КУРСОВА РАБОТА

Тема: Симулатор на орбити

Изготвил:
Ясен Ефремов

СОФИЯ
2024 г.

Глава 1

Увод

1.1 Описание и идея на проекта

Идеята на проекта е да представлява много опростена версия на програми като STK (Systems Tool Kit) и GMAT (General Mission Analysis Tool), които позволяват планиране и симулиране на космически мисии. Тези програми са изключително сложни, затова текущият проект се стреми да реализира само задаването, пресмятането и рисуването на орбитата на сателит, обикалящ около идеално сферично тяло. Крайната програма трябва да представлява графично Desktop приложение с полета за задаване на параметрите на симулацията и прозорец показващ как би изглеждала симулираната орбита.

1.2 Цел и задачи на разработката

Основните функционалности на програмата са разпространението на сателита в зададена орбита и визуализирането на получените данни.

Глава 2

Преглед на предметната област

2.1 Основни дефиниции, концепции и алгоритми, които ще бъдат използвани

Всяка орбита може да бъде описана напълно от 6 стойности. Има два начина на описание:

- Чрез 6-те орбитални елемента, задаващи Кеплерова орбита:
  - Ексцентрицитет e
  - Голяма полуос a
  - Наклон на орбитата i
  - Дължината на възходящия връх Ω
  - Параметър на перихелия ω
  - Истинска аномалия ν
- Чрез вектори на орбиталното състояние, задаващи позицията и скоростта на тялото:
  - Вектор на позицията $\vec{r} = (r_x, r_y, r_z) \in \mathbb{R}^3$
  - Вектор на скоростта $\vec{v} = (v_x, v_y, v_z) \in \mathbb{R}^3$

Самата орбита представлява конично сечение, като обикаляното тяло се намира в единия от неговите фокуси.

За простота приемаме, че обикаляното тяло е идеална сфера с маса и гравитационна константа .

Разполагаме с диференциалните уравнения, описващи начина, по който се променя скоростта и позицията на сателита с времето:

$$\frac{d\vec{v}}{dt} = -\frac{MG}{|\vec{r}|^2}\frac{\vec{r}}{|\vec{r}|}$$
$$\frac{d\vec{r}}{dt} = \vec{v}$$

Да намерим аналитично решение на такива уравнения е прекалено трудно (дори често невъзможно). Можем обаче да намерим тяхно числено решение. Това означава, че разполагаме с начални стойности за търсената функция. След това избираме дискретна времева стъпка и с нейна помощ намираме следващата стойност на функцията. Така получаваме апроксимация на търсената функция. Този метод се нарича метод на Ойлер. При него обаче се натрупва грешка. Тя може да бъде минимизирана чрез методите на Верле или на Рунге-Кута.

2.2 Потребителски изисквания и качествени изисквания

Програмата трябва да предоставя графичен потребителски интерфейс (GUI), чрез който да могат да се задават начални стойности на симулацията.

