Programmazione ad Oggetti

Relazione progetto

"SpaccaBolle"

Sofia Bagagli
Yuri Collini
Michele Nardini
Elisa Simoni

Indice

I. Analisi

II. Design

III. Sviluppo

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.

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 livelli di gioco
- Caricamento livello di gioco precedentemente salvato

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 (non disponibile nella versione demo)
- chiudere l'applicazione

Una volta premuto play, all'utente verrà mostrata una schermata di gioco composta da una mappa contenente bolle di diverso colore, disposte sulla mappa di gioco, un cannone spara bolle che si muove in maniera costante da destra a sinistra (con velocità di movimento variabile tramite le impostazioni). 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 almeno un 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 ed 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.

Si noti che si è deciso che le bolle che scoppiano in orizzontale prevalgono sulle bolle che scoppiano in verticale.

Schema punteggi per bolle che cadono in catena

1 – 20 (Prima bolla caduta)

2 – 40

3 – 80

4 – 160

5 – 320

6 – 640

7 – 1280

8 – 2560

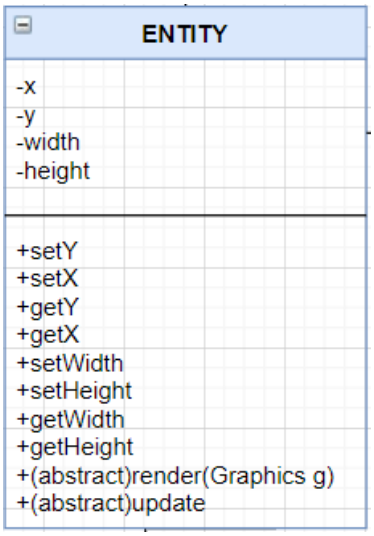
9 – 5120

La gestione del punteggio delle bolle cadono a catena si basano su di una sommatoria di potenze di due: $\sum_{i=0}^{n} 2^i$. Il punteggio massimo raggiungibile è 9999

Requisiti non funzionali

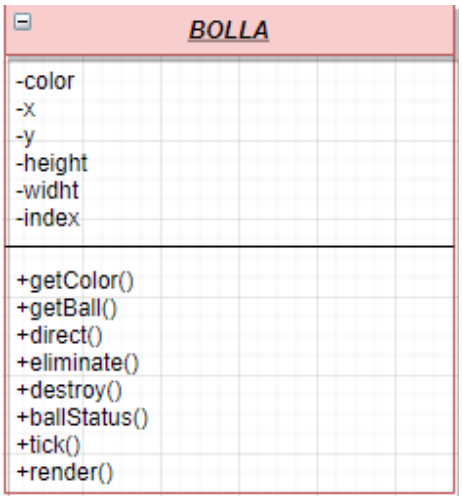
Il gioco dovrà mantenere una fluidità per quanto riguarda il lancio delle bolle e non presentare alcun tipo di lag. Inoltre il codice prodotto dovrà essere il più pulito possibile e facilmente comprensibile.

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.

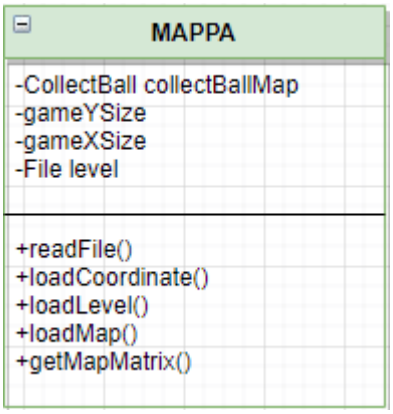
1.2.1. ENTITA' BOLLA



L'entità **"Bolla"** implementata prevede la definizione di un colore e di una dimensione specificata dal raggio. Il colore della bolla viene scelto in

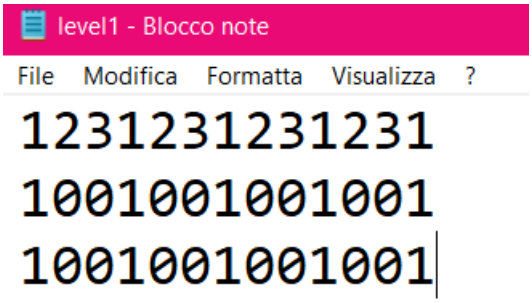
modo casuale tramite una semplice funzione che usa la libreria random. Il raggio della bolla è utile per effettuare il caricamento delle bolle sulla mappa (algoritmo che verrà discusso in seguito) , definirne distanza e possibili collisioni in fase di gioco. Ogni bolla creata, possiede una funzione denominata ***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.

1.2.2. ENTITA' 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. Ogni livello deve contenere 9 righe e 13 colonne di cui soltanto 11 potranno essere usate. (vengono escluse a fini

grafici la prima colonna e l'ultima, infatti non sono raggiungibili dal cannone)



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

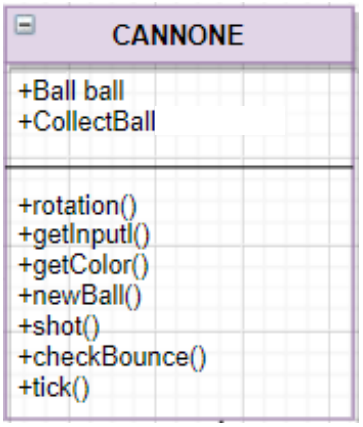
Per caricare le bolle all'interno della mappa occorre effettuare la lettura del file .txt. Per fare ciò si è deciso di suddividere l'operazione di lettura in due parti. La prima parte di questo processo è la lettura del colore, che ci permette di associare ad un numero un'entità bolla del medesimo colore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

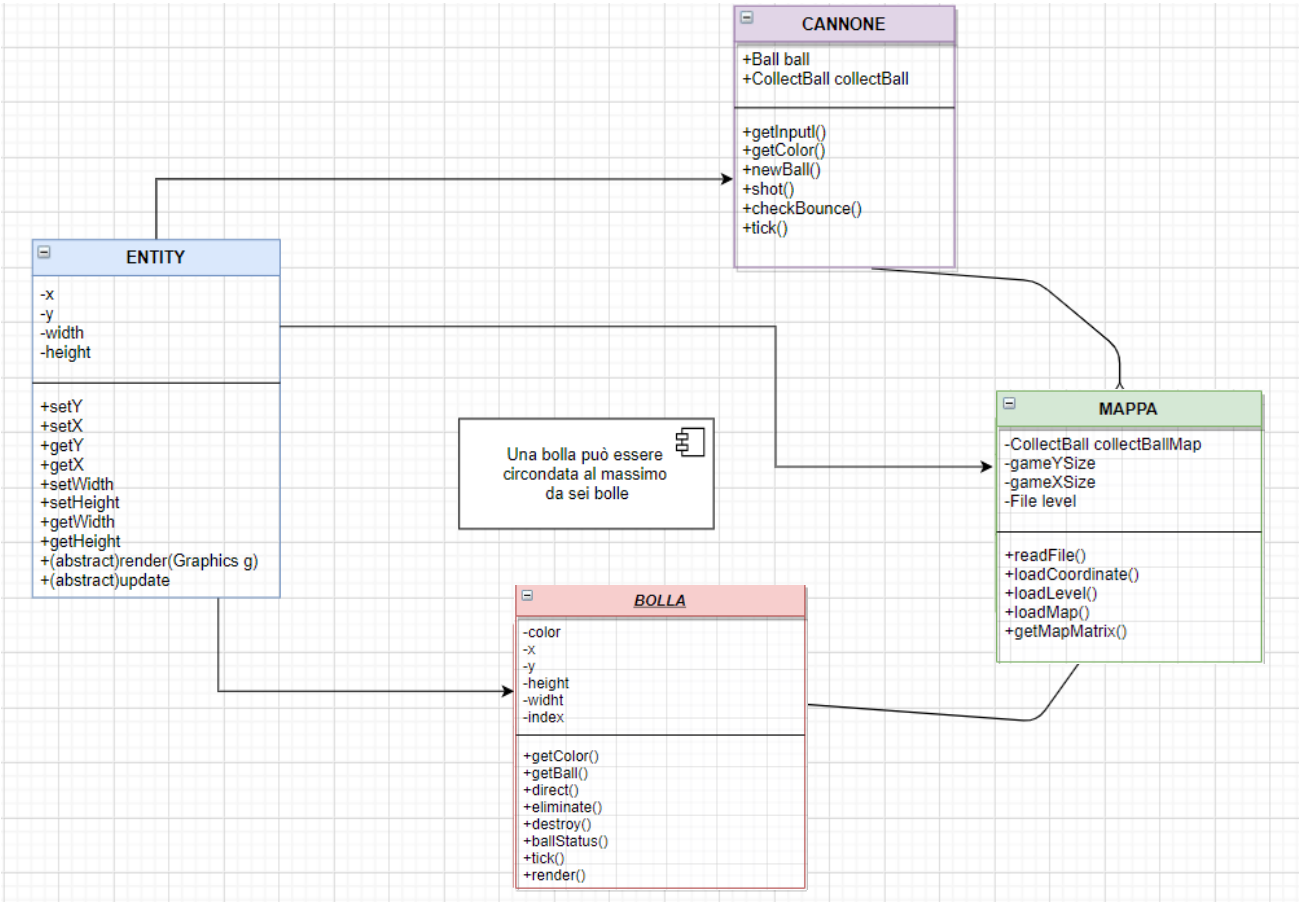
1.2.3. ENITA' CANNONE

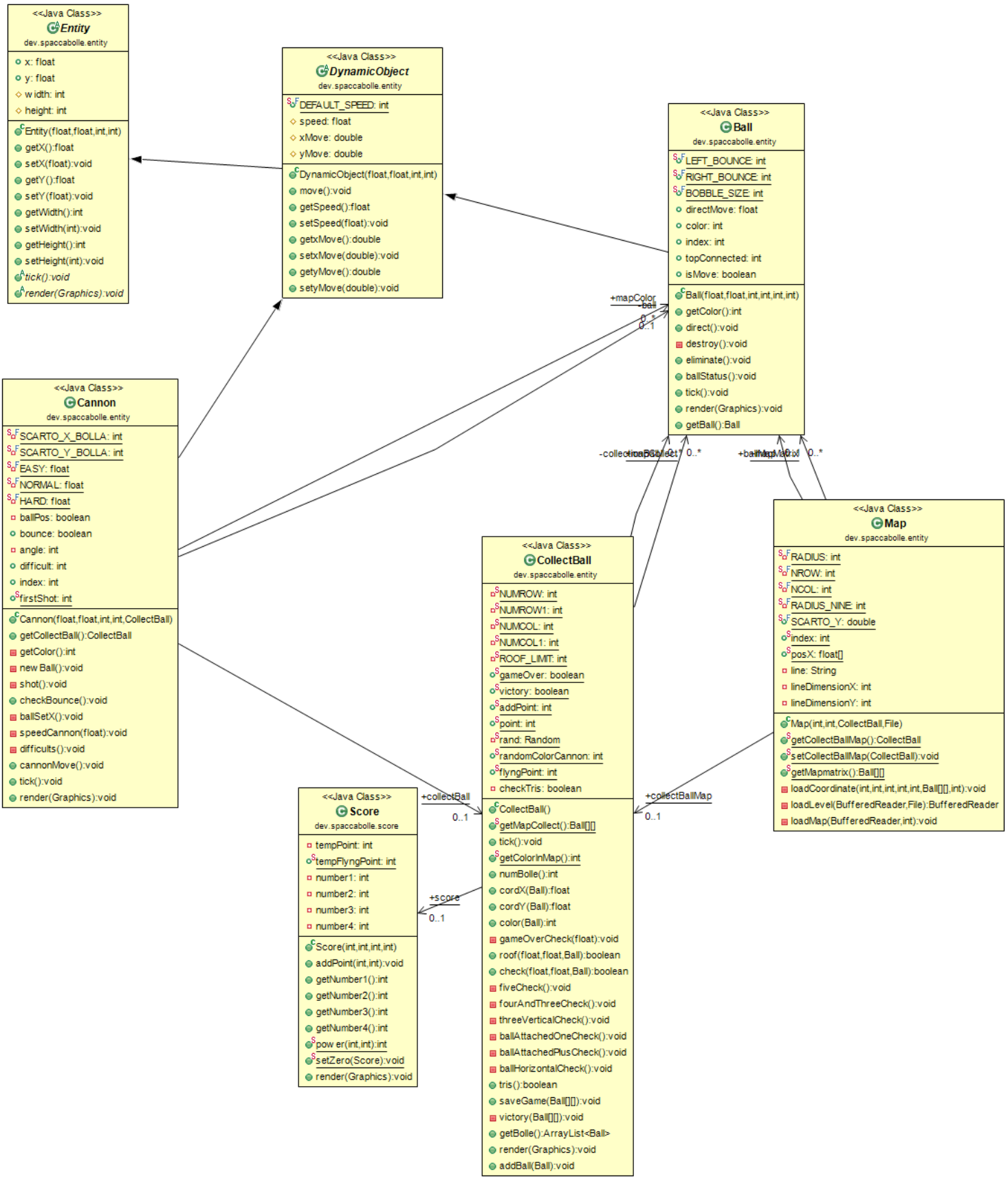
Il cannone è l'entità che permette effettivamente al player di giocare. Il cannone si muove in maniera costante lungo un'asse orizzontale da destra a sinistra. Al centro del cannone è presente una bolla di colore casuale che può essere lanciata sull'asse Y.



