Gioco Mangia-Biscotti per Progetto Sveglia

Dettagli di sviluppo

Un prototipo utilizzabile del gioco pensato per la sveglia è stato sviluppato con Java, un linguaggio che si presta bene allo sviluppo di minigiochi, partendo da una proposta già trovata nel percorso di studi, rivisitando e implementando però nuovi aspetti per adattarsi al contesto voluto.

Sono state definite diverse classi: una per ogni tipologia di biscotto presente nel gioco e una per il "Ghiottone" che è la pallina gialla comandata dall'utente. Tutte si ritrovano ad interagire con il *Main*, la funzione principale, luogo in cui vengono anche gestiti gli input dell'utente tramite degli *EventHandlers* (Sia con mouse, che potrebbe simulare un tasto analogico, sia con le freccette della tastiera). La scena di JavaFx è aggiornata da un'altra classe specifica con *metodi statici.* I diversi biscotti secondo delle probabilità predefinite vengono scelti a sorte e archiviati prima dell'inizio della partita in una *<ArrayList>* pubblica, comparendo mano a mano ogni volta che un biscotto viene mangiato dall'utente. Di default, i biscotti in partenza quando viene avviato il gioco sono tre.

La barra superiore azzurra mostra tre informazioni principali: i biscotti mangiati, i punti totalizzati, aggiornati ogni volta che viene mangiato un biscotto tramite la funzione ridefinita per ogni classe *eat* , e il tempo. Quest'ultimo è aggiornato da una funzione di update ad ogni movimento del "Ghiottone", e per semplicità nel prototipo è stimato come 1 secondo= 8 iterazioni.

Alcuni tratti del codice

```
10 usages
@Override
public void eat() {
    super.eat();
    app.setPoints(app.getPoints() + 10);
    Main.SceneX.changeBackground(this.app.n_sec, safety_check: false);
```

Esempio di funzione eat ridefinita per il biscotto di Halloween

```java
a.a.setOnKeyPressed(new EventHandler<KeyEvent>() {
    @Override
    public void handle(KeyEvent keyEvent) {
        if (!gameover) {
            double newX = ghiottone.getCenterX();
            double newY = ghiottone.getCenterY();

            // Verifica quale freccia è stata premuta e aggiorna le coordinate
            switch (keyEvent.getCode()) {
                case UP:
                    newY -= Ghiottone.STEP;
                    break;
                case DOWN:
                    newY += Ghiottone.STEP;
                    break;
                case LEFT:
                    newX -= Ghiottone.STEP;
                    break;
                case RIGHT:
                    newX += Ghiottone.STEP;
                    break;
                default:
                    break;
            }

            // Controllo dei limiti del movimento
            if (newX < Ghiottone.RADIUS)
                newX = Ghiottone.RADIUS;
            else
                newX = Math.min(newX, WIDTH - Ghiottone.RADIUS);

            if (newY < Ghiottone.RADIUS)
                newY = Ghiottone.RADIUS;
            else
                newY = Math.min(newY, HEIGHT - Ghiottone.RADIUS);
```

Gestione del movimento del Ghiottone tramite tasti

```java
2 usages
public void setTimeRemaining() {
    --this.n_sec;
    this.txttime.setText("Secondi Rimenenti:" + this.n_sec);
    if (this.n_sec == 0) {
        this.fineGioco();
    }

}
```

Funzione setTimeRemaining che aggiorna i secondi rimasti. Quando i secondi finiscono, chiama la funzione fineGioco()

```
    Cookie c = null;
    switch (random.nextInt( bound: 5)) {
        case 0:
            Random d = new Random();
            int l = d.nextInt( bound: 250);
            if (l == 0) {
                c = new BiscottoORO( app: this);
                break;
            }
            if (l <= 100) {
                c = new BiscottoNero( app: this);
                break;
            }
            if (l <= 200) {
                c = new SpecialHalloween( app: this);
                break;
            } else c = new BiscottoRN( app: this);
            break;
        case 1:
            c = new BiscottoLimone( app: this);
            break;

        case 2:
            c = new BiscottoZenzero( app: this);
            break;

        case 3:
            c = new BiscottoPrugna( app: this);
            break;
        case 4:
            c = new NormalCookie( app: this);
            break;
    }
    cookies.add(c);
    playground.getChildren().add(c);
    do {
        c.setPosition(random.nextDouble(c.getRadius(),  bound: WIDTH - c.getRadius()),
                random.nextDouble(c.getRadius(),  bound: HEIGHT - c.getRadius()));
    } while (overlap_cookie(c));
}
```

Generazione dei biscotti che appariranno prima dell'inizio della partita usando funzioni random con probabilità ponderate

Biscotti

La parte centrale sia dello sviluppo che poi dell'esperienza diretta di gioco sono i biscotti. In tabella si riportano le informazioni principali su di essi. In arancione un'idea di biscotto inserito per l'aggiornamento di Halloween.

Nome	Colore	Punti	extra	Rarità
Biscotto Normale	Sandybrown	10	-	**20%**
Biscotto Limone	Greenyellow	3	genera 3 biscotti in più	**20%**
Biscotto Nero	Nero	-12	toglie punti	**8%**
Biscotto Oro	Goldenrod	100	quasi impossibile da trovare	**0,08%**
Biscotto Prugna	Viola	0	sposta il biscotto più vicino all'angolo dello schermo	**20%**
Biscotto Rosso e Nero	Nero bordo rosso	-2	toglie punti e 5 secondi	**3,92%**
Biscotto Zenzero	Marrone	5	-	**20%**
Biscotto Zucca (Halloween special Update)	Arancione bordo marrone	10	cambia dimensione - fa calare la notte nel campo da gioco per 10 secondi (lo sfondo diventa nero)	**8%**