Processing (Java): Bolle

Introduzione

Il gioco consiste nel calcolare velocemente la somma di due numeri che scendono dall'alto e scrivere tale somma. I numeri nel gioco originale (TuxMath) erano racchiusi in una bolla che scoppia se tocca terra. Nel tutorial ci si limita a far scendere i due numeri senza particolari effetti grafici.

Descrizione del gioco

Due bolle appaiono in cima allo schermo (sul lato sinistro) con due numeri dentro. Cominciano a cadere. Ci sono due aree rettangolari sul lato destro dello schermo. In quella inferiore il giocatore deve digitare la somma dei due numeri. La risposta dell'utente appare sotto alla frase che chiede la somma. Quando il giocatore preme `Invio/Return` una scritta nella casella di testo superiore indica se la risposta è esatta, ripetendo la risposta esatta proprio sotto oppure, se è sbagliata, dando la risposta esatta sotto. C'è un contatore per le risposte esatte e per il numero totale di risposte date dal giocatore fino alla fine del gioco. Dopo ciascuna risposta delle nuove bolle appariranno sullo schermo con due nuovi numeri. Il gioco termina quando il giocatore non riesce a dare una risposta e le bolle arrivano in fondo allo schermo. Allora, nell'area rettangolare inferiore, appare una scritta che dice che il gioco è terminato e dà il numero di risposte esatte e il numero totale di tentativi nel gioco.

Devi scrivere il gioco in maniera iterativa, cioè prima devi creare e inizializzare le variabili di base, scrivere la funzione `setup` per creare lo schermo, e la funzione `draw` che implementa l'animazione. Dopo cambia il codice dove necessario, aggiungi ancora una o due funzioni per non ripetere il codice e ottieni il gioco finale. Prima prova a scrivere il codice da solo, senza guardare il codice di esempio qui sotto!

Una possibile soluzione in Processing

Dal momento che questo gioco può essere programmato in molti modi differenti, qui diamo una soluzione di base. Dopodiché, usando i suggerimenti nella sezione "Suggerimenti per miglioramenti", puoi modificare il programma nel modo che vuoi.

Oggetti utilizzati

Gli oggetti usati nel gioco sono:

- due bolle (due cerchi) con numeri scritti al loro centro che appaiono in cima allo schermo e iniziano a cadere giù
- due aree rettangolari nelle quali il testo deve essere scritto dinamicamente durante il gioco

Variabili

Abbiamo scelto di usare delle variabili globali e di scrivere delle funzioni che non hanno bisogno del passaggio di variabili come input/output. Le variabili che devono essere create sono:

- le variabili che definiscono le coordinate `x` e `y` delle due bolle
- i due numeri che devono apparire dentro a tali bolle, e la loro somma

- una variabile per memorizzare la risposta data dal giocatore
- una variabile di stato per tenere traccia dello stato del gioco (nuovo gioco, risposta esatta, risposta errata, game over) che viene usata per decidere quali scritte fare apparire nelle aree rettangolari e per l'inizializzazione dei numeri che devono apparire nelle bolle
- due contatori, uno per le risposte esatte, uno per il numero totale di risposte date dal giocatore
- una variabile per memorizzare i messaggi dati nelle due aree di testo

```
// variabili per il posizionamento delle bolle
float circleY = 25;
float circleX1 = 50;
float circleX2 = 150;

// i valori nelle bolle e la loro somma
int num1 = int(random(100));
int num2 = int(random(100));
int realSum = num1 + num2;

// per tenere traccia delle riposte date dal giocatore
int count = 0;   // conteggio delle risposte esatte
int total = 0;   // totale delle addizioni nel gioco

String solution = "";   // usata per memorizzare l'input del giocatore - la somma
                        // dei due numeri - come tipo String
int numSolution;        // versione integer dell'input del giocatore

int state = 0;       // stato del gioco:
                     // 0: un nuovo insieme di numeri appare nelle bolle
                     // 1: il giocatore ha fornito una risposta esatta
                     // 2: il giocatore ha fornito una risposta esatta
                     // 3: le bolle hanno raggiunto il fondo dello schermo senza
                     // una risposta dal giocatore

String message1 = "", message2 = "";   // messaggi da mostrare al giocatore
                                       // message1: quando state = 0 oppure 1,
                                       // appare nella casella di testo inferiore
                                       // message2: quando state = 2 oppure 3,
                                       // appare nella casella di testo superiore
```

