

Lorenzo Genesi - 868898  
Politecnico di Milano  
28/02/2017  
lorenzo.genesi@gmail.com

Genesi Lorenzo - SCARABEO

Sommario

[Analisi dei requisiti 3](#_Toc475974369)

[Regole 3](#_Toc475974370)

[Richieste specifiche 4](#_Toc475974374)

[Soluzione 5](#_Toc475974378)

[Strategia 5](#_Toc475974379)

[Strutture dati 6](#_Toc475974382)

[Algoritmi e codifica 7](#_Toc475974385)

[Situazioni di gioco 10](#_Toc475974391)

# Analisi dei requisiti

## Regole

Lo Scarabeo può essere giocato da due a quattro giocatori, su una griglia di 17x17 caselle e con 130 tessere rappresentati le lettere dell’alfabeto più due jolly.

Si comincia a giocare distribuendo a ogni giocatore 8 lettere pescate a caso. Il primo giocatore dovrà comporre una parola di senso compiuto utilizzando le proprie lettere e disponendole nella griglia. Le lettere utilizzate permettono di calcolare, una volta composta la parola, il punteggio ottenuto.

Sono presenti nella griglia caselle speciali che permettono di incrementare il punteggio altrimenti ottenuto.

Alla fine del proprio turno il giocatore pesca un numero di lettere pari a quelle appena utilizzate, in modo da averne sempre 8. Il turno passa poi al giocatore successivo che deve a sua volta comporre una parola, con il vincolo di intersecare la propria con almeno una delle lettere già presenti sulla griglia, come nelle parole incrociate.

Il gioco prosegue fino a che non sono finite tutte le lettere e uno dei giocatori non ha finito le proprie. Le tessere jolly possono essere utilizzate in sostituzione di qualsiasi altra lettera.

Se un giocatore possiede la lettera corrispondente a un jolly presente nella griglia, al proprio turno può appropriarsene sostituendola.

Scopo del gioco è ottenere il maggior punteggio possibile una volta finite le lettere disponibili. L'ultimo giocatore ad aver composto una parola ottiene come punteggio anche la somma di tutte le lettere rimanenti agli altri giocatori e non ancora utilizzate.

### Caselle speciali

* 2L moltiplica per 2 il punteggio della lettera
* 3L moltiplica per 3 il punteggio della lettera
* 2P moltiplica per 2 il punteggio della parola
* 3P moltiplica per 3 il punteggio della parola

### Punteggi lettere

* 1 punto per ogni A, C, E, I, O, R, S, T
* 2 punti per L, M, N, 3 per P
* 4 punti per B, D, F, G, U, V
* 8 punti per H e Z
* 10 punti per Q

### Punteggi bonus

* 10 punti bonus se si compone una parola di 6 lettere
* 30 punti bonus se la parola è di 7 lettere
* 50 punti bonus se è formata da tutte e 8 le lettere
* 100 punti bonus se si riesce a scrivere la parola "scarabeo" o "scarabei"

A questi punteggi vanno sommati ulteriori 10 punti se non si sono utilizzati jolly nel comporre le parole.

## Richieste specifiche

Viene esplicitamente richiesto di:

* Controllare la validità delle parole di lunghezza superiore a 3 tramite un dizionario della lingua italiana
* Fornire un comando per ottenere il miglior suggerimento di una parola da inserire nella griglia
* Controllare la correttezza dei verbi declinati e di parole con suffissi/prefissi, e l’utilizzo di un’interfaccia grafica a finestre per l’interazione con l’utente.

### Validità

Per quanto riguarda il controllo della validità non ci sono considerazioni particolari da fare, una parola risulta infatti valida se è contenuta nel dizionario.  
Di conseguenza sarà sufficiente verificare la presenza della parola inserita dall’utente, ed eventualmente di altre nuove parole intersecate nella costruzione.

### Suggerimento

Per il suggerimento la questione è più complicata, bisogna infatti tenere in considerazione:

* Le lettere a disposizione dell’utente
* Le lettere presenti sul campo da gioco
* La posizione delle lettere sul campo da gioco con eventuali caselle bonus
* La ipotetica posizione delle lettere dell’utente sul campo da gioco (la stessa parola in punti diversi può dare punteggi diversi)
* La ipotetica posizione delle lettere dell’utente sul campo da gioco in relazione alla posizione delle lettere già presenti sul campo da gioco
* La validità della parola suggerita all’utente
* La validità delle eventuali parole intersecate dalla parola suggerita, con tutte le ripercussioni del caso (nuovi lettere coinvolte e di conseguenza diversi punteggi in base a diverse posizioni sul campo da gioco)
* Eventuali punteggi bonus sulla parola suggerita e sulle eventuali parole intersecate
* Naturalmente tutti i punti precedenti devono rispettare le regole del gioco

### Verbi declinati, suffissi/prefissi e interfaccia

Per ovviare a questi problemi, lavoro direttamente su un dizionario che contiene già i verbi declinati ed eventuali parole con suffissi/prefissi, e per quanto riguarda l’interfaccia, utilizzo i form offerti dall’ambiente di sviluppo QT.

# Soluzione

## Strategia

Per questo progetto ho deciso di separare concettualmente ciò che vede l’utente, cioè tutta la parte grafica, dalla gestione vera e propria della partita, tramite l’utilizzo della programmazione ad oggetti.

A livello pratico si hanno 2 entità principali che comunicano tra di loro per poter garantire lo svolgimento della partita. L’utente potrà quindi eseguire le sue azioni tramite l’interfaccia, così il risultato delle azioni stesse sarà poi convalidato, comunicando di nuovo all’interfaccia i cambiamenti.