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 6ofps 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.

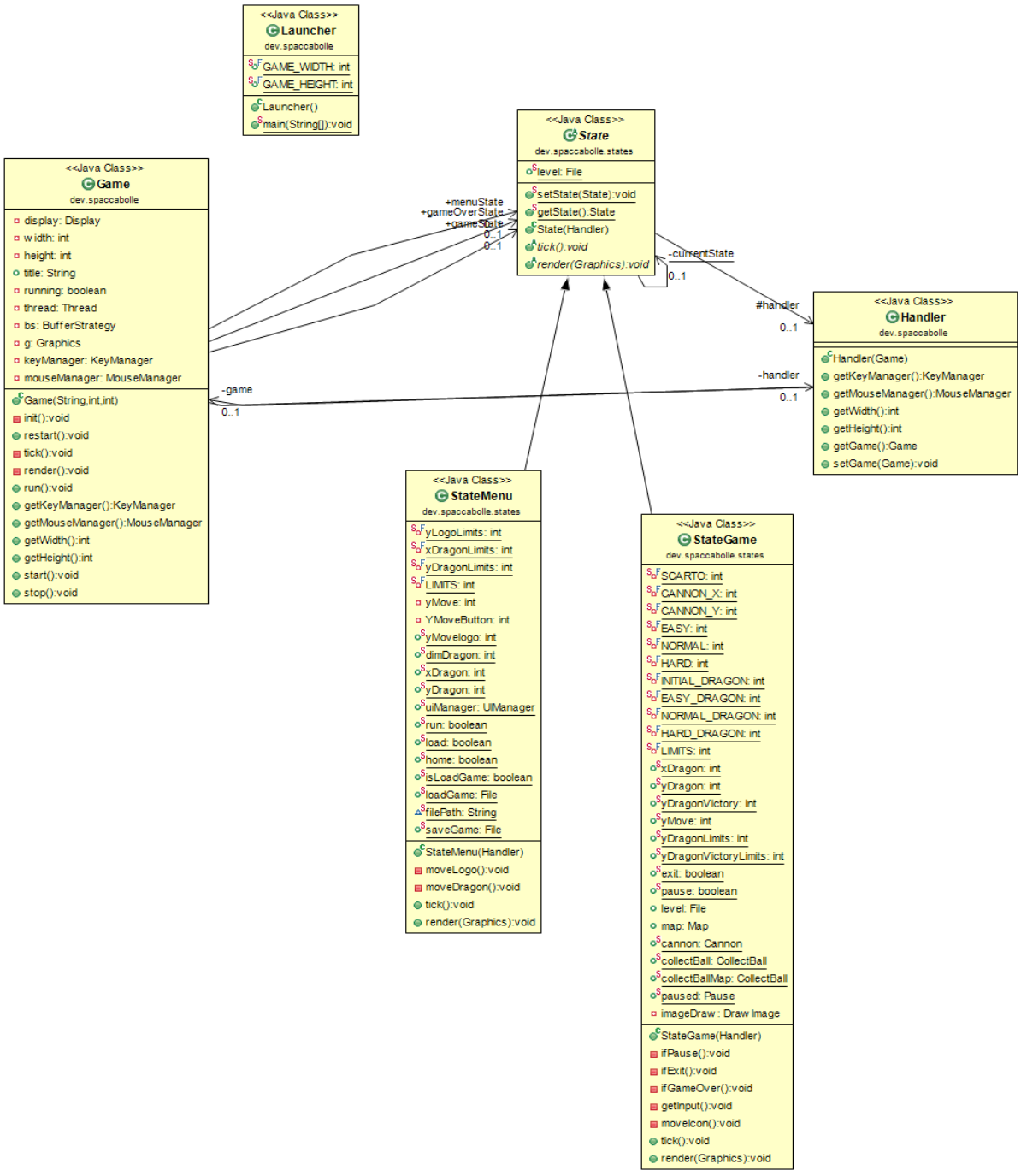



<<Java Class>>
Entity
dev.spaccabolle.entity
x: float
y: float
width: int
height: int
Entity(float,float,int,int)
getX():float
setX(float):void
getY():float
setY(float):void
getWidth():int
setWidth(int):void
getHeight():int
setHeight(int):void
tick():void
render(Graphics):void
<<Java Class>>
DynamicObject
dev.spaccabolle.entity
DEFAULT_SPEED: int
speed: float
xMove: double
yMove: double
DynamicObject(float,float,int,int)
move():void
getSpeed():float
setSpeed(float):void
getxMove():double
setxMove(double):void
getyMove():double
setyMove(double):void
<<Java Class>>
Ball
dev.spaccabolle.entity
LEFT_BOUNCE: int
RIGHT_BOUNCE: int
BOBBLE_SIZE: int
directMove: float
color: int
index: int
topConnected: int
isMove: boolean
Ball(float,float,int,int,int,int)
getColor():int
direct():void
destroy():void
eliminate():void
ballStatus():void
tick():void
render(Graphics):void
getBall():Ball
+mapColor
ball
0..1
<<Java Class>>
Cannon
dev.spaccabolle.entity
SCARTO_X_BOLLA: int
SCARTO_Y_BOLLA: int
EASY: float
NORMAL: float
HARD: float
ballPos: boolean
bounce: boolean
angle: int
difficult: int
index: int
firstShot: int
Cannon(float,float,int,int,CollectBall)
getCollectBall():CollectBall
getColor():int
new Ball():void
shot():void
checkBounce():void
ballSetX():void
speedCannon(float):void
difficults():void
cannonMove():void
tick():void
render(Graphics):void
0..*
0..*
<<Java Class>>
CollectBall
dev.spaccabolle.entity
NUMROW: int
NUMROW1: int
NUMCOL: int
NUMCOL1: int
ROOF_LIMIT: int
gameOver: boolean
victory: boolean
addPoint: int
point: int
rand: Random
randomColorCannon: int
flyngPoint: int
checkTris: boolean
CollectBall()
getMapCollect():Ball[][]
tick():void
getColorInMap():int
numBolle():int
cordX(Ball):float
cordY(Ball):float
color(Ball):int
gameOverCheck(float):void
roof(float,float,Ball):boolean
check(float,float,Ball):boolean
fiveCheck():void
fourAndThreeCheck():void
threeVerticalCheck():void
ballAttachedOneCheck():void
ballAttachedPlusCheck():void
ballHorizontalCheck():void
tris():boolean
saveGame(Ball[][]):void
victory(Ball[][]):void
getBolle():ArrayList<Ball>
render(Graphics):void
addBall(Ball):void
<<Java Class>>
Map
dev.spaccabolle.entity
RADIUS: int
NROW: int
NCOL: int
RADIUS_NINE: int
SCARTO_Y: double
index: int
posX: float[]
line: String
lineDimensionX: int
lineDimensionY: int
Map(int,int,CollectBall,File)
getCollectBallMap():CollectBall
setCollectBallMap(CollectBall):void
getMapmatrix():Ball[][]
loadCoordinate(int,int,int,int,int,Ball[][],int):void
loadLevel(BufferedReader,File):BufferedReader
loadMap(BufferedReader,int):void
<<Java Class>>
Score
dev.spaccabolle.score
tempPoint: int
tempFlyngPoint: int
number1: int
number2: int
number3: int
number4: int
Score(int,int,int,int)
addPoint(int,int):void
getNumber1():int
getNumber2():int
getNumber3():int
getNumber4():int
power(int,int):int
setZero(Score):void
render(Graphics):void
+collectBall
0..1
+score
0..1
+collectBallMap
0..1

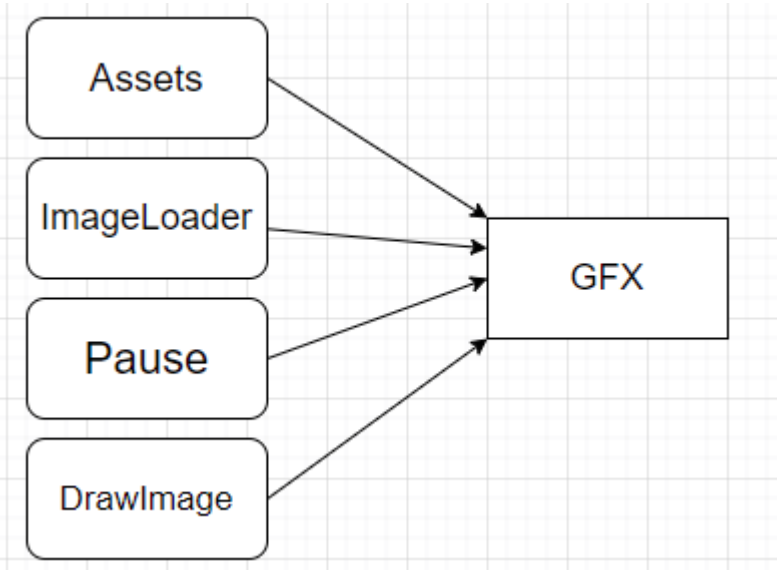
Design

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***.



2.1.1 Yuri Collini



Sfruttando la libreria ***java.gfx*** ci occupiamo della renderizzazione di tutte le immagini nonché delle loro animazioni. Nella classe ***Assets***, presente all'interno del package ***dev.spaccabolle.gfx***, sono presenti i riferimenti assoluti delle immagini dei vari bottoni presenti nel menù, del cannone, delle bolle e di tutte le altre immagini della schermata principale di gioco nonché di quella di benvenuto.

