

UNIPAC-MAN

Bellosi Jacopo (m.1081058)

Longhi Lara (m. 1079261)

Poloni Luca (m. 1078817)

START!



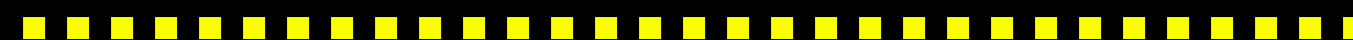
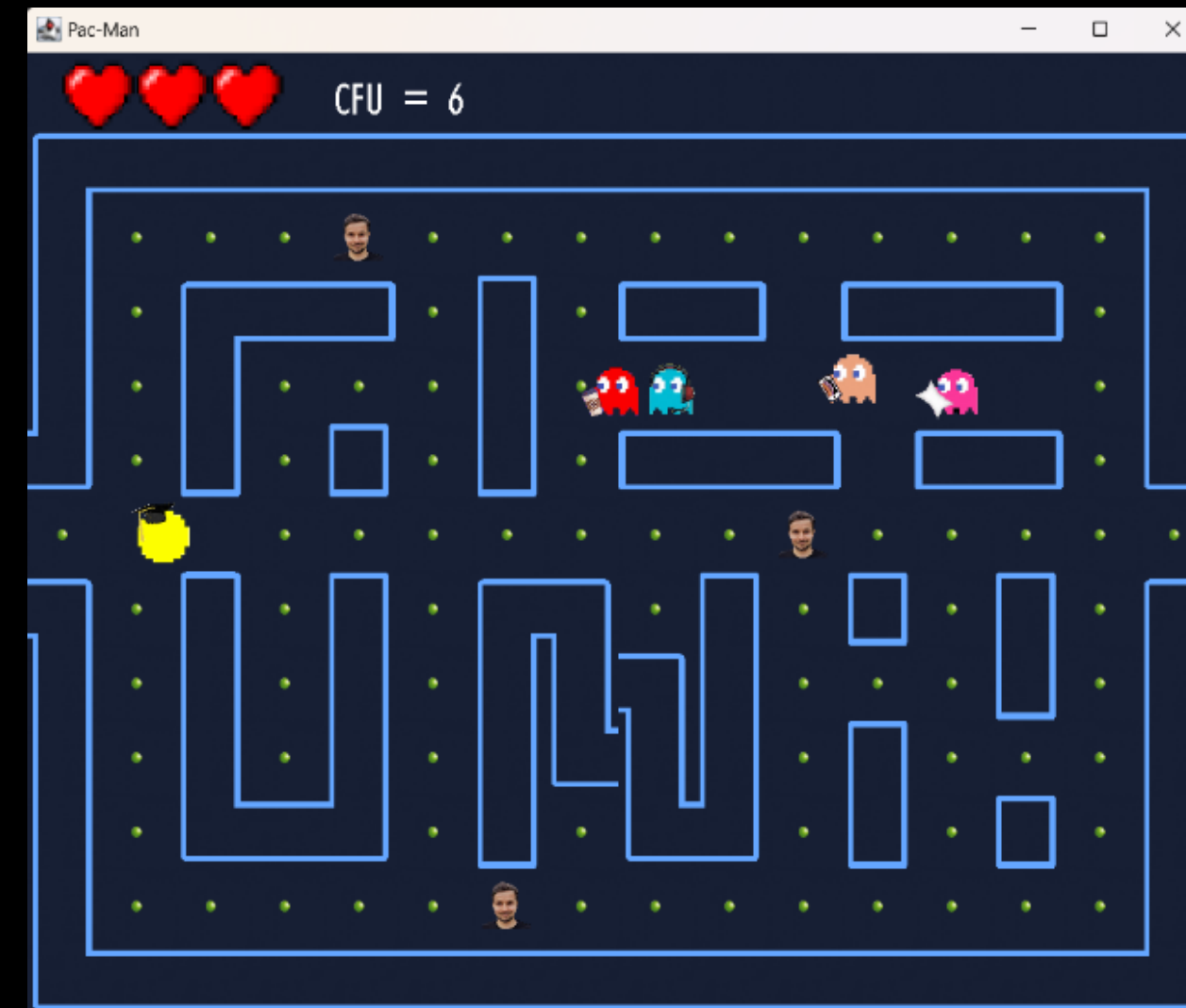
OBIETTIVO



UniPac-Man è un applicativo Java che vuole emulare il celebre gioco arcade anni '80 rivisitandolo in chiave universitaria. Il gioco è composto da tre livelli (anni accademici) e per potersi laureare bisogna raccogliere tutti i CFU presenti in ogni anno. Ma attenzione perchè i fantasmi della procrastinazione cercheranno in tutti i modi di rallentarti per non farti laureare in corso.

DIFFICOLTÀ INCONTRATE

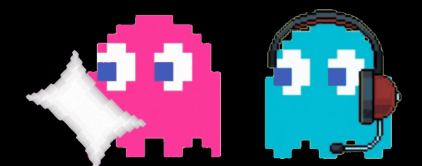
- Prendere confidenza con la libreria grafica SWING.
- Gestire il movimento dei fantasmi.
- Comprendere il funzionamento di Git
- A livello di comunicazione non ci sono state problematiche



PARADIGMI UTILIZZATI



L'applicativo è stato scritto interamente in linguaggio **JAVA** con l'utilizzo di oggetti supportata dalla libreria grafica **SWING**. Abbiamo utilizzato **Eclipse** supportato da plugin come **UcDetector** e **CodeMR**. La modellazione è avvenuta tramite il software **StarUML** utilizzando il linguaggio di modellazione **UML**.



