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 gioco Puzzle Bobble ( SpaccaBolle ). Il gioco è strutturato in 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 che spara bolle di colore diverso. Per eliminare le bolle dalla mappa è necessario che queste vengano colpite da una bolla del medesimo colore.

**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 e 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 d’apertura dove è possibile:

* iniziare una nuova partita partendo dal primo livello
* caricare un livello precedentemente salvato
* uscire dal gioco

Una volta premuto play o caricato un livello si presenterà una schermata di gioco composta da una mappa di 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 (salvare la partita in corso), pause (per fermare il gioco in corso), exit (per uscire dal gioco senza salvare) Il cannone 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 e evitare di accumulare così tante bolle da far si che tocchino il terreno. Se le palline toccano il cannone o il terreno il gioco finisce.

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.

I punti vengono calcolati in base a quante bolle vengono scoppiate o fatte cadere per la precisione:

Lo scoppio di bolle dello stesso colore: ogni bolla vale 10 punti.

Bolle fatte cadere: La prima vale 20 punti, ogni bolla addizionale raddoppia il punteggio.

Punteggi per le bolle che cadono in catena

1 - 20

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

Punti Bonus: sono assegnati in base alla velocità con cui si completa lo schema. Assegnazione punti in base al tempo con cui completi lo schema:

0-5 secondi 50, 000 . . .

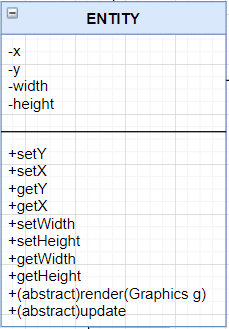
. . . 64 secondi 440

65 o più secondi 'NO BONUS'

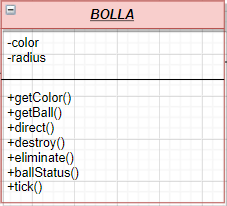
**Requisiti non funzionali**

Il gioco dovrà mantenere una fluidità per quanto riguarda il lancio delle bolle e non presentare lag.

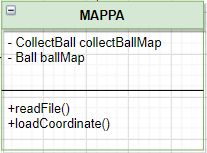
**1.2 Analisi e modello del dominio**



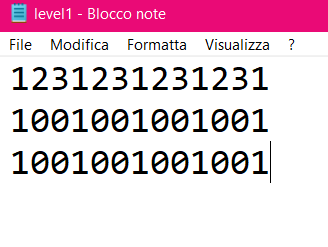
L’entità più importante del gioco è ***“Entity”***, questa entità definisce tutti i tipi di elementi presenti nel gioco definendone la grandezza, la posizione e permettendone un update costante e una renderizzazione grafica. Entity verrà utilizzata nel programma come punto di partenza per la creazione del cannone, delle bolle e della mappa.



L’entità ***“Bolla”*** prevede la definizione di un colore e di una dimensione specificata dal raggio. Il colore viene scelto in modo randomico tramite una semplice funzione che usa la libreria random. Il raggio è utile per caricare le bolle sulla mappa e definirne distanza e collisione. Ogni bolla possiede una funzione getColor() che permette di ricavare il colore della bolla presa in considerazione utile per capire che tipo di bolla viene lanciata dal cannone e per 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 funzione destroy() e attraverso la funzione eliminate() verrà cancellata e sostituita da una bolla che d’ora in poi verrà considerata una bolla della mappa tramite la funzione ballStatus() (da bolla sospesa a bolla mappa). La funzione tick() permette un continuo aggiornamento sulle collisioni in x e y dell’entità. In particolare viene controllata la collisione con altre bolle e con il cannone stesso in caso di game over.