Le immagini delle bolle sono caricate su di un vettore di tipo ***BufferedImage*** dove, per ogni posizione è presente il percorso assoluto dell'immagine. Stessa cosa vale per i bottoni START, EXIT, LOAD. Per questi bottoni sono disponibili due tipologie di immagini. Il primo tipo è l'immagine vista appena viene caricata la schermata mentre il secondo tipo è la stessa immagine con una colorazione più scura. Questo secondo tipo di immagine viene mostrata ogni qual volta il mouse è sopra al bottone.

Nella classe Assets, per evitare magic number sono state utilizzati variabili di tipo private static final in modo tale da rendere il codice prodotto più sicuro e facilmente mantenibile. La classe ***ImageLoader***, anch'essa è presente all'interno del package ***dev.spaccabolle.gfx***, si occupa del buffering di tutte le immagini presenti sul gioco tramite ***BufferedImage*** attraverso l'utilizzo di ***java.awt*** .

2.1.2 Sofia Bagagli

La casse ***DrawImage*** è una classe utilizzata per poter disegnare sul gioco vari componenti durante l'esecuzione di esso, mentre si trova nello stato game.

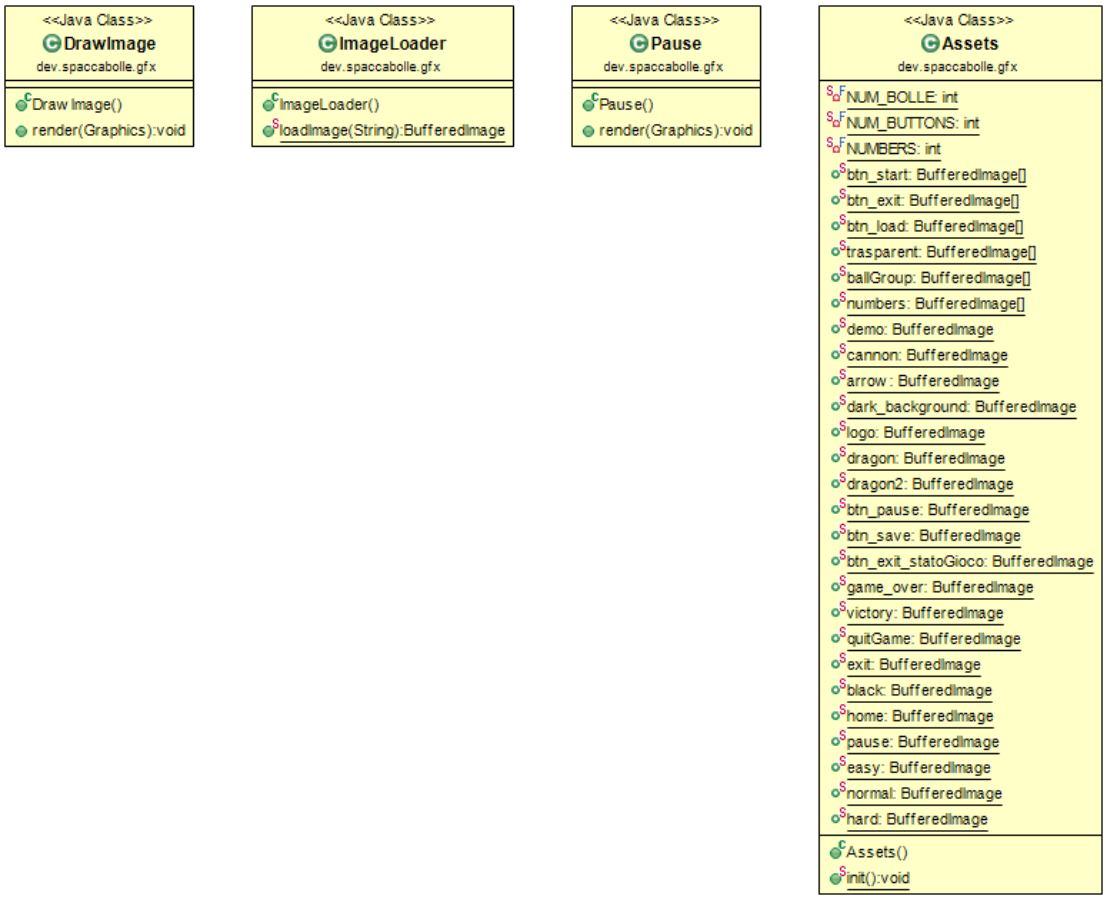
Tramite questa classe è possibile visualizzare nello stato gioco tutti i suoi componenti come il background, la mappa del gioco e le relative bolle, il cannone, i pulsanti per poter salvare la partita, per metterla in pausa e per poter uscire.

Oltre a questi componenti grafici di base, è possibile visualizzarne altri quando determinate condizioni si verificano.

Ovvero quando il gioco viene messo in stato di pausa viene visualizzato a schermo un menù con il quale l'utente può interagire, oppure quando si vuole uscire dal gioco viene visualizzata un popup in cui viene chiesta una conferma sul fatto se si è sicuri di voler abbandonare il gioco oppure no.

Quando si verifica la condizione di vittoria anche in quel caso viene mostrata un'immagine nella quale si informa l'utente di premere il tasto "E" per uscire e chiudere il gioco. In fine quando un utente perde viene mostrato un pop-up nel quale il giocatore può scegliere se ricominciare oppure terminare.

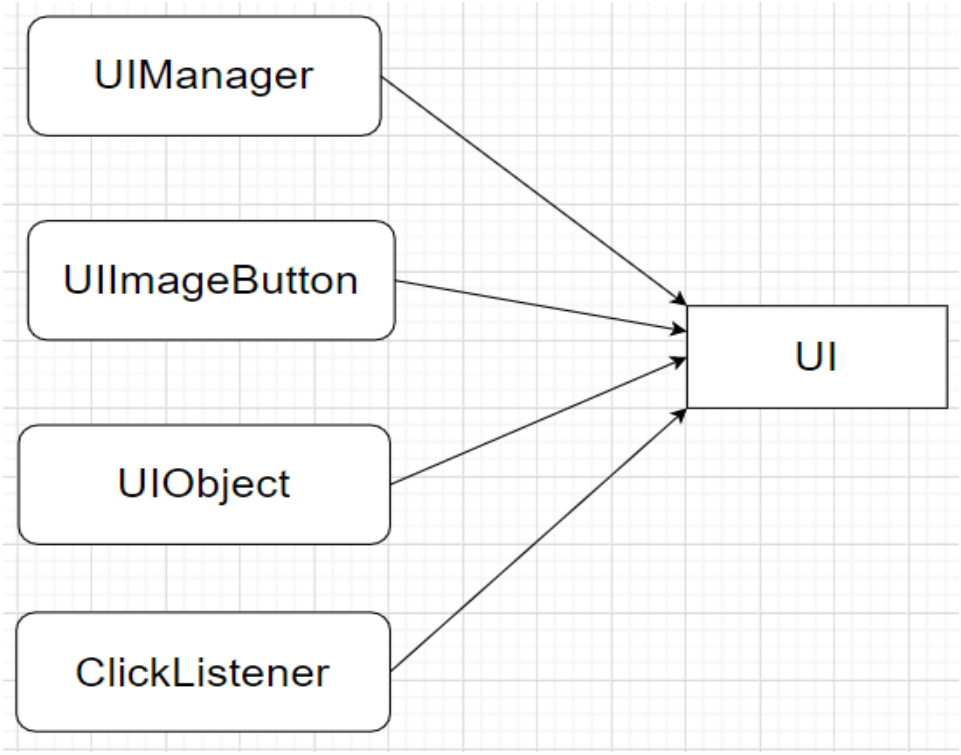
La classe Pause è una classe utilizzata per la realizzazione del menu di pausa, richiamata dalla classe DrawImage quando necessaria, che consiste nel visualizzare la scelta delle tre difficoltà di gioco disponibili, di poter tornare alla schermata iniziale e quindi ricominciare il gioco, di poter riprendere la partita o di terminare la partita.



NOTA AGGIUNTIVA

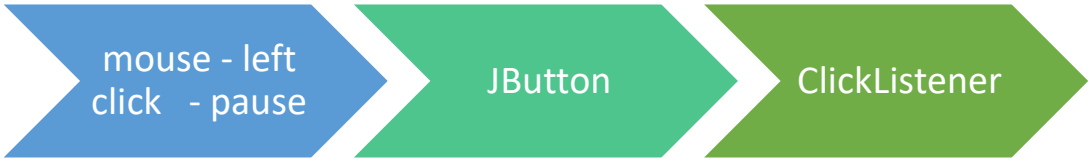
Tutta la grafica presente nel videogioco è stata interamente realizzata dal gruppo mediante l'ausilio di alcuni strumenti come ad esempio photoshop e canva.com

2.1.3 Elisa Simoni



Le interazioni con la GUI sono gestite tramite eventi. Per gestire questi eventi abbiamo usato il sistema più classico.

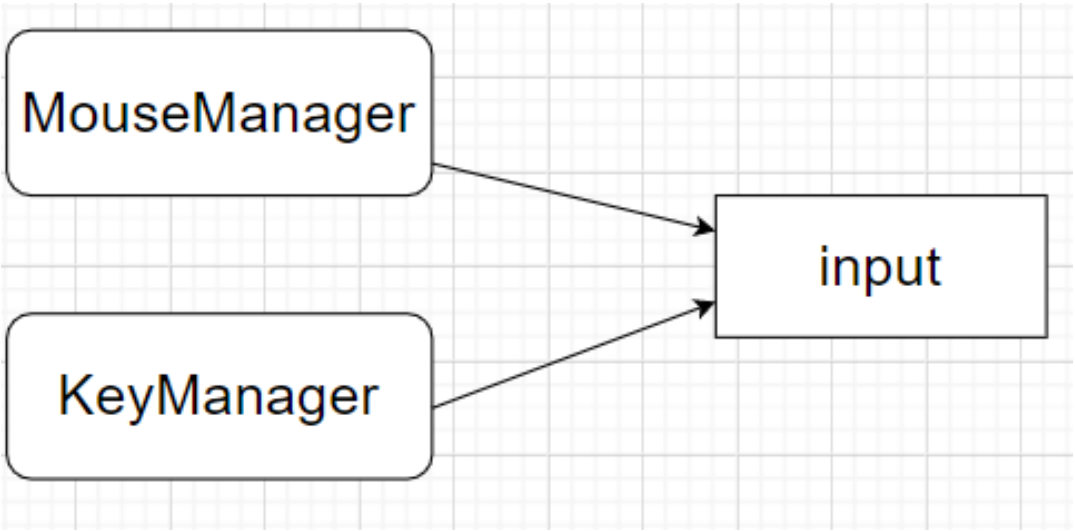
- Viene istanziato l'evento
- Un componente della GUI diventa sorgente dell'evento (MouseManager, KeyManager)
- E un listener diventa la destinazione finale dell'evento (ClickListener)



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:

