Programmazione 2 – a.a. 2021-22

Esercitazione Java: Un gioco di bocce

1. Descrizione
2. Modello matematico (semplificato) del moto delle bocce
3. L'interfaccia Gioco

1 Descrizione

Abbiamo un campo di gioco rettangolare con sponde.
Nel campo ci sono da una a 12 bocce, di cui una distinta come boccino. Le bocce hanno in generale ciascuna un diametro diverso, il boccino ha il diametro più piccolo di tutte le altre.
Il campo ha anche da una a quattro buche rotonde di diametro maggiore di quello di qualunque boccia.
Il giocatore tira il boccino, questa operazione imprime al boccino una velocità. Finché non incontra ostacoli, la boccia in moto (inizialmente il boccino) prosegue in linea retta. Ma durante il suo moto può incontrare degli ostacoli:

- Se il centro della boccia in moto (che è anche il suo centro di massa) viene a trovarsi dentro una buca, la boccia cade in buca ed esce dal gioco.
- Se la boccia in moto urta contro una sponda del campo, rimbalza indietro.
- Se la boccia in moto urta un'altra boccia, la boccia che era in moto si ferma e parte l'altra boccia, che si comporterà a sua volta nello stesso modo.

Nota: in ogni momento del gioco abbiamo al più una sola boccia (che potrebbe essere il boccino) in movimento.
Il movimento della boccia, in assenza di urti, rallenta progressivamente col tempo, per cui dopo un po' il gioco sarà di nuovo fermo. Allora il giocatore potrà tirare di nuovo il boccino, ecc. ecc.
Il giocatore ha a disposizione un numero massimo di tiri, che determina il livello di difficoltà del gioco.
Il gioco finisce quando il boccino finisce in buca, oppure quando il giocatore ha esaurito il numero di tiri a disposizione. Il punteggio del giocatore è pari al numero di bocce, diverse dal boccino, finite in buca.
La simulazione del gioco avviene in modo discreto a scatti, con un intervallo di tempo Δ fissato (piccolo) tra uno scatto e il successivo.

Che cosa viene dato

a) Viene data un'interfaccia Java **Gioco** che specifica tutte le funzioni che la vostra classe principale dovrà implementare.

b) Viene dato anche un visualizzatore grafico del gioco. Il visualizzatore è scritto in Java e assume di lavorare con un oggetto di una qualche classe (sarà la vostra classe) che implementa l'interfaccia **Gioco**.

Che cosa dovete fare

Scrivere una classe che implementa l'interfaccia **Gioco** data realizzando in tal modo il motore del gioco.
Una volta sostituita la vostra classe nel visualizzatore, avrete un gioco funzionante.
Naturalmente la vostra classe principale, che implementa l'interfaccia **Gioco**, userà varie altre vostre classi che rappresenteranno i vari oggetti coinvolti nel gioco e forniranno i metodi per realizzare sotto-parti del funzionamento del gioco. Queste classi saranno legate tra loro da gerarchie di ereditarietà e da relazioni di clientela, secondo i principi della programmazione orientata a oggetti.
E' possibile utilizzare classi predefinite di Java e tutti gli strumenti che Java offre e che sono stati presentati nel corso.