Ogni mappa viene creata attraverso la lettura di un file txt (esempio immagine 1.4)

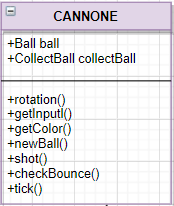


(1.4)

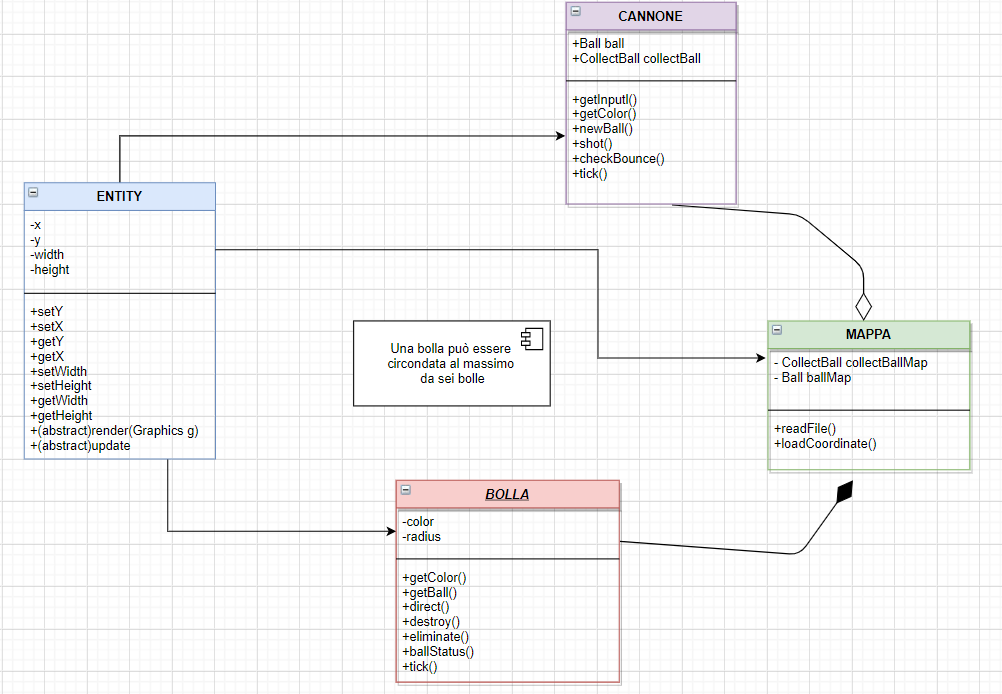
A ogni colore corrisponde un numero:

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

La lettura del file txt e suddivisa in due parti. La prima parte è la lettura del colore che ci permette di associare ad un numero un entità bolla di colore corrispondente (new Ball (..., Color = 1,2,3,4….)). La seconda parte è quella che ci permette di disporre le bolle sulla mappa in determinate coordinate. Ogni bolla e posizionata orizzontalmente fino al raggiungimento di massimo 8 bolle sulla stessa riga, una volta raggiunto il limite le bolle vengono inserite nella linea successiva a scendere fino a 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 chiamata collectBallMap utile ad una lettura successiva della mappa.

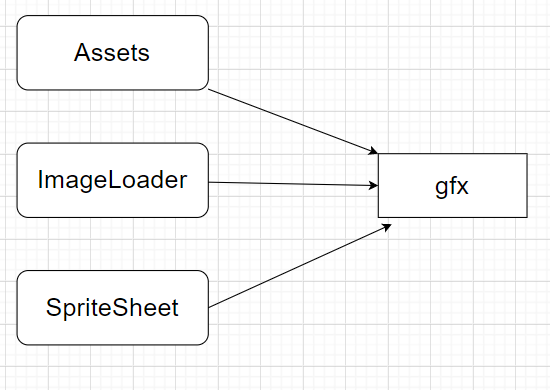


Il cannone è l’entità che permette effettivamente al player di giocare. Si muove in maniera costante da destra a sinistra e al suo centro è presente una bolla di colore casuale che può essere lanciata sull’asse Y. La difficoltà del gioco sta nel lanciare la pallina nel punto preferito dal giocatore nonostante 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. Con la funzione getBounce() si controlla la collisione della base del cannone contro i lati della mappa di gioco in modo che il cannone possa muoversi avanti e indietro.



