Relazione

Progetto *“State Quiz”*

C orso di Programmazione ad Oggetti 2014/15

Autori: Alexander Saenko

Marco Pannunzio

Data: 26 Febbraio 2015

Indice

Capitolo 1

Analisi

1.1 Requisiti

Lo scopo del progetto consiste nel realizzare un gioco che fornisca degli indizi su uno Stato attraverso domande teoriche o immagini di bandiere dei Paesi da indovinare, classificate secondo 3 livelli di difficoltà (facile, medio, difficile). Il compito di ogni giocatore consisterà nello scegliere lo Stato giusto fra le 4 opzioni fornite. Il sistema ricompenserà il giocatore con dei punti mantenendo una classifica: la quantità dei punti assegnati varierà in base alla difficoltà di ciascuna domanda.

Il gioco prevede due modalità:
- *Single player*: lo scopo di questa modalità consiste nel fare più punti possibili per scalare la classifica;
- *Multiplayer*: lo scopo di questa modalità, invece, consiste in una sfida tra 2-4 giocatori (sulla stessa macchina), in cui vincerà colui che al termine di un numero predefinito di turni avrà un punteggio maggiore.

Al termine di ogni partita si potranno consultare anche le classifiche globali, divise in base al numero di domande.

1.2 Problema

Dato che si tratta di un'applicazione grafica, la difficoltà primaria consisterà nello scegliere una tecnica di progettazione che si presti meglio al raggiungimento degli obiettivi del futuro software.

Un altro problema sarà quello di dover gestire dati di natura diversa: domande teoriche, elenco degli stati, immagini di bandiere degli Stati ecc. Da questa problematica, di conseguenza, ne scaturisce un'altra, rappresentata dalla necessità di mettere in contatto questi dati e di predisporre strutture dati adeguate.

Un altro punto chiave consiste nella gestione di ogni turno di una partita: in particolare, in questa fase l'applicazione dovrà essere in grado di determinare il tipo di domanda, il suo livello di difficoltà, il relativo punteggio ecc.

Infine, questo software dovrà essere progettato in modo da poter permettere ad ogni giocatore di consultare anche le classifiche globali, contenenti dati di partite giocate in precedenza.

Capitolo 2

Design

2.1 Architettura

Questo software, dal punto di vista architetturale, è stato realizzato con l'utilizzo del pattern Model-View-Controller (MVC):

- Il modello si occupa della gestione dei dati dell'applicazione, che costituiscono il cuore del gioco.
- La vista, invece, si occupa dell'interazione con i giocatori.
- Il controllore, infine, è destinato al coordinamento tra le prime due parti.

Mentre il design dell'interazione tra tutte e 3 le parti sembra progettato in modo adeguato (permette sostituzioni in blocco della vista, senza causare grosse modifiche alla parte relativa al controllore), le singole parti non garantiscono una buonissima riusabilità. Questo fatto è determinato soprattutto dal limitato tempo a disposizione dei progettisti. Infatti, la suddivisione delle suddette parti in frammenti più leggeri ed estendibili richiedeva molto più tempo e, a quel punto, si è deciso di puntare al completamento di un software ben funzionante e in linea con gli obiettivi prefissati.