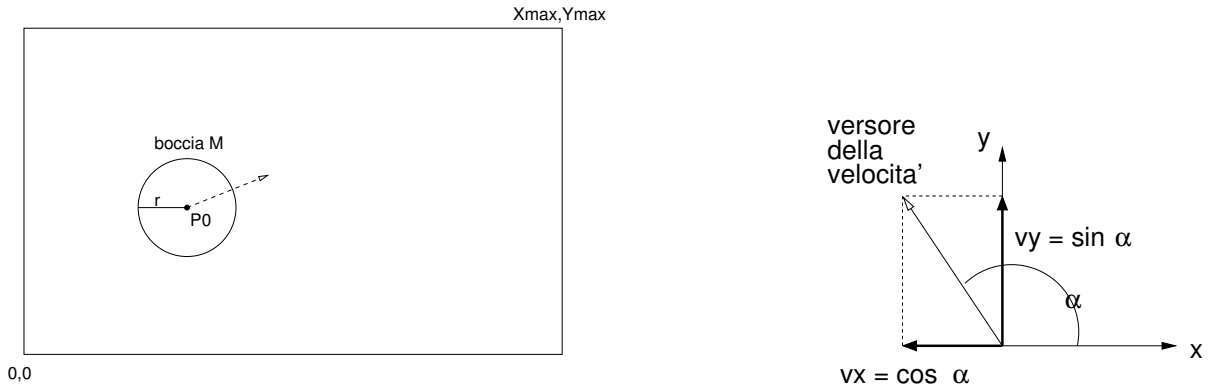

Figure 1: A sinistra: il campo di gioco con il suo sistema di coordinate e una boccia M (le dimensioni della boccia sono volutamente esagerate rispetto a quelle del campo).
A destra: le due componenti del versore della velocità.

Ordini di grandezza coinvolti

Assumendo di esprimere le lunghezze in centimetri:

- larghezza e lunghezza del tavolo sono dell'ordine di (pochi) metri (indicativamente tra 150 e 250 cm)
- i diametri delle bocce sono dell'ordine della decina di centimetri (indicativamente tra 5 e 20 cm)
- il diametro delle buche è alcuni centrimetri più grande rispetto a quelli delle bocce (massimo 25 cm)
- la simulazione del movimento di gioco avviene "a scatti" con intervallo di tempo Δ fissato (tramite la costante `DELTA_T` nell'interfaccia Gioco) pari a 20 millisecondi (= 0.02 secondi)
- la velocità iniziale del boccino è di qualche decina di centimetri al secondo (massimo 50) e, siccome la velocità non aumenta, questa sarà anche la velocità massima di qualsiasi boccia in movimento
- il decremento di velocità ad ogni scatto della simulazione (in assenza di urti) è fissato (tramite la costante `DECREMENTO` nell'interfaccia Gioco) pari a 0.05 cm / secondo.

Da quanto sopra segue che, in uno scatto di simulazione, la boccia in moto si sposta meno di un centimetro. Questo significa che la simulazione a scatti è sufficiente a rilevare qualsiasi urto.

2 Modello matematico (semplificato) del moto delle bocce

Il campo di gioco ha un sistema di coordinate con origine $(0, 0)$ nell'angolo in basso a sinistra ed unità pari a 1 centimetro. L'angolo in alto a destra ha coordinate (x_{max}, y_{max}), dove x_{max} e y_{max} sono le due dimensioni del campo. Un esempio si ha in figura 1, parte sinistra.
Tutte le lunghezze saranno espresse in centimetri come unità di misura. Identificheremo la posizione di una boccia con quella (x, y) del suo centro, nel sistema di coordinate del campo di gioco.
Per il movimento delle bocce, gli urti e i rimbalzi ricorreremo ad un modello matematico molto semplificato, ma sufficiente al nostro scopo.

Moto rettilineo

Sia M l'unica boccia in movimento e r il suo raggio. Nell'istante immediatamente precedente a questo scatto della simulazione, la boccia M ha centro in $P_0 = (x_0, y_0)$. La posizione di partenza P_0 si assume che sia valida, ovvero corrispondente a boccia collocata dentro al campo, non in buca, non intersecante un'altra boccia. Vedere figura 1, parte sinistra.
Per rappresentare la velocità con cui M si muove, che è un vettore, faremo riferimento a:

- il **modulo** $|v|$ del vettore velocità (con $|v| \geq 0$, e se $|v| = 0$ significa che la boccia è ferma)

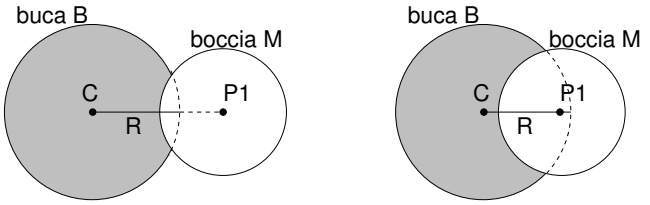

Figure 2: Nella situazione a sinistra la boccia M non cade nella buca B perché il suo centro P_1 si trova fuori dalla buca. Nella situazione a destra invece M cade nella buca B.

