Il progetto GOP.pdf
PROGETTO “GIOCO DELL’OCA PAZZA”
Il progetto GOP, realizzato da Riccardo Preite, Massimo Monacchi e Roberto Barbone, su indicazione del professore Angelo Di Iorio, è stato portato a compimento ispirandosi ad una tematica prevalentemente medievale, la quale dona originalità al classico gioco dell’oca inserendovi ambientazioni misteriose, oggetti magici, antichi mezzi di trasporto e temibili mostri.
Il gioco, le cui fondamenta rimangono sempre basate sull’avanzamento del giocatore attraverso le caselle al fine di tagliare per primo il traguardo, presenta delle novità maggiori soprattutto in queste ultime: esse infatti conterranno decine di effetti che possono variare dalla comparsa di un mostro da sconfiggere alla possibilità di cadere in un dirupo e rimanere quindi fermo per svariati turni. Il lancio dei dadi sarà determinante per la sorte dei giocatori in qualsiasi momento, nelle sfide contro i mostri, nell’avanzamento o nella retrocessione, e naturalmente nella vittoria.
Al fine di concretizzare il voluto tocco dinamico al gioco, si è resa necessaria un’implementazione chiaramente strutturata in diverse classi, che potessero definire appieno le specifiche del gioco. Tali classi, in ordine di esposizione, sono le seguenti: Giocatori, Persona, Dado, Tabellone, Casella, Type, Deck e Coda. Per rendere più cristallina l’implementazione, abbiamo preferito utilizzare i metodi set e get, agevolando così l’interazione fra le diverse funzioni delle classi. Inoltre, abbiamo scelto di includere tutti i file .h in un unico header, in modo da centralizzarli e sfruttare più rapidamente il loro utilizzo.
Per facilitare l'avvio, in modo da evitare l'utilizzo di comandi di compilazione ed esecuzione, quali "make" e "./GOP", è stato creato un file bash contenente entrambi.
Partendo da ciò che concerne gli utenti, abbiamo optato per la creazione di una classe Giocatori, la cui funzionalità tratta la gestione di una lista i cui nodi non sono altro che un’ulteriore classe, chiamata Persona, la quale rappresenta il “player” fisico, e di conseguenza viene richiamata tante volte quanti sono gli utenti che decidono di prendere parte al gioco. La classe Giocatori contiene inoltre delle funzioni che permettono attraverso il lancio dei dadi di decidere chi sarà il fortunato ad iniziare il gioco e quindi si incaricherà di effettuare il passaggio di turno e di stampare alle fine di ognuno la relativa posizione dell’utente nel tabellone. Dall’altro lato, nella classe Persona rientrano dei metodi chiamati stop e Sconfiggi, i cui utilizzi riguardano rispettivamente il bloccare il giocatore in una casella per determinati turni, indicati opportunamente da un contatore, e la possibilità di sconfiggere un mostro, dopo aver pescato la relativa carta bonus per sconfiggerlo, la quale si attiva o meno in funzione di una variabile booleana.
Brevemente spiegata, la classe Dado non contiene altro che i due campi relativi ai risultati dei singoli dadi e due funzioni che permettono di generare un risultato in maniera pseudocasuale di uno o due dadi, in base alla richiesta specifica dell’effetto di una casella.
Successivamente, parlando della classe Tabellone, è possibile riscontrare analogie con la classe Giocatori: i suoi compiti sono infatti quelli di creare una lista composta da nodi della classe Casella, generati ad ogni partita in un numero pseudocasuale e collegati tra loro da una funzione append.
Il tabellone di gioco si contraddistingue per le caselle colorate di ciano, mentre la casella 0, ossia lo start, presenta il colore rosso. I giocatori vengono simboleggiati da un cancelletto colorato a loro scelta e se più utenti contemporaneamente si trovano allo start, tale cancelletto sarà colorato anch'esso di rosso, ad indicare la molteplicità degli stessi. Inoltre, nel momento in cui vengono stampate le frasi riferite alla posizione dei giocatori, quest'ultime saranno rosse se l'utente segnalato si trova all'inizio, ciano come il tabellone altrimenti.