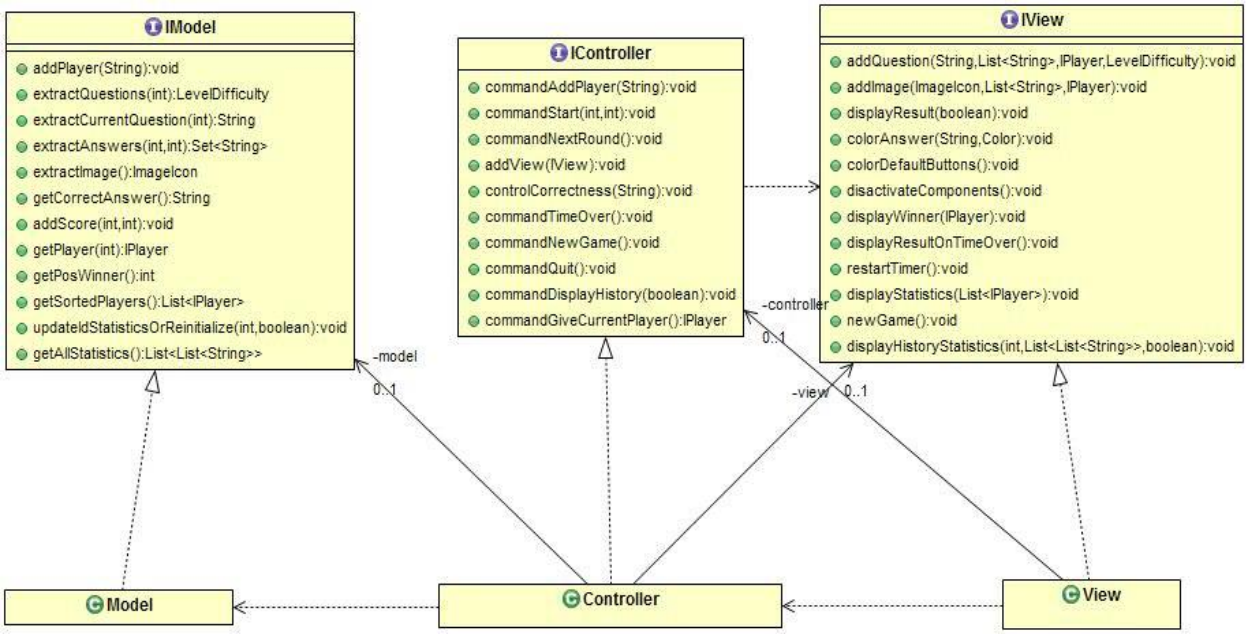

Figura 2.1 Schema UML architetturale basato sul pattern architetturale MVC.

2.2 Design dettagliato

In *Figura 2.2* è mostrato uno schema UML più dettagliato della parte relativa al modello dell'applicazione. La classe Model, che implementa l'interfaccia IModel, presenta 2 metodi (updateIdStatisticsOrReinitialize e getSortedPlayers) che, per ordinare le specifiche collezioni di dati, utilizzano il pattern Strategy (Ordinamento con comparatori), e altri metodi, che al loro interno utilizzano il pattern Iterator. Inoltre, la classe Player, volta alla rappresentazione di un giocatore, è realizzata utilizzando il pattern creazionale Builder.

