Programmazione ad Oggetti

Relazione progetto

“SpaccaBolle”

Sofia Bagagli

Yuri Collini

Mattia Gullotto

Michele Nardini

Elisa Simoni

**Indice**

**I. Analisi** 3

1.1 Requisiti . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

1.2 Analisi e modello del dominio . . . . . . . . . . . . . . . . . . . . . . .

**II. Design**

2.1 Architettura . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.2 Design dettagliato . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

**III. Sviluppo**

3.1 Testing automatizzato . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

3.2 Metodologia di lavoro . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

3.3 Note di sviluppo . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

**IV. Commenti finali**

4.1 Autovalutazione e lavori futuri . . . . . . . . . . . . . . . . . . . . . . .

4.2 Difficoltà incontrate e commenti per docenti . . . . . . . . . . .

**A. Guida utente**

**Analisi**

**1.1 Requisiti**

Il gruppo si pone l’obbiettivo di implementare una versione basilare del famoso gioco Puzzle Bobble ( SpaccaBolle ). Il gioco è strutturato in vari livelli, con difficoltà crescente. L'implementazione del gioco è caratterizzata dalla presenza di bolle di colore diverso, le quali devono essere eliminate totalmente entro un tempo predefinito, prima che le bolle riempiano tutta la schermata di gioco, ovvero quando almeno una bolla tocca la base. Sulla base è presente un cannone direzionabile che spara bolle di colore diverso. Per poter eliminare le bolle sulla mappa è necessario che queste vengano colpite da una bolla del medesimo colore sparata dal cannone presente nella base.

**Funzionalità Minimali Ritenute Obbligatorie**

* Creazione di una mappa
* Gestione del cannone spara bolle con direzionamento
* Gestione del punteggio
* Algoritmo controllo bolle sulla mappa
* Interfaccia di benvenuto ed impostazioni
* Menù di navigazione moderno
* Salvataggio Partita

**Funzionalità Opzionali**

* Gestione delle combo
* Gestione palline speciali
* Inserimento di ostacoli
* Gestione livelli di gioco

**Challenge Previste**

* Comprensione delle meccaniche di gioco
* Creazione delle interfacce per ogni fase del gioco
* Algoritmo per movimento bolle
* Pulizia del codice
* Efficienza degli algoritmi

**Requisiti funzionali**

Il gioco si presenta con una schermata di benvenuto tramite la quale è possibile effettuare una serie di operazioni, tra le quali:

* iniziare una nuova partita partendo dal primo livello
* caricare un livello precedentemente salvato
* chiudere l’applicazione

Una volta premuto play o caricato un livello precedentemente salvato, all’utente verrà mostrata una schermata di gioco composta da una mappa contenente bolle di diverso colore, un cannone spara bolle che si muove in maniera costante da destra a sinistra. Nella schermata sono inoltre presenti dei pulsanti: *SAVE* ( permette di salvare la partita in corso), *PAUSE* (permette di fermare il gioco in corso), *EXIT* (permette di uscire dal gioco senza salvare). I pulsanti appena descritti rappresentano il menù di navigazione interno al gioco. Il cannone posto alla base della schermata di gioco genererà in maniera semi-casuale delle bolle pronte per essere lanciate sul tabellone. La difficoltà del gioco sta nel formare tris di bolle dello stesso colore ed evitare di accumulare così tante bolle da far sì che tocchino la base. Se le palline toccano il cannone o il terreno il gioco finisce e l’utente perde.

Ogni volta che il giocatore perde dovrà ricominciare da capo il livello e in caso non venisse salvato nessun livello il giocatore dovrà ricominciare dal primo livello di gioco.

Il punteggio finale viene calcolato in base a quante bolle l’utente riesce a far scoppiare o a far cadere. Per la precisione, per lo scoppio di bolle dello stesso colore l’utente guadagnerà 10 punti per ogni bolla scoppiata. Per le bolle fatte cadere il meccanismo di attribuzione del punteggio è leggermente diverso. Infatti la prima bolla fatta cadere permette di guadagnare 20 punti, mentre ogni bolla addizionale raddoppia il punteggio come descritto meglio nello schema successivo.