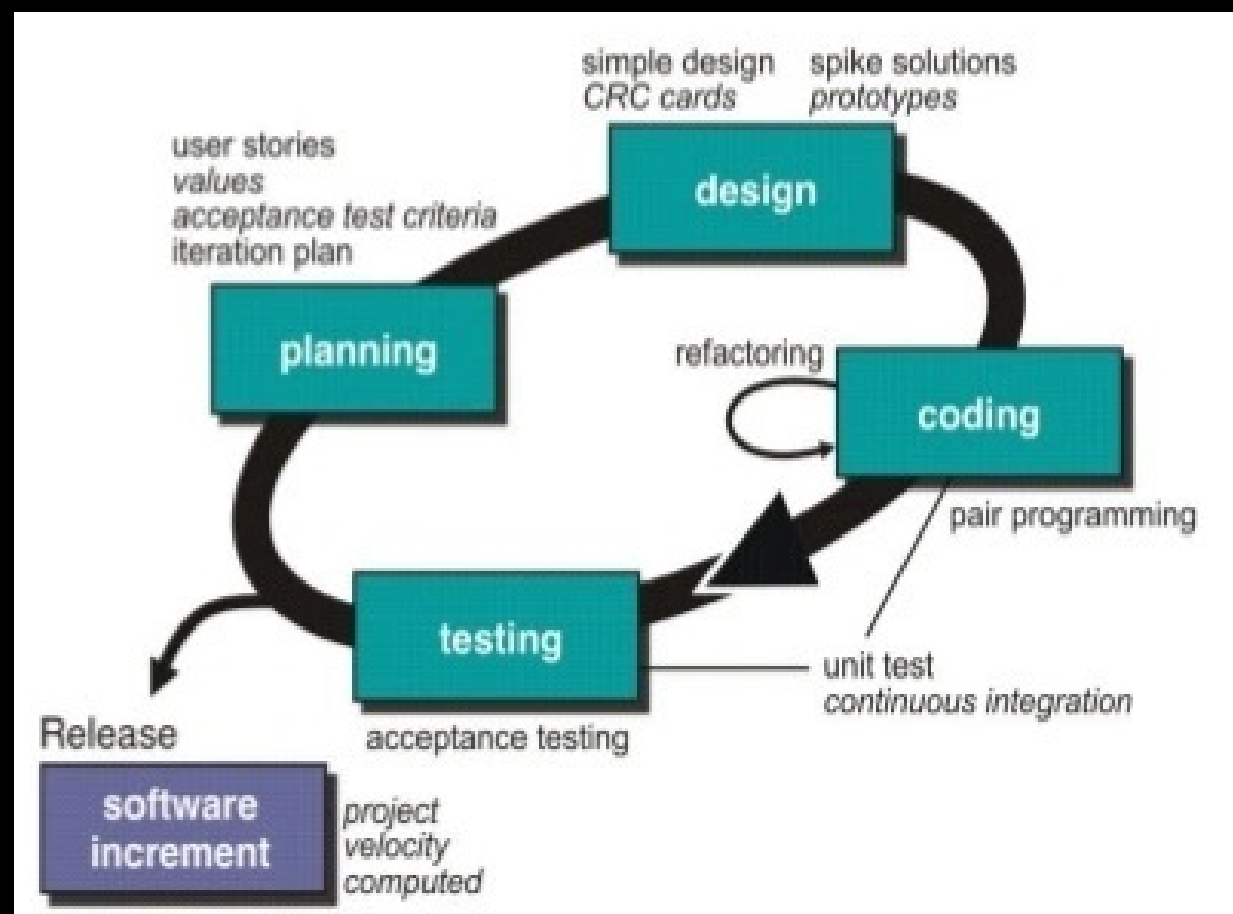
SOFTWARE CONFIGURATION MANAGEMENT



Abbiamo utilizzato **GitHub** come software configuration management. Ogni membro del team ha potuto tenere traccia dell'avanzamento del progetto, potendo caricare la propria versione della repository locale (**PUSH** request) oppure scaricarne una aggiornata (**PULL** request).

Per modificare parti sostanziose del codice e per il refactoring sono stati svolti dei **BRANCH**.

SOFTWARE LIFE CYCLE

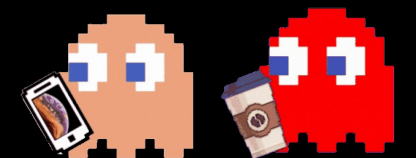


Il modello di processo per l'applicativo è un metodo AGILE, nello specifico extreme Programming:

- ♥ Whole Team
- ♥ Cliente in loco
- ♥ Pair programming
- ♥ Proprietà collettiva
- ♥ Sviluppo guidato dai test

ANALISI DEI REQUISITI

Abbiamo individuato e suddiviso i requisiti del nostro applicativo secondo il criterio **MoSCoW**:



