvoch telies s retardáciou treba nahradiť pravú stranu rovnice (1) silou $\boldsymbol{F}_{21}(\boldsymbol{r},t)$. Veličiny $\boldsymbol{R}(t')$ a $R(t')$ získame buď numerickým riešením rovnice (5), alebo z približných vzťahov (6)-(7) .

Analogicky vypočítame silu $\boldsymbol{F}_{12}(\boldsymbol{r},t)$, ktorou teleso 1 pôsobí na teleso 2 v mieste $\boldsymbol{r}$ v čase t. Pritom vo všetkých vzťahoch aj v texte vyššie zameníme $1 \longleftrightarrow 2$. Silou $\boldsymbol{F}_{12}(\boldsymbol{r},t)$ potom nahradíme pravú stranu rovnice (2).

Spôsob 2

Inou jednoduchou možnosťou je pridanie retardácie do newtonovského gravitačného potenciálu telesa 2

$$\varphi(\boldsymbol{r},t) = -\frac{Gm_2}{R(t')}, \tag{11}$$

odkiaľ pre gravitačnú silu $\boldsymbol{F}_{21}$, ktorou teleso 2 pôsobí na teleso 1 dostávame

$$\boldsymbol{F}_{21}(\boldsymbol{r},t) = -m_1\nabla\varphi(\boldsymbol{r},t) = -\frac{Gm_1m_2}{R^3(t')}\frac{\boldsymbol{R}(t')}{1-\dfrac{\boldsymbol{R}(t')}{R(t')}\cdot\frac{\boldsymbol{v}(t')}{c}}, \tag{12}$$

kde $\boldsymbol{v}(t') = \left.\frac{d\boldsymbol{r}_2(t)}{dt}\right|_{t=t'}$. Ďalší postup je rovnaký ako bolo vysvetlené v spôsobe 1.

Poznámka V rámci newtonovskej gravitácie je rýchlosť telies malá voči rýchlosti svetla $\frac{v}{c} \ll 1$. Ak pre retardované veličiny používame približné vzťahy (6)-(7), tak pre $\frac{\boldsymbol{v}(t')}{c}$ v (12) máme

$$\frac{\boldsymbol{v}(t')}{c} \approx \frac{\boldsymbol{v}}{c} - \frac{\boldsymbol{a}}{c^2}R \approx \frac{\boldsymbol{v}}{c}, \tag{13}$$

lebo člen s $\frac{1}{c^2}$ môžeme zanedbať voči $\frac{\boldsymbol{v}}{c}$. V približnom výpočte teda v (12) môžeme namiesto $\frac{\boldsymbol{v}(t')}{c}$ použiť $\frac{\boldsymbol{v}(t)}{c}$.

Rýchlosť šírenia gravitácie

Pretože retardáciu pridávame do newtonovskej gravitácie, rýchlosť telies v musí byť malá voči rýchlosti svetla c, t.j. musí platiť $\frac{v}{c} \ll 1$. V rovnici pre retardáciu (5) sme ako rýchlosť šírenia gravitácie použili tiež rýchlosť svetla c. Rýchlosť šírenia gravitácie je teda veľká voči rýchlosti telies, čím sa efekt retardácie len málo odlišuje od štandardnej newtonovskej gravitácie, v ktorej je rýchlosť šírenia gravitácie nekonečne veľká.

Aby sme lepšie uvideli vplyv retardácie, môžeme v tomto modeli experimentovať s veľkosťou rýchlosti šírenia gravitácie c_g, ktorá môže byť aj menšia ako rýchlosť svetla. Ak vo vzťahoch pre retardované veličiny namiesto c použijeme c_g, kde napr. $c_g = \frac{c}{2}$ alebo $c_g = \frac{c}{10}$, atď, lepšie uvidíme zmeny, ktoré do problému dvoch telies prináša retardácia.

Astronomické jednotky

Aby sme v našich numerických výpočtoch udržali presnosť v rámci rozsahu mantisy štandardného reálneho dátového typu s dvojitou presnosťou (15 – 16 desiatkových cifier), výpočty by mali pracovať s číslami, ktorých veľkosť zostáva blízko číslu jedna (10^0). Pre problém dvoch telies s parametrami ako sa vyskytujú v našej Slnečnej sústave je preto prirodzené počítať v astronomických jednotkách, v ktorých je jednotkou *dĺžky* astronomická dĺžková jednotka (au), t.j. stredná vzdialenosť Zeme od Slnka, jednotkou *času* je jeden deň (d) a jednotkou *hmotnosti* je hmotnosť Slnka ($M_\odot$). Pre rýchlosť svetla c a gravitačnú konštantu G v týchto jednotkách platí

$$c = 173.1446327 \frac{au}{d} \tag{14}$$

$$G = 2.9591 \times 10^{-4} \frac{au^3}{M_\odot d^2}. \tag{15}$$

Automaticky potom dostávame vzdialenosti v au a časy v d. Pre presnosť výpočtov je výhodné, že gravitačná konštanta v astronomických jednotkách má veľkosť rádu 10^{-4}, ktorá je o 7 rádov bližšia k 10^0 oproti jej hodnote v sústave SI.