**Schema dei punteggi per le bolle che cadono in catena**

1 – 20 ( Prima bolla caduta )

2 – 40

3 – 80

4 – 160

5 – 320

6 – 640

7 – 1280

8 – 2560

9 – 5120

10 – 10240

11 – 20480

12 – 40960

13 – 81920

14 – 163840

15 – 327680

16 – 655360

17 o più 1310720

**Attribuzione Punteggi Bonus**

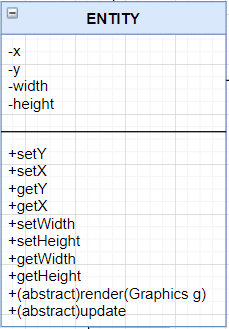
I punti bonus sono assegnati all’utente in base alla velocità con cui si completa lo schema. L’assegnazione dei punti bonus in base al tempo con cui si completa lo schema è così schematizzata :

* 0-5 secondi 50, 000 . . .
* . . . 64 secondi 440
* 65 o più secondi 'NO BONUS'

**Requisiti non funzionali**

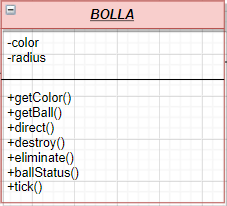
Il gioco dovrà mantenere una fluidità per quanto riguarda il lancio delle bolle e non presentare alcun tipo di lag. Inoltre il codice prodotto dovrà essere il più pulito possibile e facilmente comprensibile.

**1.2 Analisi e modello del dominio**



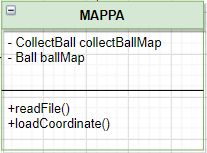
L’entità più importante all’interno del codice di gioco è denominata ***“Entity”***. Questa entità definisce tutti i tipi di elementi presenti all’interno del gioco definendone varie caratteristiche quali: grandezza, posizione. Permettendone inoltre un costante update e renderizzazione grafica. “Entity” è stata utilizzata all’interno del programma realizzato come punto di partenza per la creazione di vari elementi di gioco quali: il cannone, le bolle ed in fine la mappa di gioco.

**ENTITA’ BOLLA**

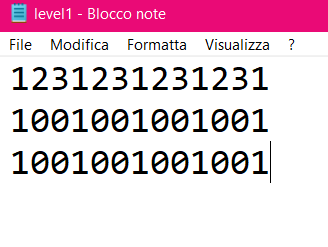


L’entità ***“Bolla”*** implementata prevede la definizione di un colore e di una dimensione specificata dal raggio. Il colore della bolla viene scelto in modo casuale tramite una semplice funzione che usa la libreria random. Il raggio della bolla è utile per effettuare il caricamento delle bolle sulla mappa ( algoritmo che verrà discusso in seguito ) , definirne distanza e possibili collisioni in fase di gioco. Ogni bolla creata, possiede una funzione denominata ***getColor()*** la quale restituisce il colore della bolla, sulla quale viene effettuata la chiamata a tale funzione, utile per capire che tipo di bolla viene lanciata dal cannone. Inoltre è utile per poter controllare se bolle adiacenti possano permettere una collisione (tre dello stesso colore). La funzione ***getBall()*** ci permette di richiamare la bolla presa in considerazione ed analizzarne la posizione. Ogni bolla può essere lanciata dal cannone sulla dimensione y e in caso di collisione essa fermerà il suo andamento attraverso la l’ulteriore funzione ***destroy()***. Tramite la funzione ***eliminate()*** la bolla verrà cancellata e sostituita da una bolla che d’ora in poi verrà considerata una bolla della mappa tramite la funzione ***ballStatus()*** ( la bolla passa dallo stato sospesa allo stato di bolla della mappa). La funzione ***tick()*** permette un continuo aggiornamento sulle collisioni nelle coordinate in x e y dell’entità. In particolare, viene controllata la collisione con altre bolle presenti sulla mappa e con il cannone stesso in caso di game over.

**ENITA’ MAPPA**



Nel gioco, è stato deciso in fase di progettazione di realizzare ogni mappa tramite la lettura di un semplice file di testo come si può vedere nell’immagine riporta di seguito.



