Home sweet Home

# Introduzione

Home Sweet Home è un gioco realizzato in Java. Lo scopo del gioco è proteggere la casa dagli Zombie impedendogli di sfondare la porta. Per ogni Zombie ucciso il giocatore guadagna dei punti.

Il gioco è suddiviso in livelli di difficoltà crescente e termina solo con l’uccisione di tutti gli Zombie o se gli Zombie distruggeranno completamente la porta della casa.

# Istruzioni

* Per muoversi si usa: W/↑ S/↓ A/← D/→
* Per utilizzare il martello contro gli zombie o riparare la porta si usa: ENTER/SPACE
* Per iniziare il gioco: SPACE

# Composizione del gioco

Il gioco è realizzando seguendo, ove possibile, il pattern MVC.

### L’interfaccia utente

Le classi **BackgroundView**, **GameOverView**, **GamePauseView**, **PlayerView** e **ZombieView** si occupano di disegnare varie parti dell’interfaccia grafica.

La classe **GameView**, con estensione Panel, svolge il ruolo fondamentale di unire gli output di tutte le altre view per realizzare, con la funzione paint, l’interfaccia grafica completa. Inoltre, implementa **Runnable** per utilizzare un Thread con lo scopo di “rendere il gioco animato” tramite vari repaint.

### La gestione del gioco

Le classi che hanno un ruolo fondamentale nella gestione del gioco sono:

* **GameSettings**: utilizzata all’inizio del gioco, per dare la possibilità al Giocatore di selezionare la dimensione della finestra e il nome utente
* **GameWindow**: estende JFrame, per creare la finestra e implementa KeyListener, per ricevere i comandi da tastiera
* **GameLevel**: controlla il passaggio del gioco ad un livello successivo
* **GameScore**: gestisce il punteggio

### Le classi Player

Le classi che riguardano il Player sono:

* **PlayerController** estende PlayerView. La sua funzione è controllare e animare il player.
* **PlayerModel** contiene le variabili del Player e dei metodi get & set per accedervi (coordinate, vita, forza, velocità).

### Le classi riguardanti gli Zombie

Le classi che riguardano gli Zombie sono:

* **ZombieModel** è un “nodo" all’interno di una collezione che contiene le variabili del singolo Zombie e dei metodi per interagirvi (coordinate, vita, forza, velocità).
* **ZombiesController** implementa Runnable, il suo compito è quello della creazione e gestione della lista di **ZombieModel** mediante l’uso di un Thread. Inoltre, dà l’input di fine e di inizio livello.