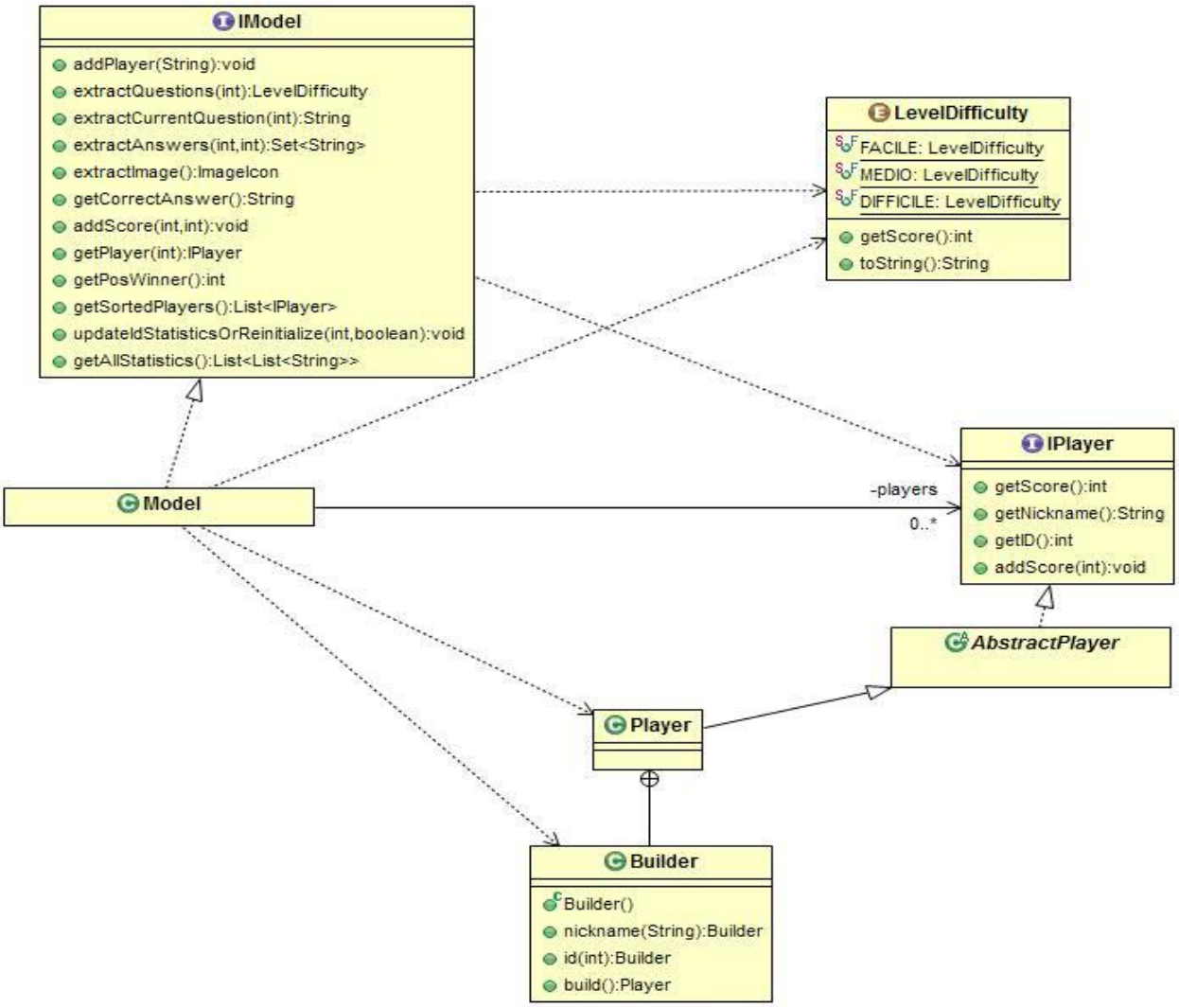

Figura 2.2 Schema UML di dettaglio del modello.

In *Figura 2.3* è mostrato uno schema UML di dettaglio della parte relativa al controllore dell'applicazione. Da questo schema si può dedurre che la classe Controller sia realizzata attraverso l'utilizzo di un altro pattern creazionale, ovvero del pattern Singleton. Questa scelta è stata influenzata dalla necessità di avere soltanto un controllore.

Il controllore viene menzionato nelle classi StatisticGUI, View e MenuGUI: la prima si occupa della visualizzazione delle classifiche globali, la seconda rappresenta l'interfaccia principale, ovvero la GUI del gioco e, infine, la classe MenuGUI si riferisce all'interfaccia grafica del menù del gioco, in cui vengono scelte alcune impostazioni, tra cui il numero di giocatori e il numero di domande.