La struttura di questo file testuale sembra inizialmente incomprensibile ma è in realtà molto semplice. Infatti ad ogni numero presente nel file testuale corrisponde un determinato colore come riportato nello schema qui di seguito:

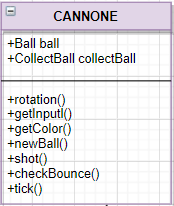
* 0 🡪 Bolla nulla/posizione vuota
* 1 🡪 **Rosso**
* 2 🡪 **Blu**
* 3 🡪 **Verde**
* 4 🡪 **Giallo**

Per caricare le bolle all’interno della mappa occorre effettuare la lettura del file .txt . Per fare ciò si è deciso di suddividere l’operazione di lettura in due parti. La prima parte di questo processo è la lettura del colore, che ci permette di associare ad un numero un entità bolla del medesimo colore a quello appena letto come mostrato qui di seguito :

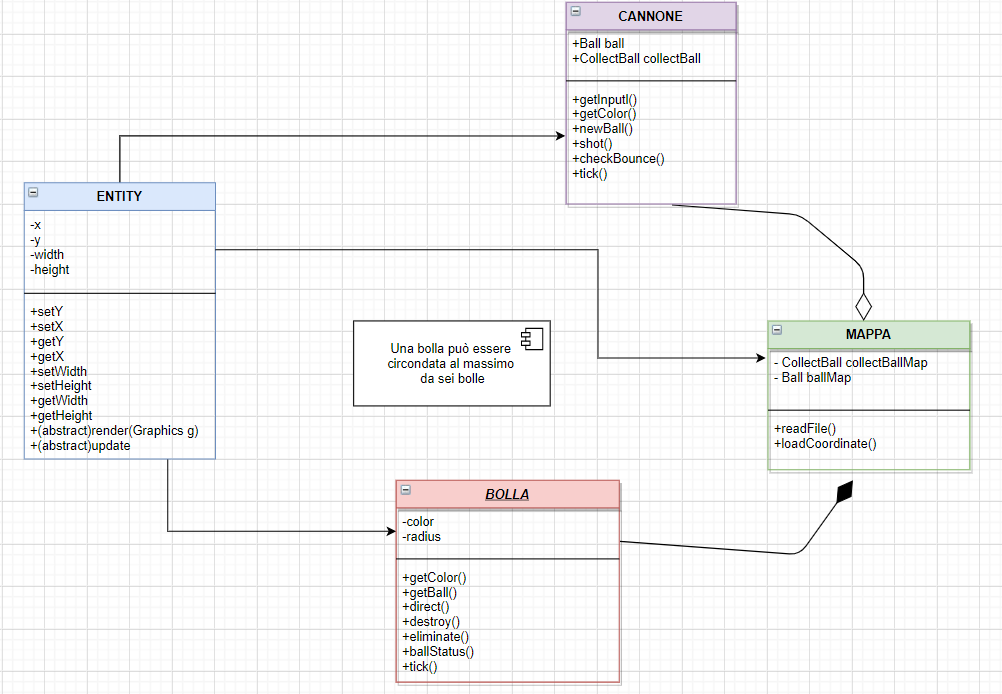
* new Ball (..., Color = 1,2,3,4….) )

La seconda parte è quella che ci permette di disporre le bolle sulla mappa in coordinate prestabilite. Ogni bolla infatti è posizionata orizzontalmente fino al raggiungimento di massimo 8 bolle sulla stessa riga. Una volta che viene raggiunto il limite massimo, le bolle vengono inserite nella linea successiva a scendere fino ad un massimo di 13 righe. La quattordicesima riga è quella di game over. Ogni volta che una bolla viene creata viene inserita in una lista denominata ***collectBallMap*** utile ad una lettura successiva della mappa.

