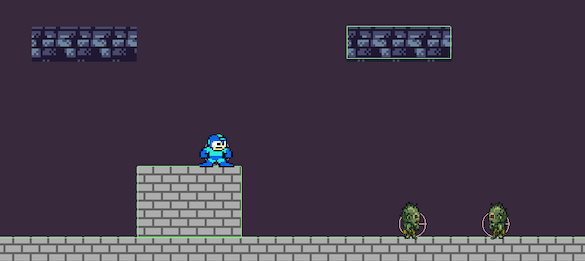
Università degli studi di Modena e Reggio Emilia

Facoltà di Ingegneria “Enzo Ferrari”

Corso di laurea in Ingegneria Informatica

A.A. 2021/2022

*Death Ops*



Grandi Andrea (144719)

## Indice

### Specifica dei Requisiti del Software (SRS)

1. **Introduzione…………………………………………………………………………………..3**
   1. **Obiettivo……………………………………………………………………………..3**
   2. **Campo d’applicazione………………………………………………………………3**
   3. **Definizioni, acronimi e abbreviazioni……………………………………………...3**
   4. **Struttura document SRS……………………………………………………………3**
2. **Descrizione generale…………………………………………………………………………..3**
   1. **Inquadramento………………………………………………………………………3**
      1. **Interfaccia Sistema/utente……………………………………………………...3**
      2. **Interfaccia Hardware…………………………………………………………...4**
      3. **Interfaccia Software…………………………………………………………….4**
   2. **Funzioni principali del Software……………………………………………………4**
   3. **Caratteristiche degli utenti………………………………………………………….4**
   4. **Vincoli generali………………………………………………………………………4**
   5. **Assunzioni e dipendenze…………………………………………………………….4**
3. **Specifica dei requisiti………………………………………………………………………….5**
   1. **Requisiti funzionali…………………………………………………………………..5**
      1. **Interfacce esterne ……………………………………………………………….5**
      2. **Caratteristiche del sistema: Personaggio (Player)…………………………….5**
      3. **Caratteristiche del sistema: Nemici…………………………………………….6**
      4. **Caratteristiche del sistema: Partita…………………………………………….7**
      5. **Caratteristiche del sistema: Impostazioni……………………………………...8**
   2. **Requisiti non funzionali……………………………………………………………...8**

**Unified Modelling Language (UML)**

1. **Class diagram – Scenario generale…………………………………………………………...10**
2. **Object diagram – Personaggi………………………………………………………………….12**
3. **Pakage diagram………………………………………………………………………………...13**
4. **Use case diagram……………………………………………………………………………….14**
5. **Activity diagram…………………………………………………………………......................16**
6. **State diagram – Stati del personaggio………………………………………………………...18**
7. **Sequence diagram………………………………………………………………………………19**

**Design Pattern**

1. **Decorator Pattern – Aggiunta personaggi…………………………………………………….20**

### SPECIFICA DEI REQUISITI DEL SOFTWARE (SRS)

1. **Introduzione**

La presente sezione ha lo scopo di riportare la visione globale dell’intero documento di Specifica dei requisiti. La struttura del documento è quella suggerita dallo standard IEEE 830 noto come SRS (Software Requirements Specifications).

* 1. Obiettivo

Lo scopo di questa sezione è rappresentare nel modo più preciso, consistente e chiaro, il progetto, realizzato insieme a Montagnani Luca, per il corso di Programmazione ad oggetti. Più precisamente, la specifica dei requisiti del sistema intende spiegare le funzioni del software per la realizzazione di un videogioco in 2D Scrolling Background.

L’approccio seguito per lo sviluppo del software è di tipo prototipale, quindi nuovi requisiti potranno essere introdotti in seguito, oltre a vincoli che fino a questo momento non sono ancora stati individuati.

* 1. Campo d’applicazione

Carattere puramente ludico.

* 1. Definizioni, acronimi e abbreviazioni

|  |  |
| --- | --- |
| Videogioco 2D | Videogioco sviluppato attraverso tecniche di visualizzazione bidimensionale. |
| Scrolling Background | L’azione di gioco è ripresa da una vista laterale; la transazione dello sfondo può avvenire sia orizzontalmente che verticalmente. |

* 1. Struttura del documento SRS

Questo documento prosegue con una spiegazione generale delle funzionalità del software; in particolare verranno indicati i vincoli sull’hardware, dopodichè verranno elencati e descritti i requisiti dell’applicativo secondo un’organizzazione *by feature,* che è composta da tre parti: la prima parte riguarda le interfacce esterne del software, nella seconda parte vengono indicati i requisiti funzionali, infine nella terza parte vengono indicati i requisiti non funzionali.

1. **Descrizione generale**
   1. Inquadramento

“Death ops” è un software di gioco 2D, realizzato per il superamento dell’esame di Programmazione ad oggetti (A.A. 2020/2021), insieme al collega Montagnani Luca. Il gioco si caratterizza come uno sparatutto con scrolling in 2D.

* + 1. Interfaccia Sistema/Utente

Il software nasce per una utenza non necessariamente esperta in campo informatico, orientata all’aspetto ludico del programma.

Riveste per tale motivo un’elevata importanza l’accessibilità generale al programma.

Interfaccia utente

Occorre prestare attenzione alla progettazione di una interfaccia utente graficamente accattivante, ma allo stesso tempo semplice da utilizzare (user friendly) in modo tale da massimizzare il numero di possibili utenti.

Navigabilità

Il menù iniziale deve essere chiaro e non ambiguo, semplice da utilizzare.

Si deve comporre di una finestra nella quale dovranno essere presenti un numero limitato di bottoni per iniziare una nuova partita, uscire dal gioco, settare le impostazioni audio/video e visionare i crediti. L’utente deve avere la possibilità di scegliere tra due tipi di personaggio da utilizzare in partita.

Usabilità

Si vuole che gli utenti devono poter compiere le operazioni in modo semplice ed intuitivo, senza dover di particolari tipi di conoscenze.

Vincoli di installazione

Il software non deve richiedere l’installazione da parte dell’utente, ma deve essere contenuto in una cartella compressa (.zip) ed una volta scompattata, l’utente deve essere in grado di lanciare il programma facendo doppio click su un file eseguibile con il nome del gioco.

* + 1. Interfaccia Hardware

Non è previsto l’uso di particolari controller esterni al di fuori di quelli normalmente disponibili, ovvero mouse e tastiera.

* + 1. Interfaccia Software

Il programma deve essere eseguibile su più piattaforme, come Windows, Linux e MacOS

* 1. Funzioni generali del software

Le macro funzionalità di “Death ops” sono:

1. Creazione di una nuova partita
2. Scelta tra due personaggi (Skin)
3. Terminazione di una partita in corso
4. Creazione di una classifica locale basata sui punteggi
5. Settaggio delle impostazioni di gioco come audio

Questi sono solo alcuni delle funzioni generali del software che verranno meglio specificate nella parte dei requisiti funzionali.