La funzione `setup`

In questa versione di base del gioco la funzione `setup` viene usata per creare lo schermo per il gioco.

```
void setup(){
  size(600, 400);
}
```

La funzione `draw`

Presupposti: le due bolle sono create alla stessa coordinata `y`, nello stesso momento e cadono giù in parallelo alla stessa velocità.

1. Inizia con un colore a tua scelta per: lo sfondo, le bolle e le due aree di testo. Puoi usare dei colori fissi o dei colori casuali.
2. Crea le bolle e scrivi i numeri all'interno delle bolle.
3. Per ottenere l'effetto delle bolle che cadono, cosa devi fare? Scrivi il codice che anima le bolle.

4. Come controlli se le bolle hanno raggiunto il fondo dello schermo? Controlla se hanno raggiunto il fondo. Cosa succedere se e quando raggiungono il fondo?

```
void draw(){
  background(200);    // grigio

  if (state == 1 || state == 2) { // il giocatore ha dato una risposta giusta o sbagliata
    delay(2000);
    state = 0;
  }
  fill(255, 10, 10); // rosso; puoi cambiarlo con qualsiasi colore
  rect(300, 250, 300, 30);
  myTextBox();

  fill(0, 255, 0); // verde
  ellipse(circleX1, circleY, 20, 20);
  fill(0, 0, 0); // nero
  text(num1, circleX1-7, circleY+5);
  fill(0, 255, 0);
  ellipse(circleX2, circleY, 20, 20);
  fill(0, 0, 0);
  text(num2, circleX2-7, circleY+5);
  circleY = circleY + 1;
  if (circleY > height && (key != RETURN || key != ENTER)) {  // le bolle arrivano in fondo
    state = 3;
    myTextBox();
    noLoop();   // termina il ciclo di disegno
  }
}
```

Hai visto la funzione `delay()` nel codice qua sopra? È necessario interrompere l'animazione per un periodo di tempo in modo che i messaggi nei campi di testo possano essere visti. Per fare in modo che la funzione `delay()` lavori correttamente dobbiamo aggiungere una riga in testa al nostro programma che includerà una libreria.

```
import processing.serial.*; // abbiamo bisogno di questa per inserire alcuni ritardi nel gioco
```

La funzione `keyPressed`

Presupposti: ipotizziamo che il giocatore prema solo tasti numerici, `Invio` oppure `Return`. Questa ipotesi può essere cambiata in un'altra versione del codice dove i tasti premuti possono essere controllati perché siano numerici e un avvertimento viene mostrato se il giocatore preme qualsiasi altro tasto.

Dobbiamo tenere traccia dei tasti premuti dal giocatore. Questa funzione è usata anche per decidere se la risposta dell'utente è giusta o sbagliata.

Scrivi una funzione che:

- controlla se il tasto premuto è `Invio` o `Return` (significa che ha finito di scrivere la risposta)
- se *sì*, allora dobbiamo controllare se la risposta è esatta.
  - se *sì*, allora dobbiamo emettere un messaggio che dice cha la risposta è esatta, fare un po' di conti (aggiornare il numero di risposte esatte e di tentativi totali), e ricominciare il gioco con numeri nuovi
  - se *no*, allora dobbiamo emettere un messaggio che dice cha la risposta è errata, dare la risposta esatta, fare un po' di conti (aggiornare il numero di tentativi totali), e ricominciare il gioco con numeri nuovi

- se *no* (e il tasto premuto è numerico), aggiungiamo il nuovo carattere corrispondente al tasto alla stringa degli altri tasti premuti prima.