**ENITA’ CANNONE**

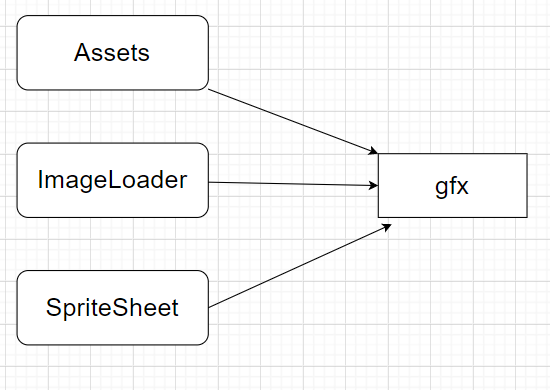


Il cannone è l’entità che permette effettivamente al player di giocare. Il cannone si muove in maniera costante lungo un’asse orizzonatale da destra a sinistra. Al centro del cannone è presente una bolla di colore casuale che può essere lanciata sull’asse Y. La difficoltà del gioco consiste nel lanciare la pallina nel punto preferito dal giocatore nonostante il movimento del cannone. Per lanciare basta premere il tasto ***ENTER***. Attraverso la funziona ***getInput()***, della tastiera e la pallina si muoverà a 60fps verso la destinazione scelta. Mediante la funzione ***getBounce()*** si controlla la collisione della base del cannone contro i lati della mappa di gioco in modo tale che il cannone possa muoversi lateralmente da desta a sinistra.



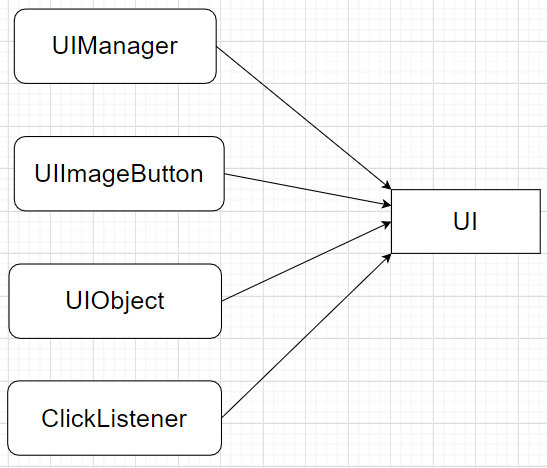
**Design**

GFX



Sfruttando la libreria ***java.gfx*** ci occupiamo della renderizzazione delle immagini nonché delle loro animazioni. Nella classe ***Assets***, presente all’interno del package ***dev.spaccabolle.gfx***, sono presenti i riferimenti assoluti delle immagini dei bottoni del menù, del cannone, delle bolle e di tutte le altre immagini della schermata principale nonché di benvenuto. Le immagini delle bolle sono caricate su di un vettore dove, per ogni posizione è presente il percorso assoluto dell’immagine. La stessa cosa viene fatta per i bottoni in modo tale che alla pressione del pulsante l’immagine diventi leggermente più scura. La stessa cosa vale anche quando il cursore è posizionato sopra il bottone. La classe ***ImageLoader***, anch’essa è presente all’interno del package ***dev.spaccabolle.gfx***, si occupa del buffering delle immagini tramite ***BufferedImage*** attraverso l’utilizzo di ***java.awt***. L’ultimo elemento del package ***dev.spaccabolle.gfx*** è la classe ***SpriteSheet*** ovvero la codificazione delle animazioni del gioco. Gli spritesheets vengono utilizzati per accelerare il processo di visualizzazione delle immagini sullo schermo. Infatti, è molto più veloce recuperare un'immagine e visualizzare solo una parte di tale immagine, piuttosto che recuperare molte immagini e visualizzarle tutte insieme.

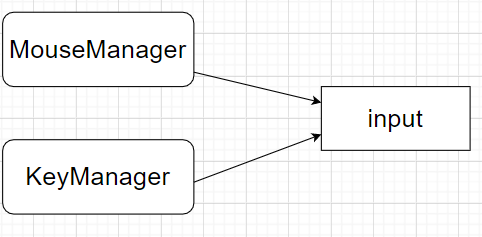
UI



La parte di User-Interface viene gestita da quattro entità distinte. La classe ***UIManager*** presente all’interno del package ***dev.spaccabolle.ui*** è la più importante di tutte le entità. Attraverso la libreria ***java.awt*** ci permette di creare object e mediante l’utilizzo di un handler di gestirlo. Per i bottoni presenti sulle varie schermate di gioco, esiste un interfaccia ***UIImageButton*** che consente di dare caratteristiche comuni ad ogni bottone quali la dimensione, la forma e la gestione ***onClick()*** del bottone. La classe ***ClickListener*** è essenziale per la ricezione degli input da tastiera dati dall’utenti. Gli input vengono elaborati e sfruttati dall’utente per giocare. I comandi utili sono:

* click del mouse
* P (pausa)
* ESC (per uscire).

