# Documentazione progetto *battaglia navale* in C++

Mauro Riva [1053644]

A.A. 2021/2022

# 1 Introduzione

Il progetto si pone l'obiettivo di realizzare il celebre gioco da tavolo *battaglia navale* sulla console di Windows tramite un programma sviluppato in C++. Il gioco consiste in uno scontro tra due giocatori, ognuno dei quali ha a disposizione una tabella 10x10 sulla quale deve posizionare verticalmente o orizzontalmente 6 navi di varie dimensioni: la più grande è la portaerei che occupa 5 caselle, seguita dalla corazzata con 4, due incrociatori da 3, un sottomarino da 3 e un cacciatorpediniere da 2. Ad ogni turno il giocatore seleziona una casella da colpire dalla tabella dell'avversario. Il colpo può avere tre risultati: *mancato* quando nella casella non è presente nessuna nave, *colpito* quando una nave viene centrata e *affondato* se la nave è stata colpita in tutte le caselle che occupa. La partita prosegue finché un giocatore affonda tutta la flotta dell'avversario.

# 2 Classi

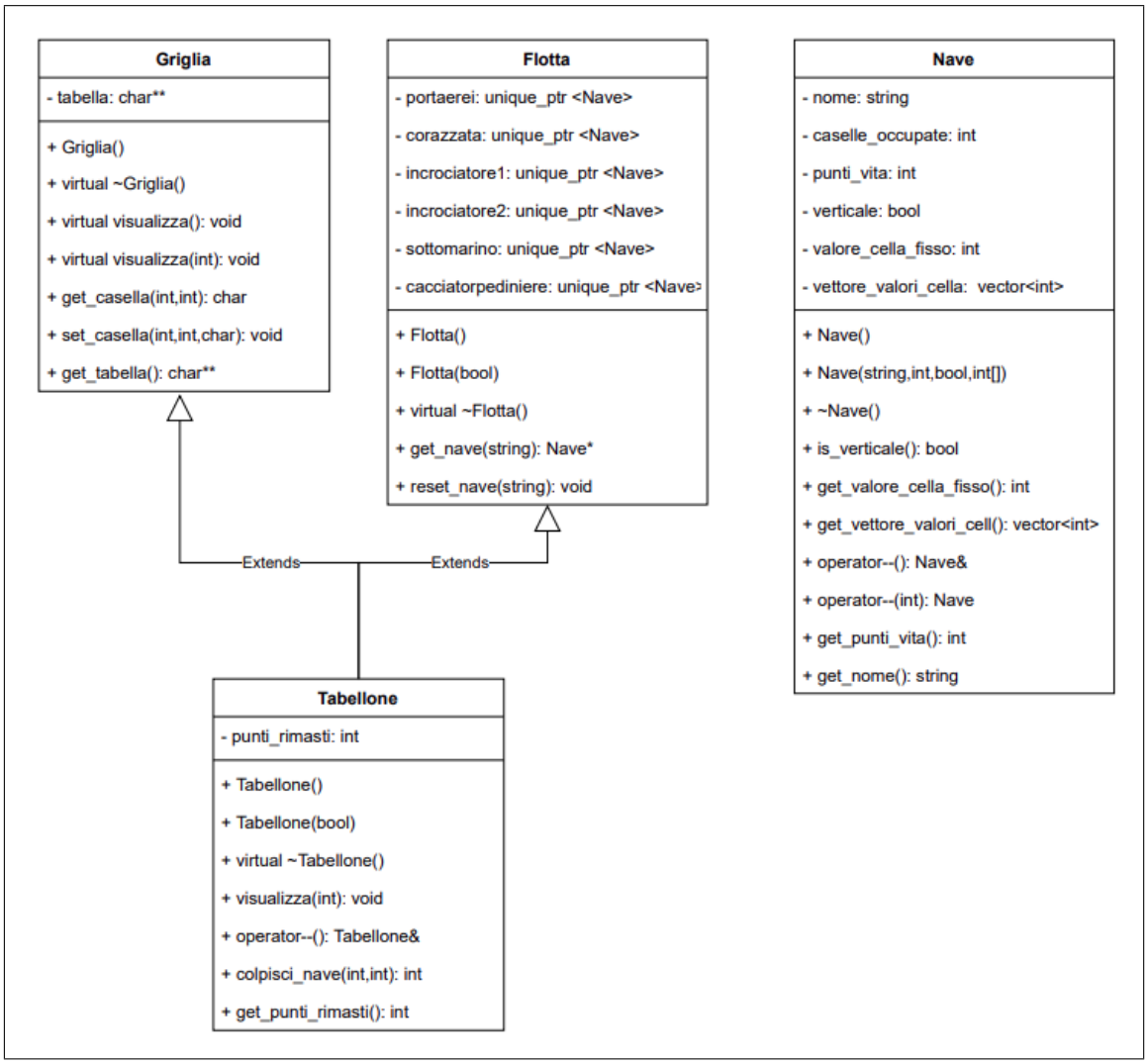


Figura 1: Diagramma UML delle classi

La classe *Tabellone* sfrutta il principio dell'ereditarietà multipla del C++ essendo sottoclasse sia di *Griglia* sia di *Flotta*. L'ereditarietà di *Tabellone* è di tipo pubblico per entrambe le classi.

# 3 Menù e selezione modalità di gioco


Figura 2: Menù principale

All'avvio del programma viene richiesto all'utente di scegliere tra le modalità *singleplayer* e *multiplayer* inserendo il numero 1 o il numero 2: questa parte dell'applicazione è gestita della funzione *int menu()* dichiarata nel file *Battaglia navale.cpp*.

# 4 Posizionamento della flotta


Figura 3: Inserimento di una nave

Dopo aver scelto la modalità di gioco vengono create le due istanze della classe *Tabellone*: il posizionamento delle navi da parte dell'utente o l'inserimento casuale da parte del computer sono gestite dal costruttore della classe *Flotta*. L'inserimento della riga e della

colonna dalla console utilizza una funzione template *input_cella(T)* che restituisce un intero tra 0 e 9 qualora il parametro passato in input (che può essere un intero compreso tra 1 e 10 o un carattere tra 'A' e 'J') sia del valore corretto, -1 altrimenti.