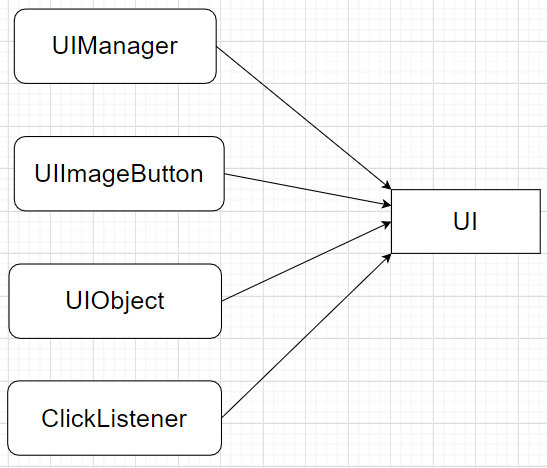
**Design**

GFX



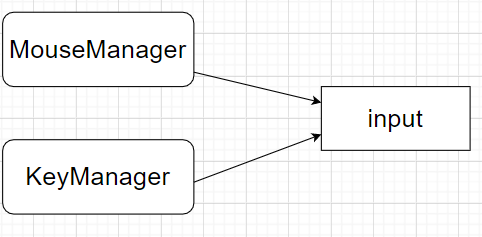
Sfruttando la libreria java.gfx ci occupiamo della renderizzazione delle immagini e delle loro animazioni. In Assets sono presenti i riferimenti assoluti delle immagini dei bottoni, del cannone, delle bolle e altre immagini della schermata prinicipale. Le immagini delle bolle sono caricate su di un vettore dove per ogni posizione è presente il percorso assoluto dell’immagine. La stessa cosa viene fatta per i bottoni in modo che alla loro pressione l’immagine diventi leggermente più scura. ImageLoader si occupa del BufferedImage attraverso java.awt. L’ultimo elemento del gfx è lo SpriteSheet ovvero la codificazione delle animazioni del gioco. Gli spritesheets vengono utilizzati per accelerare il processo di visualizzazione delle immagini sullo schermo; È molto più veloce recuperare un'immagine e visualizzare solo una parte di quell'immagine piuttosto che recuperare molte immagini e visualizzarle

UI



La parte di user interface è gestita da quattro entità. L’ UIManager è il più importante di tutti, attraverso la libreria java.awt ci permette di creare object e attraverso un handler di gestirlo. Per i bottoni presenti sulle varie schermati esiste un interfaccia UIImageButton che ci consente di dare caratteristiche comuni ad ogni bottone quali la dimensione, la forma e la gestione onClick(). Il ClickListener è essenziale per la ricezione degli input da tastiera dati dall’utenti. Gli input verranno elaborati e sfruttati dall’utente per giocare. I comandi utili sono: click del mouse, p (pausa) e esc (per uscire).

INPUT



MouseManager 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.

KeyManager è responsabile della ricezione in input dei ???

**2.1 Architettura**

Il pattern architetturale di SpaccaBolle è un classico MVC. Le view permettono al gioco di assumere stati diversi e dargli una rappresentazione grafica. Ognuna di esse possiede un controller che ne permette la gestione e la manipolazione. La prima view visualizzata sarà quella della schermata principale (StatoMenu). 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 per verificare il corretto funzionamento di SpaccaBolle.

• **TestCannon**

Questo test permette di verificare il corretto funzionamento del cannone, in particolare ne verifica la posizione durante il movimento costante da destra a sinistra, e se effettivamente genera nuove bolle

• **TestBall**

Questo test permette di verificare la corretta creazione delle bolle, controllare che le posizioni siano allocate in modo corretto e 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.

• **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 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.

Per poter lavorare in contemporanea sul progetto abbiamo utilizzato GitHub attraverso la creazione di branch personali.