Hmotnosti telies zadávame v jednotkách hmotnosti Slnka $M_\odot$. Napríklad hmotnosť sústavy Zem+Mesiac v týchto jednotkách je[2]

$$M_{♁☽} = 3.04043263333 \times 10^{-6} M_\odot. \tag{16}$$

Hmotnosť samotnej Zeme

$$M_♁ = 3.00348959632 \times 10^{-6} M_\odot \tag{17}$$

a samotného Mesiaca

$$M_☽ = 1.23000383 \times 10^{-2} M_♁ = 3.69430371 \times 10^{-8} M_\odot. \tag{18}$$

Stredná rýchlosť obehu Zeme okolo Slnka je

$$v_♁ = 0.01720209895 \frac{au}{d}, \tag{19}$$

a teda obežná doba (za predpokladu presne kruhovej dráhy) je

$$T = \frac{2\pi(1au)}{v_♁} = 365.2568983 d.$$

[2] pozreté 30.4.2021: https://en.wikipedia.org/wiki/Planetary_mass

Príloha C: Používateľská príručka Problém jedného telesa

Program má niekoľko atribútov, ktoré sú meniteľné počas chodu programu (behom módu init, ktorý je opísaný nižšie):

- „AutoScale“, ktoré uvádza, či program má automaticky škálovať.
- „FixedScale“, ktoré uvádza, že program nemá dovoliť škálovať ani počítaču a ani pomocou klávesnice.
- „ForceCircularOrbit“, ktoré uvádza, či program má upraviť rýchlosť tak, aby dráha bola kruhová.
- „StepsForward“ je hodnota krokov, ktoré má počítač vykonať bez animácie (na začiatku a aj na príkaz).
- „FixedCentre“ je nastavenie na to, aby stred “kamery” bol stále namierený do bodu 0,0.
- „WriteExcelFiles“ - nastavenie, či program má zapisovať údaje do excelovských súborov.
- „HowManyPositionsBack“ - nastavenie, koľko pozícií dozadu si program má pamätať.
- „RememberWholeOrbit“ - nastavenie, či si program má pamätať všetky pozície od začiatku.
- „SpeedMovePixel“ - slúži na určenie rýchlosti pohybu po obrazovke.

Program má 4 módy (fázy):

- „init“ - pomocou šípok sa dá nastaviť krok a epsilon simulácie a iné premenné, užívateľ si tu môže vybrať atribút, ktorý chce zmeniť, a následne ho môže zmeniť. Zobrazuje sa iba vybraná premenná, v tejto fáze sa objavujú rozličné varovania (warningy), ktoré upozorňujú na nenačítané dáta, alebo na únikovú rýchlosť - warning

o rýchlosti sa objaví iba v prípade, že nie je nastavené „ForceCircularOrbit“ na „true“.

- „animation–stop“ - epsilon sa meniť nedá, tak ako ani ďalšie atribúty. Animácia je stopnutá a ani výpočet sa nevykonáva automaticky, môže sa však nastaviť krok a krokovať .

- „animation–play“ - animácia beží sama.

- „compute“ - zobrazuje v polsekundových intervaloch, koľko už počítač vypočítal, tento mód sa vykresľuje iba pre veľké „StepsForward“, pre malé výpočet trvá tak krátko, že sa vykreslí na okamih čierna obrazovka, ale tá sa po polsekunde znova prekreslí novým stavom animácie.

Program sa štartuje dvojklikom - nie je nutné ho spúšťať cez príkazový riadok. Program automaticky načíta config.txt.

Program z config.txt načíta okrem meniteľných atribútov aj také, ktoré sa v móde init nedajú meniť: „PositionX“, „PositionY“, „VelocityX“,„VelocityY“, ktoré určujú začiatočnú polohu, „Scale“, ktorý označuje začiatočné škálovanie a „SpeedScaledPercent“, ktoré určuje rýchlosť škálovania v percentách. Keď sa načíta príliš veľké percento (nad 100), tak program dosadí default(50); to isté pre nižšie ako 0.

Počiatočný stav sa kontroluje v config.txt. Program akceptuje riadky so správne zadaným menom, znamienkom rovnosti a hodnotami správneho typu.

