Università degli Studi di Trento — 26 luglio 2019

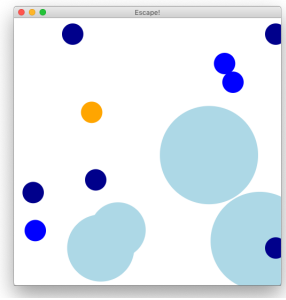
Linguaggi di Programmazione — Programmazione 2

Prova al calcolatore — *Prof. Gian Pietro Picco*

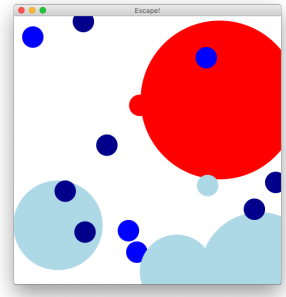
Obiettivi e funzionalità richieste

Si vuole realizzare *"Escape!"*, un semplice videogioco in cui il giocatore deve guidare una palla colorata (di seguito `User`) evitandone la collisione con altre palle nemiche (di seguito `Enemy`) il cui movimento è contemporaneamente guidato dal computer.

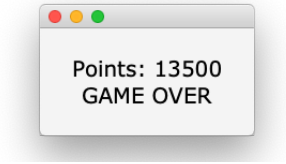
- Il movimento di `User` avviene mediante i tasti freccia, la cui pressione fa muovere `User` di 10 pixel nella direzione corrispondente. Ogni movimento di `User` (pressione del tasto) costituisce una "iterazione" del gioco.
- Successivamente al movimento di `User`, determinato dall'utente, nella stessa iterazione il computer aggiorna la posizione e le caratteristiche delle palle `Enemy` da esso pilotate. Queste possono essere di tre tipi:
  - `Striker` sceglie una direzione a caso e procede lungo di essa per tutta la durata del gioco;
  - `Wanderer` cambia direzione, a caso, ogni 5 iterazioni;
  - `Bubbler` si comporta come `Wanderer` e, in aggiunta, a ogni iterazione aumenta il proprio raggio del 20% con una probabilità del 10%.
- Il gioco inizia con un elemento per ogni tipologia: `User`, `Striker`, `Wanderer` e `Bubbler`. Questi vengono posizionati in un punto a caso dell'area di gioco.
- Man mano che il gioco procede, il computer aggiunge nuovi `Enemy` all'area di gioco (Figura 1(a)) rendendo il compito dell'utente progressivamente più difficile. Un nuovo `Enemy` viene aggiunto ogni 10 iterazioni: la sua tipologia (`Striker`, `Wanderer`, o `Bubbler`) viene determinata casualmente con la stessa probabilità.
- Quando uno degli elementi (`User` o `Enemy`) esce da uno dei lati dall'area di gioco, esso viene automaticamente riposizionato sul lato opposto. Ad esempio, se l'utente causa l'uscita di `User` dal lato superiore della finestra di gioco, la palla corrispondente riappare dal lato inferiore.
- A ogni iterazione senza collisioni vengono assegnati 100 punti. Il punteggio viene visualizzato in una finestra secondaria, *diversa* da quella in cui il gioco si svolge.
- Il gioco termina quando si verifica una collisione fra `User` e un `Enemy`: le due palle coinvolte diventano di colore rosso (Figura 1(b)) e il gioco termina, visualizzando la scritta "GAME OVER" nella finestra secondaria (Figura 1(c)).



(a) Fase di gioco.



(b) Collisione.



(c) Finestra secondaria alla fine del gioco (non in scala).

Figura 1: Escape!

Ulteriori dettagli e specifiche a cui attenersi:

- L'area di gioco è di 500×500 pixel, la finestra secondaria è di 100×200 pixel.
- Il raggio iniziale di ogni palla è 20 pixel.
- I colori (`Color`) da utilizzare (pena riduzione punti) per i vari elementi sono: `User`: `ORANGE`; `Striker`: `BLUE`; `Wanderer`: `DARKBLUE`; `Bubbler`: `LIGHTBLUE`.
- Ogni qualvolta un `Enemy` viene generato e posizionato a caso sull'area di gioco, si deve fare in modo che questo non generi immediatamente una collisione con `User`.
- Le direzioni a caso di un `Enemy` sono 8: 2 orizzontali, 2 verticali, 4 diagonali.
- Nonostante la specifica usi il termine "iterazione", l'implementazione deve utilizzare la gestione degli eventi di JavaFX e *non* un ciclo di lettura da tastiera.

Requisiti

- Si usi, quando evidente/utile, una gerarchia di ereditarietà, sfruttando ove possibile il polimorfismo.
- Si usino costanti ove ragionevole, e si badi alla pulizia del codice (es., linee guida Java) evitando duplicazioni e codice inutilmente complesso.
- È richiesto il *class diagram* UML. È sufficiente la vista di alto livello (no attributi/metodi) ma devono essere mostrate le associazioni più importanti, oltre a quella di ereditarietà, analogamente a quanto fatto a lezione. Il diagramma UML deve essere consegnato su un foglio protocollo indicando nome, cognome, numero di matricola.

Suggerimenti

- Si presti attenzione al fatto che i valori del centro e raggio di un `Circle` sono `double`.
- Per la generazione di numeri casuali si possono usare i metodi `nextDouble()` e `nextInt(int)` della classe `Random` nel package `Math` oppure, in alternativa, il metodo `Math.random()`.
- La realizzazione del gioco non necessita di layout particolari e/o complicati.
- Per i tasti freccia si raccomanda di usare i codici definiti in `KeyCode`.