* 1. Caratteristiche dell’utenza

L’utenza media non possiede particolari conoscenze in ambito informatico.

I principali utenti del programma sono i Player, ovvero chi effettivamente utilizzerà il software per scopi puramente ludici.

* 1. Vincoli generali

Il software non ha la necessità di una connessione ad internet, perché non c’è nessuna funzionalità che lo richiede.

Tutte le operazioni di base del software, come la creazione di una partita o settaggio delle impostazioni di gioco, possono essere eseguite da qualsiasi tipo di utenza poiché nessuna funzionalià presenta aspetti critici per il sistema. Per l’accesso e la modifica del codice sorgente è invece riservata ad utenti muniti di una password.

* 1. Assunzioni e dipendenze

Non vi sono particolari meccanismi o nozioni che l’utente deve essere a conoscenza per il corretto utilizzo del software.

1. **Specifica dei requisiti**
   1. Requisiti funzionali

In questo paragrafo verranno spiegati tutti i requisiti funzionali; per ogni requisito verrà indicata l’importanza e la stabilità utilizzando una scala da 1 a 5, dove 1 è il minimo e 5 il massimo.

* + 1. Interfacce esterne

1. Interfaccia utente: l’interfaccia utente è realizzata in java ed è ottimizzata per una risoluzione dello schermo 500x500 pixel (richiama l’interfaccia di un retrò game).
2. Interfaccia Hardware: i requisiti minimi dell’hardware del computer sono: Pentium 4 (o equivalente) e 512 Mb di memoria ram.
3. Interfaccia Software: il programma, per funzionare, ha la necessità che nel computer sia presente una Java Virtual Machine (JVM), almeno aggiornata alla Standard Edition 7.
   * 1. Caratteristiche del sistema: Personaggio

|  |  |
| --- | --- |
| Numero del requisito | RF001 |
| Descrizione | Personaggio – Movimento |
| Input | Tastiera |
| Processo | Il personaggio, all’interno della finestra di gioco, è “bloccato” al centro della finestra, esegue solamente le animazioni che corrispondono al movimento a sinistra o a destra, sarà poi lo sfondo a muoversi di conseguenza.  Tasto D: movimento a destra  Tasto A: movimento a sinistra |
| Output | Il personaggio esegue il movimento, il background scorre |
| Importanza | 5 |
| Stabilità | 5 |
| Riferimento | n/a |

|  |  |
| --- | --- |
| Numero del requisito | RF002 |
| Descrizione | Personaggio – Salto |
| Input | Tastiera |
| Processo | Il personaggio deve essere in grado di saltare degli ostacoli, o nemici, all’interno del gioco.  Tasto W: salto  Nel momento in cui il personaggio è “in aria”, non può eseguire un altro salto. |
| Output | Il personaggio esegue il movimento |
| Importanza | 5 |
| Stabilità | 5 |
| Riferimento | n/a |

|  |  |
| --- | --- |
| Numero del requisito | RF003 |
| Descrizione | Personaggio - Sparo |
| Input | Tastiera |
| Processo | Il personaggio deve essere in grado di sparare palle infuocate in grado di colpire i nemici ed eliminarli.  Barra spaziatrice: sparo  Il personaggio ha a disposizione infiniti spari. |
| Output | Il personaggio spara |
| Importanza | 5 |
| Stabilità | 5 |
| Riferimento | n/a |

|  |  |
| --- | --- |
| Numero del requisito | RF004 |
| Descrizione | Personaggio - Vita |
| Input | n/a |
| Processo | Il personaggio ha a disposizione una vita, descritta da piccoli cuori rossi. Ogni volta che si inizia una partita, il personaggio parte con tre cuori (massimo della vita), ed ogni volta che si scontra con un nemico, perde una vita.  Quando il personaggio ha zero vite la partita termina con un game over. |
| Output | Punti vita |
| Importanza | 5 |
| Stabilità | 5 |
| Riferimento | n/a |

|  |  |
| --- | --- |
| Numero del requisito | RF005 |
| Descrizione | Personaggio - Punteggio |
| Input | n/a |
| Processo | Il personaggio ha un punteggio assegnato ad ogni partita, ogni volta che un nemico viene eliminato il personaggio accumula 10 punti nel suo punteggio personale. |
| Output | Punteggio personaggio aggiornato |
| Importanza | 4 |
| Stabilità | 5 |
| Riferimento | n/a |

* + 1. Caratteristiche del sistema: Nemici

|  |  |
| --- | --- |
| Numero del requisito | RF006 |
| Descrizione | Nemici - Movimento |
| Input | n/a |
| Processo | I nemici (“Zombie”) devono essere in grado di muoversi in maniera autonoma sul ground di gioco. I nemici si muovono solo a destra e a sinistra, cambiano direzione appena incontrano un ostacolo. La velocità dei nemici è sempre uguale all’interno di ogni livello, la velocità aumenta proseguendo nei livelli di gioco. |
| Output | n/a |
| Importanza | 5 |
| Stabilità | 5 |
| Riferimento | n/a |

|  |  |
| --- | --- |
| Numero del requisito | RF007 |
| Descrizione | Nemici - Vita |
| Input | n/a |
| Processo | I nemici non possiedono dei punti vita, appena li si colpisce con uno sparo, vengono eliminati. |
| Output | n/a |
| Importanza | 4 |
| Stabilità | 5 |
| Riferimento | n/a |

* + 1. Caratteristiche di sistema: Partita

|  |  |
| --- | --- |
| Numero del requisito | RF008 |
| Descrizione | Menù di gioco |
| Input | Tastiera |
| Processo | Durante una partita il giocatore deve essere in grado di poter invocare il menù di gioco. Quando il menù è attivo, la partita entra in uno stato di pausa.  Il menù di gioco deve avere queste funzionalità:  Uscita da gioco  Ritorno alla partita in corso  Settaggio audio |
| Output | Schermata del menù di gioco |
| Importanza | 4 |
| Stabilità | 5 |
| Riferimento | n/a |

|  |  |
| --- | --- |
| Numero del requisito | RF009 |
| Descrizione | Messa in pausa |
| Input | Tastiera |
| Processo | Il giocatore premendo il tasto ESC durante una partita si entra nella modalità pausa, in questa fase si ha la completa sospensione del gioco fin tanto che non verrà ripremuto il tasto ESC. Nella schermata di pausa è possibile cambiare i settaggi audio e attivare o disattivare i suoni. |
| Output | n/a |
| Importanza | 5 |
| Stabilità | 4 |
| Riferimento | n/a |

|  |  |
| --- | --- |
| Numero del requisito | RF010 |
| Descrizione | Terminazione di una partita |
| Input | n/a |
| Processo | La partita termina nel momento in cui il personaggio non ha più punti vita, oppure finiscono i livelli di gioco. |
| Output | Schermata di Game Over |
| Importanza | 5 |
| Stabilità | 5 |
| Riferimento | n/a |