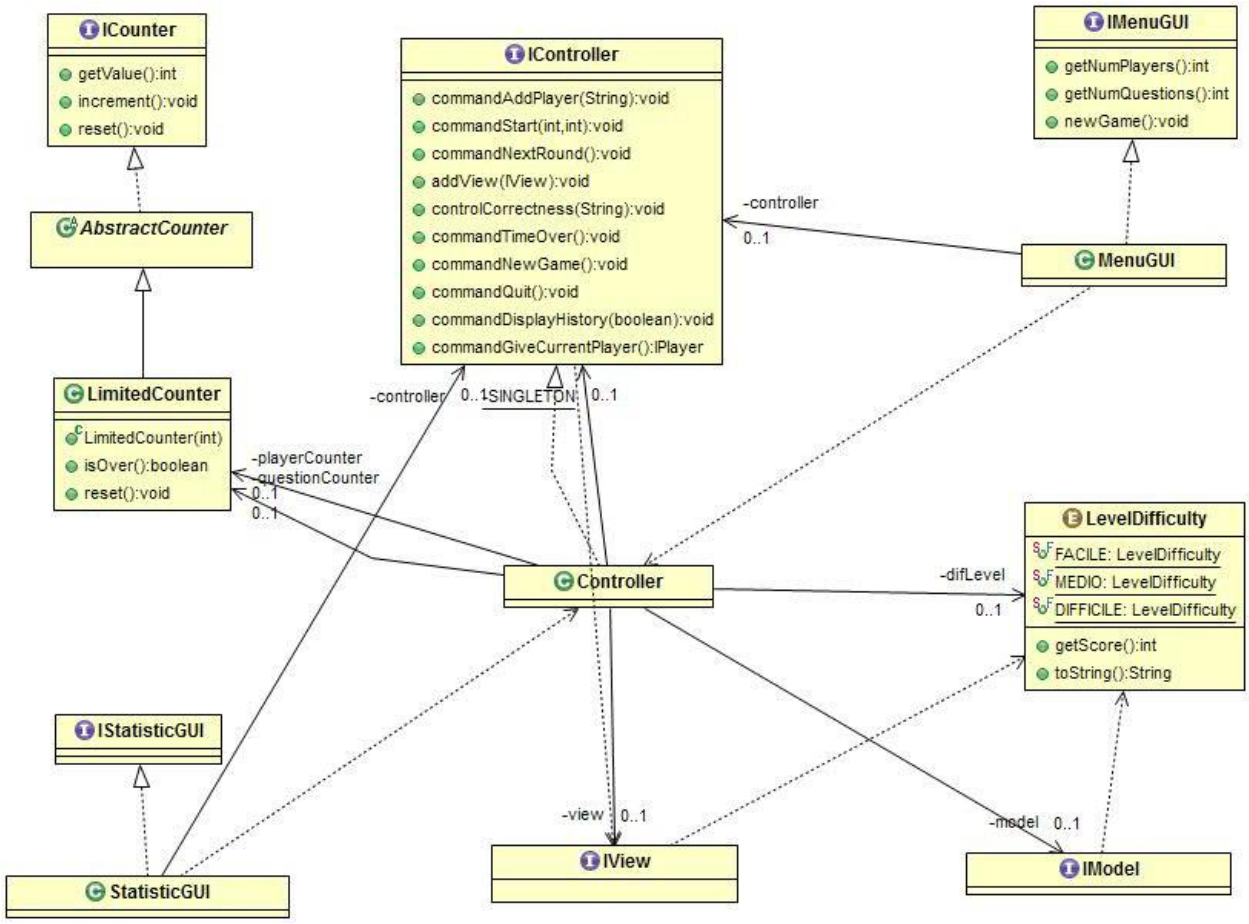

Figura 2.3 Schema UML di dettaglio del controllore.

In *Figura 2.4*, invece, è mostrato uno schema UML della vista. L'interfaccia grafica principale, ovvero quella della classe View, è rappresentata da una composizione gerarchica di pannelli e componenti (pattern Composite). Inoltre, gli handler di eventi nelle classi View, MenuGUI e StatisticGUI sono realizzati per mezzo di pattern Adapter.

Sia il menù di gioco, sia la visualizzazione della storia di tutte le statistiche, divise in base al numero di domande di ogni partita (quest'ultimo viene impostato all'inizio di ogni partita, tramite la schermata del menù, e può assumere un valore scelto tra quelli memorizzati nell'apposita ComboBox), sono realizzati attraverso le rispettive classi MenuGUI e StatisticGUI che entrambe estendono la classe JDialog.