In aggiunta questa classe si occupa di stampare ad ogni turno tutte le caselle del tabellone, coi i rispettivi avanzamenti dei giocatori. Altra funzionalità della classe è quella di gestire gli effetti caselle di modo che ad ogni passo effettuato dal player corrisponda l’adeguato evento, il quale può concretizzarsi nella classe Type o in Deck, nel caso in giocatore dovesse pescare una carta.
Ultima chicca di questa classe, è la gestione dell’evento che si concretizza nel momento in cui due giocatori si trovano sulla stessa casella: attraverso la funzione duello, i giocatori coinvolti devono necessariamente combattere e solo chi farà il punteggio più alto coi dadi potrà sostare in tale casella, mentre l’altro, sconfitto, ritornerà al punto di inizio.
Conseguentemente, la classe Casella incorpora le informazioni relative al proprio tipo, al proprio numero di posizione all’interno del tabellone, una funzione che permette di puntare alla casella successiva e infine quella che rileva se all’interno vi è o meno un giocatore, e se presente, restituisce il suo nome.
Come già menzionato, la classe Type si contraddistingue nell’operare con i 4 principali tipi di casella, ossia mostri, mezzi, oggetti e luoghi. Invece però di inserire questi quattro all’interno della classe effettiva, abbiamo preferito implementare altrettante strutture, contenute nel file header, in modo da avere una distinzione più chiara degli stessi. Per ciò che concerne i luoghi, gli

oggetti e i mostri, abbiamo programmato le funzioni utilizzando uno switch il quale, come verrà spiegato successivamente opportunamente relazionato al valore generato casualmente della posizione della casella nel tabellone, identifica nello specifico tutte le casistiche dei possibili luoghi, mostri, ecc..; tutte le informazioni relative a queste casistiche vengono prese leggendo da file, e stampando solo quelle necessarie al momento opportuno.
Strettamente collegata alla classe Type, si presenta la classe Deck, creata appositamente per gestire un mazzo di carte, le relative pescate e gli esaurimenti degli effetti. Tale classe si avvale di una funzione, chiamata catch_card, la quale attraverso una generazione di numeri casuali richiama la funzione call_effect, composta essa stessa da dieci diversi casi, ognuno di essi facente riferimento ad una diversa funzione, la cui finalità varia da effetto a effetto.
L'ultima classe di cui tratteremo riveste un'importanza fondamentale per la buona riuscita del gioco, in quanto evita indesiderate o eccessive ripetizioni di determinati eventi a discapito di altri. Seguendo questa filosofia, la classe Coda ha la finalità di impedire ad un particolare effetto di ripetersi finchè non viene rimosso dalla coda, nello specifico rappresentata da un vettore, il quale ovviamente rispetta le caratteristiche FIFO, permetteremo quindi a tutti i tipi di caselle di succedersi regolarmente. Nel momento della creazione del tabellone, e quindi delle singole caselle, prima di assegnare un determinato effetto, la funzione Find, andrà a compiere un'iterazione sul vettore, di 4 celle per il tabellone, e nel caso esso fosse già presente impedirà a tale effetto di essere assegnato a quella casella; altrimenti, nel caso esso non compaia all'interno della coda, la funzione Shift andrà ad inserire tale evento nella coda, spostando di conseguenza tutti gli altri di una posizione verso destra, liberando quindi il primo elemento inserito.
Filo conduttore di tutte le classi altro non è che il main, chiamato appunto Gop.cpp, il quale unendo le funzionalità delle classi appena descritte, si occupa di tutti i procedimenti necessari dall'avvio del gioco alla proclamazione del vincitore. Con semplici cicli infatti, inizialmente chiede agli utenti le informazioni di base, il numero di giocatori, quali sono i loro nomi e i colori da loro scelti. L'intero gioco è controllato da un ciclo while,
la cui guardia è definita da un booleano settato a falso fino a quando un giocatore non arriva sull'ultima casella. Verrà richiesto se si vuole cominciare una nuova partita e in caso di risposta negativa il gioco termina.