|  |  |
| --- | --- |
| Numero del requisito | RF011 |
| Descrizione | Livelli di gioco |
| Input | n/a |
| Processo | Durante una partita il giocatore può percorrere più livelli. I livello di gioco deve essere visibile nell’interfaccia di gioco nell’angolo in alto a destra durante tutta la partita.  Il giocatore supera un livello e passa quindi al successivo quando termina la mappa di gioco. |
| Output | Aggiornamento livello |
| Importanza | 5 |
| Stabilità | 5 |
| Riferimento | n/a |

|  |  |
| --- | --- |
| Numero del requisito | RF012 |
| Descrizione | Mappe di gioco |
| Input | n/a |
| Processo | Durante una partita, quando si passa da un livello ad un altro, la mappa di gioco cambia (ambientazione ed ostacoli diversi). La mappa cambia quando si passa da un livello ad un altro. |
| Output | Aggiornamento mappa di gioco |
| Importanza | 5 |
| Stabilità | 5 |
| Riferimento | n/a |

* + 1. Caratteristiche di sistema: Impostazioni

|  |  |
| --- | --- |
| Numero del requisito | RF013 |
| Descrizione | Settaggio impostazioni |
| Input | Mouse e tastiera |
| Processo | Il giocatore deve essere in grado di settare le impostazioni audio della partita in qualunque momento, sia prima di iniziare la partita, sia durante. |
| Output | Aggiornamento settaggi |
| Importanza | 4 |
| Stabilità | 5 |
| Riferimento | n/a |

* 1. Requisiti non funzionali

|  |  |
| --- | --- |
| Numero del requisito | RNF001 |
| Descrizione | Menù iniziale: funzionalità necessarie |
| Input | n/a |
| Processo | Il gioco deve disporre di un menù iniziale con queste funzionalità:  Inizio di una nuova partita  Uscita dal programma  Visione dei crediti |
| Output | Aggiornamento livello di gioco |
| Importanza | 5 |
| Stabilità | 4 |
| Riferimento | n/a |

|  |  |
| --- | --- |
| Numero del requisito | RNF002 |
| Descrizione | Menù iniziale: settaggio audio |
| Input | n/a |
| Processo | Nel menù delle impostazioni deve essere disponibile la possibilità di abilitare/disabilitare la musica e gli effetti sonori. |
| Output | Aggiornamento livello di gioco |
| Importanza | 4 |
| Stabilità | 4 |
| Riferimento | n/a |

|  |  |
| --- | --- |
| Numero del requisito | RNF003 |
| Descrizione | Menù di fine partita: funzionalità necessarie |
| Input | n/a |
| Processo | Al termine di una partita deve essere presente un menù che abbia almeno le seguenti funzionalità:  Uscita dal programma  Inizio di una nuova partita |
| Output | n/a |
| Importanza | 4 |
| Stabilità | 4 |
| Riferimento | n/a |

|  |  |
| --- | --- |
| Numero del requisito | RNF004 |
| Descrizione | Riproduzione suoni |
| Input | n/a |
| Processo | Durante una partita e nei vari menù, devono essere presenti almeno i seguenti effetti audio:  Musica di sottofondo (per menù e per ogni livello)  Sparo della sfera infuocata  I suono sono in formato .mp3 e devono essere memorizzati in una determinata directory all’interno della cartella di gioco. |
| Output | n/a |
| Importanza | 4 |
| Stabilità | 4 |
| Riferimento | n/a |

Mappatura dei tasti:

Immagine che contiene testo, elettronico, tastiera

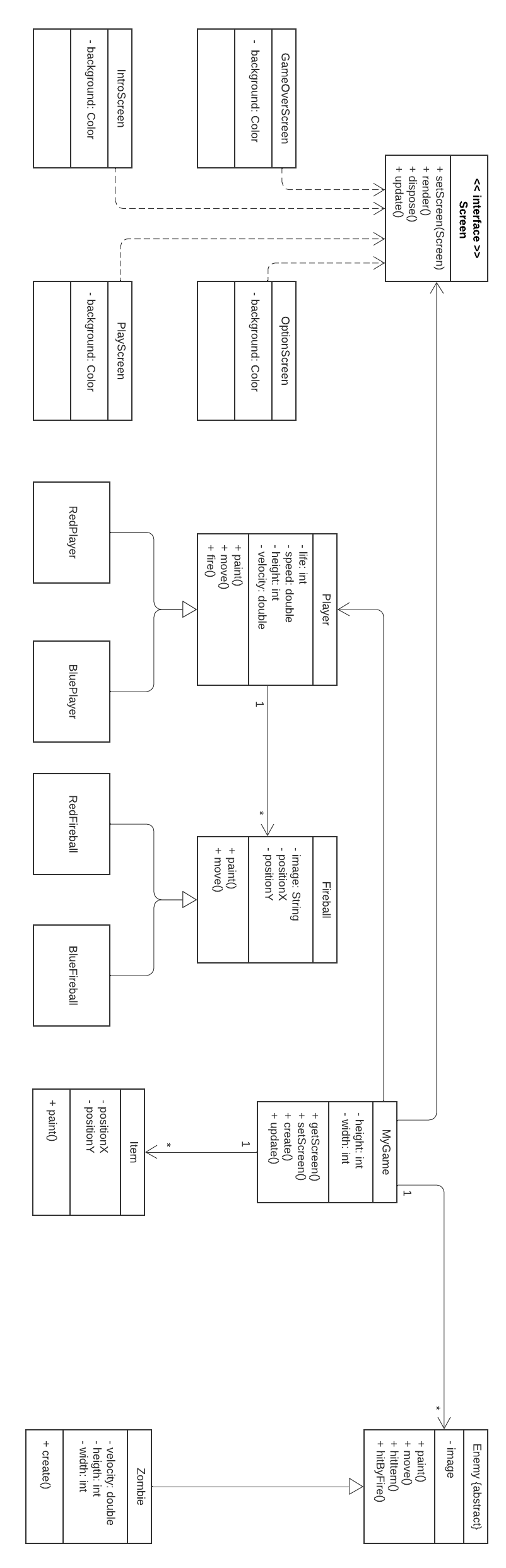
Descrizione generata automaticamente

**Unified Modelling Language (UML)**

1. **Class Diagram**

Questo diagramma ha lo scopo di rappresentare in maniera generale la struttura del sistema, la tipologia degli oggetti e le relazioni statiche che esistono tra di loro. Descrive inoltre le proprietà e i metodi principali di ciascuna classe coinvolta.

