Sistemi Multimediali e Laboratorio

Roberto Ranon

Esercitazione n.2 del 06/03/2018: particelle e animazioni con leggi fisiche.

Nota: *per svolgere questi esercizi, dovete aver scaricato e installato l'ultima versione di Processing (attualmente, la 3.3, ma qualsiasi versione dalla 2.x dovrebbe andare bene). Se state lavorando su una delle macchine del laboratorio, dovreste trovare Processing tra i programmi disponibili in ambiente Windows. Se ci sono problemi, avvisatemi durante la lezione o con un mail a roberto.ranon@uniud.it.*

Le slide che presentano brevemente i concetti e i comandi necessari sono disponibili all'indirizzo ***https://users.dimi.uniud.it/~roberto.ranon/slides/processing-intro.pdf*** *(o sul sito* ***elearning.uniud.it****). Per maggiori informazioni, consultate la guida di riferimento di Processing: basta scrivere nell'editor l'istruzione, selezionarla, cliccare su di essa con il tasto destro, e scegliere "Find in Reference".*

Ci aspettiamo che, al termine del laboratorio, abbiate completato almeno gli esercizi fino al n. 5. Se non è così, cercate di completarli a casa.

1. Partite dal codice di esempio (**Es2-startingCode.zip** sul sito e-learning). Il codice definisce una classe `Particle` (per rappresentare una particella), dotata di una posizione nella finestra, rappresentata tramite un oggetto della classe `PVector` chiamato `position`.

   a. consultate la guida di riferimento di Processing relativa a `PVector`. In breve, si tratta di una classe per rappresentare vettori a 2 e 3 dimensioni, e le operazioni tra essi.

   b. completate il metodo `Display` della classe `Particle` in modo che la particella sia disegnata da un'ellisse del diametro di 20 pixel, di un colore a piacere, e centrata in `position`.

2. L'uso di leggi fisiche permette di realizzare, in modo semplice, animazioni fluide e realistiche. Se aggiungiamo alla nostra particella l'informazione **velocità**, allora possiamo modificare la sua posizione ad ogni frame secondo la relazione

   nuova posizione = vecchia posizione + velocità

   Si tratta della stessa relazione che abbiamo utilizzato per muovere l'automobilina nella scorsa esercitazione. L'uso di vettori per rappresentare posizione e velocità, e della somma tra vettori, ci permette però di esprimere questa relazione in maniera più compatta, gestendo contemporaneamente le componenti x e y della velocità e della posizione[1].

   Modificate ora l'esercizio precedente in modo che:

   a. la particella possieda una posizione e una velocità, entrambe impostabili tramite il costruttore;

---

[1] **(facoltativo)** La velocità che abbiamo impostato verrà aggiunta alla posizione ad ogni ridisegno del frame, cioè circa 60 volte al secondo. Quindi, la velocità effettiva sarà velocità*60/sec. Naturalmente, se per qualche motivo la nostra applicazione disegna 30 frame per secondo, la velocità effettiva sarà dimezzata (lo sketch sarà "più lento"). Questo, in generale, non è desiderabile. Per avere invece una velocità indipendente dal numero di frame per secondo, potete usare la funzione `millis` (https://processing.org/reference/millis_.html), che ritorna il numero di millisecondi trascorsi dall'esecuzione del programma. Pensate a come utilizzare la funzione `millis` per fare in modo di esprimere la velocità in pixel/sec.

b. la posizione venga aggiornata, in base alla velocità, nel metodo `Update()`;

c. la particella possieda inizialmente una velocità casuale compresa tra -2 e 2 in entrambe le componenti.

3. Modificate l'esercizio precedente in modo che la particella "rimbalzi" quanto raggiunge uno dei bordi della finestra. **Suggerimento:** modificare il metodo `Update` in modo che: (i) verifichi il raggiungimento di un bordo, e (ii) inverta la giusta componente della velocità.

4. Per modificare la velocità nel tempo in modo graduale (ad esempio, per far curvare la nostra particella), si usa il concetto di **accelerazione** e la sua relazione con la velocità:

   nuova velocità = *vecchia velocità* + *accelerazione*

   Dal momento che esprimeremo anche l'accelerazione come un vettore, (i) il modulo del vettore indicherà di quanto varia il modulo della velocità, e (ii) la direzione del vettore indicherà la direzione verso cui vogliamo che la nostra particella si diriga.

   Dal momento che l'accelerazione incrementa la velocità ad ogni frame, è buona pratica limitare la velocità risultante per evitare che la nostra particella diventi troppo veloce.

   Modificate l'esercizio precedente in modo che:

   a. la particella possieda, oltre a posizione e velocità, anche un'accelerazione, e una velocità massima (da rappresentare come uno scalare);

   b. nel metodo `Update`, la velocità venga modificata in base all'accelerazione, e successivamente limitata in modulo ad un valore massimo. A questo proposito, potete usare il metodo `limit` della classe `PVector` (consultate la guida di riferimento di Processing)

   c. sempre nel metodo `Update`, la prima istruzione deve impostare l'accelerazione in modo che questa sia uguale al vettore tra la posizione della particella e la posizione corrente del mouse (in pratica, vogliamo che la particella si diriga verso il mouse). **Suggerimento:** il vettore tra il punto A e il punto B si può calcolare sottraendo A da B.

   d. Dopo aver svolto i punti precedenti, fate girare lo sketch. La particella dovrebbe seguire il mouse. Per rimuovere l'oscillazione rapida della particella intorno alla posizione del mouse, limitate anche il modulo dell'accelerazione, ad esempio al valore 0.2.

5. Definite un array di 20 particelle, con posizione iniziale casuale all'interno della finestra, e velocità iniziale nulla, che si comportano tutte come la particella dell'esercizio 4.

6. Definite un array di 20 particelle, ognuna con le seguenti caratteristiche:

   a. la particella viene creata al centro dello schermo, con velocità casuale tra -2 e 2 in entrambe le direzioni, e con un colore di riempimento casuale;

   b. la particella accelera verso il basso con modulo dell'accelerazione pari a 0.2 (come se subisse la forza di gravità), con velocità limitata a 5 pixel per frame;

   c. la particella che esce dallo schermo viene ricreata al centro, con velocità casuale tra -2 e 2

   d. la particella, nel suo movimento, lascia una "scia". Per realizzare questo tipo di effetto, una tecnica è quella di usare le istruzioni

      ```
      fill(0, alpha);
      rect(0, 0, width, height);
      ```

      anziché

      `background(0)` o altro colore.

      L'influenza dell'effetto "scia" dipende dal valore di `alpha`. Valori più vicini a 0 aumentano l'effetto, valori prossimi a 255 lo riducono.