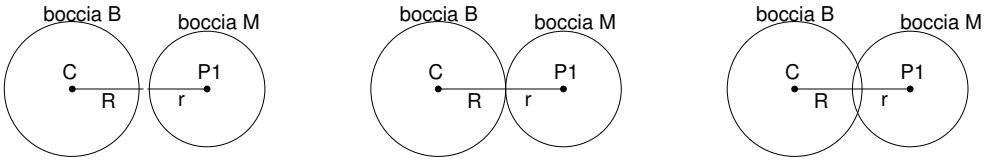

Figure 3: Nella situazione a sinistra la boccia M non urta la boccia B. Nelle altre due situazioni invece l'urto avviene. Nella situazione a destra l'urto è in realtà avvenuto qualche istante prima, ma tolleriamo una minima sovrappsizione tra le due bocce.

- le due componenti (v_x, v_y) del **versore** del vettore velocità (vettore di lunghezza unitaria ottenuto dividendo ciascuna componente per il modulo)

La velocità verrà fornita (mediante la funzione `preparaBoccino` dell'interfaccia Gioco) passando il modulo e l'angolo α formato dal vettore velocità con la direzione positiva dell'asse x (vedere figura 1, parte destra). Le due componenti del versore si ricavano mediante:

$$v_x = \cos\alpha$$
$$v_y = \sin\alpha$$

Durante lo scatto di simulazione, che corrisponde ad un intervallo di tempo Δ (stabilito da una costante nell'interfaccia Gioco), la boccia M si muove e raggiunge una nuova posizione $P_1 = (x_1, y_1)$ espressa da:

$$P_1 = P_0 + \Delta \cdot |v| \cdot (v_x, v_y)$$

Dopo avere calcolato la posizione $P_1 = (x_1, y_1)$ controlleremo se la boccia incontra qualche ostacolo e, nel caso, agiremo di conseguenza.

Caduta in buca

Sia B una buca, C il centro e R il raggio della buca B. La boccia M cade in buca se il suo centro P_1 sta dentro o sul bordo della buca (vedere figura 2) ovvero se:

$$\text{dist}(P_1, C) \leq R$$

dove $\text{dist}(\_, \_)$ denota la distanza tra due punti.
Se la boccia M finisce in buca, viene eliminata. Essendo che questa era l'unica boccia in moto, il gioco si ferma e riprenderà quando l'utente avrà tirato di nuovo il boccino (a meno che la boccia appena finita in buca non sia il boccino: allora il gioco è finito).

Urto con un'altra boccia

Sia B una boccia diversa da M e siano C ed R il centro e il raggio della boccia B. La boccia M in moto urta la boccia B (vedere figura 3) se e solo se:

$$\text{dist}(P_1, C) \leq r + R$$

Nota: precisamente bisognerebbe intercettare l'urto nell'istante in cui vale l'uguaglianza. Noi che lavoriamo con una simulazione a scatti ammettiamo di tollerare pochi millimetri di sovrapposizione tra le due bocce.

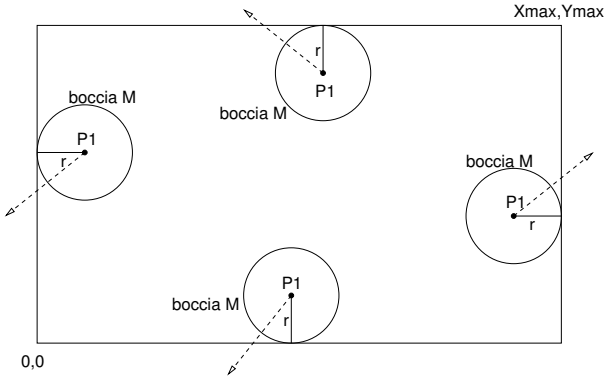

Figure 4: Situazioni in cui la boccia M urta ciascuna delle quattro sponde del campo (tratteggiati i vettori velocità).

In caso di urto, il moto si trasferisce da M all'altra boccia B, la quale acquista la stessa velocità che aveva la boccia M. La boccia M si ferma.
Nota: questo modello di comportamento non corrisponde alla realtà, ma per i nostri scopi va bene. Anche la boccia urtante M dovrebbe rimanere in moto e rimbalzare, la direzione di moto della boccia urtata B dovrebbe variare a seconda dell'angolo di incidenza, l'intensità della velocità dovrebbe essere ripartita tra le due bocce secondo le loro dimensioni... ma evitiamo!