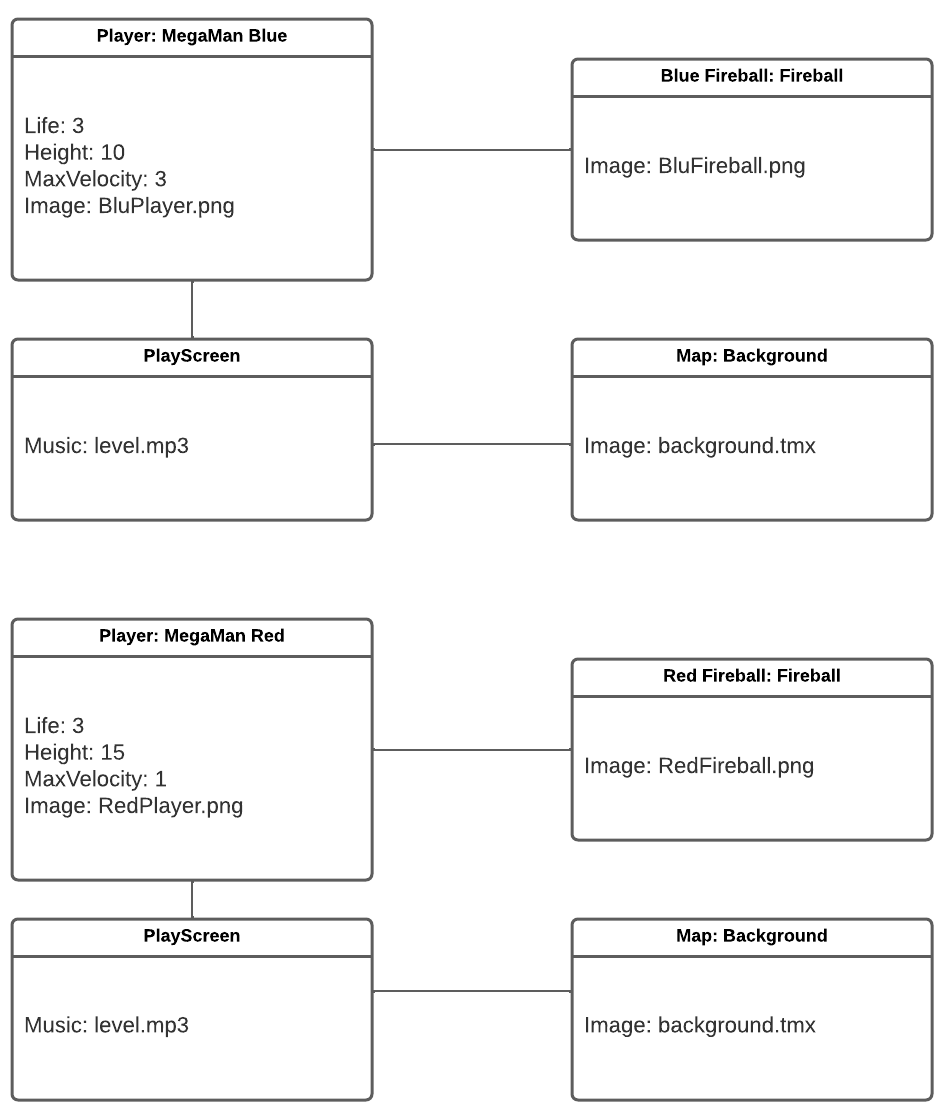
Nello specifico il cuore del sistema è la classe “MyGame” che coordina e gestisce tutte le classi a lui associate in modo diretto e indiretto. Altri elementi cardine sono “Screen” che è l’interfaccia per la gestione di tutte le schermate utilizzate (schermata di menù, schermata di gioco, schermata di game over e schermata delle impostazioni), “Player” che definisce le caratteristiche del personaggio, come stato del personaggio, vita, salto, ecc…



1. **Object Diagram**

L’Object Diagram ha il compito di rendere più facilmente comprensibili le associazioni del Class Diagram, tramite la descrizione della configurazione degli oggetti in un determinato istante, vengono quindi mostrate delle istanze particolari delle classi coinvolte.

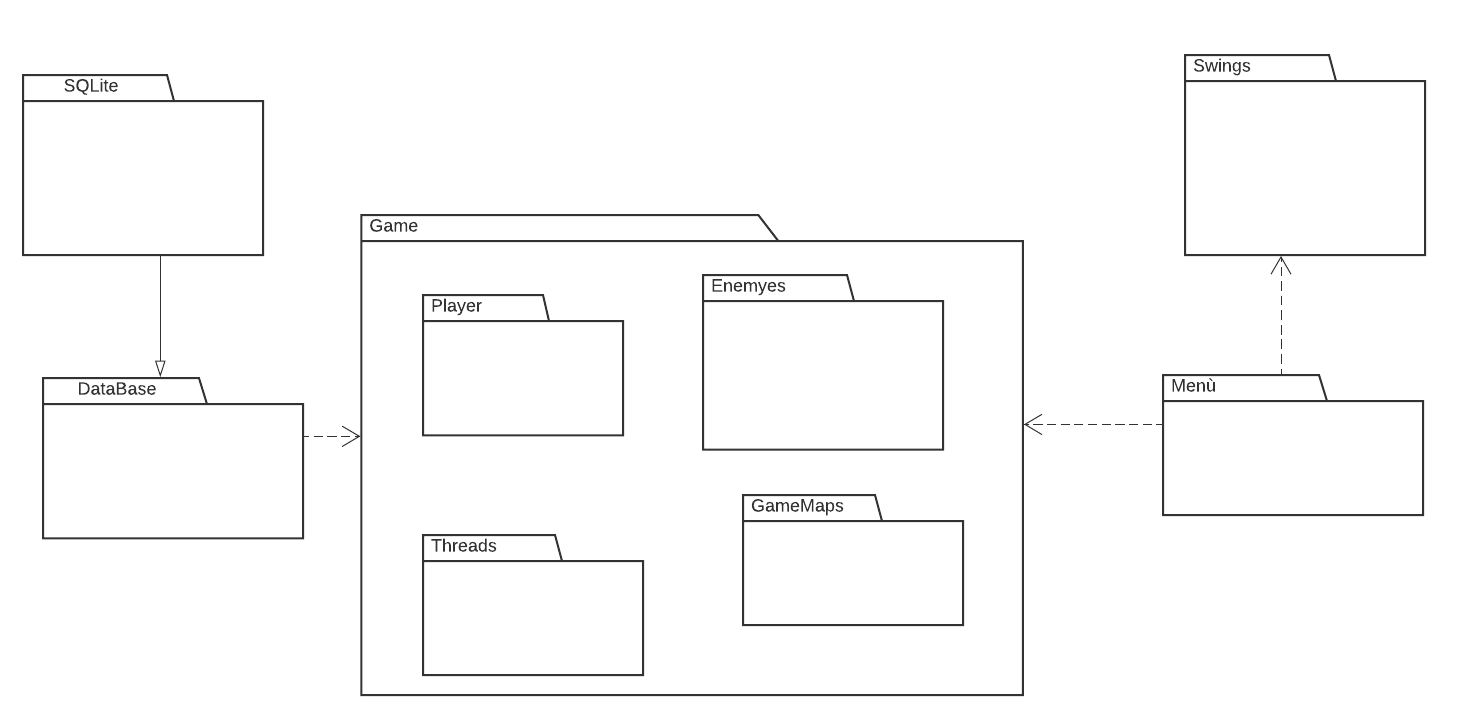
In particolare vengono evidenziate le differenze nella creazione dei due tipi di Player, infatti come da requisiti, si possono selezionare due tipi di personaggio (differenti solo come dimensioni, velocità e aspetto).