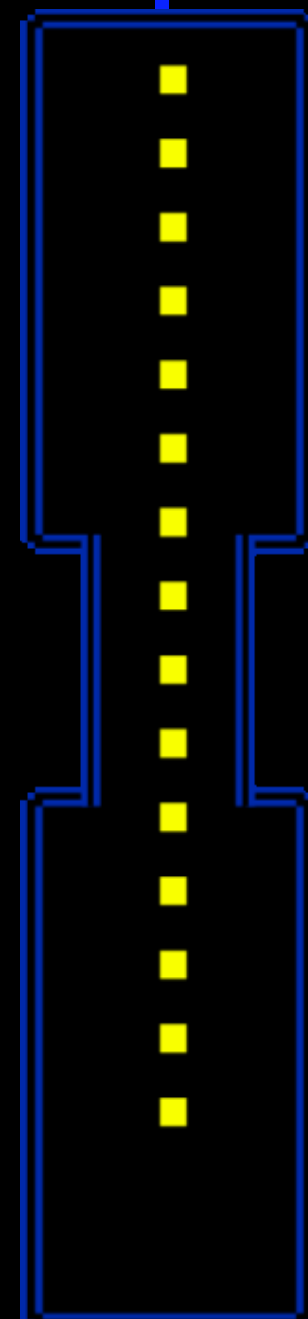
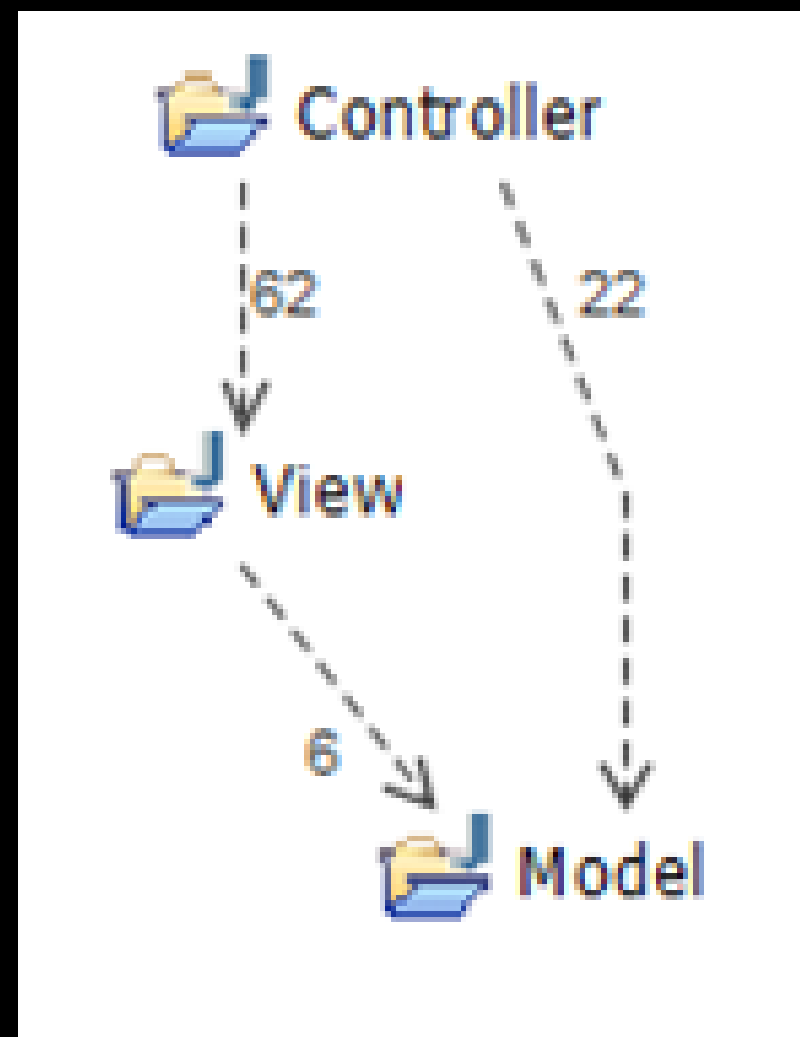
IMPLEMENTAZIONE

Rispetto ai requisiti siamo riusciti ad implementare le seguenti funzionalità e dinamiche:

- ♥ Interfaccia funzionante
- ♥ 3 livelli (uno per anno universitario)
- ♥ Suoni e colonne sonore
- ♥ Power-up
- ♥ Mappe personalizzate
- ♥ Tunnel
- ♥ Movimento randomico fantasmini della procrastinazione
- ♥ Movimento intelligente fantasmini della procrastinazione
- ♥ Classifica, punteggi e storico partite

ARCHITETTURA

Stile architetturale MVC (Model-View-Controller)



DESIGN PATTERN

Singleton Pattern : Pattern principale, utilizzato nella classe Player per garantire la presenza di un solo PacMan all'interno del gioco. È accessibile da tutte le classi per poter condividere informazioni di gioco.

Factory Pattern : Introdotto nella classe SetSetter per creare tutti gli oggetti di gioco nella fase di caricamento.