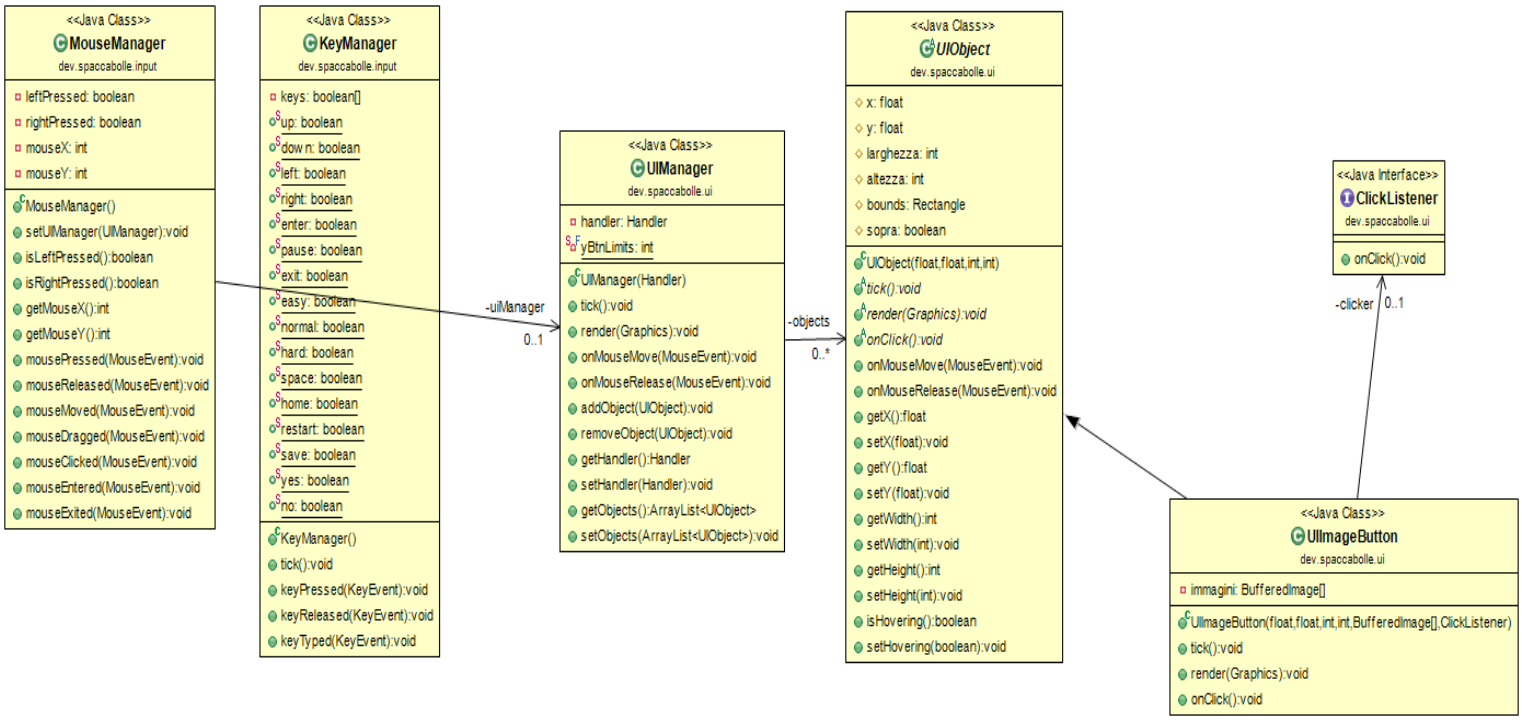
- click del mouse
- P (pausa)
- ESC (per uscire)
- S (salva)
- Y (yes)
- N (No)
- E (Exit)



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



2.1.4 Michele Nardini

Per la creazione degli oggetti di gioco abbiamo usato le classi Cannon e Ball. Tali classi ereditano i metodi della classe astratta DynamicObject contente tutti i metodo che permettono il movimento dinamico degli oggetti sulla mappa.

Utilizzeranno un metodo tick() per la costante esecuzione di metodi specifici quali per il cannone:

- **shot()** = spara la bolla posizionate

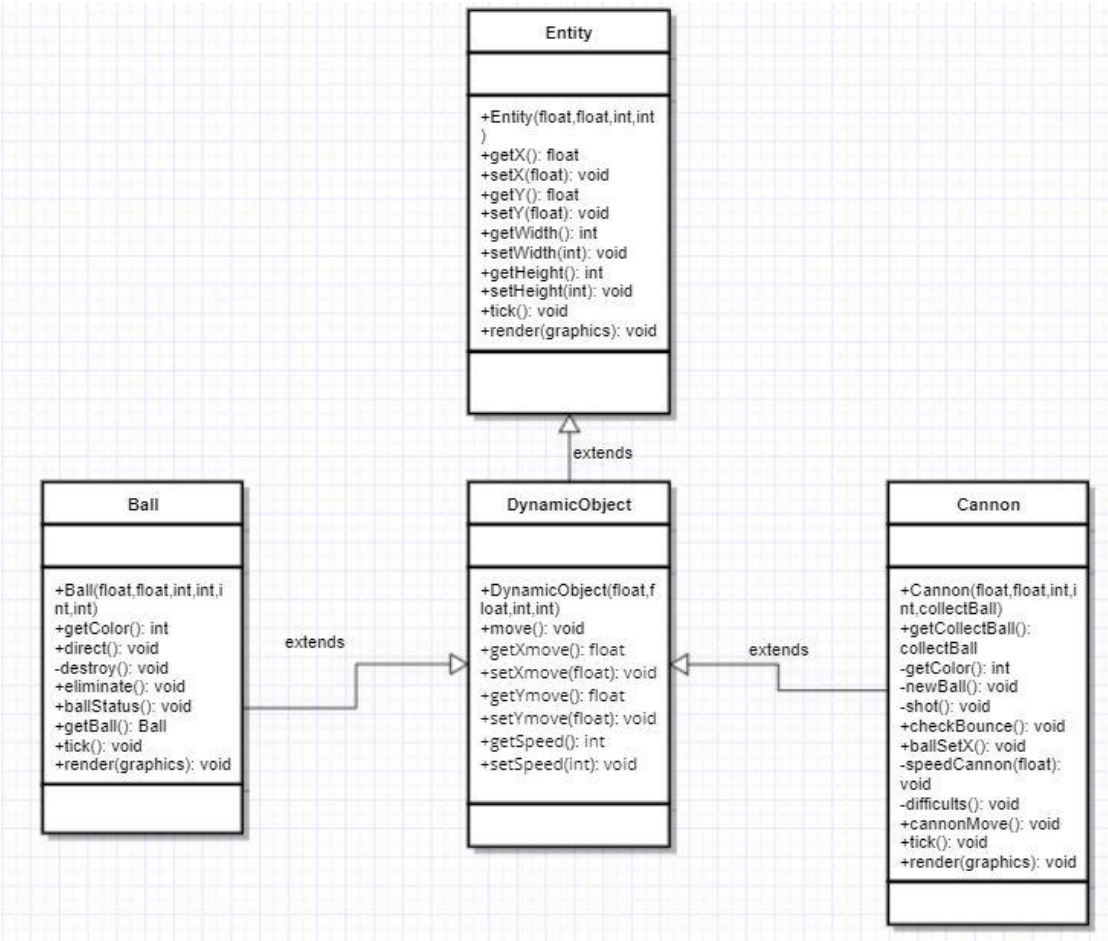
- **cannonMove()** = provvede al movimento del cannone e alla sua velocità

- **checkBounch()** = controlla quando invertire la direzione del cannone

- **newBall()** = provvede alla creazione di una nuova bolla da sparare a determinate condizioni

Per la bolla ci sono continui controlli una volta sparata per verificare quando quest'ultima dovrà fermarsi per essere posizionata correttamente della mappa evitando di sovrapporsi con le altre bolle.

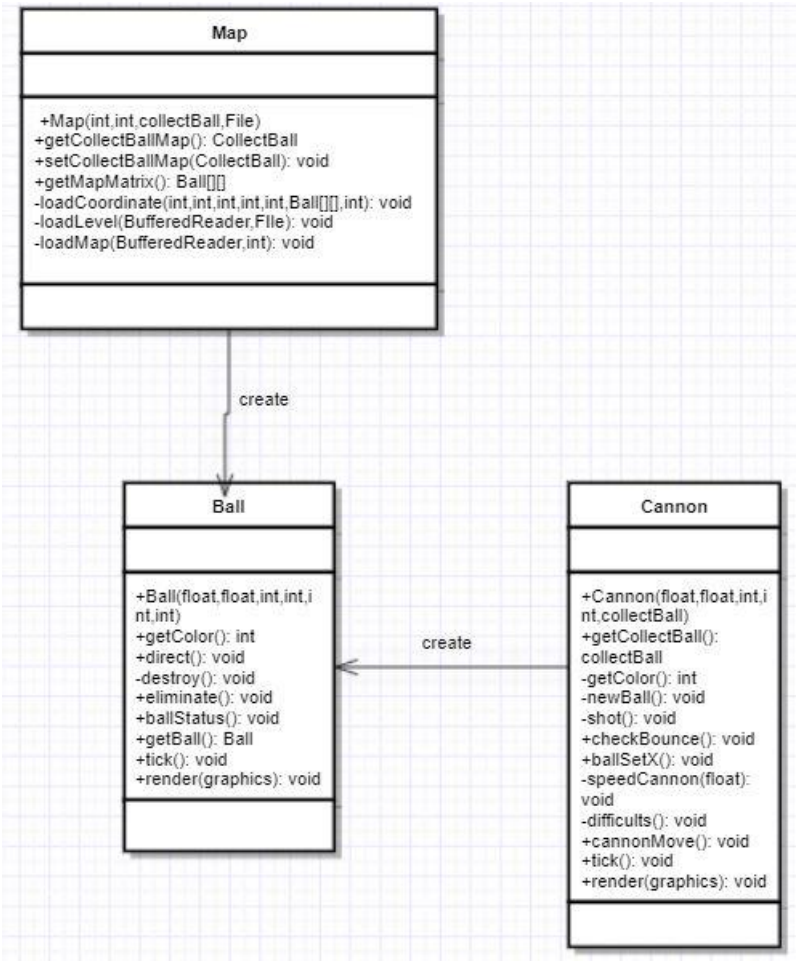
Entrambi gli oggetti verranno renderizzati da un metodo render(graphics).



DynamicObject è stato esteso dalle seguenti classi:

• Cannon, oggetto che servirà per far muovere le bolle e per la loro stessa creazione

• Ball, oggetti che il giocatore dovrà cercare di eliminare mediante la formazione di coppie vicine dello stesso colore formate da 3 o + bolle



Le classi incaricate alla creazione degli oggetti Ball saranno Cannon e Map. Cannon, classe che viene istanziata dalla classe StateGame, creerà una bolla ogni qualvolta lo spazio nel cannone sarà libero e la bolla sparata in precedenza sarà stata posizionata nella mappa.

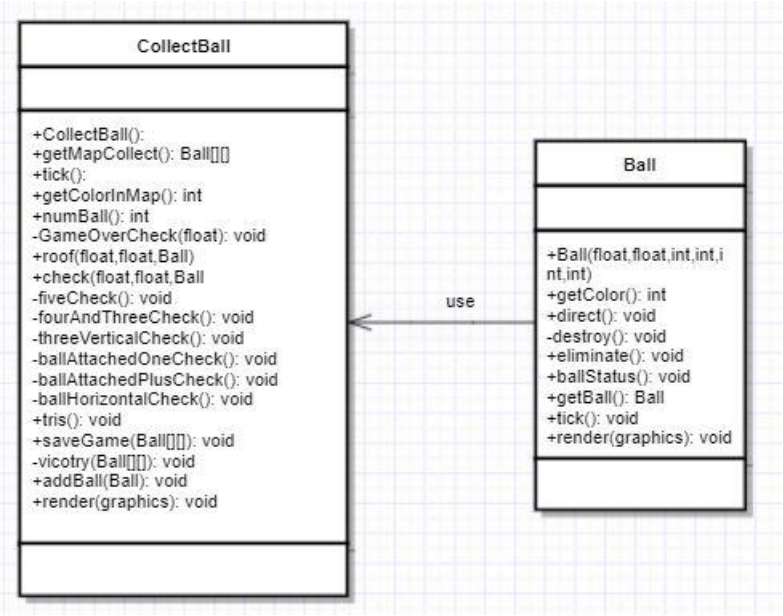
La classe Map sarà la classe che definirà la mappa di gioco tramite l'utilizzo di una matrice bidimensionale che andrà a definire le posizioni delle bolle nella mappa.