1. **Package Diagram**

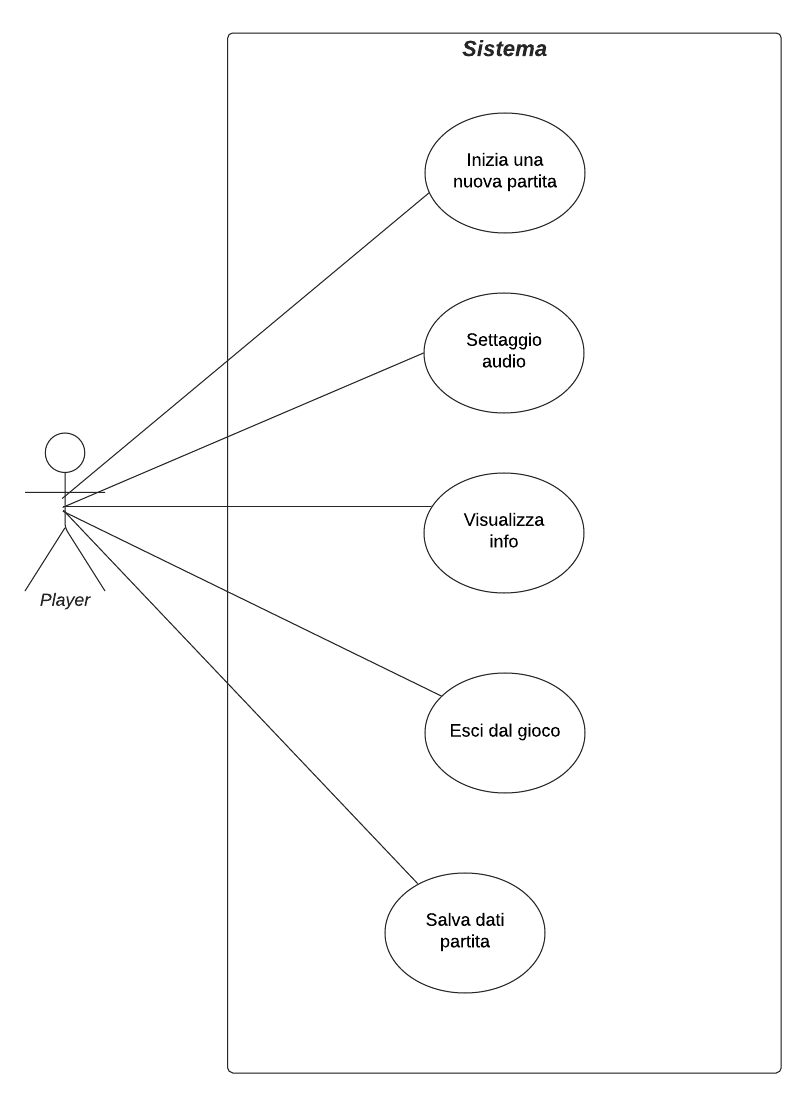
Il Package Diagram permette di definire i pacchetti che raggruppano oggetti con caratteristiche in comune tra di loro. Questo Diagram è utile per evidenziare le dipendenze che si instaurano e osservare il sistema da un livello logico più alto.

Qui in particolare vengono mostrati i packages che riguardano l’interfaccia grafica , utilizzata per creare il menù di gioco, il database, utilizzato per creare un file con i dati di ogni partita, e la finestra di gioco, con al suo interno i pacchetti chiave.



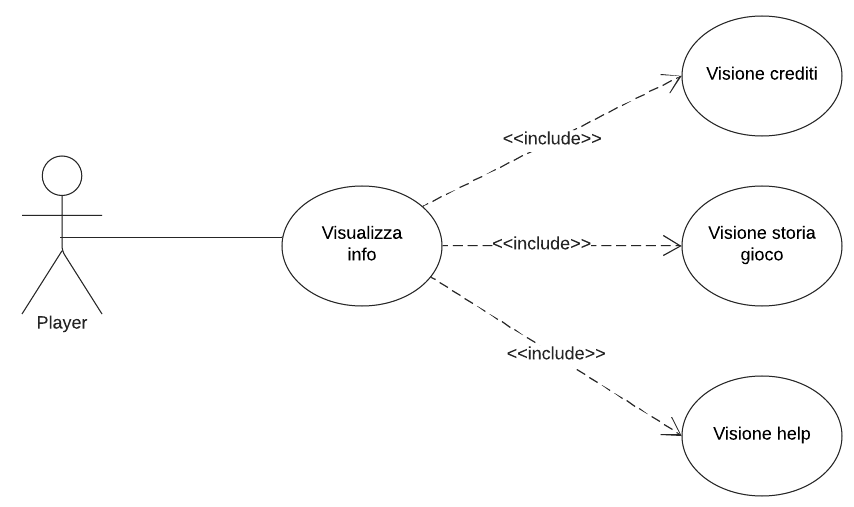
1. **Use Case Diagram**

Lo Use Case Diagram presenta gli attori principali del sistema, ovvero gli utenti, e le varie funzionalità a cui sono legati.

