Specyfikacja implementacyjna

Filip Choromański
Mateusz Karwowski

1 Informacje ogólne

Program jest aplikacją okienkową, więc aby go uruchomić wystarczy dwukrotnie nacisnąć na ikonę z nazwą programu. Okno programu pojawi się na środku ekranu.

2 Opis pakietów

2.1 Core

Pakiet, którego zawartością będą interfejsy klas. Jego zastosowanie ułatwi dostosowanie poszczególnych funkcjonalności programu do własnych potrzeb. Będzie zawierał miedzy innymi interfejs do wczytywania danych dzięki czemu będziemy mogli dostosować program do pobierania danych z różnych źródeł oraz interfejs służący do rozwiązywania równań różniczkowych, ułatwi to wykorzystywanie różnych metod matematycznych.

2.2 Logic

W tym pakiecie będzie znajdował się cały tak zwany "backend" naszego projektu. Będą w nim zawarte przede wszystkim klasy odpowiedzialne za symulację, wczytywanie danych, czy też ich przechowywanie.

2.3 GUI

Pakiet będzie przechowywać klasy implementujące oprawę graficzną programu,a w szczególności interfejs użytkownika.

3 Opis klas

3.1 BodiesReader

Interfejs należący do pakietu *Core*, zawiera metodę służącą do wczytywania danych o ciałach, która zwraca tablicę obiektów *Body*.

3.2 DiffSolver

Interfejs należący do pakietu *Core*, zawiera metodę służącą do rozwiązywania równania różniczkowego.

3.3 FileBodiesReader

Klasa implementująca interfejs `BodiesReader`. Służy do wczytywania danych o ciałach z pliku tekstowego.

3.4 EulerMethod

Klasa implementująca interfejs `DiffSolver`. Rozwiązuje równanie różniczkowe metodą Eulera.

3.5 Vectors

Klasa będzie udostępniała kilka metod odpowiadających za dokonywanie działań na wektorach potrzebnych w algorytmie odnajdywania kolejnych położeń ciał.

3.6 Body

Klasa będzie zawierała informacje o ciele, to jest jego masę, oraz wektory położenia i prędkości przechowywane w tablicach typu `double` .

3.7 Simulator

Klasa bezpośrednio odpowiadająca za odnajdywanie nowego położenia ciała. Główną metodą będzie `findLocation` . Działać ona będzie zgodnie z opisanym poniżej algorytmem.

Algorytm odnajdywania kolejnych położeń:

1. Obliczenie sił działających na ciała

$$\mathbf{F}_i = -\sum_{j=1, i\neq j}^{N} \frac{G\mathbf{r}_{i,j} m_i m_j}{(r_{i,j})^3},$$

obliczenie wektora

$$\mathbf{r}_{i,j} = \mathbf{x}_i - \mathbf{x}_j,$$

oraz jego długości

$$\mathbf{r}_{i,j} = ||r_{i,j}||.$$

2. Obliczenie przyśpieszenia każdego z ciał

$$\mathbf{a}_i = \frac{\mathbf{F}_i}{m_i}.$$

3. Obliczenie prędkości każdego z ciał, jako pochodnej przyśpieszenia po czasie, wykorzystując metodę Eulera:

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \mathbf{a}_i \Delta t$$

4. Obliczenie położenia, analogicznie jak w poprzednim punkcie:

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \Delta t$$

3.8 InputDataPanel

Klasa zawierająca komponenty interfejsu graficznego odpowiadające za wczytywanie danych.

3.9 CoordinateSystemPanel

Klasa zawierająca panel z układem współrzędnych, w którym będą rysowane kolejne położenia poszczególnych ciał.

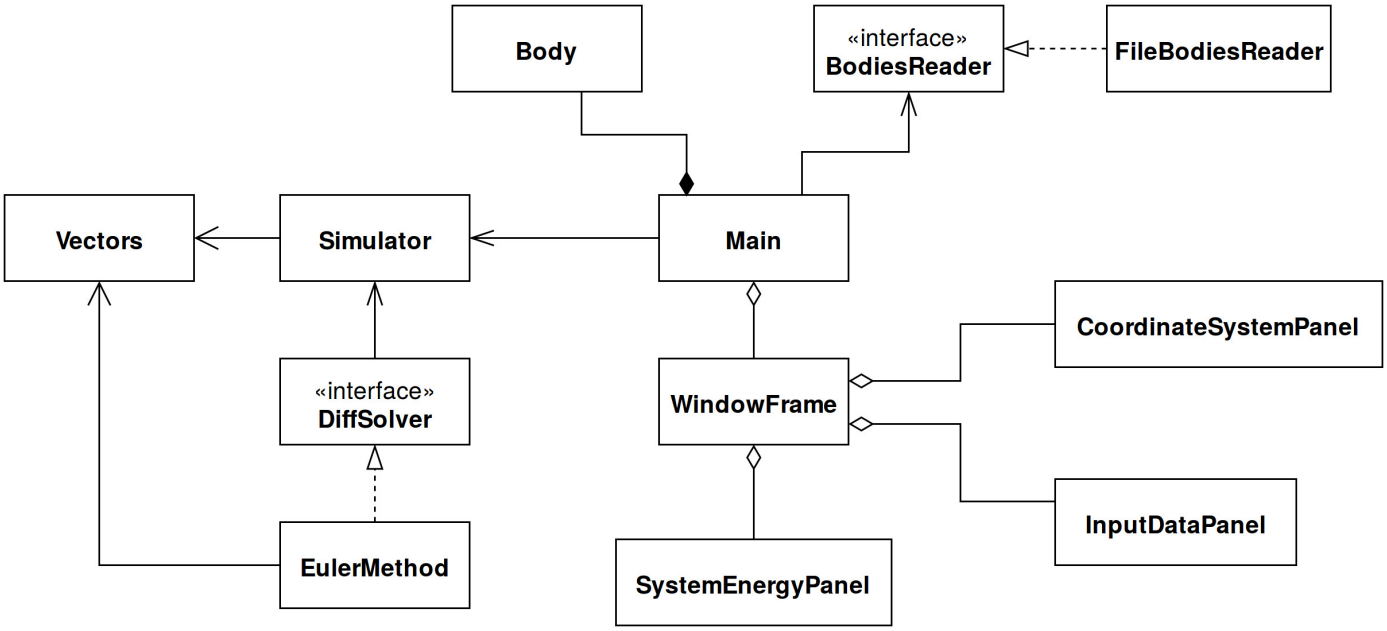
3.10 SystemEnergyPanel

Część interfejsu gdzie będą wyświetlane energia całkowita, kinetyczna i potencjalna układu.

3.11 WindowFrame

Klasa grupujące wszystkie komponenty GUI w jednym oknie.

4 Diagram Klas



Rysunek 1: Diagram Klas UML

5 Testy

5.1 Testy klas

5.1.1 Vectors

Celem testów będzie sprawdzenie implementacji działań algebraicznych na wektorach. Będą one polegać na ocenie prawidłowości wyników tych działań dla sporządzonych przykładów.

5.1.2 Simulator

Test tej klasy będzie polegał na wydrukowaniu pliku z kolejnymi położeniami dla symulacji z konkretnymi danymi i porównanie ich z wynikami uzyskanymi przy użyciu programu `nbodysimulator` napisanym w języku C.

5.2 Testy całościowe

Testy całościowe zweryfikują komunikację pomiędzy klasami. Poprawność działania symulacji sprawdzimy monitorując energię całkowitą układu. W tym celu zaimplementujemy dodatkową klasę testową liczącą tą energię.