Tale classe verrà istanziata tramite la classe StateGame, che caricherà oltre alle dimensioni della mappa un livello di default che la classe

leggerà sfruttando il metodo loadLevel() e che permetterà leggendo i singoli valori numerici contenuti nel file di posizionare le bolle di un determinato colore nella posizione prescelta nella matrice.

Un metodo per scegliere il livello al posto di caricarne uno di default sarà utilizzare un tasto load nel menu di gioco che permetterà all'utente di scegliere uno dei file contenuti in locale e caricarne il contenuto.

La classe Map così leggendo il file riga per riga creerà le singole bolle che inserirà nella propria matrice bidimensionale e nella matrice CollectBall contente tutte le bolle create.

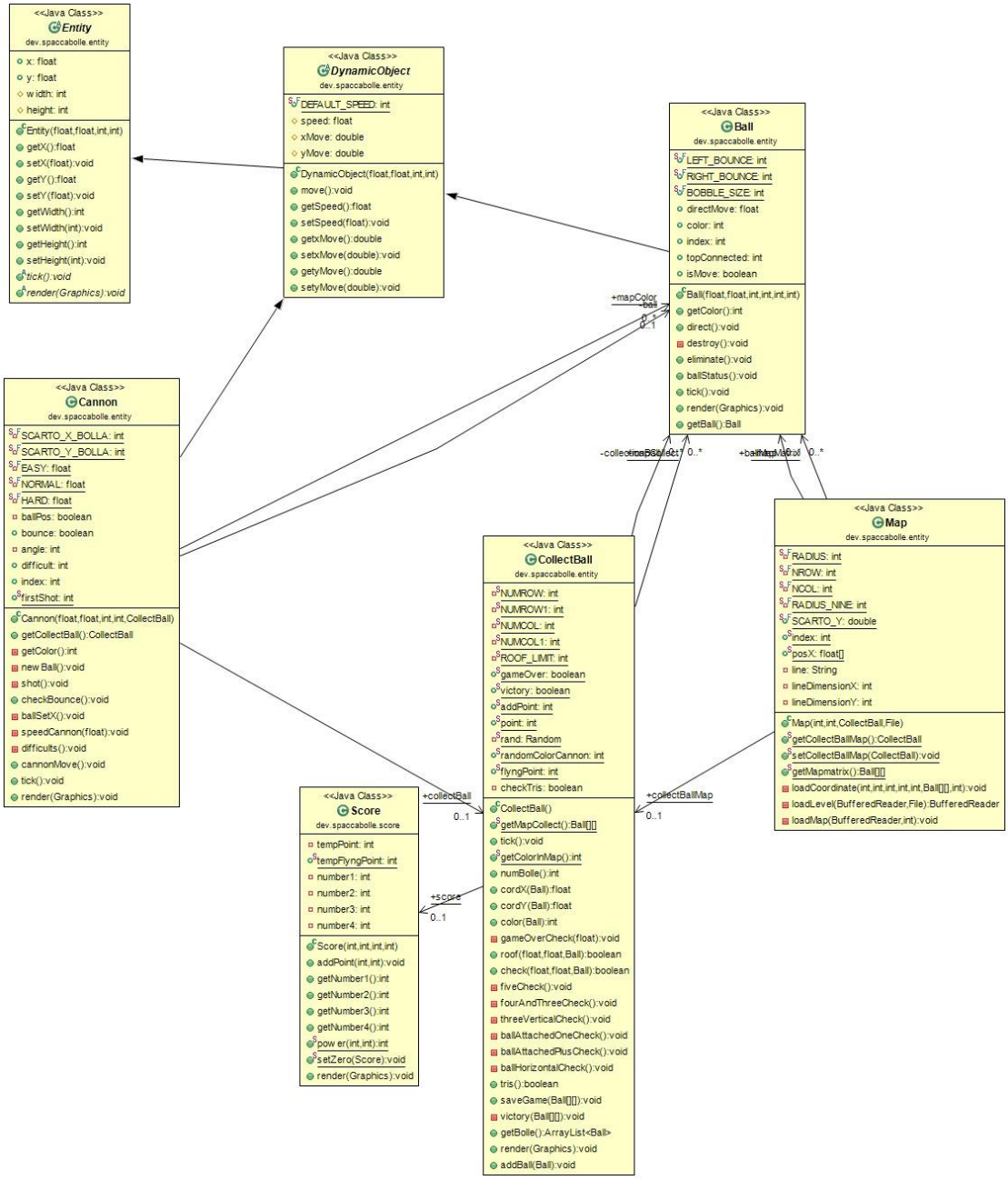


La classe CollectBall, anche essa istanziata da StateGame verrà sfruttata da Ball che la utilizzerà per effettuare tutti i controlli affinché la bolla sparata non viaggi all'infinito. Con il metodo roof() e check() verranno controllate le collisioni della bolla in movimento con il soffitto della mappa e con le altre bolle in modo tale da fermarla nel momento opportuno e salvarla nella matrice mappa nella posizione adeguata.

Oltre a ciò controllerà quando una bolla dovrà scoppiare tramite il metodo tris() e quelli richiamati da quest'ultima che farà in modo che al formarsi di coppie da 3 o più bolle dello stesso colore queste poi esplodano ed inoltre verificherà che non ci siano bolle sospese in aria senza alcun appoggio in tal caso, se presenti, queste bolle "esplodono".

La classe è anche provvista di un metodo saveGame() che permetterà il salvataggio automatico del livello in un file save.txt dove verranno salvati i valori della matrice collectBallMap che serviranno per ricostruire la mappa nel caso si voglia caricare tramite il bottone Load il livello salvato.

Schema generale oggetti di gioco



Materiale Pubblico Utilizzato & Fonti di Ispirazione :

- StackOverflow
  https://stackoverflow.com/
- Video su YouTube
  https://www.youtube.com/watch?v=dEKs-3GhVKQ&list=PLah6faXAgguMnTBs3JnEJY0shAc18XYQZ
- Documentazione di Java
  https://docs.oracle.com/javase/7/docs/api/

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, di controllare l'assegnazione del corretto colore, controlla le coordinate x e y della bolla, e infine ne controlla la sua eliminazione.

• TestScore

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.

• TestCollectBall

Questo test permette di verificare le corretta generazione dell'ArrayList CollectBall, controlla che la dimensione dell'arrayList sia corretta, controlla le caratteristiche delle bolle una volta inserite all'interno del gioco, e in fine controlla l'aggiunta di una nuova bolla nell'arrayList.

• TestGame

Questo test permette di verificare il corretto caricamento del gioco, verifica il corretto funzionamento degli input e controlla la lunghezza e la larghezza del gioco.

- TestMap

Questo test permette di verificare la corretta generazione della mappa, controlla il corretto caricamento della bolla nella mappa e attraverso il test testGetMapMatrix ci assicuriamo che essa possa essere esportata.

- TestState

Questo test permette di verificare la corretta assegnazione dello stato al gioco.

3.2 Metodologia di lavoro

L'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, si noti che a volte per problemi tecnici il codice veniva condiviso tramite email ed altre tecnologie ed in seguito un membro del gruppo lo caricava su GitHub.

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.

NOTA PER IL DOCENTE

" Come comunicato anche via mail, lo studente Mattia Gullotto, per impedimenti personali sopraggiunti nell'ultimo periodo non è più riuscito a proseguire nello sviluppo del progetto e per tale motivo non fa più parte del gruppo. I suoi compiti sono stati suddivisi tra i vari membri."

I compiti sono stati così suddivisi :

3.2.1. Sofia Bagagli

- Interfaccia Grafica menu
- Gestione e grafica Cannone
- Grafica Bolle della Mappa
- Test JUnit

Durante la progettazione del lavoro ho preso in carico la parte riguardante la gestione del cannone, la grafica e la gestione degli input da parte dell'utente.

All'inizio avevo progettato il cannone in modo che la freccia di esso, si muovesse entro un certo limite, seguendo il puntatore del mouse.

Per venire in contro alle esigenze del giocatore, il mio compagno di lavoro, Michele Nardini, l'aveva invece implementata tramite l'utilizzo dei tasti della tastiera AWSD, in questo modo il giocatore durante la partita avrebbe potuto scegliere in quale modalità giocare.

Tuttavia, il team ha poi pensato, per rendere il gioco più originale, di far muovere autonomamente il cannone lungo un'asse X. Ho poi pensato per rendere il gioco più difficile, e magari più interessante, da giocare tre livelli di difficoltà (Easy, Normal e Hard), dove il cannone si muove sempre in modo più veloce.

Per la parte grafica sono stata accompagnata nello sviluppo dal mio collega di progetto Yuri Collini, ho lavorato nel caricamento dell'immagini, alla loro rappresentazione a livello visivo nel gioco e alla loro vera e propria realizzazione.

Per rendere l'interfaccia utente il più User-Friendly possibile, ho cercato di rendere la grafica dell'immagini intuibile, di bell'aspetto, semplici e chiare. Dando in più al giocatore la possibilità di scegliere se interagire con il gioco tramite i pulsanti della tastiera (opportunatamente implementati con le varie funzioni da svolgere, in base al tasto premuto) oppure cliccando sopra le immagini del gioco, per poter salvare, mettere in stato di pausa, uscire dal gioco oppure ricaricare il gioco dall'inizio. Questo però è stato reso possibile solo nello stato di gioco, mentre nel menu iniziale l'utente può interagire solo tramite il click dell'immagine con il cursore del mouse. Infine ho interfacciato tutta la parte di listener

Classi su cui ho lavorato :

- Cannon
- Package Listener
- Display
- StateMenu
- StateGame

3.2.2 Yuri Collini

- Menù di navigazione
- Interfaccia di benvenuto
- Gestione Punteggio
- Pulizia del Codice

Nello sviluppo del progetto mi sono occupato della progettazione e realizzazione dell'interfaccia di benvenuto ma anche del menù di navigazione in essa contenuto. Per questo compito ho lavorato a stretto contatto con la mia collega Sofia Bagagli.
Nella fase di progettazione dell'interfaccia di benvenuto, ovvero quella che viene mostrata all'avvio del gioco, abbiamo cercato di renderla più pulita, semplice ed intuitiva possibile ma al contempo abbiamo cercato di renderla anche accattivante dal punto di vista estetico.
Per la gestione del menù di navigazione presente nella schermata di avvio, ho pensato di inserire solamente 3 bottoni, questo per rispettare le linee guida che mi ero prefissato facendo particolare attenzione alla semplicità. Per caricare i bottoni nella schermata ho lavorato sulla classe Assets, di cui è presente una spiegazione più dettagliata più in alto in questa relazione, ma anche sulle classi ImageLoader e quelle per la gestione dello stato.
Per poter integrare l'interfaccia di benvenuto, presente nella classe StateMenu ho lavorato sulla classe Game, la classe Handler e la classe Launcher.
Per la gestione dei punteggi ho collaborato con la mia collega Elisa Simoni, la quale mi ha aiutato ad integrare la classe Score da me realizzata, e presente nel package dev.spaccabolle.score, con la classe

CollectBall. La classe Score mediante l'utilizzo del metodo render() consente di mostrare a video il punteggio realizzato dall'utente aggiornato in ogni istante della partita. Per far ciò mi sono avvalso dell'utilizzo della classe Assets, precedentemente sviluppata e presente nel package dev.spaccabolle.gfs. Va fatto notare che il punteggio visualizzato a video è stato gestito mediante l'utilizzo di quattro variabili private intere, le quali rappresentano il punteggio finale scomposto in unità, decine, centinaia ed in fine migliaia. Ovviamente sono stati creati metodi getter per poter acquisire il valore di ognuna di queste variabili in qualsiasi istante.

