2ª Trabalho – PPD – 2014/1

N-Body Simulation

Entregar em 30/04/2015, código até às 09:00 e a apresentação às 16:00

Faça um programa que:

- Leia dois argumentos de entrada T e ΔT;
- Leia a descrição de um universo por meio e um arquivo de entrada em formato de texto;
- Simule o universo começando de um tempo $t = 0.0$ e continue enquanto $t < T$.
- Imprima a saída no mesmo formato de entrada para que seja criada a animação em opengl.

Requisitos:

- Cada grupo deverá apresentar quatro soluções para o problema, sendo uma serial e outras paralelas implementadas com os seguintes padrões: MPI, Pthreads e OpenMP.
- Linguagem: qualquer uma, desde que use paralelismo e que gere a saída.
- A apresentação deverá conter uma análise da proposta de paralelização do problema e do desempenho (speedup e eficiência) de cada solução. Lembre-se de testar com entradas de diferentes tamanhos e/ou diferentes precisões.

Formato de Entrada: A entrada será um texto com a descrição do universo a ser simulado.

1. O primeiro valor será um inteiro **N** que representa o número de partículas do sistema.
2. O segundo valor é um valor real **R** que representa o raio do universo, este valor será usado na animação.
3. As outras N linhas contem cinco valores cada. Os primeiros dois valores são as coordenadas em x e y da posição inicial (px e py); os próximos dois valores são as componentes do vetor inicial (vx e vy]); o quinto valor é a massa do corpo (m)

```
N       5
R       2.50e+11
        1.4960e+11  0.0000e+00  0.0000e+00  2.9800e+04  5.9740e+24
        2.2790e+11  0.0000e+00  0.0000e+00  2.4100e+04  6.4190e+23
        5.7900e+10  0.0000e+00  0.0000e+00  4.7900e+04  3.3020e+23
        0.0000e+00  0.0000e+00  0.0000e+00  0.0000e+00  1.9890e+30
        1.0820e+11  0.0000e+00  0.0000e+00  3.5000e+04  4.8690e+24
          px          py          vx          vy          m
```

Simulação:

A movimentação das partículas é definida pela lei da gravitação universal de Newton, onde pelo cálculo da força de atração entre os corpos podemos definir a direção e velocidade com que irão se movimentar.

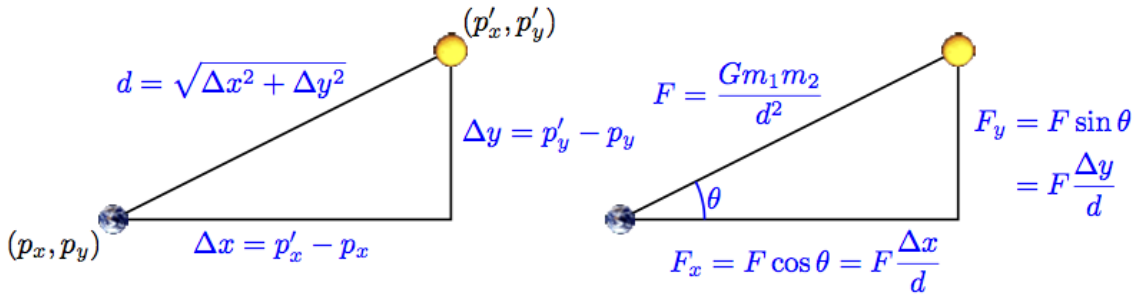
Apesar de o movimento real ser contínuo, na simulação devemos determinar o movimento de forma discreta, considerando pequenos "passos de tempo" representados por ΔT. Assim, quanto menor o ΔT, mais precisa será a simulação, mas consequentemente demandara mais tempo para ser calculada.

Loop de tempo:

Para criar o efeito de movimentação o programa receberá o tamanho do passo, ΔT, e o tempo total da simulação, T, então o programador deve calcular quantas iterações devem ser feitas para completar o tempo total, de forma que o programa se adapte a precisão passada pelo operador.