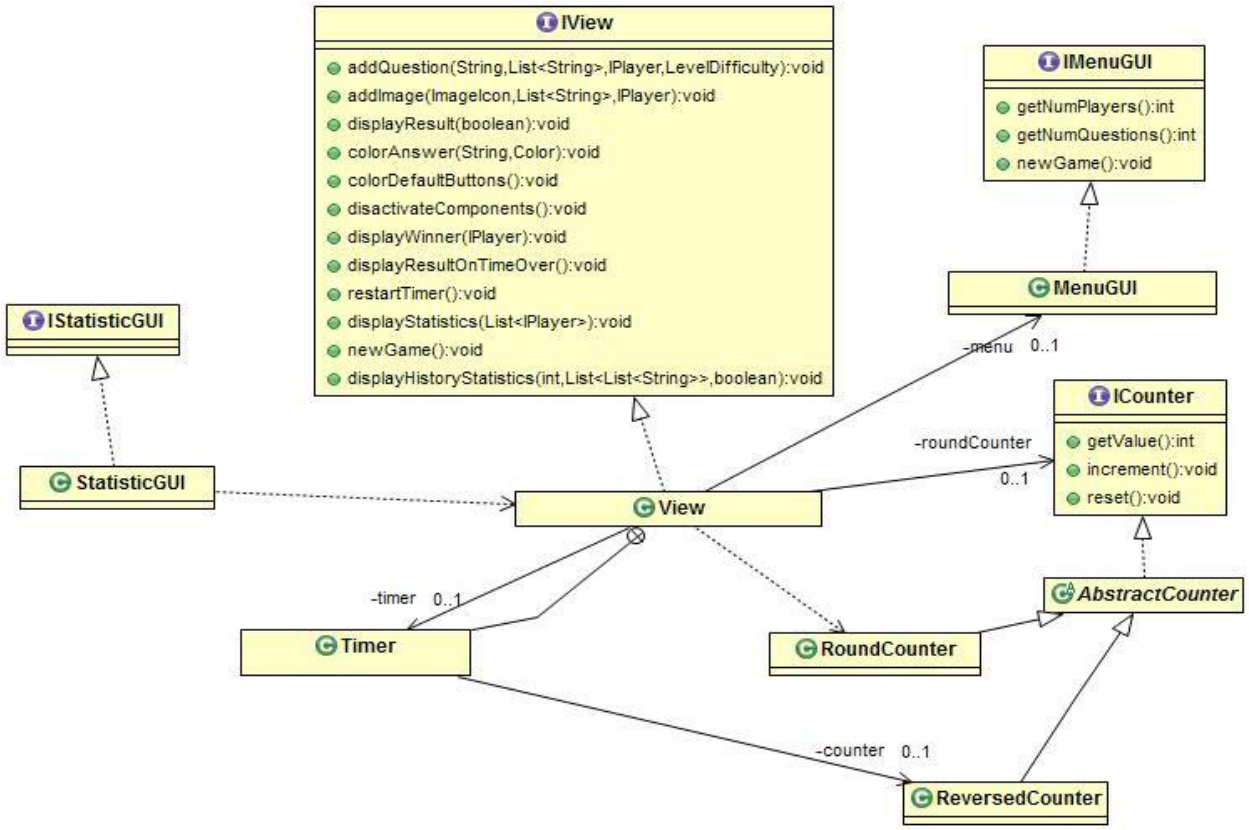

Figura 2.4 Schema UML di dettaglio della vista.

Inoltre, il nostro quiz, come la maggior parte di giochi di questo tipo, si avvale di un timer che regola l'avvicendamento dei turni. Esso è stato realizzato per mezzo della classe innestata Timer all'interno della classe View che estende la classe Thread.

Capitolo 3

Sviluppo

3.1 Testing automatizzato

Le classi che sono state sottoposte a test automatizzati sono le seguenti:

1) Model;
2) Player;
3) LimitedCounter, RoundCounter e ReversedCounter;
4) Controller.

Tutte queste classi sono state testate interamente utilizzando JUnit ad eccezione della classe Controller, in cui soltanto le eccezioni lanciate dai suoi metodi sono state sottoposte ad un test automatico, mentre il resto è stato testato manualmente. La ragione di questa scelta risiede nel fatto che la maggior parte dei suoi metodi, anche se indirettamente, agisca sulla GUI tramite i messaggi inviati alla View. Di conseguenza, anche tutto quello che riguarda la vista è stato testato manualmente.

Il nostro software è stato testato sui seguenti sistemi operativi:

- Windows 7
- Windows 8.1
- Mac OS X Yosemite

3.2 Divisione dei compiti e metodologia di lavoro

I compiti sono stati divisi nel seguente modo:

Marco Pannunzio:

- Tutto quello che riguarda la parte grafica dell'applicazione (package statequiz.view):
    1) Il menù di gioco;
    2) L'interfaccia principale;
    3) La schermata di consultazione delle statistiche;
- Il timer di gioco;
- Creazione, selezione e preparazione di tutti i dati necessari al corretto funzionamento dell'applicazione (file di testo contenenti i nomi di tutti gli Stati del mondo, diversi file, divisi in base al livello di difficoltà, contenenti le domande teoriche, immagini di bandiere di tutti gli Stati ecc.).

Alexander Saenko:

- Il modello dell'applicazione;
- Il suo controllore;
- E, di conseguenza, tutto quello che è contenuto nei package statequiz.counter (ICounter, AbstractCounter, LimitedCounter, RoundCounter, ReversedCounter), statequiz.player (IPlayer, AbstractPlayer, Player), statequiz.difficulty (l'enum LevelDifficulty).