# 5 Turno giocatore

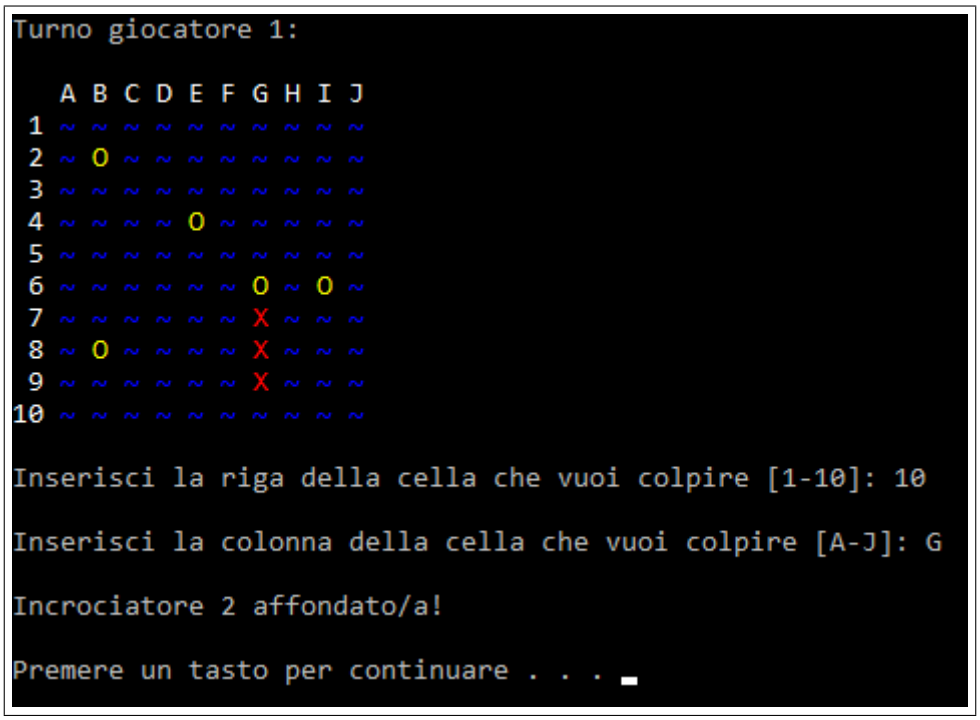


Figura 4: Turno giocatore

Ad ogni turno viene chiesto ai giocatori di selezionare una casella da colpire. La tabella mostrata al giocatore 1 è quella del giocatore 2 (e viceversa). La visualizzazione della tabella senza mostrare la posizione delle navi è stata implementata nelle classe *Tabellone* facendo l'overriding del metodo virtual *visualizzazione(int)* di *Griglia*. Le caselle selezionate che risultano vuote sono contrassegnate con una 'O' in giallo, mentre quelle dove è presente una nave con una 'X' rossa. Quando una nave viene affondata si procede a resettare il relativo smart pointer, de-allocando quindi dallo HEAP la memoria occupata dall'oggetto puntato dal puntatore.

# 6 Turno computer


Figura 5: Turno computer

Per rendere la modalità *singleplayer* sfidante è stato necessario implementare un algoritmo che permetta al computer di avere uno stile di gioco simile a quello di un giocatore umano: se il computer si fosse limitato a colpire caselle in modo casuale sarebbe stato praticamente impossibile per il giocatore perdere la partita. Inizialmente la scelta della casella avviene in modo casuale, nell'esempio riportato in figura 5 il primo colpo in I10 è andato a vuoto, mentre il secondo in C5 ha colpito quella che noi sappiamo essere la nostra portaerei. A questo punto la modalità di scelta casuale delle caselle viene sostituita da una modalità di scelta privilegiata. Ci sono due liste di interi, una per le righe e una per le colonne, per rappresentare le caselle. L'associazione tra riga e colonna di una casella memorizzata nelle liste è garantita dalla posizione dell'elemento all'interno delle liste stesse. Quando una nave viene colpita per la prima volta vengono memorizzate le 4 caselle che circondano il colpo andato a segno, tranne se eccedono i limiti della tabella o se sono già state colpite in precedenza. Nei turni successivi il