Ak program nerozpozná správne meno v riadku dosadí, oznámi to používateľovi po otvorení aplikácie hláškou „warning“, ktorý riadok a čo bolo v tom riadku. Program na začiatku vytvorí defaultné dáta, ktoré potom prepisuje načítaním z configu.

Ak program načíta samé nuly, dosadí default.

POZOR: epsilon musí byť vo formáte .0xyz.... Ak je epsilon väčšie ako 1.00 (prípadne číslo menšie ako $10^{-}13$), tak sa epsilon nastaví na default.

Hneď po spustení sa program spustí v „init“-móde.

Po potvrdení sa spustí mód „compute“, ktorý vypočíta niekoľko krokov bez animácie, pričom počet krokov závisí od premennej „StepsForward“. Po skončení „compute“ módu sa objaví animácia v móde „animation–stop“.

V „animation–stop“ a „animation–play“ sa pod animáciou zobrazuje pozícia x, y. Nad animáciou sa zobrazujú informácie: ktorá metóda je to, zvolené epsilon, čas a krok.

Animáciu (v „animation–stop“ aj „animation–play“) možno dočasne prerušiť úplne uvedením programu do výpočtového režimu („compute“ módu), stlačením „c“ na klávesnici. V módoch „animation–stop“ aj „animation–play“ sa dá meniť škálovanie, ale musia byť vypnuté dve funkcie (nastavené na „false“): „FixedScale“ a „AutoScale“.

V módoch „animation–stop“ aj „animation–play“ sa dá meniť stred „kamery“, ale musí byť nastavený atribút „FixedCentre“ na „false“.

Ak je nastavená premenná „WriteExcelFiles“ na „true“ a zároveň „RememberWholeOrbit“ je nastavená na „true“, tak sa po skončení ešte nejaký čas (závisí od veľkosti nazhromaždených údajov) bude zapisovať do xsxl súborov, ktorých názov pozostáva z názvu metódy a epsilonu, ktoré bolo zvolené v programe. Súbory sa zapíšu do adresára files; keď program zistí, že taký adresár nie je, tak ho vytvorí. Na ľavom boku animácie sa zobrazuje select voľba, ktorá slúži na to, aby sa vedela vybrať plocha, ktorá sa „pozametá“ sprievodičom. Na pravom boku je legenda „pozametaných plôch“ aj s hodnotami, ktorá sa objaví až po zmeraní prvej plochy.
Ovládanie:

- Tlačidlo escape slúži na zatvorenie aplikácie.
- Tlačidlo enter na začiatku slúži na potvrdenie epsilon a začatie simulácie.
- Tlačidlom p sa spúšťajú/zastavujú všetky metódy do automatického módu.
- Medzerníkom sa dá nastaviť, či sa bývalé pozície telesa budú vykresľovať ako body alebo ako úsečky.
- Tlačidlami plus a mínus sa ovláda, aký veľký je krok.
- V móde „init“ sa tlačidlami doprava a doľava dá meniť parameter epsilon a iné atribúty, v móde „animation–stop“ a „animation–play“ sa týmito tlačidlami pohybuje "kamerou"(doprava a doľava).
- V móde „init“ sa tlačidlami hore a dole vyberá, ktorý atribút užívateľ chce zmeniť, v móde „animation–stop“ a “animation–play” sa týmito tlačidlami pohybuje "kamerou"(hore a dole).
- Tlačidlom e sa vymieňajú pohľady na metódy.
- Tlačidlom k sa krokuje animácia v zastavenom móde.
- Tlačidlom s sa vyberá pred stlačením k, ktoré plochy má rátať a ktoré nie.
- Tlačidlom c sa zastaví animácia, program sa prepne do režimu „compute“ a vykoná „StepsForward“ krokov, potom sa tento režim vráti do režimu krokovania.
- Tlačidlami n a m sa ovláda škálovanie. n vzďaľuje a m približuje.

Príloha D: Používateľská príručka Problém dvoch telies bez retardácie

Spúšťa sa dvojklikom, po spustení sa objaví „hlavné menu“ s možnosťami „load“, ktorý načíta starý výpočet (ešte má malú vadu, dráha sa objaví až po kliknutí next, popísané nižšie), alebo „new“, ktorý presmeruje na formulár.

Formulár síce má prednastavené hodnoty, ale dajú sa zmeniť kliknutím na políčko a písaním. Po odkliknutí „done“ sa skontroluje formát všetkých údajov, keď niečo nesedí, tak vyčervení políčko, ktoré má zlý formát, pod formulárom sú tlačidlá a pred nimi je označenie „generate random orbit:“, tlačidlá generujú náhodné dráhy podľa názvu.

