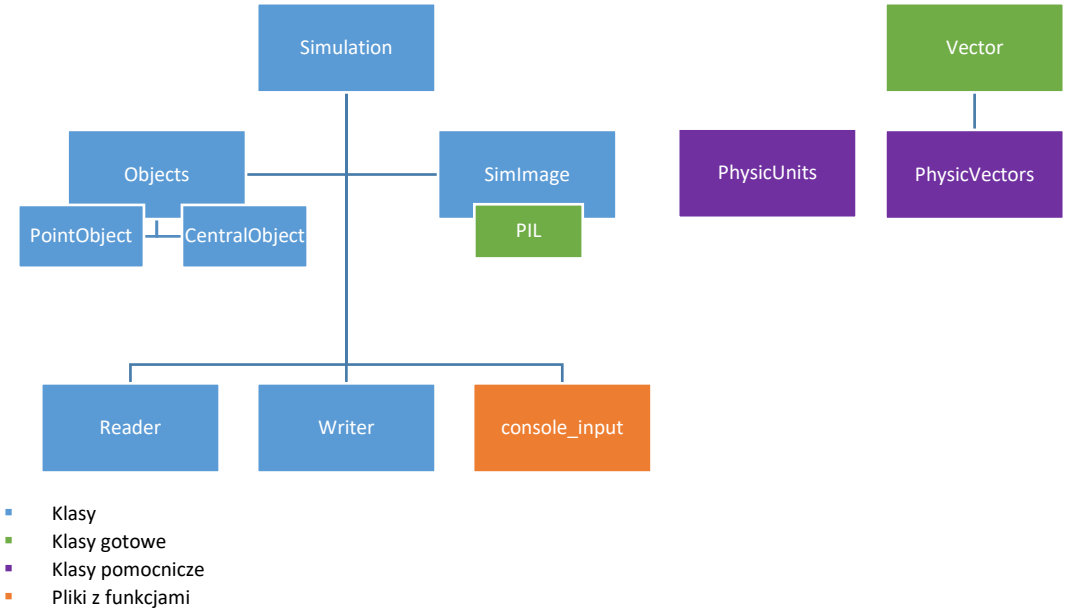
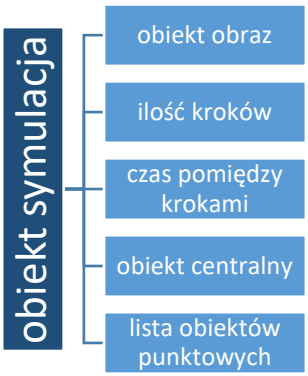
Symulator ruchu obiektów w polu grawitacyjnym

Opis działania programu:

Program przedstawia graficznie ruch obiektów w polu grawitacyjnym innego masywnego obiektu jak np. planeta. Wynik graficzny jest zapisywany w postaci zdjęcia png, gdzie obiekty są reprezentowane za pomocą pikseli. Program również sprawdza czy obiekty zderzają się ze sobą i przestaje je dalej symulować jeżeli do takiego zderzenia dojdzie.

Schemat klas:

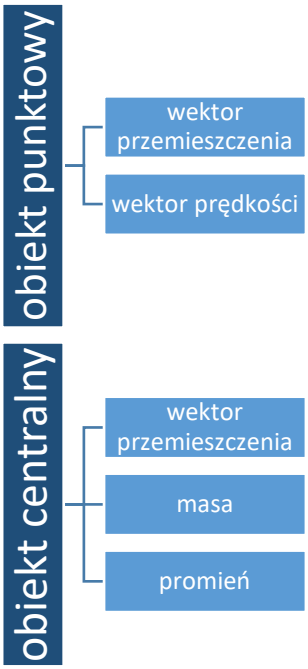




Cała symulacja jest reprezentowana przez klasę simulation. Przetrzymuje ona wszystkie obiekty, właściwości obrazu, ilość kroków symulacji i czas pomiędzy krokami.

Klasa simulation zawiera również wszystkie niezbędne metody które są wywoływane w „main'nie" (np.: aktualizowanie pozycji obiektów lub detekcja zderzeń obiektów). Pozwala również na zapis symulacji i wczytanie jej (o tym później).

Klasa obraz zawiera obiekt obraz utworzony dzięki bibliotece PIL, rozmiar obrazu, a także skalę tego obrazu (ilość metrów na jeden piksel).