****

**Immagine che contiene testo, elettronico, compact disc

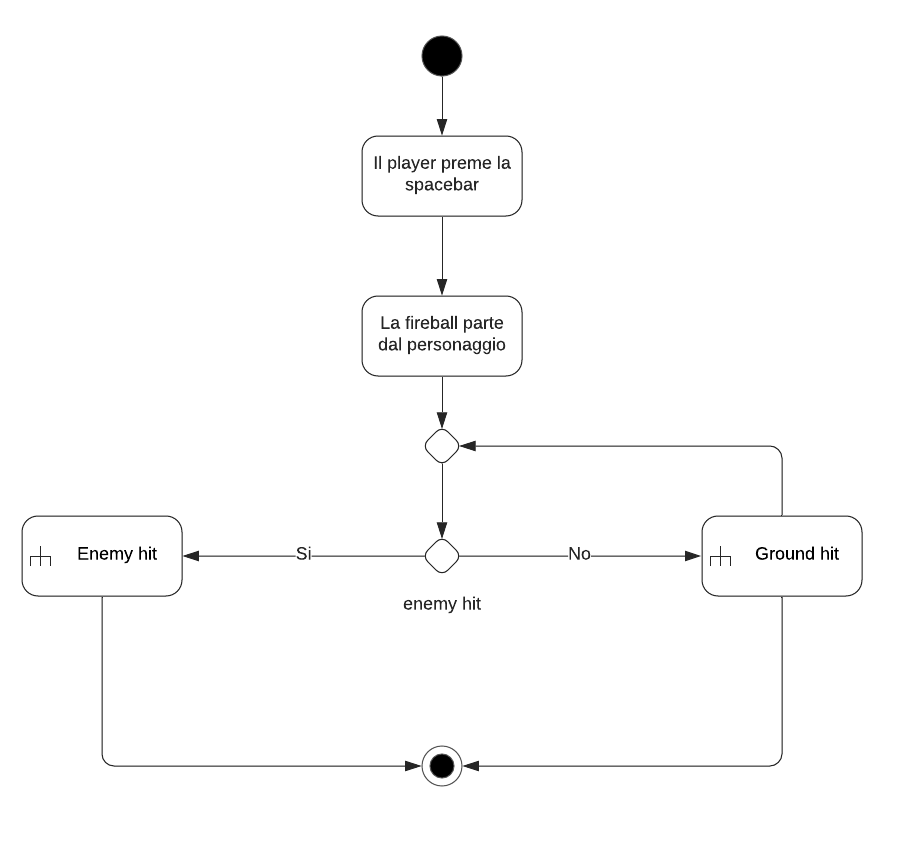
Descrizione generata automaticamente**

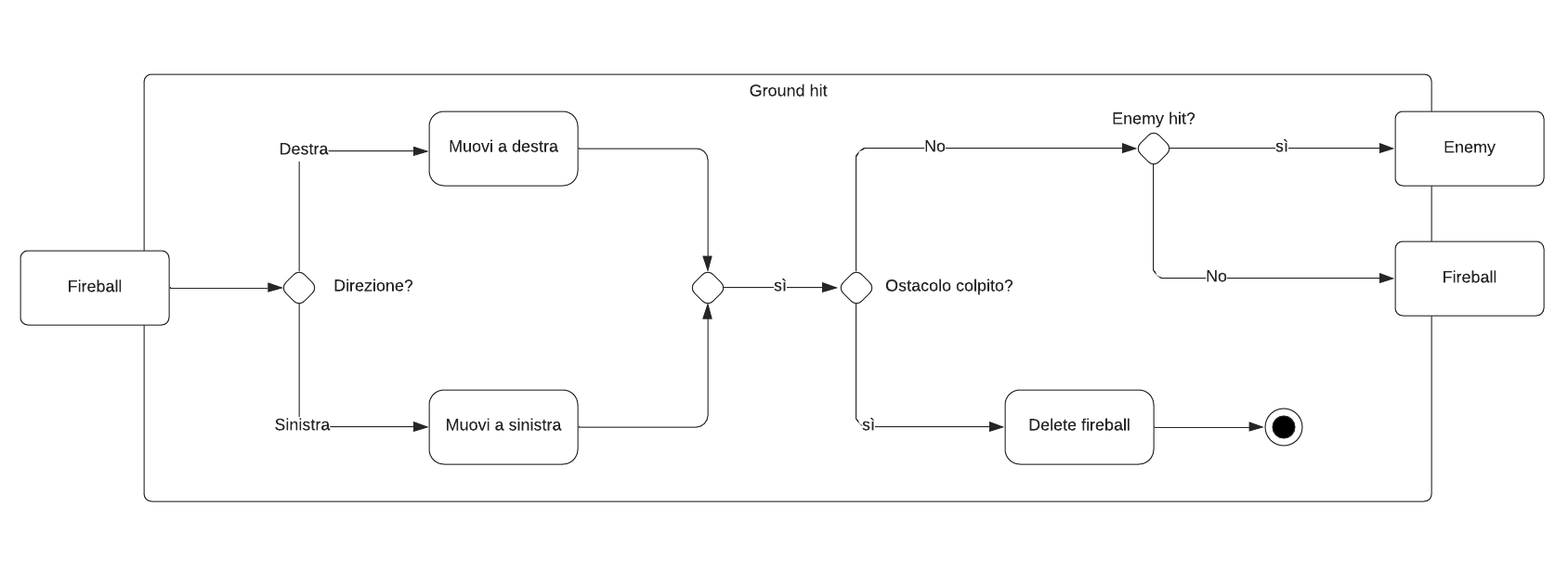
****

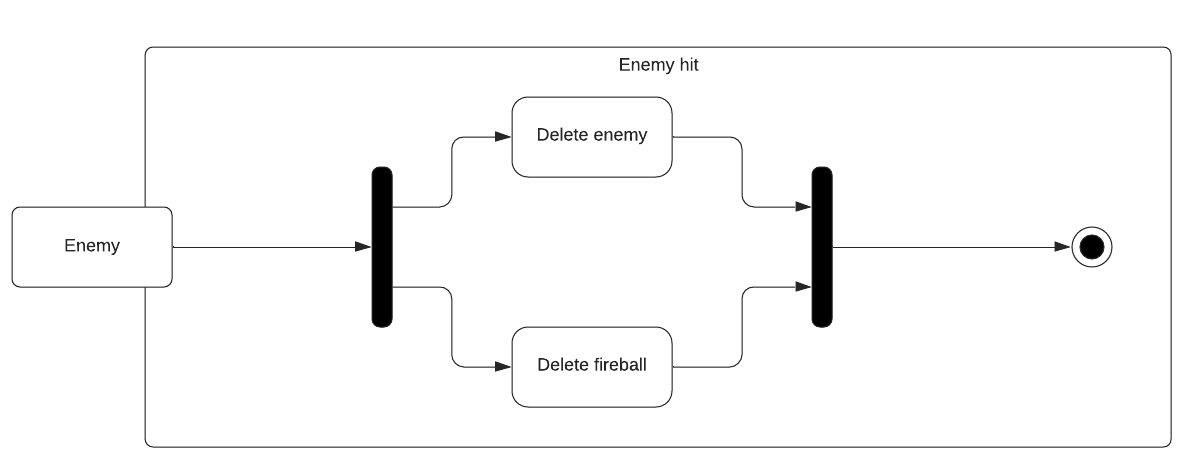
1. **Activity Diagram**

L’Activity Diagram è utile per descrivere una procedura logica attraverso un flusso di lavoro.

Nel diagramma vengono prese in analisi le attività e le azioni che il sistema esegue quando si spara, in particolare quando la palla infuocata colpisce il terreno di gioco (ground) oppure quando colpisce in nemico o un ostacolo, queste due attività sono state scomposte per descrivere in maniera più analitica l’algoritmo di controllo.