Urto contro due (o più) bocce

Evento poco probabile. Nel caso, processiamo un solo urto, con una sola delle bocce.

Urto con una sponda

Il controllo se la boccia M urta una sponda è semplificato dal fatto che le sponde non possono avere inclinazione arbitraria, ma sono paralelle agli assi cartesiani. Vedere figura 4.

- per la sponda sinistra del campo (avente $x = 0$), l'urto può avvenire solo se $v_x < 0$, ed avviene se e solo se $x_1 - r \leq 0$
- per la sponda destra (avente $x = x_{max}$), l'urto può avvenire solo se $v_x > 0$, ed avviene se e solo se $x_1 + r \geq x_{max}$
- se $v_x = 0$ la boccia non può urtare una sponda verticale, al più potrebbe muoversi rasente lungo di essa
- per la sponda in basso (avente $y = 0$), l'urto può avvenire solo se $v_y < 0$, ed avviene se e solo se $y_1 - r \leq 0$
- per la sponda in alto (avente $x = y_{max}$), può avvenire solo se $v_y > 0$, ed avviene se e solo se $y_1 + r \geq y_{max}$
- se $v_y = 0$, la boccia non può urtare una sponda orizzontale, al più potrebbe muoversi rasente lungo di essa

Nota: precisamente bisognerebbe intercettare l'urto nell'istante in cui vale l'uguaglianza. Noi che lavoriamo con una simulazione a scatti ammettiamo di tollerare pochi millimetri di sconfinamento oltre la sponda.
In caso di urto, la direzione di moto della boccia M (che sarà usata per eseguire il prossimo scatto di simulazione) cambia. Dobbiamo eseguire una riflessione simmetrica rispetto alla normale alla sponda.
I due angoli tra la normale alla sponda (diretta verso l'interno del campo) e l'opposto del vettore velocità incidente, e tra la normale alla sponda e il vettore velocità riflessa, sono uguali fra loro (vedere figura 5).
Ricordiamo che, come le sponde, anche le normali alla sponde sono parallele agli assi cartesiani (ciascuna perpendicolare alla sua sponda e diretta verso il centro del campo).
Nella riflessione, la componente della velocità di M in direzione tangente alla sponda (perpendicolare alla normale) resta invariata, mentre l'altra componente cambia di segno (vedere figura 5, parte sinistra):

- nell'urto con una sponda orizzontale le due componenti del versore della nuova velocità sono $v'_x = v_x$ e $v'_y = -v_y$

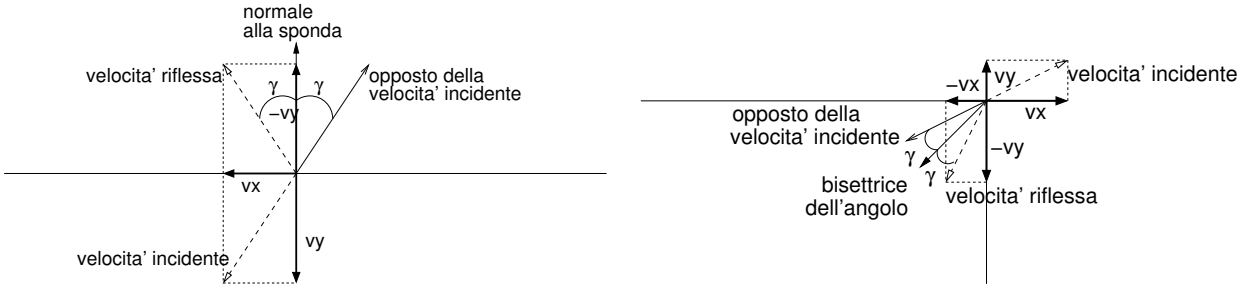

Figure 5: A sinistra: riflessione del moto della boccia che urta contro la sponda inferiore del campo. A destra: riflessione del moto della boccia che urta contemporaneamente contro la sponda in alto e la sponda destra del campo.