Inizialmente avevamo intenzione di realizzare la parte relativa al controllore insieme, ma, dopo aver analizzato bene il carico di lavoro di ciascuno di noi, abbiamo deciso di dividere i compiti nel modo descritto sopra. Dunque, i compiti assegnati sono stati portati a termine in modo quasi del tutto indipendente: solo piccoli frammenti di codice sono stati sviluppati su consigli dell'altro membro o, direttamente, dall'altro componente del gruppo.

Per garantire una miglior collaborazione tra entrambi i membri del gruppo, ci siamo avvalsi dell'uso del repository BitBucket, aggiornando la versione del software con una frequenza media pari a un giorno solare.

3.3 Note di sviluppo

La classe MyTableCellRenderer è stata presa da questo forum *http://www2.mokabyte.it/forum/thread.jsp?forum=15&thread=5276*. Il suo scopo consiste nel modificare l'aspetto grafico di determinate celle di una JTable. In particolare, noi l'abbiamo usata affinché il testo di ogni cella venga posizionato al centro.

Capitolo 4

Commenti finali

4.1 Conclusioni e lavori futuri

Probabilmente, dal punto di vista di qualità del software, la nostra applicazione non è stata sviluppata nel modo migliore (lo si evince dalla poca riusabilità delle singole parti del pattern MVC). Questo fatto è legato in gran parte al limitato tempo a nostra disposizione, ma anche alla poca esperienza in questo ambito.

Tuttavia, gli obiettivi prefissati in partenza sono stati raggiunti tutti e, nel complesso, siamo soddisfatti del lavoro svolto in queste settimane.

Appendice A

Guida utente

Nonostante l'interfaccia grafica del nostro quiz sia molto intuitiva, noi vorremmo comunque illustrare i suoi elementi chiave.

All'avvio del gioco si apre un menù, in cui il/i giocatore/i possano scegliere la modalità di gioco, *Single Player* o *Multiplayer*, ed il numero di domande, di cui sarà costituita la nuova partita.

- *Single Player*: ovviamente in questa modalità c'è un unico giocatore;
- *Multiplayer*: se è stata scelta questa modalità, la prossima operazione da effettuare consisterà nell'impostare il numero di giocatori (da 2 a 4).

Dopodiché, ogni giocatore dovrà digitare il proprio nickname. Due o più giocatori possono avere lo stesso nickname, ma ad ogni giocatore viene assegnato un ID univoco.

Una volta completate queste semplici operazioni, inizia la partita vera e propria. Il campo di gioco è strutturato nel seguente modo: in basso troviamo la domanda corrente con le 4 risposte, di cui solo una è corretta, in alto, invece, vengono visualizzate le informazioni relative al giocatore corrente (Id, nickname, score), quelle relative al turno (Es. Round 3/10) e alla difficoltà della domanda attuale, e, infine, il timer, il cui valore viene decrementato ogni secondo.

A ciascun giocatore spetta il proprio turno e l'avvicendamento dei turni può avvenire in due casi:

1) quando il giocatore corrente dà una risposta e viene informato del suo esito;

2) in caso di mancata risposta al termine del tempo regolato dal timer (30 secondi).

Ogni partita può terminare in 3 modi diversi:

- Modalità *Single Player*: al termine di ogni partita si apre una nuova schermata, nella quale vengono visualizzata la statistica globale (contenente risultati di tutte le partite, appartenenti sia a questa sessione di gioco, sia a tutte le sessioni precedenti), selezionata in base al numero di domande scelto in partenza; questa interfaccia permette di consultare anche tutte le altre classifiche (è sufficiente scegliere il numero di domande specifico tramite l'apposita ComboBox e premere il tasto Display);
- Modalità *Multiplayer*: se al termine di una partita si ha un vincitore, ovvero colui che ha ottenuto il punteggio più alto, il gioco propone ai giocatori la classifica della partita appena giocata;
- Modalità *Multiplayer*: se vi sono più vincitori, la partita termina regolarmente.

In tutti questi casi, si ha la possibilità di consultare le statistiche un numero illimitato di volte, oppure di iniziare una nuova partita, dovendo però ripetere tutti i passi descritti all'inizio di questo capitolo.