computer andrà a colpire le caselle prioritarie finché le avrà esaurite tutte o avrà affondato una nave. Si inizia con la casella a destra, che nell'esempio è D5. Dato che il colpo non è andato a segno si prova con la casella sotto, C6. In questo caso abbiamo colpito nuovamente la nave senza affondarla (a meno che si tratti di una seconda nave) e sappiamo che è posizionata in verticale. Possiamo quindi procedere ad eliminare tutte le caselle presenti nelle liste che hanno una colonna diversa

da C. Questo processo è effettuato tramite 2 iteratori che scorrono le liste parallelamente. Prima di passare al turno successivo si aggiunge alle liste la casella sottostante quella colpita, ossia C7. In C7 la nave viene nuovamente colpita, quindi al turno seguente si prova con C8, in cui non è presente la nave. A questo punto il computer torna a scorrere le caselle rimaste nella lista, selezionando C4, che è l'ultima necessaria ad affondare la portaerei. Ora che la nave è stata affondata le liste vengono svuotate e al turno successivo si ritorna alla modalità casuale.

# 7 Costrutti utilizzati

- Costruttori e Distruttori: Presenti in tutte le classi implementate;
- Campi pubblici e privati: Attributi definiti privati, metodi definiti pubblici;
- Membri virtual: Distruttori di *Griglia*, *Flotta* e *Tabellone*, metodo *visualizza(int)* di *Griglia*. Si noti la funzione *visualizza_tabella(Griglia*)* in *Battaglia navale.cpp* a cui viene passato un puntatore a un oggetto *Tabellone*;
- Overloading: Metodi *visualizza* nella classe *Griglia* e costruttori;
- Overloading degli operatori: Effettuato sull'operatore - - nelle classi *Nave* e *Tabellone*;
- Ereditarietà multipla: La classe *tabellone* eredita da *Griglia* e da *Flotta*;
- STL: Uso delle strutture dati vector e di list in *Nave.cpp*, *Battaglia navale.cpp* e *Tabellone.cpp*;
- Smart Pointer: Impiego di *unique_ptr*;
- Templates: Funzione *input_cella(T)* in *Flotta.cpp*.