INPUT



***MouseManager*** presente all’interno del package ***dev.spaccabolle.input*** è una classe che riceve una notifica quando si verifica una modifica nello stato del mouse (***MouseListener***). Le modifiche del mouse possono avvenire premendo, facendo clic e rilasciandolo. Può anche entrare o uscire dall'area della finestra. Questa interfaccia listener può essere ottenuta dal pacchetto ***java.awt.event***.

La classe ***KeyManager*** , anch’essa presente nel package ***dev.spaccabolle.input***, è responsabile della ricezione degli input inviati dall’utente mediante tastiera. Gli input ricevibili da tastiera sono :

* UP
* DOWN
* LEFT
* RIGHT
* ENTER

**2.1 Architettura**

Il pattern architetturale utilizzato per la realizzazione di SpaccaBolle è il classico MVC ( Model – View – Controller ). Le view permettono al gioco di assumere stati diversificati e dare una rappresentazione grafica al gioco. Ognuna di esse possiede un controller che ne permette la gestione nonché la manipolazione. La prima view che verrà mostrata all’avvio del gioco sarà quella della schermata principale ( ***StatoMenu*** ). Mediante la pressione del tasto ***PLAY*** ci permette di passare dalla view ***StatoMenu*** alla view ***StatoGame***. Il tasto ***PAUSE*** presente nella view in game permette di passare alla view ***MenuInGame***.

**3.1 Testing automatizzato**

Questa sezione spiegherà i test eseguiti dal team per verificare il corretto funzionamento di tutto il gioco SpaccaBolle.

• **TestCannon**

Questo test ci permette di verificare il corretto funzionamento del cannone, in particolare ne verifica la posizione durante il movimento costante da destra a sinistra quindi lungo l’asse orizzontale, ed inoltre verifica se effettivamente genera nuove bolle da sparare.

• **TestBall**

Questo test ci permette di verificare la corretta creazione delle bolle, controllare che le posizioni di queste ultime sulla mappa siano allocate in modo corretto ed in fine controllare che l'assegnazione del colore delle bolle sia corretta.

• **TestPoint**

Questo test permette di verificare la corretta generazione dei punti dati dallo scoppio della bolla su una specifica collisione in base ai criteri stabiliti che sono anche stati spiegati in precedenza in questa relazione.

• **TestCollision**

Questo test consente l'utilizzo di restrizioni cartografiche nonché bolle appena avviate e preesistenti per verificare la corretta gestione della raccolta.

**3.2 Metodologia di lavoro**

La suddivisione dell’implementazione del gioco è stata suddivisa in modo tale da permettere ad ogni membro del gruppo lo sviluppo di una parte logica e una parte grafica. La suddivisione punta ad assegnare i compiti in base alle capacità, esperienze pregresse e punti di forza di ogni membro del gruppo di lavoro.

Per poter lavorare in contemporanea sul progetto abbiamo utilizzato GitHub attraverso la creazione di branch personali per facilitare la condivisione del codice prodotto da ciascuno.

I punti critici del progetto sono stati gestiti in collaborazione e sinergia in modo da ottenere il migliore risultato possibile non ostante le oggettive difficoltà incontrate durante lo sviluppo del gioco.

I compiti sono stati così suddivisi:

