Gravity Simulation

A program célja:

„Égitestek” mozgásának szimulálása Newton $F = G\frac{m_1 * m_2}{r^2}$ gravitációs törvénye alapján, 2 dimenzióban.

Felhasználóval való kommunikáció:

A szimulációs program végső változata a felhasználóval a standard input/output-on kívül grafikus felületen is kommunikál (ez lesz az elsődleges felület).

Bemenetek:

A lehelyezni kívánt égitest adatait a felhasználó az egér és a billentyűzet segítségével állíthatja majd. A lehelyezés előtt a képernyőn megjelenő prototípus mutatja majd, hogy hogyan fog kinézni és milyen tulajdonságokkal rendelkezik majd az objektum.

Az állítható tulajdonságok a következők:

- Égitest helye – az egér bal klikkjével helyezhető le az objektum a játéktérre
- Égitest mérete (sugara) – egér görgőjével
- Égitest súlya – fel-le nyilakkal a billentyűzeten
- Égitest kezdeti sebességének nagysága és iránya – egér segítségével, Angry Birds-szerű célzással.
- Égitest gyorsulásának nagysága és iránya – nem megadható, program számolja
- Elpusztulhat-e az égitest egy ütközésben – ’i’ betűvel váltogatható az állapot

Kimenetek:

A program által generált játéktér, melyen érvényesülnek a kölcsönhatások. Erre futás során bármikor lehelyezhető egy új objektum, így ez az input is egyben.

Fontosabb függvények:

```
void RenderObjects(std::vector<PlanetaryObject> s)
```

A grafikus felületért (megjelenés és interakciók) felelős függvény. Itt van meghívva a legtöbb függvény. A grafikus ablak megnyitásán és bezárásán kívül a bolygók sebességének, gyorsulásának, pozíciónak állításáért felelős függvények is itt hívódnak meg.

Osztályok:

```
class Position {
	double x;
	double y;

};
```

Tagfüggvényei:

- két paraméteres konstruktor

- double getX(), double getY()
- bool operator== (Position p)
- Position operator+ (Position p)
- Position operator+= (Position p)

```
class Vector {
	Position direction;

};
```

Tagfüggvényei:

- default konstruktor, egy paraméteres konstruktor
- setDirection(Position p)
- Position getDirection()
- bool operator== (Vector v)
- Vector operator+ (Vector v)
- Vector operator+= (Vector v)

```
class PlanetaryObject {
protected:
	Position position;
	Vector velocity;
	Vector acceleration;
	double mass;
	double radius;

};
```

Tagfüggvényei:

- default konstruktor, 4 paraméteres konstruktor
- minden adattagjához egy getter és egy setter tagfüggvény
- operator==, operator!=
- double getDistance (PlanetaryObject p) – két objektum középpontja közti távolságot számolja ki.
- Vector ParticularAcceleration(PlanetaryObject p) – kiszámolja a newtoni képlet alapján, hogy az objektumot mekkora erővel húzza egy másik.

A grafikus kezelés bizonytalansága miatt, és mivel nagyrészt SFML beépített osztályok és függvények használata is szükséges a megvalósításhoz, ezért ezekből egyelőre csak egyet írok a specifikációba.

```
class RenderPlanetaryObject : public PlanetaryObject {
	sf::CircleShape render_shape;
};
```

Tagfüggvényei: default, 1 változós (PlanetaryObject-ből) és 4 változós konstruktor

A render_shape a grafikus megjelenítés során magát az objektumot jelzi, az SFML-lel ennek tagfügvényeit használva lehet kirajzolni az objektumot.

UML osztálydiagram: