Symulacja ruchu N-ciał

Szymon Bugaj, Lei Peng

29 grudnia 2015

Streszczenie

Sprawozdanie z pierwszego etapu projektu z przedmiotu RIM. Tematem projektu jest symulacja ruchu N-ciał (N punktów materialnych pod działaniem siły grawitacji).

Spis treści

1 Wstęp teoretyczny

Projekt stanowi symulacja i wizualizacja praw fizyki klasycznej newtonowskiej. Szczególnie 3 praw dynamiki Newtona oraz siły grawitacji pomiędzy zbiorem N punktów materialnych.

Newtonowska siła grawitacji pomiędzy dwoma punktami materialnymi:

$$\vec{F_G} = G\frac{m_1 m_2}{\|r\|^2}\frac{\vec{r}}{\|r\|}$$

W fizyce klasycznej zależnością łączącą masę przyśpieszenie i siłę działającą na ciało jest:

$$\vec{F} = \vec{a}m$$

Związek między położeniem, prędkością a przyśpieszeniem jest następujący:

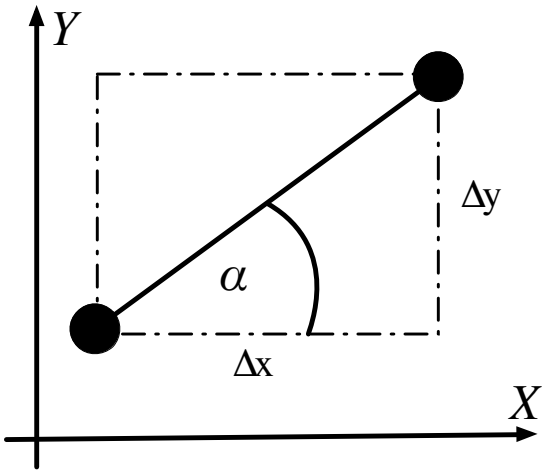
$$v(t) = \frac{dp(t)}{dt}$$
$$a(t) = \frac{dv(t)}{dt}$$

Mając dane położenia wszystkich N cząstek oraz ich prędkości dla danego momentu, korzystając z powyższych równań możemy obliczyć położenie dla dowolnego punktu w dowolnym innym momencie.

2 Rozkładanie siły grawitacji na składowe względem osi współrzędnych

W przypadku 2D należy rozpatrzyć prostokąt o bokach równoległych do osi X i Y, gdzie w przeciwległych wierzchołkach leżą dwa rozpatrywane ciała.

Rozkład wektora siły na składowe



$$a_x(t) = sin\alpha * a$$
$$a_y(t) = cos\alpha * a$$

Dla przypadku 3D należy rozpatrzyć w analogiczny sposób prostopadłościan (ciała powinny znajdować się w wierzchołkach po przeciwnych ścianach, po przeciwnych rogach).

3 Model numeryczny

Interesują nas położenia ciał w kolejnych dyskretnych punktach czasu. Przyjmujemy, że w przeciągu jednostki czasu wielkości prękość, przyśpieszenie są stałe. Aktualizujemy interesujące nas wielkości w kolejności:

- 1) aktualizacja prędkości
- 2) aktualizacja położenia
- 3) aktualizacja przyśpieszenia

Co należy podkreślić, wynika z tego, iż do aktualizacji prędkości brana jest poprzednia wartość przyśpieszenia.

```
/*
    Naming scheme
    @d <- dim index: x == 0, y == 1 or z == 2
    @i <- body index
    @p <- position
    @v <- velocity
    @a <- acceleration

    update variables in order: 1) v, 2) p, 3) a
*/
void NBodiesSystem::step( time_type delta_t ) {
    p_prev = p_curr;
    v_prev = v_curr;

    for (int d = 0; d < D; ++d)
        for (int i = 0; i < N; ++i) {
            v_curr[d][i] = v_prev[d][i]  +
                a[d][i] * delta_t;
            p_curr[d][i] = p_prev[d][i]  +
                (v_prev[d][i] + v_curr[d][i])
                * 0.5 * delta_t;

            //Bouncing off the wall
            if (p_curr[d][i] > 1 || p_curr[d][i] < -1)
                v_curr[d][i] = -v_curr[d][i];
        }

    step_a();
}
```

```
void NBodiesSystem::step_a () {
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < N; ++j) {
            if ( i == j ) continue;

            position_type* r_axis =
                new position_type[D];
            position_type r_squared = 0;
            for (int d = 0; d < D; ++d) {
                r_axis[d] =
                    (p_curr[d][i] - p_curr[d][j]);
                r_squared +=
                    r_axis[d] * r_axis[d];
            }

            //To not divide by very small number or zero
            position_type a_scalar =
                G * m[0][j] /
                pow(r_squared + efactor, 1.5);

            for (int d = 0; d < D; ++d)
                a[d][i] =
                    -1 * a_scalar *
                    (r_axis[d]/r_squared);

            delete [] r_axis;
        }
    }
}
```