Movimentação dos corpos:

Força gravitacional: O primeiro passo é calcular a força que cada corpo sofrerá devido à atração aos outros corpos, para isso o programa deve calcular a soma das forças exercidas sobre cada planeta (princípio da superposição). Para cada corpo deverá ter um loop para somar a força de atração com cada um dos outros corpos do sistema. Isso deve ser feito com cada um dos corpos. A força deve ser calculada usando a lei da gravitação universal de Newton que considera a constante gravitacional $G = 6.67\ x\ 10^{-11} Nm^2/kg^2$. A imagem abaixo ilustra a equação usando como exemplo o planeta Terra e o Sol.



Componentes x e y da força: O programa deve calcular a força segundo as componentes x e y, para isso deve-se usar as equações de F_x e $F_y$descritas na figura.

Δx e Δy: Cuidado na hora de calcular a distancia entre os corpos, pois elas podem ser negativas e isso é normal, pois significará uma força para o "lado negativo".

Aceleração: Após obter as componentes da força resultante da atração de cada corpo no sistema é necessário calcular a aceleração do corpo. Para isso usamos as equações $a_x = F_x/m$ e $a_y = F_y/m$, onde m é a massa da partícula.

Velocidade: Após a aceleração ser computada é preciso atualizar a velocidade do corpo (v_x, v_y) para $(v_x + \Delta t * a_x, v_y + \Delta t * a_y)$.

Posição: Após calcular a nova velocidade de todos os corpos devemos atualizar a posição de todos de (p_x, p_y) para $(p_x + \Delta t * v_x, p_y + \Delta t * v_y)$. Lembre-se que para atualizar as posições é preciso que a velocidade de cada corpo já tenha sido calculada para manter a ilusão de que todos os planetas estão se movimentando juntos.

Animação:

Após obter o arquivo de saída vocês podem usar e alterar o código que eu fiz para animar o resultado em opengl. Lembrando que o código possui uma configuração da velocidade da animação.

Teste:

Para testar se os cálculos estão sendo feitos corretamente irei passar um arquivo de entrada "planets.txt" que deve obter as seguintes respostas:

dT = 25000.0 T = 25000.0
1.495963e+11 7.450000e+08 -1.482009e+02 2.980000e+04 5.974000e+24
2.278984e+11 6.025000e+08 -6.385972e+01 2.410000e+04 6.419000e+23
5.787527e+10 1.197500e+09 -9.893309e+02 4.790000e+04 3.302000e+23
3.308672e+01 0.000000e+00 1.323469e-03 0.000000e+00 1.989000e+30
1.081929e+11 8.750000e+08 -2.832940e+02 3.500000e+04 4.869000e+24

dT = 25000.0 T = 50000.0
1.495889e+11 1.489982e+09 -2.964037e+02 2.979926e+04 5.974000e+24
2.278952e+11 1.204996e+09 -1.277197e+02 2.409983e+04 6.419000e+23
5.782579e+10 2.394488e+09 -1.978872e+03 4.787953e+04 3.302000e+23
9.926174e+01 2.819781e-01 2.647001e-03 1.127913e-05 1.989000e+30
1.081788e+11 1.749943e+09 -5.665973e+02 3.499771e+04 4.869000e+24

dT = 25000.0 T = 31557600.0
1.495858e+11 -2.398052e+09 4.776588e+02 2.979538e+04 5.974000e+24
-2.216595e+11 -4.867230e+10 5.117596e+03 -2.365417e+04 6.419000e+23
3.572805e+10 4.501624e+10 -3.764498e+04 3.020125e+04 3.302000e+23
5.957242e+05 6.231631e+06 -5.862224e-02 1.632314e-01 1.989000e+30
-7.436673e+10 -7.879158e+10 2.523958e+04 -2.417890e+04 4.869000e+24

Para ver como a animação deve ficar irei passar o arquivo de saída do último teste.

Pontos extras:

O grupo que criar seu próprio sistema de forma criativa ganhará pontos extras.

Observação:

Me inspirei em um trabalho da universidade de Princeton, se quiserem podem ir no link pra ler a descrição que é colocada lá.
http://www.cs.princeton.edu/courses/archive/spr15/cos126/assignments/nbody.html