Classi su cui ho lavorato :

- Score
- Assets
- StateMenu
- Game
- Handler
- Launcher
- ImageLoader
- State

3.2.3 Michele Nardini

- Gestione collisioni ed esplosioni
- Algoritmo Bolle sulla mappa
- Salvataggio partita
- Test JUnit

All'interno del progetto mi sono occupato di sviluppare e gestire i vari aspetti dinamici del gioco, quali la gestione del movimento del cannone e delle bolle, delle collisioni tra le bolle e del posizionamento di quest'ultime nella mappa; infine, mi sono occupato del salvataggio e del caricamento di una partita.
Per quanto concerne l'aspetto degli oggetti dinamici di gioco ho creato due classi abstract, Entity e DynamicObject, il cui scopo era gestire il

posizionamento dell'oggetto sulla finestra insieme al suo movimento verticale/orizzontale ad una data velocità. Tali classi verranno poi ereditate dalle Classi Cannon e Ball.
Il cannone avrà il compito di direzionare la bolla in una determinata colonna una volta richiamato il metodo shot(), tali bolle una volta create dal cannone vengono aggiunte in un Array di collectBall che conterrà tutte le bolle create sia ad inizio gioco durante il caricamento del livello, sia ogni volta che vengono create quest'ultime dal cannone una volta che l'ultima bolla sparata è stata posizionata nella mappa.
Insieme ad Elisa Simoni poi ho creato i metodi per il posizionamento delle bolle nella mappa; tale metodo sfrutta l'array contente tutte le bolle e di una matrice bidimensionale dove vengono salvate le bolle posizionate in una determinata riga e colonna.
Sempre insieme ad Elisa Simoni abbiamo ragionato su un metodo che permettesse di eliminare le bolle del medesimo colore che formassero coppie da 3 o + bolle, tale metodo permette in seguito di eliminare le bolle sospese in aria senza alcun tipo di attacco.
Terminato il lavoro sulle collisioni mi sono concentrato sull'aspetto del caricamento e di salvataggio della partita; per fare ciò abbiamo utilizzato un file di testo contenente una tabella di valori numerici i cui valori variavano tra 0 e 4(tali valori indicavano il colore della bolla).
La partita ha comunque un algoritmo di salvataggio automatico che memorizza ad ogni mossa la partita in un file di default, con il salvataggio invece è possibile salvare in un file creato dall'utente con il nome desiderato.
L'ultimo aspetto su cui mi sono concentrato è stata la parte di JUnit in cui ho creato dei test per il corretto funzionamento delle classi.

Classi su cui ho lavorato :

- Ball
- Cannon
- CollectBall
- DynamicObject
- Entity
- Map
- StateGame

3.2.4 Elisa Simoni

- Algoritmo Bolle sulla mappa
- Gestione collisioni ed esplosioni
- Salvataggio partita

Ho iniziato a lavorare pensando alla creazione mappa, come prima idea avevo pensato alla gestione dei livelli tramiti JSON e avevo creato una classe per il caricamento e la lettura attraverso la serializzazione. Però poi ho pensato che la lettura sarebbe stata difficoltosa in quanto avevamo di rappresentare la mappa in maniera molto semplici. Alla fine, la decisione è caduta su di un file txt rappresentate una matrice bidimensionale di numeri da 0 a 4. Ogni colore rappresentava un colore di bolla e mi è risultato poi semplice implementare attraverso matrice il sistema di collisione e scoppio delle bolle. Assieme a Michele Nardini abbiamo implementato le classi Ball e CollectBall .
La classe Ball crea l'entità più importante del gioco ed è stata creata in modo che potesse essere usata sia nel cannone che nella parte di mappa.
La classse CollectBall è stata creata per gestire ogni eccezione di lancio della bolla sulla mappa. Un'eccezione molto ostica è stata quello nel programma definita come ***roof()*** che evita alle palline di attaccarsi al soffitto. La gestione delle collisioni prevede un controllo su matrici di elementi nulli e di colori consecutivi. Per gestire la caduta di palline attaccate a quelle effettivamente da scoppiare ho proceduto attraverso un controllo a ricorsione verso l'alto che controlla che non vi siano elementi nulli al di sopra (bolle a 0 scoppiate) Dopo aver creato la logica di gioco sono passata al salvataggio della partita sfruttando le librerie di input e output stream. Il livello viene salvato costantemente durante il gioco in modo che l'utente possa avere un salvataggio di backup nonostante non salvi materialmente il livello. Insieme a Yuri Collini ho interfacciato la classe Score con quella di CollectBall in modo d'assegnare ad una variabile globale il valore adeguato.
Insieme a Sofia Bagagli abbiamo collegato il suo lavoro di parte grafica con la mia parte logica lavorando sugli Stati, InputManager e i loader grafici. Infine, notando una ripetizione inutile nella creazione delle bolle nel cannone, ho creato un algoritmo che mi permetta di leggere la

mappa alla ricerca di ogni determinato colore in modo di evitare di sparare colori assenti (questo avviene esattamente due lanci dopo la scomparsa di un colore sulla mappa, come da gioco originale).

Classi su cui ho lavorato :

- Ball
- Map
- CollectBall
- Display
- Cannon
- Score
- StateMenu

3.3 Note di sviluppo

3.3.1 Sofia Bagagli :

- Utilizzo di JUnit
- Utilizzo librerie di listener e manager per gestione bottoni
- Librerie per elementi grafici e programmi per la creazione di contenuti (awt)

3.3.2 Yuri Collini:

- Utilizzo librerie grafiche per gestione bottoni
- essenziali di gioco (Game, Handler, Launcher)
- Librerie per elementi grafici e programmi per la creazione di contenuti (awt)

3.3.3 Michele Nardini:

- Utilizzo libreria Graphics
- Utilizzo ArrayList per gestione cannone
- Utilizzo di costruttori statici

- Librerie per elementi grafici e programmi per la creazione di contenuti (awt)

3.3.4 Elisa Simoni:

- Utilizzo di ArrayList e Matrici Bidimensionali per gestione mappa
- Utilizzo di librerie per creazione di file in lettura e in scrittura (BufferedReader, File, InputStream, InputStreamReader)
- Utilizzo librerie Listener per la gestione degli input
- Librerie per elementi grafici e programmi per la creazione di contenuti (awt)

Commenti Finali

4.1 Autovalutazioni

In questa sezione finale della relazione relativa al progetto d'esame sviluppato dal gruppo, che prende il nome di Spaccabolle, sono state inserite le autovalutazioni dei vari membri del gruppo. Inoltre, verranno indicate quali sono state le difficoltà che ciascun membro ha incontrato e come sono stati risolti alcune problematiche.

Autovalutazione di Yuri Collini

Voglio iniziare dicendo che sono molto soddisfatto del progetto realizzato dal gruppo. Questo è stato un progetto molto lungo e che ha richiesto numerosi sforzi da parte di tutti i membri.

Personalmente lo sviluppo del gioco, che abbiamo soprannominato Spaccabolle, mi è servito molto per migliorare la mia conoscenza del linguaggio Java. Inoltre, grazie a questo progetto ho potuto capire quali sono le difficoltà che si possono incontrare in qualunque lavoro di gruppo, ho scoperto nuovi strumenti e tecnologie utili a migliorare la produttività e la collaborazione tra i vari membri del team. Una delle maggiori difficoltà che ho riscontrato è il fatto di dover lavorare a distanza per la maggior parte del tempo. Questa situazione, dovuta alla pandemia tutt'ora in corso, mette ancora più in difficoltà

in quanto se dovevo spiegare una mia idea, ed avevo bisogno di ricorrere all'utilizzo di schemi o esempi, capitava molto spesso che non si comprendeva a pieno cosa volessi intendere. Il fatto di lavorare a distanza ha inciso anche sulla mia produttività in maniera negativa ed inoltre portare avanti un progetto di questa portata senza avere un confronto di persona è decisamente più pesante. Una delle cose che mi è piaciuta maggiormente è stato il fatto di poter sviluppare un qualcosa, che in questo caso era un videogioco, partendo da completamente da zero. Questo ti permette di avere la massima libertà su ogni singola decisione, ti permette di dare sfogo alla tua creatività e secondo me questa, è stata un'esperienza che mi ha insegnato molto, anche perché mi ha permesso di capire meglio quali sono le difficoltà che ogni singolo programmatore deve incontrare ogni qual volta si appresta a lavorare ad un progetto che sia da solo o in team.

Autovalutazione di Elisa Simoni

Durante lo sviluppo del gioco in talune occasioni ho incontrato difficoltà, poi risolte. Di seguito ecco le principali problematiche. Nella fase di progettazione riponevo molta fiducia nelle mie capacità di programmazione e gestione logica dei problemi, ma una volta approcciato il progetto ho subito capito che il mio background scolastico non era sufficiente per l'obbiettivo che avevamo. Dopo aver colmato alcune mie lacune è stato stimolante sviluppare SpaccaBolle.

Le maggiori difficoltà le ho incontrate nella fase di gestione delle collisioni e conseguenti scoppi tra le bolle e del caricamento della mappa. Queste parti hanno richiesto uno sforzo logico in più per poterle completare con successo.

Per la prima volta nella mia carriera scolastica ho messo mano a un progetto partendo completamente da zero. Il lavoro in gruppo a volte è stato difficoltoso in quanto la mancanza di organizzazione e l'impossibilità di vedersi di persona, causata dalla pandemia in corso, hanno rallentato notevolmente lo sviluppo del gioco, ma grazie e varie piattaforme online, siamo riusciti in parte a sopperire a queste problematiche.

Voglio concludere dicendo che lo sviluppo di questo progetto mi ha aiutato a capire la fatica e l'impegno che si celano dietro un programma

ma anche quanto programmare possa essere utile nell'approcciarsi alla vita di tutti giorni

Autovalutazione di Michele Nardini

Sono molto soddisfatto dei miei risultati ottenuti durante le fasi di realizzazione di SpaccaBolle. I videogiochi sono sempre stati una parte rilevante delle mia vita e poterne creare uno per intero, partendo da zero, dove potessi mettere le mie capacità in "gioco" è stata un'esperienza unica ed eccezionale. Ogni riga di codice mi ha permesso di approfondire le mie conoscenze del linguaggio Java e mi ha dato dei buoni spunti per sviluppare altri progetti personali. La parte di gestione dell'architettura del software mi ha messo a dura prova in quanto spesso e volentieri non ottenevo i risultati sperati. Ho, inoltre, sviluppato una passione per la parte grafica dei videogiochi notando quanto un bell'aspetto combinato ad una giusta programmazione possano creare un ottimo prodotto finale. Concludo dicendo questo tipo di progetto mi ha spinto ad ampliare le mie conoscenze del linguaggio Java, in particolare all'ottimizzazione e l'automazione di eventi.