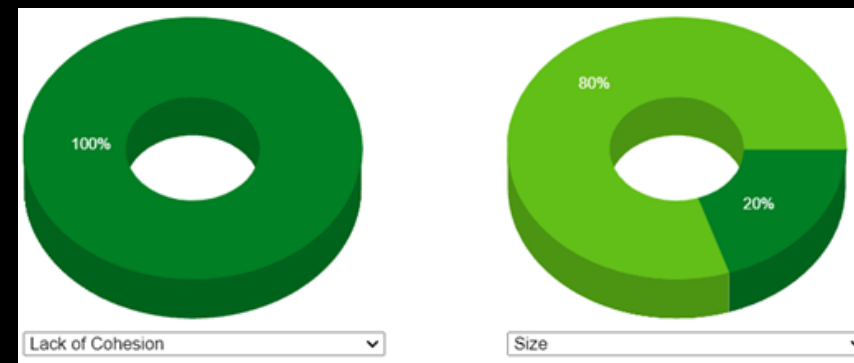
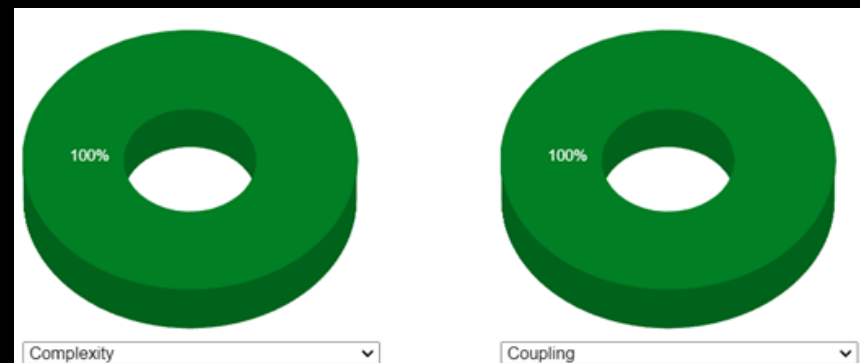
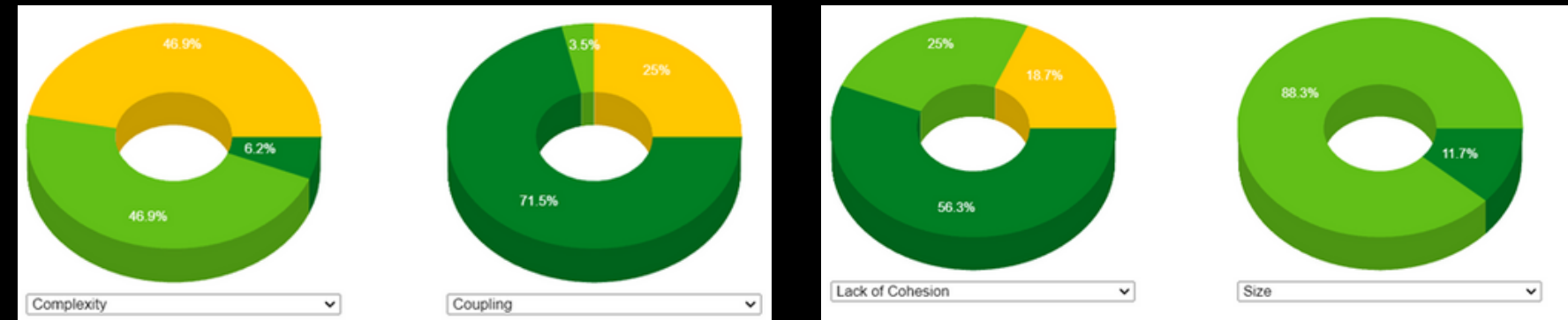
Abstract Pattern : GameEngine contiene tutti i dati principali, dai dati di gioco, alle posizioni, al passaggio dei livelli e li condivide a tutte le altre classi.



METRICHE DI QUALITÀ

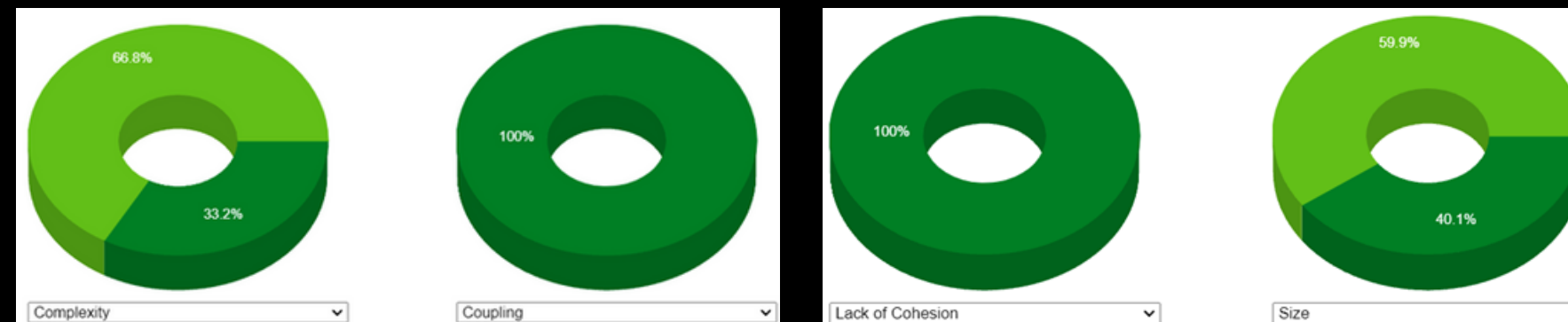
Attraverso il tool per Eclipse CodeMR abbiamo analizzato complessità, coesione e accoppiamento delle classi all'interno dei tre progetti Maven (MVC):