****

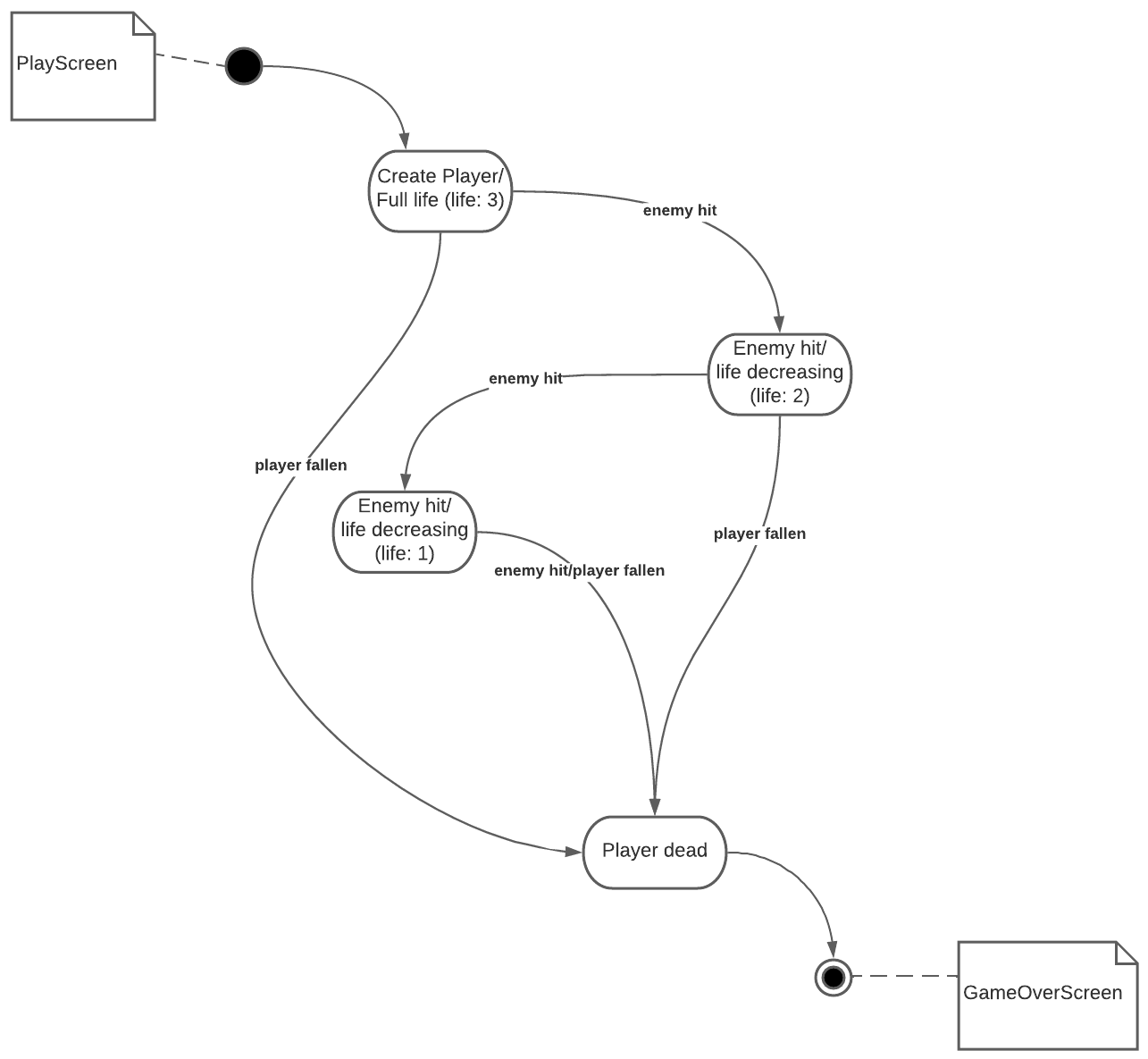
****

****

1. **State Diagram**

Lo State Diagram raffigura gli stati che un determinato oggetto assume nel tempo.

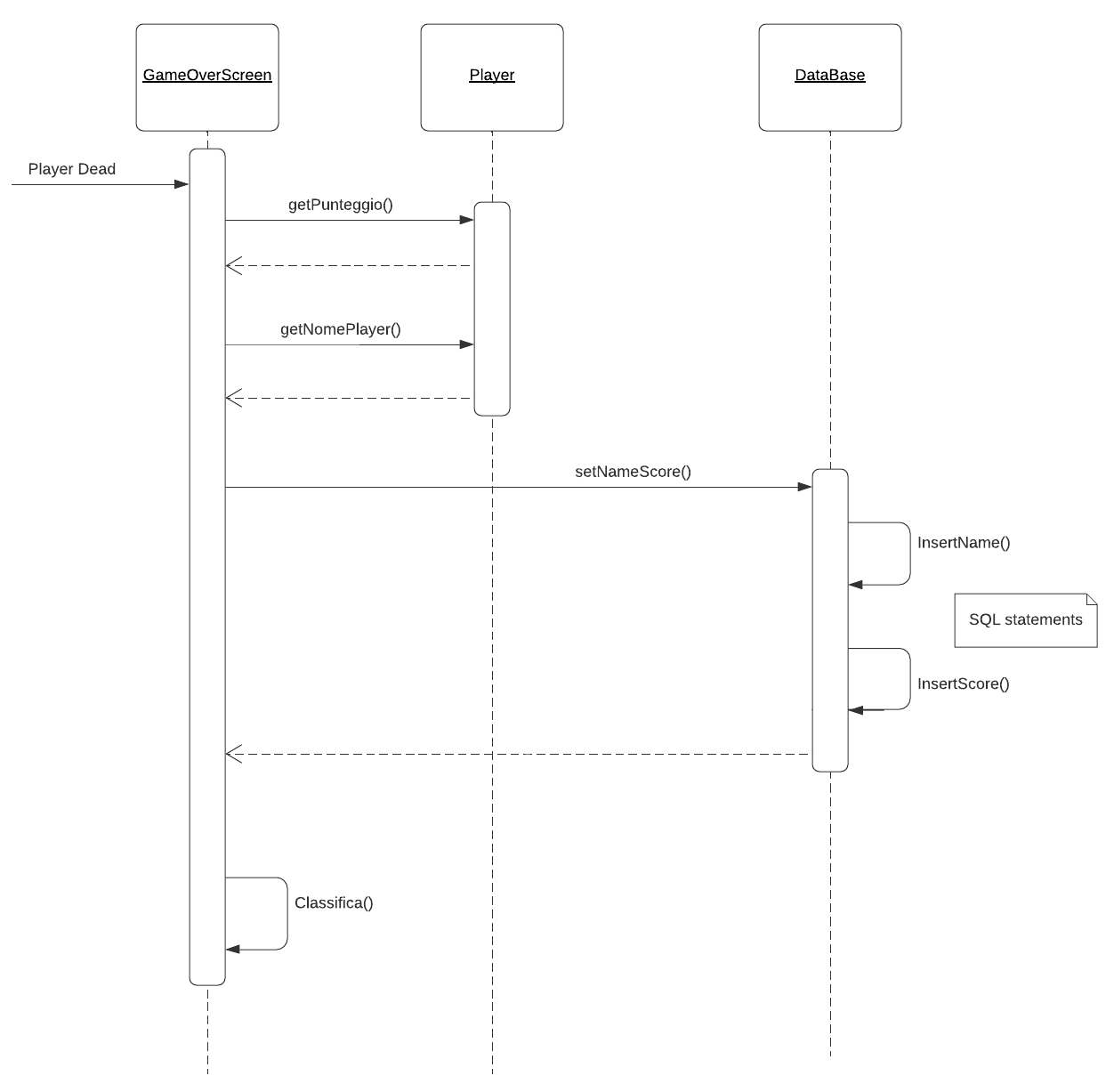
In particolare si è scelto di rappresentare gli stati che assume il Player nel corso di una partita.

