Progetto di Paradigmi di Programmazione

Realizzazione di Battaglia Navale con funzioni opz

Loris Farrauto, Stefano Alcamo

**1.1 Lettura del progetto**

Siamo partiti con il leggere i requisiti dati dal testo del progetto.

Abbiamo deciso di sviluppare una prima versione del progetto unicamente con la vista, così da avere un’idea chiara di come sarebbe dovuto funzionare il gioco.

**2.1 Sviluppo di battagliaNavaleView**

La view è composta dai 2 campi da gioco (uno dove il giocatore ha posizionato le navi e l’altro dove spara alle navi della cpu) che hanno dimensioni fisse e dai bottoni necessari per posizionare le navi e sparare a quelle della cpu.

La vista presenta un timer, gestito da un controller a parte, che simboleggia il tempo rimasto a disposizione al giocatore per fare la sua mossa; inoltre ogni interazione del gioco con l’utente è presentata da una label di testo il cui contenuto viene modificato in base alla situazione corrente.

Infine è possibile interagire con un JMunuBar per accedere alle funzionalità salva/carica/nuova\_partita e visualizzare le informazioni inerenti al gioco e agli sviluppatori.

**2.2 Sviluppo della gestione vista navi e spari**

Man mano che le navi vengono piazzate, viene mostrato l’inserimento sulla griglia di sinistra con la colorazione in grigio di un numero di quadratini pari alla lunghezza della nave e disposti in base alla direzione scelta dall’utente.

Dopo ogni colpo sparato dalla cpu o dal giocatore, vengono posizionate delle “X” sulle rispettive griglie per rappresentare il punto colpito, se tale punto risulta appartenente ad una nave colpita e affondata, tutti i suoi quadratini vengono colorati di rosso, mentre se il danno non è letale per la nave, il quadratino viene solamente colorato di grigio.

**3.1 Sviluppo di battagliaNavaleModel**

Il modello dispone di classi di controllo, classe per la creazione degli oggetti “nave”, classe astratta “sfidanti” estesa da “cpu” e “giocatore” e della classe principale del modello.

La classe principale genera gli oggetti “giocatore” e “cpu” e gestisce le meccaniche inerenti al posizionamento delle navi e degli spari sulla mappa, inoltre tiene in memoria tutti i risultati della partita che si sta svolgendo.

Le classi di controllo servono a garantire il posizionamento corretto delle navi e degli spari proposti confrontandoli con le opzioni consentite dal gioco.

La classe “nave” contiene tutte le informazioni appartenenti alle navi che vengono posizionate sul campo di battaglia (coordinate, numero di parti integre rimanenti, orientamento, tipologia e il suo id).

Le classi “cpu” e “giocatore” contengono la lista delle navi che gli appartengono e la lista degli spari effettuati.

La classe “cpu” dispone inoltre di una piccola intelligenza artificiale in grado di selezionare un bersaglio e di posizionare le navi randomicamente.

La classe “SalvaCaricaDati” contiene tutti i dati sensibili necessari all’inizializzazione di una partita caricata.

**4.1 Sviluppo di battagliaNavaleController**

Il controller è costituito da 3 classi, ”BattagliaNavaleController”, “MenuOptions” e “TimerGioco”.

La classe “BattagliaNavaleController” gestisce le azioni attribuite ai pulsanti necessari per posizionare le navi e sparare, aziona le funzionalità del model quando è necessario e in risposta aggiorna la view; inoltre contiene i metodi necessari al caricamento della partita.

La classe “MenuOptions” contiene i gestori per creare una nuova partita, per caricare una partita, per salvare una partita, per chiudere il gioco, per leggere le istruzioni del gioco e per vedere i crediti relativi agli sviluppatori del gioco; inoltre contiene i metodi effettivi per salvare e caricare una partita.

La classe “TimerGioco” gestisce il tempo che ha il giocatore per finire il proprio turno.