CONTROLLER -->

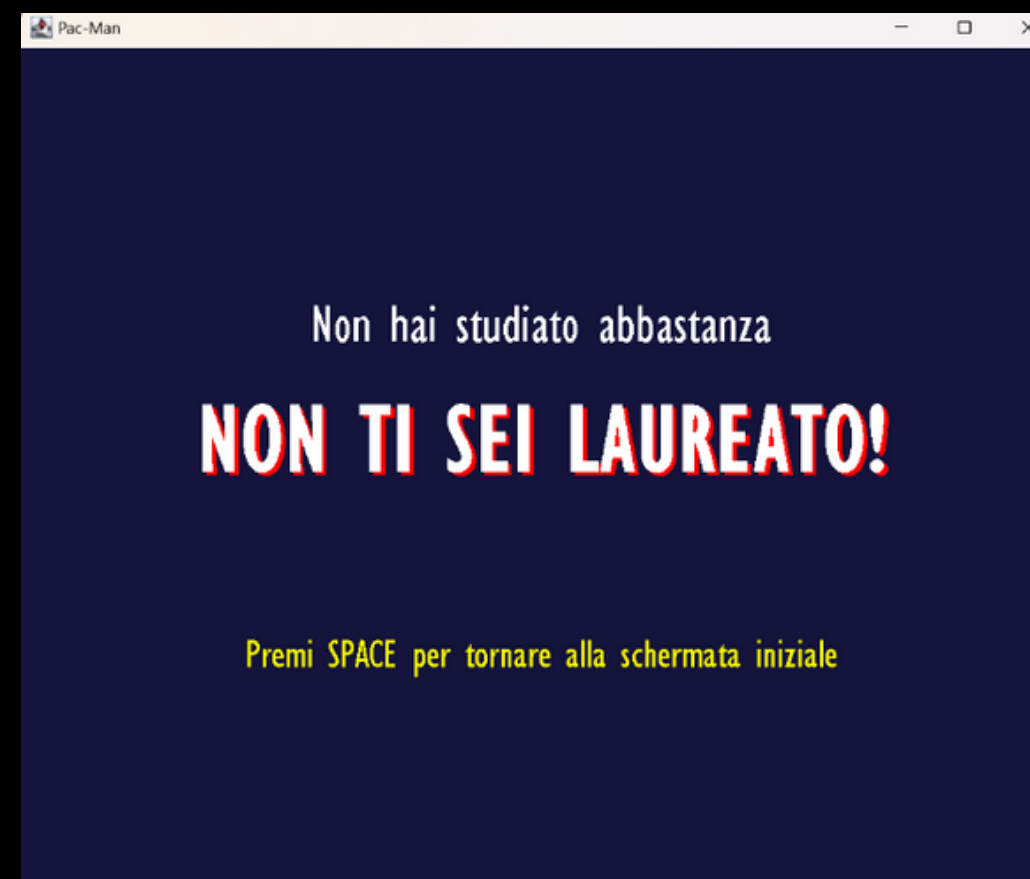
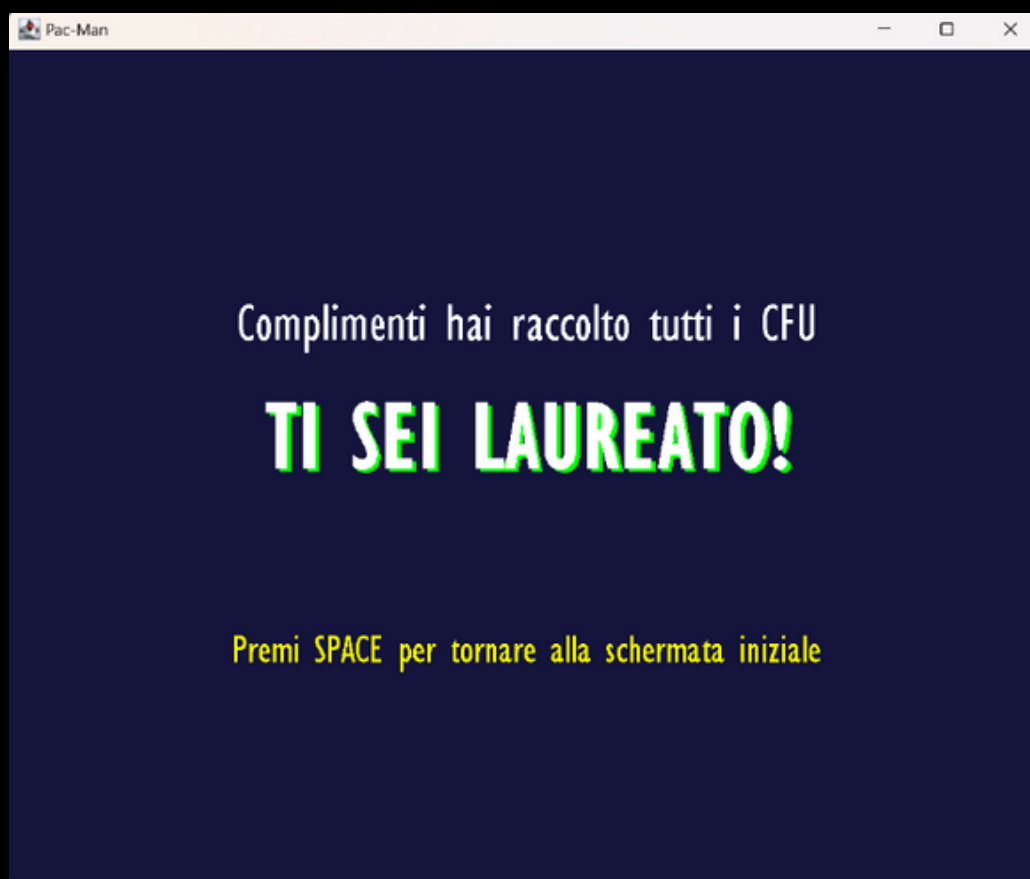
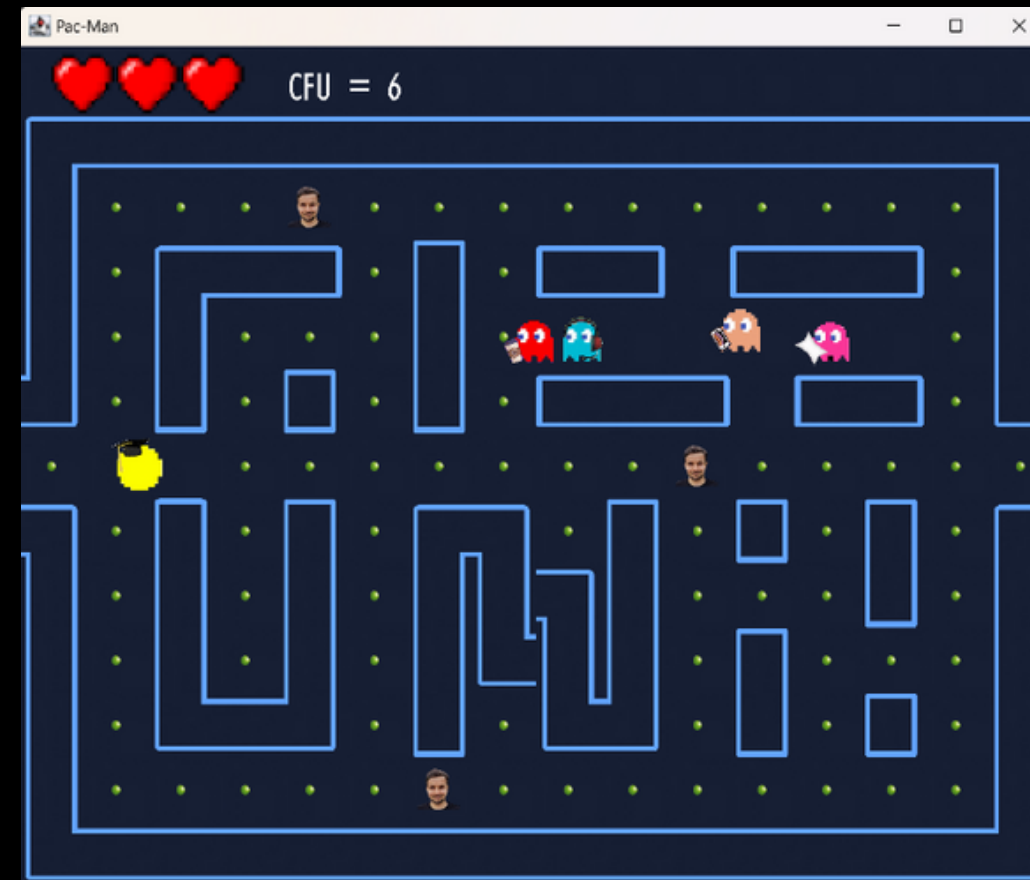