- nell'urto con una sponda verticale le due componenti del versore della nuova velocità sono $v'_x = -v_x$ e $v'_y = v_y$

Urto con due sponde

La boccia potrebbe urtare contemporaneamente con due sponde che fanno angolo, per esempio la sponda superiore e la sponda destra (vedere figura 5, parte destra).
In quel caso si applicano gli stessi calcoli descritti sopra, ma considerando la bisettrice dell'angolo (diretta verso l'interno del campo) al posto della normale alla sponda. Equivalentemente, si considera la media delle normali alle due sponde coinvolte.
Le due componenti del versore della velocità cambiano di segno entrambe e si scambiano fra loro, diventando $v'_x = -v_y$ e $v'_y = -v_x$.

Urto contro una sponda e una boccia

Evento poco probabile. Nel caso, gestiamo l'urto contro la sponda in questo scatto di simulazione, mentre l'urto contro la boccia sarà eventualmente trattato nello scatto seguente.

Se non ci sono stati urti

Se non ci sono stati urti, l'intensità della velocità della boccia M in moto diminuirà di 0.05 cm al secondo (come stabilito da una costante nell'interfaccia Gioco). Se tale diminuzione la porta a zero allora la boccia si ferma. Se la diminuzione la porta ad un valore minore di zero, viene messa a zero.

Riepilogo

Uno scatto di simulazione deve fare nell'ordine:

- avanzare la boccia M in moto portandola dalla vecchia posizione P_0 alla nuova posizione P_1
- ciclo sulle buche, che controlla se la boccia M cade in qualche buca; se la boccia M cade in una buca, allora viene eliminata e questo scatto di simulazione termina
- ciclo sulle sponde che controlla se la boccia M urta qualche sponda, bisogna contare quante sponde urta (ne può urtare fino a due); se M urta una sponda o due sponde, allora la boccia M rimbalza e questo scatto di simulazione termina
- ciclo sulle bocce, che controlla se la boccia M urta qualche altra boccia; se M urta una boccia B (può urtarne più di una, se ne considera una), allora la boccia M si ferma, si mette in moto la boccia urtata B, e questo scatto di simulazione termina
- se il flusso d'esecuzione arriva qui, si diminuisce la velocità della boccia M e lo scatto di simulazione termina

3 L'interfaccia Gioco

E' dato il file Gioco.java contenente l'interfaccia che dovete implementate. I commenti sono predisposti per generare la documentazione con javadoc. L'interfaccia contiene le costanti:

- `DELTA_T` intervallo di tempo tra due scatti di simulazione
- `DECREMENTO` decremento della velocità dopo ogni scatto di simulazione

e definisce funzioni per i seguenti scopi:

- Inizializzare il gioco leggendo da file le dimensioni del campo, il numero delle bocce e delle buche e i loro diametri

  ```
  void inizializzaLeggendo(String nome) throws Exception
  ```

- Stabilire il numero di tiri a disposizione del giocatore

  ```
  void cambiaNumeroTiri(int n) throws Exception
  ```

- Ritornare il numero di bocce e di buche

  ```
  int numeroBuche()
  int numeroBocce()
  ```

- Ritornare le dimensioni del campo e il diametro di ciascuna boccia e di ciascuna buca

  ```
  double campoX()
  double campoY()
  double bocciaX(int indiceBoccia)
  double bocciaY(int indiceBoccia)
  double bucaX(int indiceBuca)
  double bucaY(int indiceBuca)
  double diametroBoccia(int indiceBoccia)
  double diametroBuca(int indiceBuca)
  ```

- Ritornare quale boccia è il boccino

  ```
  int indiceBoccino()
  ```

- Controllare se una boccia è già caduta in buca e ritornare il numero di bocce già cadute in buca

  ```
  boolean caduta(int indiceBoccia)
  int numeroCadute()
  ```

- Controllare se, nella situazione attuale, il gioco è finito, controllare se c'è una boccia in movimento, ritornare tiri sono rimasti disponibili, ritornare il punteggio corrente

  ```
  boolean giocoFinito()
  boolean bocceFerme()
  int numeroTiri()
  int punti()
  ```

- Stabilire la velocità iniziale del boccino per il prossimo tiro

  ```
  void preparaBoccino(double intensita, double angoloDirezione)
  ```

- Eseguire uno scatto della simulazione del gioco

  ```
  boolean evoluzioneDeltaT()
  ```

  Molto importante: prima di mettervi ad implementare le funzioni, leggete attentamente i requisiti descritti nei commenti dell'interfaccia Gioco e visionabili nella sua documentazione.