PCD Assignment #01 - v 1.0-20220314

Nel repository del corso (package pcd.ass01.seq) è fornito il codice di un programma che simula il movimento di N corpi su un piano bidimensionale, soggetti a due tipi di forze:

- una *forza repulsiva*, per cui ogni corpo b_i esercita su ogni altro corpo b_j una forza in modulo pari a: $F_{ij} = k_{rep} * m_i / d_{ij}^2$, dove m_i è la massa del corpo b_i, k_{rep} è una costante data, d_{ij} è la distanza fra i due corpi. La direzione della forza è data dal versore $(b_j - b_i)$ – ovvero respingente per il corpo b_j.
- una *forza di attrito*, per cui su ogni corpo b_i che si muove ad una velocità v_i è esercitata una forza $FR_i = - k_{fri} * v_i$ che si oppone al moto, quindi in direzione opposta alla sua velocità, dove k_{fri} è una costante data.

Il programma è sequenziale, non strutturato. L'algoritmo che definisce il comportamento del simulatore - contenuta nella classe Simulator - in pseudocodice è il seguente:

```
vt = 0;      /* virtual time */
dt  = 0.01; /* time increment at each iteration */

loop {
    For each body b[i]:
      compute total force exerted by other bodies b[j] and friction
      compute the instant acceleration, given the total force and mass
      update body velocity, given the acceleration and the virtual
                                                        time elapsed dt
    Update bodies positions, given the velocity and virtual time elapsed dt
    Check boundary collisions;
    vt = vt + dt;
    Display current stage;
}
```

La semplice GUI fornita nel programma semplicemente visualizza l'andamento della simulazione (con i tasti UP and DOWN è possibile fare zoom in e out).

Nel programma:

- la classe Body rappresenta una corpo e tiene traccia della sua posizione, velocità, aggiornate ad ogni frame, oltre che della sua massa, nonché fornisce metodi per calcolo forze esercitata da un altro corpo, forza attrito e per aggiornare velocità e posizione;

- la classe Simulation rappresenta il motore della simulazione, che si occupa della generazione dei corpi e dell'esecuzione del simulation loop
- la classe SimulationView si occupa della visualizzazione di ogni frame della simulazione.

Alcuni aspetti rilevanti in merito al comportamento del programma e alla natura del problema:

- Il calcolo delle forze al tempo t avviene considerando coerentemente le posizioni dei corpi al tempo t.
- L'aggiornamento delle posizioni può avvenire solo dopo che tutte le forze sono state calcolate (e le velocità aggiornate).
- Il controllo della collisione con i confini del mondo per un corpo b_i può comportare il cambiamento della velocità e posizione del corpo.
- Nel programma, la visualizzazione dello stato corrente della simulazione o frame (via GUI) avviene in modo sincrono, per cui la successiva iterazione avviene solo dopo aver visualizzato lo stato della precedente.

CONSEGNA

1) Realizzare una versione concorrente della simulazione **senza GUI**, considerando un insieme iniziale N di corpi - e calcolando l'evoluzione temporale per un certo numero di passi Nsteps - con Nsteps fissato come parametro. Posizione e velocità iniziali possono essere definite in modo casuale. L'obiettivo è:
    a) Massimizzare le performance, sfruttando tutte le capacità di calcolo del generico sistema di elaborazione su cui è mandato in esecuzione il programma
    b) Organizzare il programma in modo modulare, estendibile.

    Analizzare le performance del programma considerando valori di N pari a 100, 1000, 5000, con Nsteps pari a 1000, 10000, 50000, calcolando lo speedup, e valutando il suo comportamento usando sia il numero ottimale teorico di thread, sia considerando prove diverse con un numero variabile di threads per verificarne la scalabilità.

    Usare JPF per verificare la correttezza del programma, considerandone la parte più significativa in merito, opportunamente semplificata.

    Definire un modello in TLA+/PLUSCAL, identificando le proprietà di correttezza che si ritengono più opportune e verificandole con TLA+ toolbox.

2) Estendere la simulazione **includendo una GUI** con pulsanti start/stop per lanciare/ fermare la simulazione e visualizzare l'andamento, includendo informazioni circa il tempo virtuale. Usare JPF per verificare la correttezza dell'estensione, identificandone un possibile modello che riproduca gli aspetti essenziali della GUI,

VINCOLI

- Adottare un approccio basato su programmazione multi-threaded, adottando, da un lato, principi e metodi di progettazione utili per favorire modularità, incapsulamento e proprietà relative, dall'altro una soluzione che massimizzi le performance e reattività.
- Come meccanismi di coordinazione prediligere la definizione e uso di monitor rispetto ad altre soluzioni di più basso livello.
- Non è consentito l'uso di librerie e classi esterne relative alla gestione degli aspetti di concorrenza: in particolare, relativamente alla libreria java.util.concurrent, è possibile utilizzare (eventualmente) solo le classi utili all'implementazione di monitor.

LA CONSEGNA

La consegna consiste in una cartella "Assignment-01" compressa (formato zip) do sottoporre sul sito del corso, contenente:

- directory **src** con i sorgenti del programma
- directory **doc** che contenga una breve relazione in PDF (report.pdf). La relazione deve includere:
    - Analisi del problema, focalizzando in particolare gli aspetti relativi alla concorrenza.
    - Descrizione della strategia risolutiva e dell'architettura proposta, sempre focalizzando l'attenzione su aspetti relativi alla concorrenza.
    - Descrizione del comportamento del sistema (ad un certo livello di astrazione) utilizzando Reti di Petri.
    - Prove di performance e considerazioni relative.
    - Identificazione di proprietà di correttezza e verifica (JPF e/o TLA+).