****

1. **Sequence Diagram**

Il Sequence Diagram va a rappresentare come le varie classi e oggetti interagiscono tra di loro nel tempo.

In particolare nel diagramma seguente si vuole descrivere il funzionamento del sistema con il database dopo aver terminato una partita.



Tutti i diagrammi mostrati in questo elaborato sono stati fatti con il software Lucidchart.

**Design Pattern**

In questa sezione verrà descritto un particolare Design Pattern per l’aggiunta di nuove tipologie di personaggi.

Il Design Pattern in questione è il Decorator Pattern:

Questo pattern è relativo agli oggetti, definito per composizione e creato a runtime (dinamico e flessibile) e di tipo strutturale. Rende possibile dare ad un oggetto nuove features dinamicamente senza apportare modifiche al codice della classe sottostante. I decoratori hanno lo stesso super tipo degli oggetti decorati. Utilizza il principio SOLID dell’Open Closed.

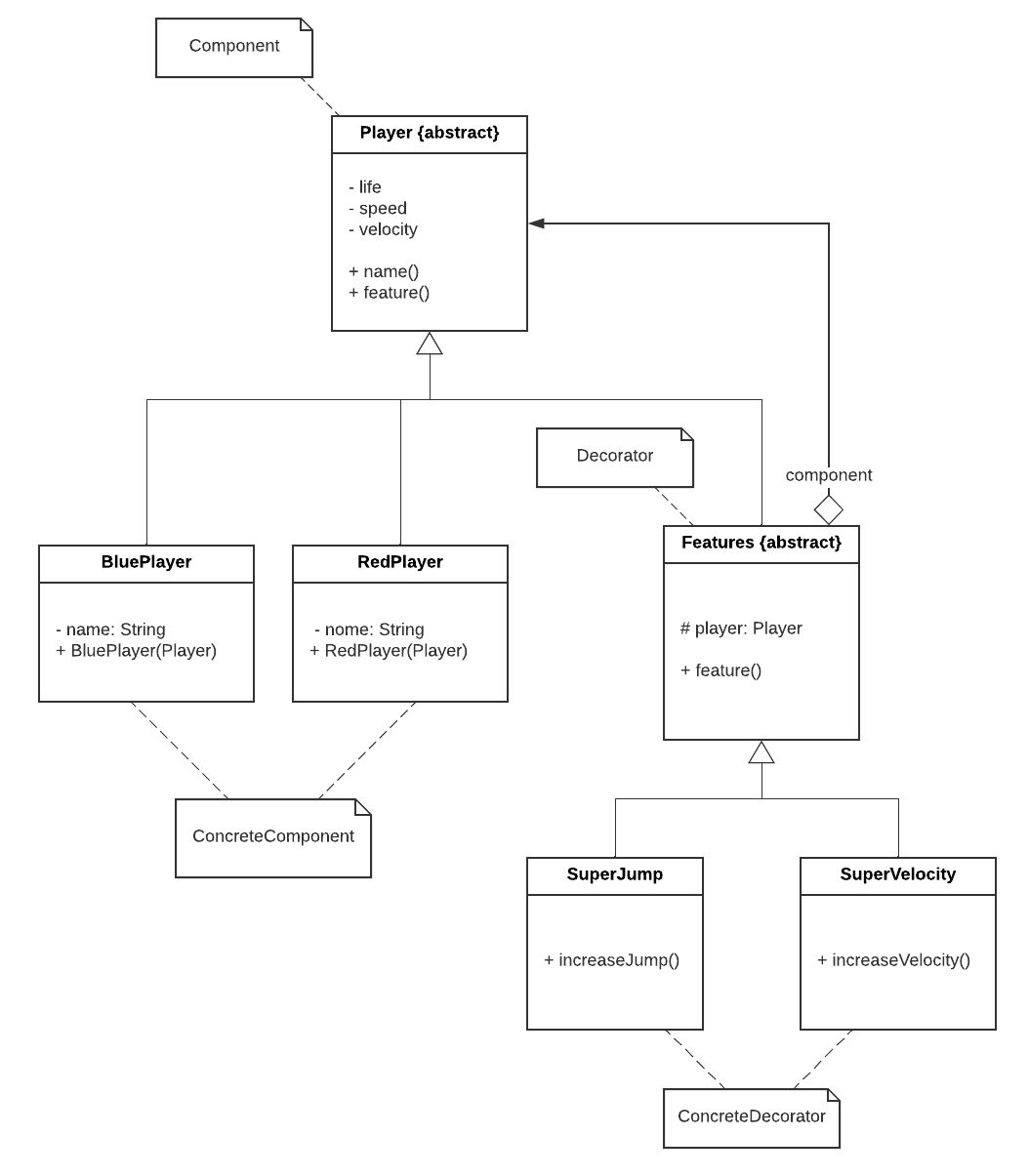
INTENT: Aggiungere funzionalità aggiuntive ad un oggetto dinamicamente, è una valida alternativa all’estensione del codice di una sottoclasse, rende maggiore la flessibilità.

APPLICABILITY: Quando ho la necessità di aggiungere funzionalità ad un oggetto dinamicamente senza avere effetti su altri oggetti. Quando l’estensione attraverso sottoclassi è impraticabile.

IMPLEMENTATION: Utilizza un’interfaccia e delle classi concrete che implementano tale interfaccia, una classe decorator astratta che implementa anch’essa l’interfaccia precedente e ne contiene un’istanza. Servirà poi anche una classe che implementerà la classe decorator per decorare gli oggetti concreti.

Immagine che contiene screenshot

Descrizione generata automaticamente



Implementazione delle classi principali, non completa, del pattern scelto:

public abstract class Player{

private int life = 3;

private int jumpHight = 6;

private int velocity = 3;

private String name = “”;

public String name(){

return name;

}

public abstract void features();

}

public abstract class Features extends Player{

Player player;

public void features(){

System.out.println(“features”);

}

}

public class BluePlayer extends Player{

private String name;

public BluePlayer(){

this.name = “Blue”;

}

public void features(){

//features

}

}

public class SuperJump extends Features{

public SuperJump(Player player){

this.player = player;

}

public String name(){

return player.name() + “superJump”;

}

public void features(){

System.out.println(player.features() + “superJump”);

}

}