Autovalutazione di Sofia Bagagli

Durante la realizzazione del videogioco ho riscontrato varie difficoltà che sono state prontamente superate grazie alla pazienza l'aiuto reciproco di tutti i membri del gruppo. All'inizio del progetto, il team aveva pensato di restare il più fedeli possibile alla versione originale del videogioco Puzzle Bobble. Nella versione originale, il cannone restava fermo al centro dello schermo con una freccia che permetteva il direzionamento, a destra ed a sinistra, della traiettoria che doveva compiere ogni bolla lanciata. Quest'ultima funzione l'avevo implementa in maniera tale che la freccia del cannone seguisse il cursore del mouse all'interno della schermata di gioco per sportarsi, mentre un altro membro del gruppo l'aveva implementata mediante la combinazione dei tasti AWSD. Uno dei problemi che ho riscontrato. Inizialmente, risiedeva nel fatto che non sapevo bene come implementare il tutto, anche perché fin dall'inizio c'era sempre qualcosa che non funzionava correttamente e si venivano a creare molti bug. Alla fine, però, ero

riuscita ad arrivare al mio scopo con successo, tuttavia, si è deciso in seguito di far spostare l'intero cannone senza permettere quindi il direzionamento della bolla. Questa scelta è stata effettuata per svariate ragioni una delle quali era quella di rendere il gioco più personale e non una mera copia esatta della versione originale, rendendo al contempo il gioco leggermente più difficile.

La cosa che mi è piaciuta maggiormente di tutto il progetto è stata la parte grafica a me assegnata. Questa è quella che più mi ha appagato, la creazione degli elementi grafici e le loro funzioni per me è stato molto divertente, stimolante inoltre mi ha permesso di scoprire nuove cose e nuovi strumenti di lavoro.

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.

Avvio dell'applicazione

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. Si noti che nella versione di DEMO tale funzionalità non è disponibile. In fine mediante il bottone "EXIT" potrà uscire e chiudere l'applicazione.

Schermata di gioco

All'avvio del gioco troverete davanti a voi una schermata come quella sopra. Il cannone si muove autonomamente e per sparare le bolle bisogna premere sulla tastiera la barra spaziatrice. Se si vuole mettere in pausa il gioco basta premere il pulsante sulla tastiera "P" oppure cliccando sulla parola "PAUSE". Dopo aver effettuato il primo lancio sarà possibile effettuare il salvataggio della partita cliccando sulla parola "SAVE" oppure premendo il tasto sulla tastiera "S". Per uscire dal gioco cliccare sulla parola "EXIT" oppure premere il tasto sulla tastiera "E",

apparirà una schermata che chiederà se si è sicuri di voler abbandonare il gioco o meno e se si vuole salvare la partita.

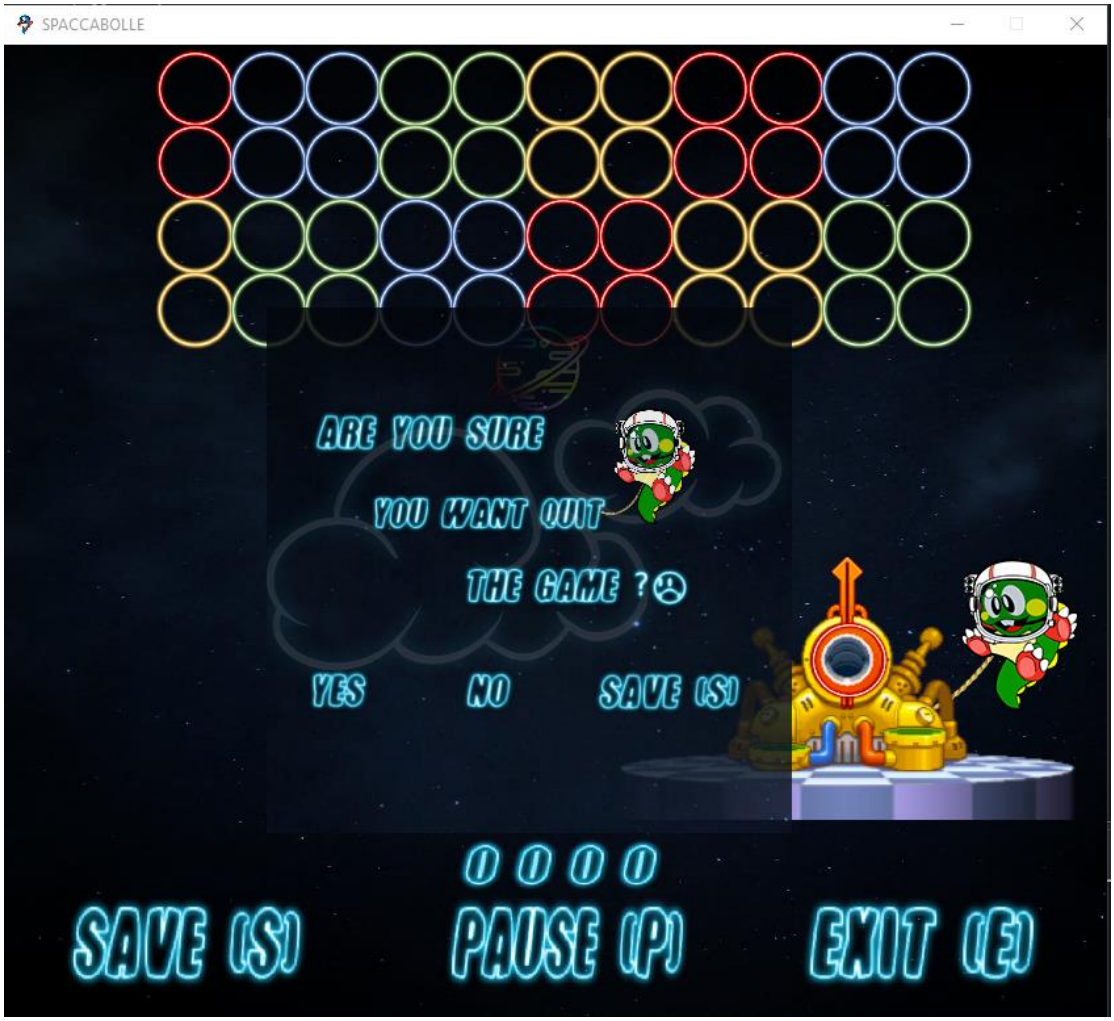
Schermata di pausa

Quando si apre il menù di pausa, all'utente verrà mostrata una schermata come quella sopra ed egli potrà scegliere di cambiare la difficoltà di gioco, premendo i tasti numerici "1, 2, 3" e "INVIO".

Se l'utente preme sull'immagine della casa si torna alla schermata principale del gioco, premendo il simbolo di pausa il gioco riprende e cliccando sull'icona power si apre un'altra finestra di conferma di uscita dal gioco.

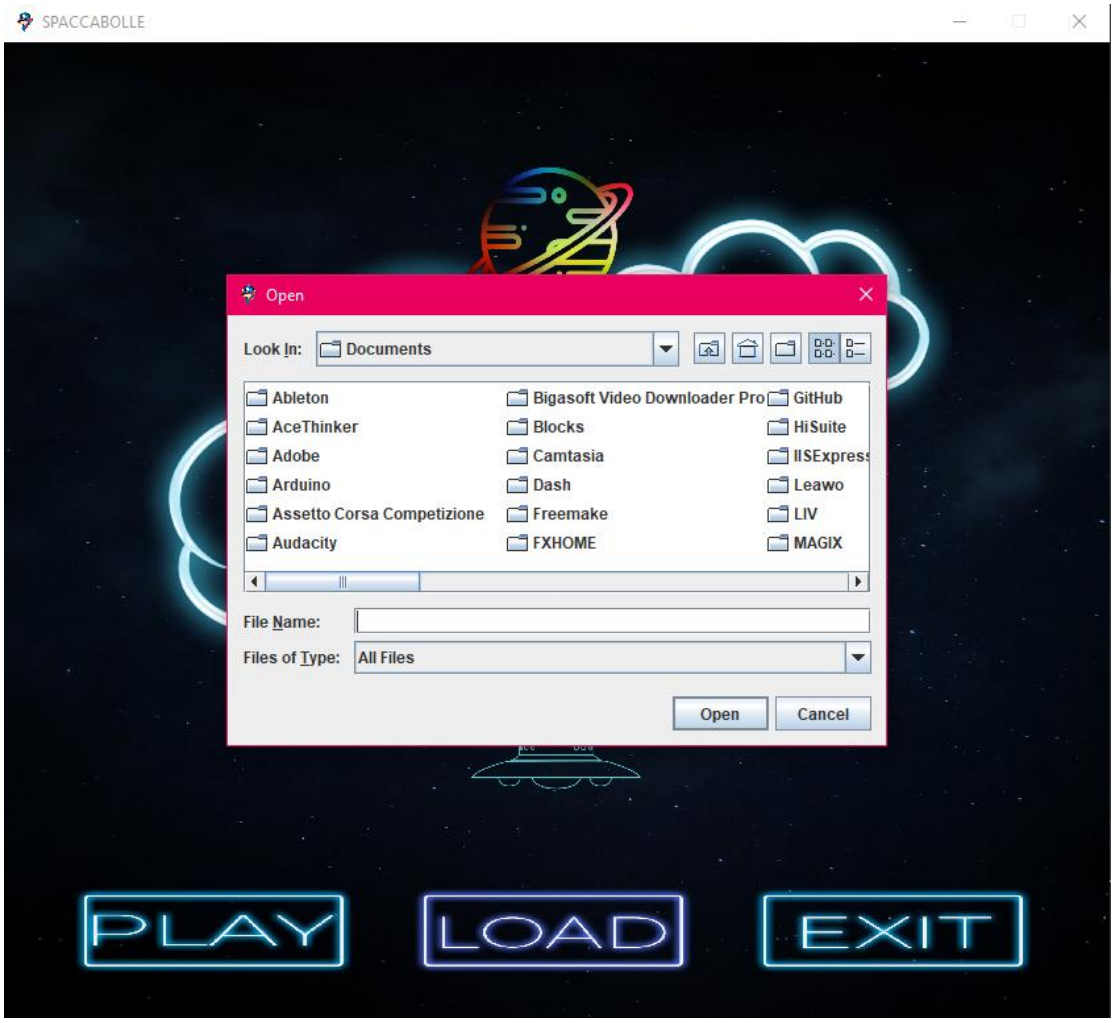
Una volta messo in pausa il gioco, il cannone si blocca e non è più possibile sparare bolle.

Schermata di uscita dal gioco



Una volta premuto il tasto "exit" all'utente verrà mostrata una schermata come quella sopra dove potrà scegliere se uscire dal gioco (premendo il tasto sulla tastiera "Y") oppure no (premendo il tasto sulla tastiera "N") oppure salvare il livello (premendo il tasto sulla tastiera "S").