**Yuri Collini**

* Menù di navigazione
* Interfaccia di benvenuto
* Gestione Punteggio e Bonus

**Michele Nardini**

* Gestione Cannone
* Grafica Personaggi

**Sofia Bagagli**

* Interfaccia Grafica impostazioni
* Gestione Cannone

**Elisa Simoni**

* Algoritmo Bolle sulla mappa
* Salvataggio Partita

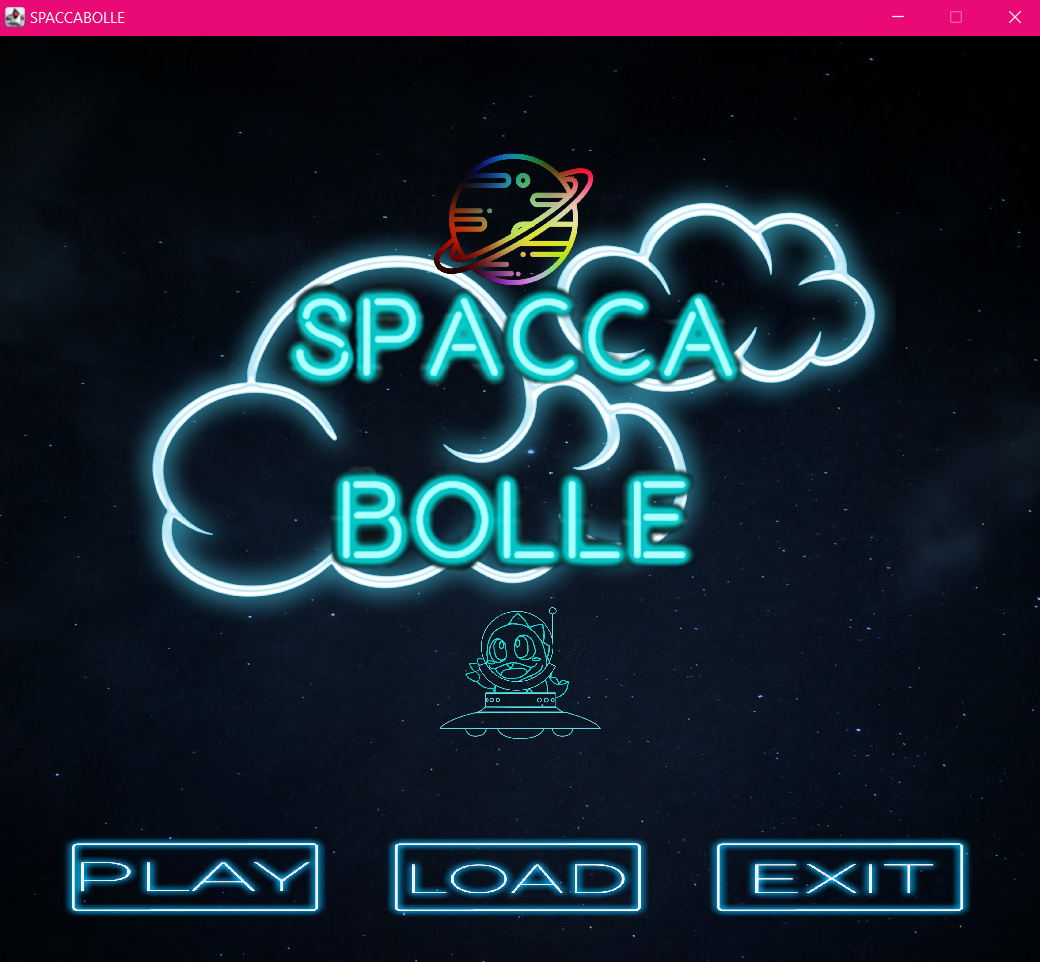
**Mattia Gullotto**

* Algoritmo Bolle sulla mappa
* Grafica Bolle della Mappa

**GUIDA UTENTE**

In questa sezione della relazione, viene riportata una breve guida utente utile a comprendere il funzionamento e tutte le dinamiche del gioco SpaccaBolle da noi sviluppato.

**Schermata di Benvenuto**



All’avvio del programma, l’utente troverà una schermata di benvenuto come quella presente nell’immagine qui sopra. Il giocatore si troverà davanti a sé tre bottoni : PLAY, LOAD, EXIT. Premendo il tasto “PLAY”, presente nel lato sinistro dello schermo, l’utente avvierà una nuova partita, mentre tramite il pulsante “LOAD” il giocatore potrà caricare e continuare a giocare una partita precedentemente salvata. In fine mediante il bottone “EXIT” potrà uscire e chiudere l’applicazione.

**Schermata di gioco** da cambiare

**Schermata di pausa** (ancora da mettere)

Schermata finale/pausa (non esiste ancora)