NB: ricordati che la somma dei due numeri è un `integer` però i tasti premuti sono caratteri. Fai la conversione di tipo necessaria per essere in grado di confrontare la risposta data con la vera somma dei due numeri.

```
void keyPressed()
{
  if (key == RETURN || key == ENTER) {  // l'input del giocatore è finito; controlla se il
                                        // risultato è esatto
    numSolution = int(solution);        // ottieni il valore integer dell'input del giocatore
    total++;                            // incrementa di 1 il numero di risposte totali date
                                        // dal giocatore
    if (numSolution == realSum) { // risposta esatta
      state = 1;
      count++;                    // incrementa di 1 il numero di risposte esatte
      myTextBox();

      // ricomincia il gioco con nuovi numeri
      num1 = int(random(100));
      num2 = int(random(100));
      realSum = num1 + num2;
      solution = "";              // inizializza la variabile vuota per mantenere i tasti premuti
      circleY = 25;
      return;
    } else { // risposta sbagliata
        state = 2;
        myTextBox();
        // ricomincia il gioco con nuovi numeri
        num1 = int(random(100));
        num2 = int(random(100));
        realSum = num1 + num2;
        solution = "";
        circleY = 25;
        return;
    }
  }
  else if (key >= '0' && key <= '9')  // il giocatore continua a scrivere
  {
    solution = solution + key;        // aggiunge l'ultimo tasto premuto alla stringa di tasti premuti
  }
}
```

La funzione `myTextBox`

Per avere codice più leggibile e compatto scriviamo una funzione che stamperà il testo giusto nella casella di testo giusta per ogni stato del gioco, cioè:

- chiede la somma (nella casella di testo inferiore) quando una nuova coppia di numeri appare nelle bolle
- stampa un messaggio (nella casella di testo superiore) quando viene data una risposta esatta
- stampa un messaggio (nella casella di testo superiore) quando viene data una risposta sbagliata
- stampa un messaggio (nella casella di testo inferiore) quando il gioco è finito

```
void myTextBox() { // per comunicare con il giocatore

  switch(state) {
```

```
    case 0: // inzio del gioco; un nuovo insieme di numeri
      fill(255); // bianco; può essere cambiato con qualsiasi colore
      rect(300, 350, 300, 30);
      fill(0);
      message1 = "Qual è la somma di quei numeri?\n";
      text(message1+solution, 310, 365);
      break;
    case 1: // soluzione esatta
      fill(255, 20, 20); // rosso; può essere cambiato con qualsiasi colore
      rect(300, 250, 300, 30);
      fill(0);
      message2 = "Esatto! \n ";
      text(message2+solution, 310, 265);
      break;
    case 2: // soluzione sbagliata
      fill(255, 20, 20); // rosso
      rect(300, 250, 300, 30);
      fill(0);
      message2 = "Risposta sbagliata! Era \n";
      text(message2+realSum, 310, 265);
      break;
    case 3: // bolle fuori dallo schermo, nessuna risposta, game over
      fill(255);  // bianco
      rect(300, 350, 300, 30);
      fill(0);
      message1 = "Tempo esaurito! Game over con "+count+"\nrisposte esatte su "+total;
      text(message1, 310, 365);
  }
}
```

Suggerimenti per miglioramenti

- Aggiungi un effetto di esplosione alle bolle quando raggiungono il fondo dello schermo.
- Aggiungi un cronometro al gioco e metti un tempo limite. Il gioco finisce quando viene raggiunto il tempo limite oppure quando le bolle raggiungono il fondo dello schermo.
- Aggiungi un nuovo livello quando viene raggiunto un certo conteggio di risposte esatte. In questo nuovo livello puoi aggiungere una nuova bolla che contiene un operatore aritmetico casuale `+` oppure `-` che sarà l'operazione da fare tra i due numeri. Oppure puoi avere un terzo numero nella nuova bolla e fare la somma dei tre numeri.
- Qualsiasi altro miglioramento che ti viene in mente?
- Qualsiasi miglioramento della grafica o delle animazioni?

Just go for it! Provaci!