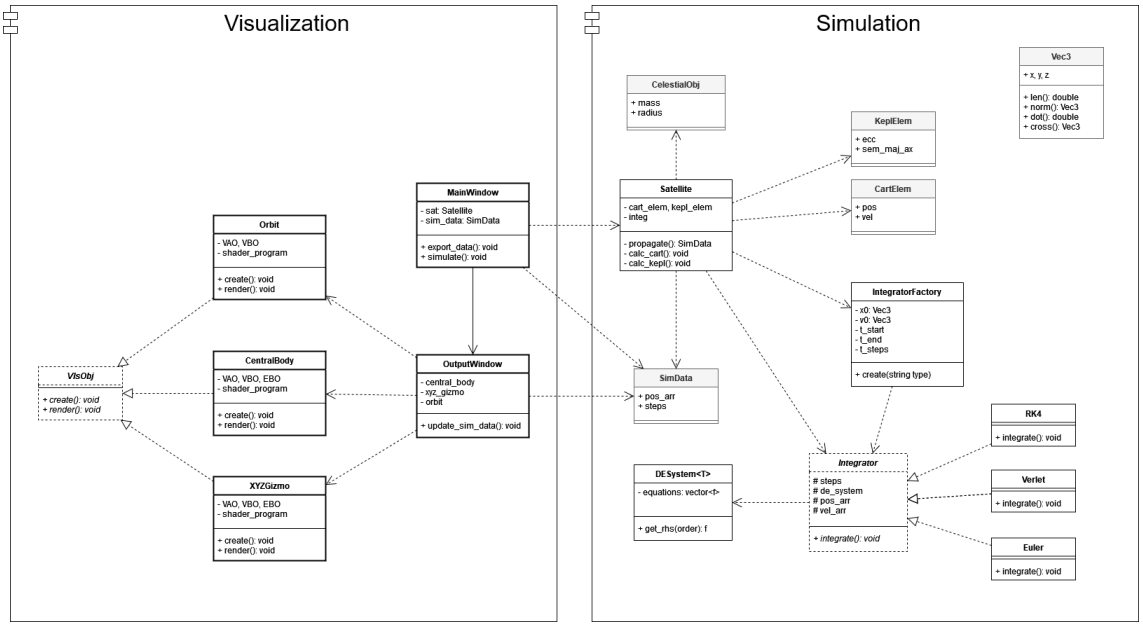
Глава 3

Проектиране

3.1 Обща архитектура – ООП дизайн

При разработката на проекта бяха използвани съвременни ООП практики като полиморфични йерархии от класове и Factory шаблонът.

3.2 Диаграми



Фигура 3.1: UML диаграма на структурата на проекта

Глава 4

Реализация, тестване

4.1 Реализация на класове

Ключови за проекта са класовете `Satellite` и `Integrator`. Класът `Integrator` е абстрактен и бива използван за реализацията на методите на Ойлер, Верле и Рунге-Кута. Класът `Satellite` съдържа описанието на орбитата си и може да превръща Кеплеровите елементи във вектори на състоянието и обратно.

4.2 Управление на паметта и алгоритми. Оптимизации.

Местата, на които бе необходимо ръчно управление на паметта, са най-вече масивите от генерирани данни от симулацията. Те се управляват от класът `Integrator`, като се освобождават в неговия деструктор според принципа RAII.

При симулацията най-важни са алгоритмите за пресмятане на позицията и скоростта на сателита, както и тези за превръщане на Кеплеровите елементи в Декартови и обратното [1] [2] [3]. Те са оптимизирани да работят със стойности без размерност, което увеличава тяхната точност. Размерността се пренася изцяло от следните три константи:

$$R_0 = R_E$$
$$V_0 = \sqrt{\frac{M_E G}{R_E}}$$
$$T_0 = \frac{R_0}{V_0}$$

След нормиране на векторите на позицията и скоростта получаваме следните диференциални уравнения

$$\vec{r} = R_0\vec{\rho} \Rightarrow \frac{d\vec{r}}{dt} = \frac{\cancel{R_0}}{\cancel{T_0}}\frac{d\vec{\rho}}{d\tau} = \cancel{V_0}\vec{u} \Rightarrow \frac{d\vec{\rho}}{d\tau} = \vec{u}$$
$$\vec{v} = V_0\vec{u} \Rightarrow \frac{d\vec{v}}{dt} = \frac{\cancel{V_0}}{\cancel{T_0}}\frac{d\vec{u}}{d\tau} = -\frac{M_E\cancel{G}}{\cancel{R_0^3}}\frac{\vec{\rho}}{|\rho|^3} \Rightarrow \frac{d\vec{u}}{d\tau} = -\frac{\vec{\rho}}{|\rho|^3}$$
$$t = T_0\tau$$

4.3 Планиране, описание и създаване на тестови сценарии

За тестване на кода бе използвана библиотеката GoogleTest. Класът `Integrator` и неговите наследници бяха тествани чрез т.н. Fixture, който позволява преизползване на инициализирани променливи в множество тестове

Глава 5

Заключение

5.1 Обобщение на изпълнението на началните цели

Основните цели на проекта, а именно

- Задаване и симулиране на орбитата на сателит
- Записване на симулираните параметри в текстов файл
- Рисуване на обикаляното тяло
- Рисуване на орбитата на сателита

бяха изпълнени, с което се предоставя работеща начална версия на софтуера.

5.2 Насоки за бъдещо развитие и усъвършенстване

Към проекта могат да бъдат добавени функционалности, които да позволяват симулиране на повече от един сателит, планиране и изпълнение на мисии, съставени от множество маневри, запазване и зареждане на симулацията във и от файл и други.

Съдържание

Библиография

[1] René Schwarz. *Cartesian State Vectors -> Keplerian Orbit Elements.* URL: https://downloads.rene-schwarz.com/download/M002-Cartesian_State_Vectors_to_Keplerian_Orbit_Elements.pdf.

[2] René Schwarz. *Keplerian Orbit Elements -> Cartesian State Vectors.* URL: https://downloads.rene-schwarz.com/download/M001-Keplerian_Orbit_Elements_to_Cartesian_State_Vectors.pdf.

[3] Bryan Weber. *Classical Orbital Elements and the State Vector.* URL: https://orbital-mechanics.space/classical-orbital-elements/orbital-elements-and-the-state-vector.html.