I punti critici del progetto sono stati gestiti in collaborazione in modo da ottenere il migliore risultato possibile.

**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

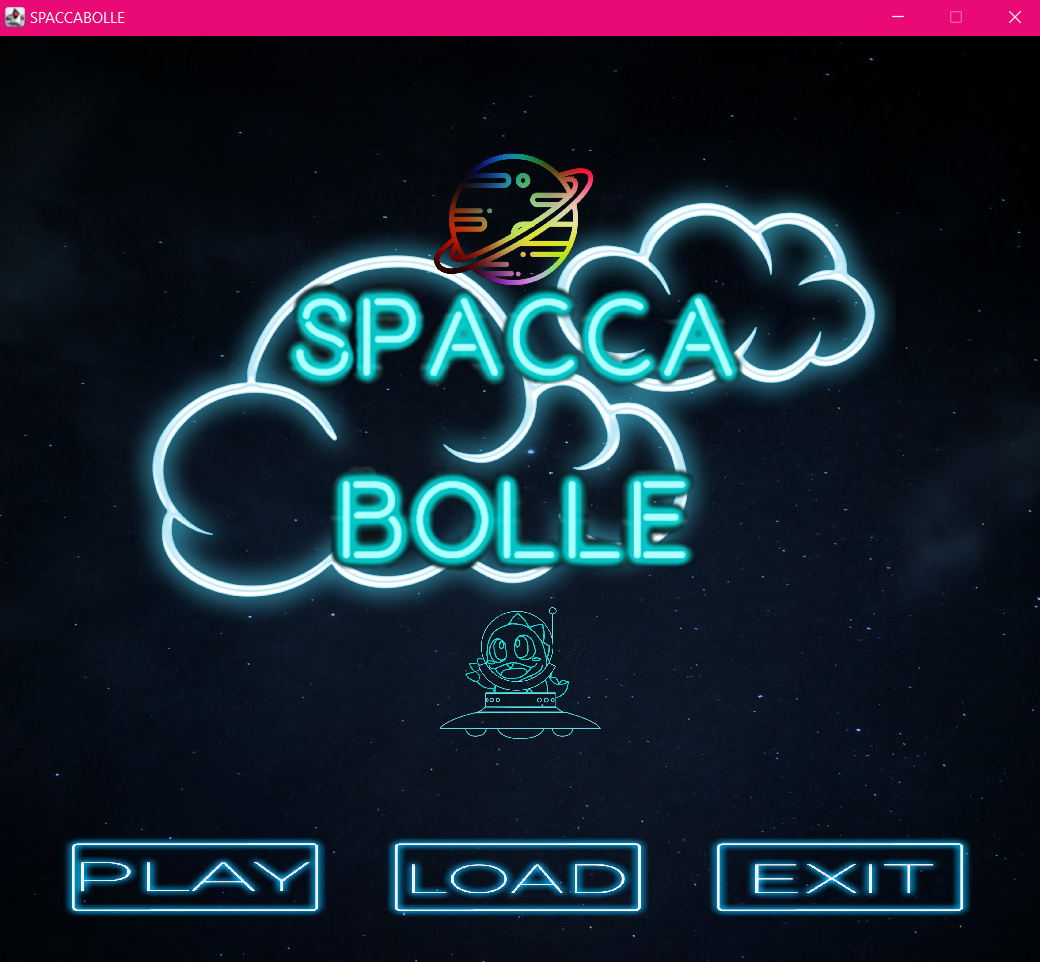
**Mattia Gullotto**

* Algoritmo Bolle sulla mappa
* Grafica Bolle della Mappa

**GUIDA UTENTE**

Di seguito un breve guida al gioco SpaccaBolle

**Schermata principale**



Col mouse cliccare uno dei bottoni per: iniziare il gioco, caricare una precedente partita oppure uscire dall’applicazione.

**Schermata di gioco** da cambiare

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

Schermata finale/pausa (non esiste ancora)