<-- MODEL

VIEW -->



DEMO



MODELLAZIONE

Attraverso StarUML abbiamo utilizzato i seguenti modelli:

Diagramma delle classi

Diagramma dei casi d'uso

Diagramma della macchina a stati

Diagramma di sequenza

Diagramma delle attività

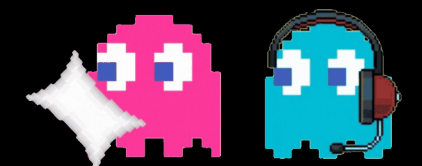


DIAGRAMMA DI STATO

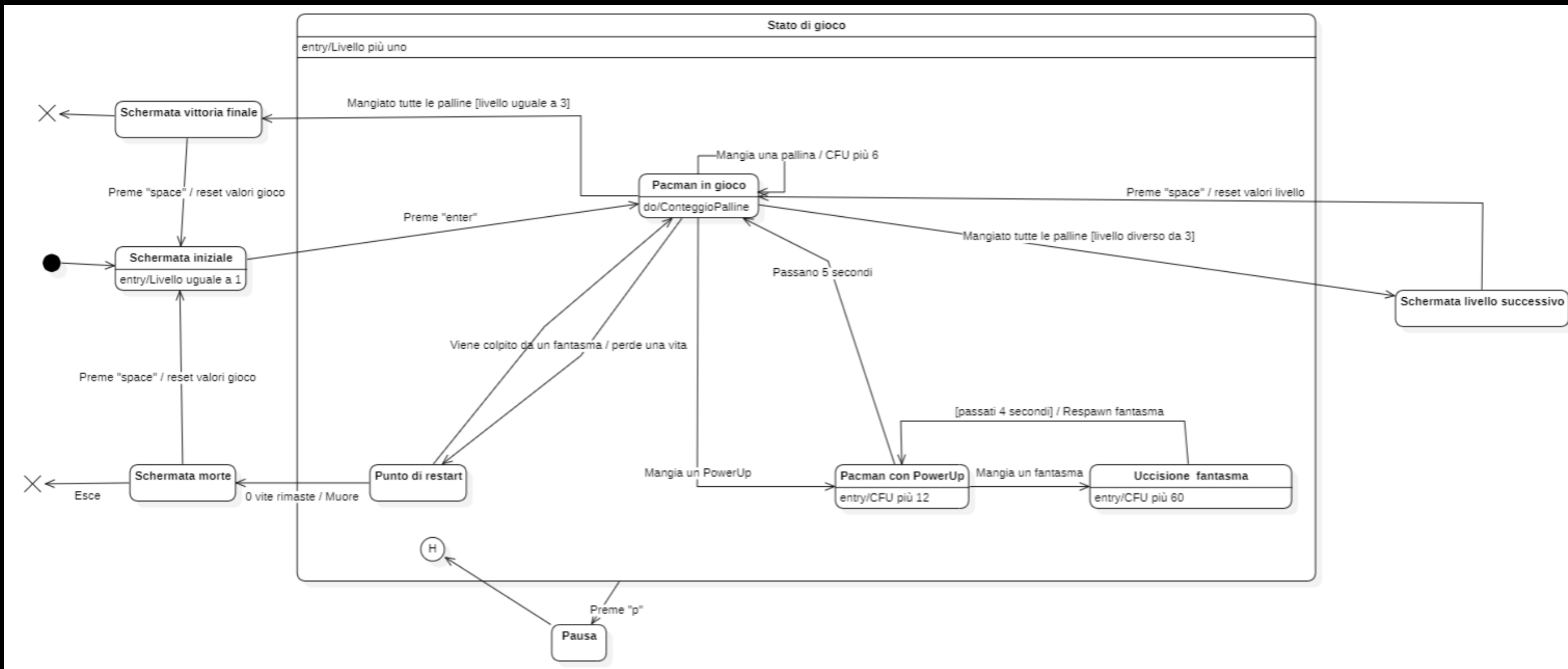


DIAGRAMMA DEI CASI D'USO

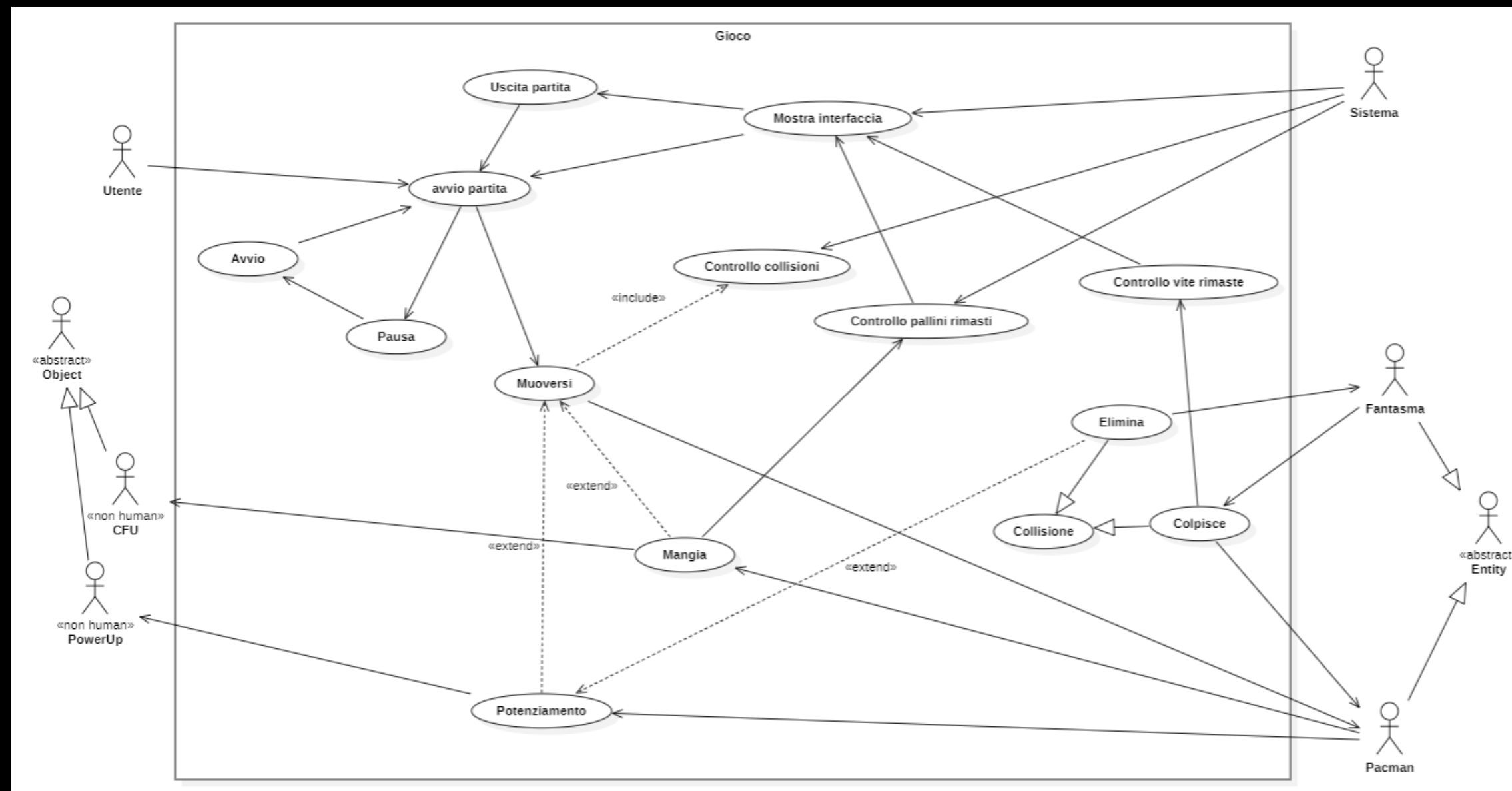
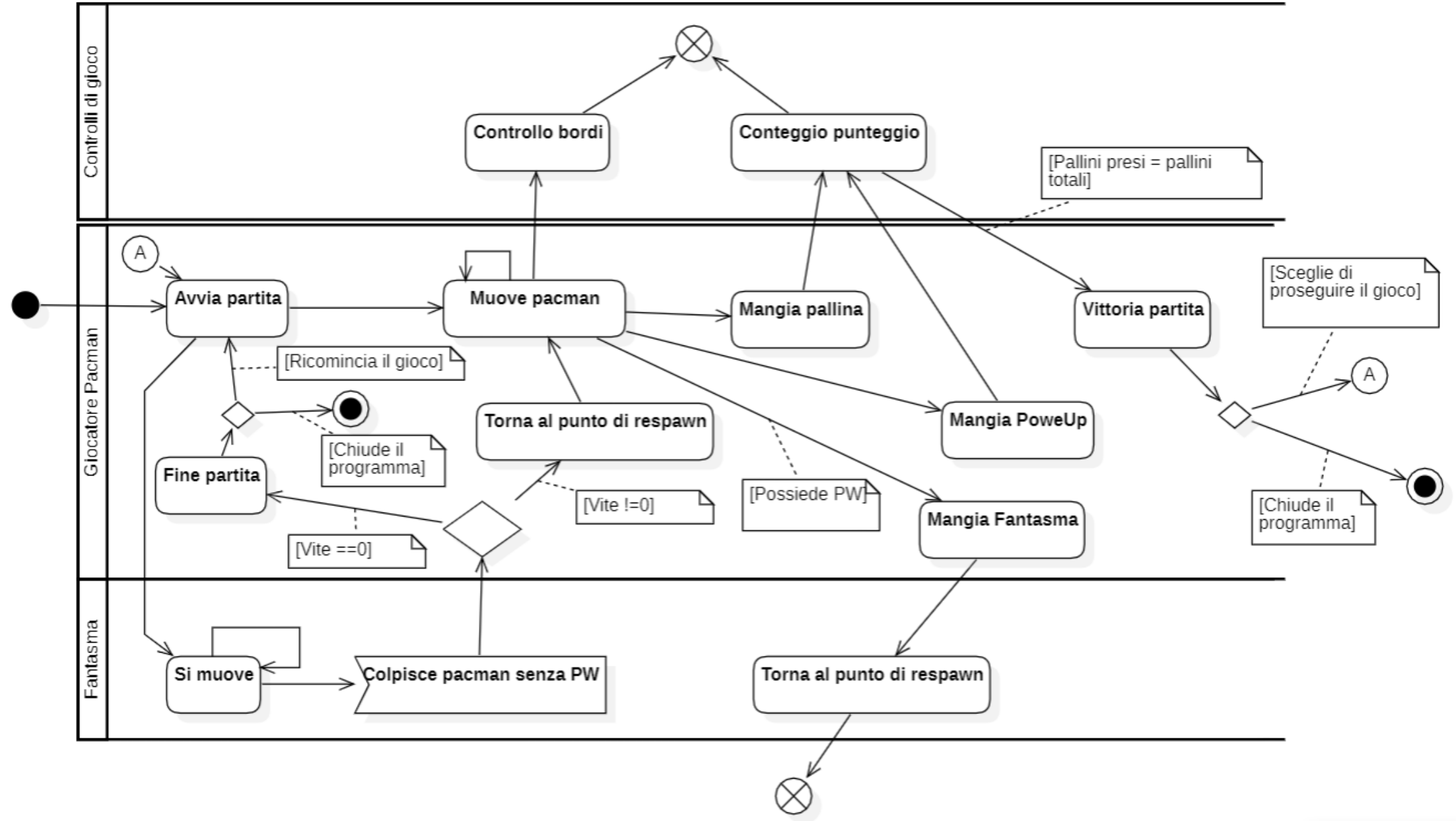


