KSI 2014/2015

Úloha 3-4: Karlík objevuje vesmír (už zase)

Jan Horáček
Gymnázium, Brno, Vídeňská 47; jan.horacek@seznam.cz

2. února 2015

1 Úvod

Ahoj milí orgové. Blíží se půlnoc, deadline, a já brutálně nestíhám, tak buďte prosím alespoň trochu shovívaví — ale pdf jsem napsal! Tak tedy, hurá na to...

2 Princip

Princip mé implementace je velmi jednoduchý: každý krok provádím následovně:

1. **Spočítám výslednici sil od všech planet.**
   To provedu tak, že projdu všechny planety cyklem (`foreach`).
   (a) Pro každou spočtu velikost gravitační síly, kterou na mě tato planeta působí (dle vztahu v zadání). Pro tyto účely jsem implementoval metodu `Vector.dist(self, second)`, která vrací vzdálenost `self` a libovolného vektoru `second`. Tuto hodnotu používám jako r. Tím jsem získal velikost gravitační síly.
   (b) Směr gravitační síly získám z pozice planety vůči raketě. To provedu pomocí vekotorového rozdílu `planet.position - self.rocket.position`. Vyjádřím si úhel, který tento vektor svírá s osou x do proměnné `alpha`.
   (c) Nyní mám k dispozici velikost i směr síly, kterou působí daná planeta na raketu, vyrobím tedy vektor této síly a tento vektor přičtu k výslednici (pomocí vektorového součtu).
   (d) Stejný postup opakuji pro každou planetu, nakonec je ve výslednici opravdu výsledná síla (plyne z principu skládání sil).
2. **Z výslednice vypočtu zrychlení.**
   A to podle vztahu $\vec{a} = \frac{\vec{F}}{m}$. $\vec{F}$ je výslednice sil, kterou jsem spočítal v předchozím kroku a hmotnost rakety m je známá. Zrychlení opět získám jako vektor.
3. **Ze zrychlení vypočtu změnu rychlosti.**
   A to podle vztahu $\Delta\vec{v} = \vec{a}\Delta t$. Δt je konstanta `timeIncrement`.
4. **Změnu rychlosti připočtu k aktuální rychlost.**
   Tím jsem úspěšně implementovat setrvačnost.

5. **Vypočtu novou polohu.**
   A to opět dle vztahu v zadání: $s = s_{old} + \Delta\vec{s}; \Delta\vec{s} = \vec{v}\Delta t$.

Toť vše. Implementace zbývajících metod je očividná.

Snad jen dodám, že pro zrychlení běhu programu si v metodě `isDestinationReached` zapamatuji planetu, do které jsem narazil a tu pak v metodě `planetReached(self)` vrátím v konstantním čase. To si můžu dovolit, protože zadání specifikuje, že před zavoláním metody `planetReached(self)` musí byt nutně volána metoda `isDestinationReached`.

3 Zamyšlení

Přesnost této metody ovlivňuje především velikost konstanty `timeIncrement`, proto je na modelování pohybů, kde působí intenzivní síly pro krátký čas, tato simulace nevhodná. Vysoká velikost časového intervalu nás omezuje v tom směru, že můžeme postihnout jen ty síly, které působí dlouhodoběji. Jak jsem říkal, pro krátké intenzivní síly je vhodné celý průběh pohybu analyzovat spíše diferenciálně.