Schermata di caricamento del livello

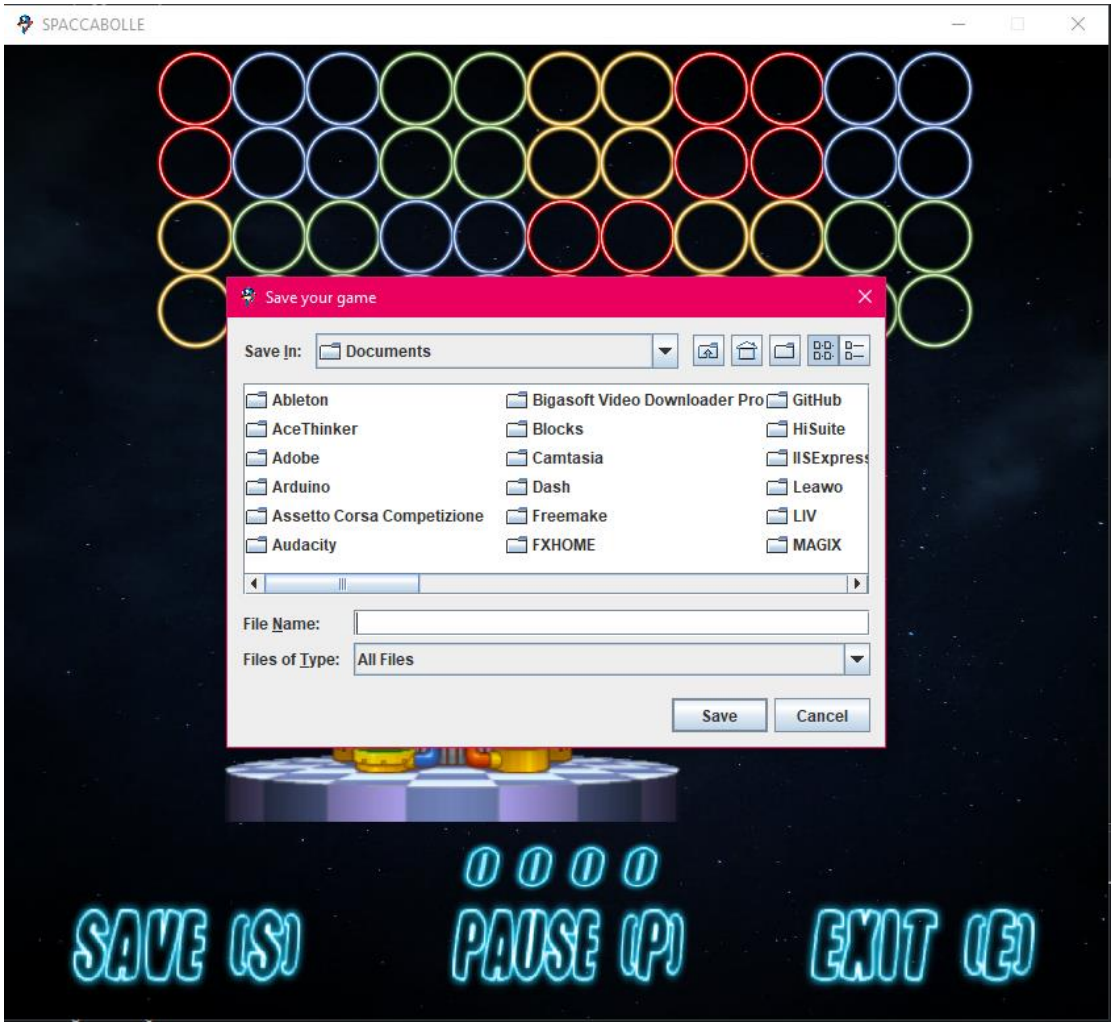


Premendo il tasto "LOAD" l'utente potrà caricare e giocare ad una partita precedentemente interrotta e salvata. La schermata che si presenterà è come quella in foto.

NOTA

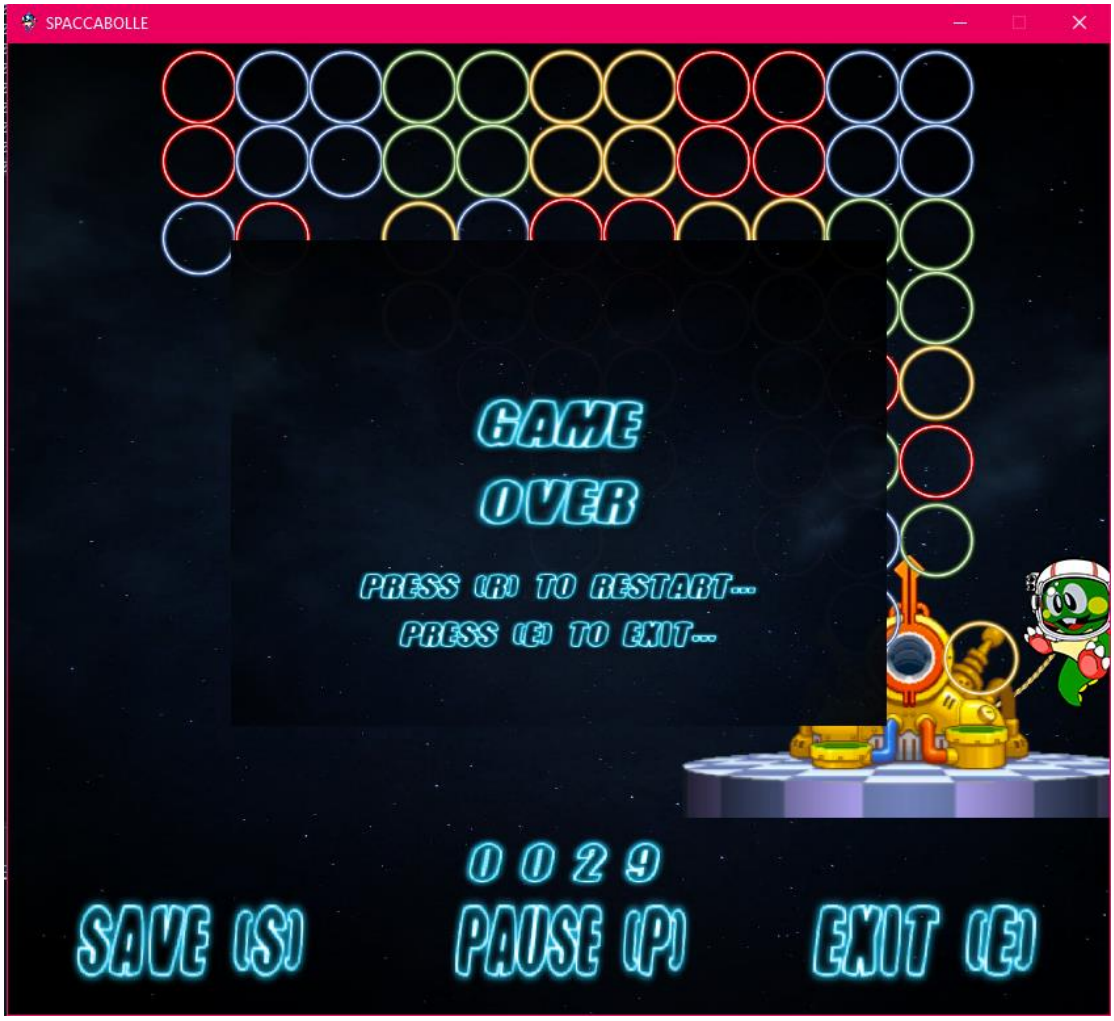
Come in precedenza, la funzionalità di caricamento di una partita precedentemente salvata non è disponibile nella versione demo.

Schermata di salvataggio del livello



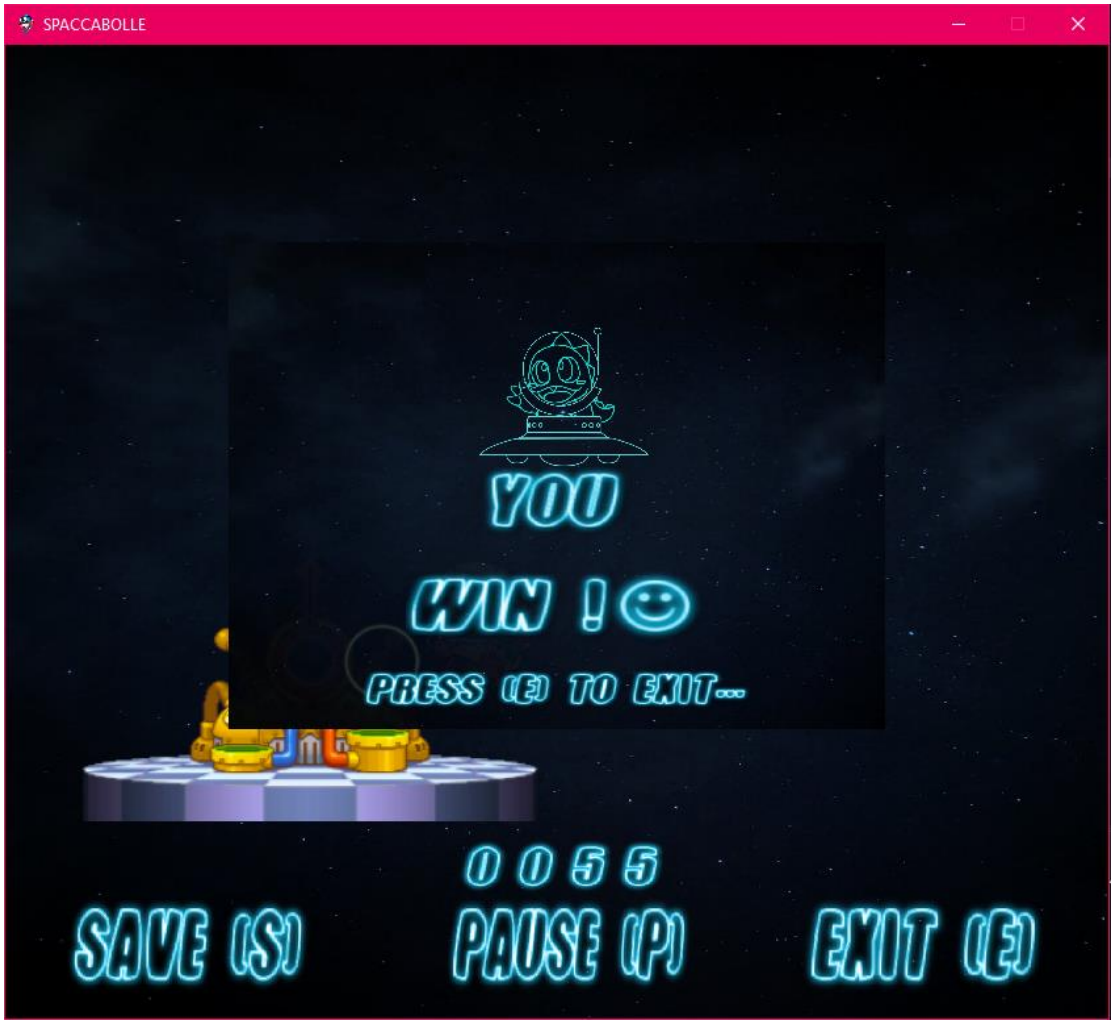
Il gioco inoltre, permette di salvare il livello di gioco in corso. L'utente deve inserire il nome ed il programma automaticamente lo salverà il livello di gioco in formato txt. Per giocare ad un livello precedentemente salvato vedi la precedente descrizione.

Schermata di Game Over



Se la pallina tocca il cannone allora il giocatore perde la partita ed appare la schermata di game over come quella qui sopra, in cui l'utente potrà scegliere se ricominciare il gioco premendo il tasto della tastiera "R" oppure se terminare il gioco ed uscire dall'applicazione premendo il tasto "E".

Schermata di Win



Se sulla mappa non sono più presenti bolle allora l'utente ha vinto! In questo caso all'utente apparirà una schermata come quella qui sopra. Il giocatore verrà fatto uscire dal programma premendo il tasto "E" dalla tastiera.