DIAGRAMMA DELLE ATTIVITÀ



MANUTENZIONE

È stato utilizzato il processo di **refactoring** per garantire un codice più ottimizzato, leggibile, poco complesso e a basso costo di manutenzione. È stato utilizzato **UCDetector** per ripulire il codice.

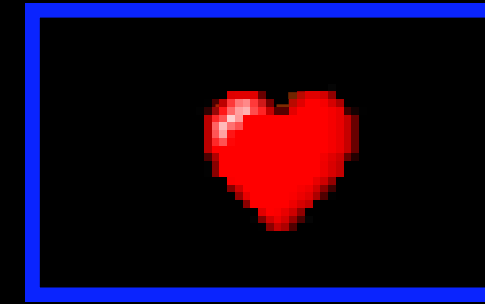
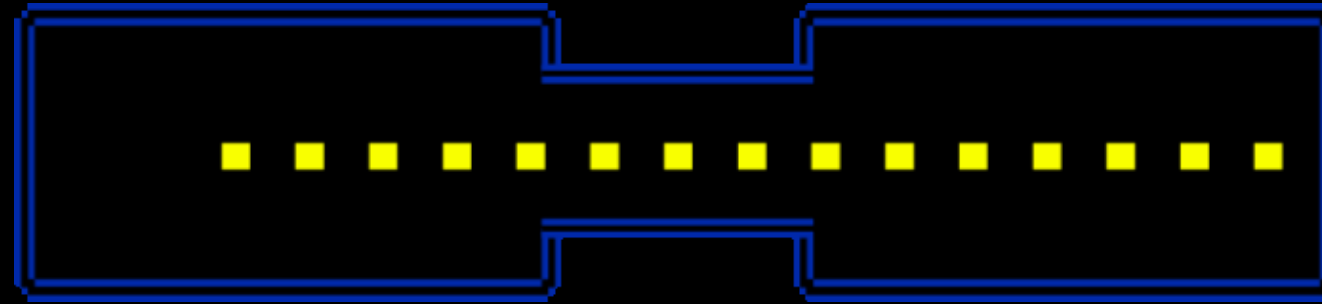
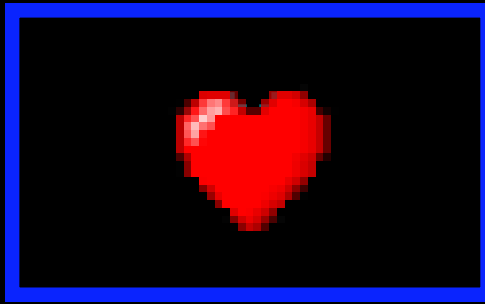


TESTING



Abbiamo svolto dei test prevalentemente unitari in **JUnit** prevalentemente sui metodi delle classi del package Controller. Ecco alcuni casi di test eseguiti:

- ♥ Movimento di Pacman
- ♥ Vite di default
- ♥ Aumento livello
- ♥ Morte fantasma
- ♥ Suoni vittoria
- ♥ Collisione fantasma-Pacman
- ♥ Default setting fantasmi
- ♥ ...



GRAZIE PER
L'ATTENZIONE

START!