Keď majú údaje správny tvar, tak sa spustí hlavná animácia. V hornom stĺpci sú dve textové polia a tlačidlo next, ktoré spúšťa ďalšie kolo výpočtu. Textové polia sú po stranách, ukazujú 1. koľkokrát už výpočet prebiehal a 2. epsilon, s ktorým výpočet prebehol, v uzavretých orbitách to označuje počet obehov. Pod touto úrovňou je naľavo hore ešte označenie – info o dráhe a o stave výpočtu. Po stranách animácie sú tlačidlá „+“ a „-“, ktoré ovládajú zoomovanie. Naspodku animácie je checkbox, ktorý ovláda, či je vidieť analytická dráha. Analytické riešenie sa dá zvoliť iba v prípade eliptickej (kruhovej) dráhy a ťažiskovej sústavy. Dve tlačidlá, ovládajúce „offset“, teda po akých krokoch má brať body, sú v ľavom dolnom rohu. Medzi offsetom a analytickou dráhou je tlačidlo, ktoré vráti na začiatok. Na pravej strane sú tlačidlá „save“ a „choose“, tlačidlom „save“ sa program prepne z počítania do okna, kde je možné uložiť počiatočné podmienky. Uloženie je v rovnakej forme ako "config.txt"(trošku iné poradie param.), takže stačí z uloženého skopírovať do "config.txt". Tlačidlom „choose“ sa užívateľ prepne do výberu orbít, kde si môže vybrať, či chce vidieť celý interval „kôl“ (pri uzavretých obehoch) alebo jednotlivé kolá, poprípade interval. V prípade, že výpočet nie je nastavený na ťažiskovú sústavu a dráha je eliptická, namiesto analytickej dráhy sa objaví checkbox, ktorý ovláda „ťažiskovú dráhu“ bez počítania; vtedy sa objaví aj analytický checkbox.

Príloha E: Používateľská príručka Problém dvoch telies s retardáciou

Spúšťa sa dvojklikom, po spustení sa objaví „hlavné menu“ s možnosťami „load“, ktorý načíta starý výpočet (ešte má malú vadu, dráha sa objaví až po kliknutí next, popísané nižšie) , alebo „new“, ktorý presmeruje na formulár.

Formulár síce má prednastavené hodnoty, ale dajú sa zmeniť kliknutím na políčko a písaním. Po odkliknutí „done“ sa skontroluje formát všetkých údajov; keď niečo nesedí, tak vyčervení políčko, ktoré má zlý formát. Pod formulárom sú tlačidlá a pred nimi je označenie „generate random orbit:“, tlačidlá generujú náhodné dráhy podľa názvu.

Keď majú údaje správny tvar, tak sa spustí hlavná animácia. V hornom stĺpci sú dve textové polia a tlačidlo next, ktoré spúšťa ďalšie kolo výpočtu. Textové polia sú po stranách; ukazujú 1. koľkokrát už výpočet prebiehal a 2. epsilon, s ktorým výpočet prebehol; pri uzavretých orbitách to označuje počet obehov. Pod touto úrovňou je naľavo hore ešte označenie – info o dráhe a o stave výpočtu. Po stranách animácie sú tlačidlá „+“ a „-”, ktoré ovládajú zoomovanie. Na spodku animácie je checkbox, ktorý ovláda, či je vidieť analytická dráha. Analytické riešenie sa dá zvoliť iba v prípade eliptickej (kruhovej) dráhy a ťažiskovej sústavy. Dve tlačidlá ovládajúce „offset“, teda po akých krokoch má brať body, sú v ľavom dolnom rohu, medzi offsetom a analytickou dráhou je tlačidlo, ktoré vráti na začiatok. Na pravej strane sú tlačidlá „save“ a „choose“. Tlačidlom „save“ sa program prepne z počítania do okna, kde je možné uložiť počiatočné podmienky, uloženie je v rovnakej forme ako „config.txt“ (trošku iné poradie param.), takže stačí z uloženého skopírovať do „config.txt“. Tlačidlom „choose“ sa užívateľ prepne do výberu orbít, kde si môže vybrať, či chce vidieť celý interval „kôl“ (pri uzavretých obehoch) alebo jednotlivé kolá, poprípade interval. V prípade, že výpočet nie je nastavený na ťažiskovú sústavu a dráha je eliptická, namiesto analytickej dráhy sa objaví checkbox, ktorý ovláda „ťažiskovú dráhu“ bez počítania. Vtedy sa objaví aj analytický checkbox. Program má 2 metódy, každá metóda má číslo, ktoré treba zadať do konfigu; ak sa zadá niečo iné, program vyberie automaticky prvú:

- 1. Podľa vzorca 10 v prilohe B
- 2. Bez retardácie