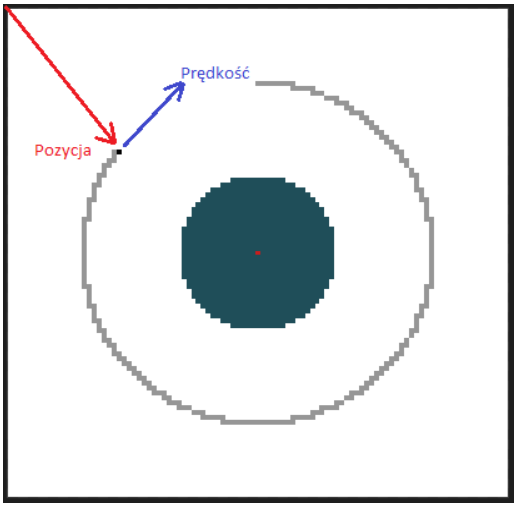
Obiekty w symulacji są podzielone na obiekty punktowe o zerowej masie i promieniu, oraz na obiekt centralny o zerowym wektorze prędkości.

Klasy obiektów zawierają metody które umożliwiają symulowanie ich ruchu w polu grawitacyjnym:

- update_velocity
- update_position

Pierwsza z nich liczy wektor prędkości uzyskany przez przyspieszenie grawitacyjne w danym czasie, a druga metoda liczy wektor przemieszczenia.

Jak działają wektory:



Wektor pozycji ma swój początek w lewym górnym rogu obrazu który jest punktem (0, 0). Dodatkowo wektor ten jest dodatni kiedy pada w kierunku obrazu. Jest to zupełnie odwrotnie niż w układzie kartezjańskim ale ma to swoje zalety. Otóż piksele w obrazie również liczy się od tego lewego górnego rogu, więc dzięki temu łatwiej było napisać funkcje rysujące piksele.

Wektor prędkości wygląda standardowo – wychodzi z obiektu i jest dodatni w I i IV ćwiartce układu.

Jednostki fizyczne:

Niektóre zmienne takie jak „steps” lub „scale” muszą być dodatnie. Ich ujemna wartość sprawi że cała symulacja nie będzie miała sensu. Tą właściwość kontrolują klasy w „physic_units”, które podnoszą wyjątki kiedy wartość jest ujemna lub nie jest typu float zamiast int.

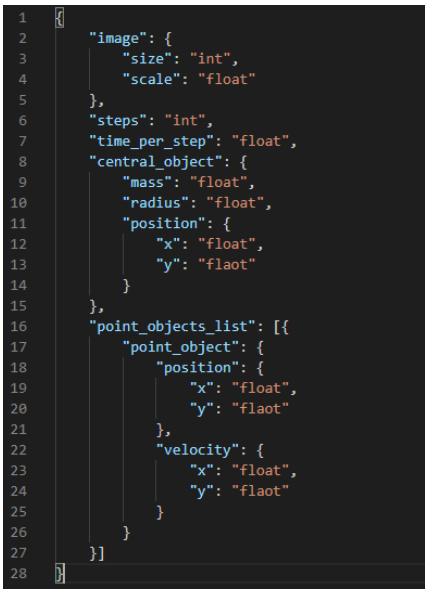
Funkcje „console_input”:

Te funkcje są odpowiedzialne za wyświetlanie w konsoli informacji jaką wartość należy podać, następnie wczytują tą wartość oraz sprawdzają czy jest odpowiedniego typu lub czy wartość ma odpowiedni znak. Jeżeli użytkownik poda niepoprawną wartość to pytanie jest ponawiane.

Ograniczenia symulacji:

- Obiekt punktowy nie może poruszać się z prędkością większą niż prędkość światła. Niestety obiekt ten nie ma masy spoczynkowej, więc i masa relatywistyczna będzie zerowa, co doprowadza do sytuacji że prędkość „c” może zostać przekroczona. Symulacja jednak informuje użytkownika jeżeli to nastąpi.
- Promień obiektu centralnego nie może być mniejszy niż promień Schwarzschilda. Dzięki temu niemożliwe będzie aby obiekt orbitował z prędkością większą niż światło.

Forma zapisu pliku:



```
{
    "image": {
        "size": "int",
        "scale": "float"
    },
    "steps": "int",
    "time_per_step": "float",
    "central_object": {
        "mass": "float",
        "radius": "float",
        "position": {
            "x": "float",
            "y": "flaot"
        }
    },
    "point_objects_list": [{
        "point_object": {
            "position": {
                "x": "float",
                "y": "flaot"
            },
            "velocity": {
                "x": "float",
                "y": "flaot"
            }
        }
    }]
}
```

Symulacja jest zapisywana w formie pliku json za pomocą klasy writer. Plik ten zawiera słownik wszystkich obiektów i ich wartości, które są używane przez klasę simulation. Przy kolejnym starcie programu możliwe jest wczytanie zapisanych symulacji za pomocą klasy reader, która jednocześnie sprawdza czy plik ma poprawną formę oraz wartości dla poszczególnych obiektów.