Invio delle azioni dell’utente

GESTIONE DELL’INTERFACCIA UTENTE

GESTIONE DELLA PARTITA

Convalida delle azioni

Restituzione del risultato

In questo modo è possibile limitare le mosse dell’utente alla cerchia di azioni consentite, e di conseguenza limitare anche la possibilità di incorrere in casi non previsti.

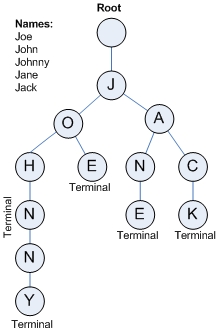
Sul piano della codifica queste due entità sono rappresentate dagli oggetti Partita e MainWindow, e comunicano tra di loro tramite il meccanismo Signals-Slots di QT.

Sempre per comodità a livello di gestione, sono state create delle altre classi, che corrispondono alle entità presenti nel programma ed utilizzate nel gioco vero e proprio, che rimangono collegate alle due principali sopracitate.  
Nello specifico:

|  |  |
| --- | --- |
| PARTITA | MAINWINDOW (interfaccia) |
| Tabellone – gestisce il tabellone di gioco | CustomButton\_carta – rappresenta graficamente una carta in possesso del giocatore |
| Mazzo – gestisce il mazzo della partita | CustomButton\_casella – rappresenta graficamente una casella del tabellone della partita |
| Giocatore – gestisce un giocatore della partita |  |
| Dict – gestisce il dizionario di riferimento della partita |  |

## Strutture dati

### Dizionario

La struttura dati più complessa è quella utilizzata dal dizionario, che mi permette, in questo caso, di gestire al meglio tutte le operazioni necessarie allo svolgimento del gioco.

1 - Esempio di struttura TRIE

In particolare:

* Inserimento delle parole nel dizionario
* Verifica dell’esistenza di una parola nel dizionario
* Calcolo del suggerimento migliore

Per questo la scelta è ricaduta su una struttura denominata TRIE.  
Questa struttura è un albero, dove ogni nodo presenta una lettera ed ha potenzialmente 26 successori ad esso collegato, uno per ogni lettera dell’alfabeto. Inoltre ogni nodo contiene un flag che indica se in corrispondenza di esso si trova la fine di una parola.

Successivamente illustrerò come questa struttura, a livello pratico, è sfruttata dal programma.

### Tabellone

La classe tabellone rappresenta il campo da gioco, ed è formato da diverse matrici parallele, di dimensione 17x17

Le matrici sono:

* Campo: rappresenta le caselle appartenenti al tabellone, con i relativi eventuali bonus.
* Lettere: sono tutte le lettere posizionate sul campo da gioco
* NuovaLettera: indica se la lettera è appena stata inserita
* Jolly: indica se la lettera è in verità un jolly posizionato

# Algoritmi e codifica

## Convalida del tabellone

Alla fine di ogni turno, prima di passare a quello successivo, viene eseguito un check del tabellone. A seconda del risultato di questa operazione, il turno verrà ripetuto, oppure sarà possibile proseguire.

Il tabellone viene considerato valido se tutte le parole maggiori di 3 presenti su di esso appartengono al dizionario.

Per fare questo esamino ogni cella appartenente al tabellone.  
Se la cella non è vuota, allora controllo se è l’inizio di una parola in orizzontale o in verticale.  
Se è l’inizio di una parola, allora la scorro tutta e la estraggo.  
A questo punto controllo se la parola (o la parola al contrario) è contenuta nel dizionario.  
Se esiste almeno una parola che non è contenuta nel dizionario, allora il tabellone non è valido.

## Ricerca dell’esistenza di una parola nel dizionario

Questa funzione ricorsiva mi permette di verificare se la una parola è contenuta nel dizionario.

Concettualmente basta scorrere ogni nodo dell’albero seguendo le lettere da cui è composta la parola stessa, e verificare che l’ultimo nodo coincida con la fine di una parola.

A livello di codice si traduce in:

int Trie::find(const string& word, int cur) const{

if(cur == word.length()) // se sono arrivato in fondo alla parola da cercare

return endOfWord ? int(Trie::WORD) : int(Trie::PREFIX); // se sono effettivamente alla fine di una parola ritorno il valore Trie::WORD, altrimenti il valore Trie::PREFIX

int index = tolower(word[cur]) - 'a'; // calcolo il successore della lettera

if(successor[index] == NULL) // se non esiste

return int(Trie::NONE); // ritorno il valore Trie::NONE

return successor[index]->find(word, cur+1); // altrimenti continuo

}

## Calcolo di tutte le parole possibili dato un rack di lettere iniziale

Questa funzione ricorsiva, a partire da un rack di lettere, permette di calcolare tutte le possibili combinazioni delle lettere stesse che formano delle parole valide, ovvero contenute nel dizionario italiano.

Concettualmente è sufficiente verificare per ogni lettera ogni possibile combinazione con le successive presenti nel rack di partenza, e per ognuna di esse verificarne la validità tramite il dizionario. Per poter ottimizzare la velocità di composizione e ricerca delle parole, controllo ogni volta se la parola a cui sono arrivato non ha più successori nella struttura. In quel caso, non è necessario continuare con le combinazioni in quella specifica direzione.

A livello di codice l’implementazione risulta meno chiara, in quanto più complessa rispetto all’esempio precedente, ma comunque, grazie alla logica ricorsiva, rimane piuttosto compatta ed elegante, ma soprattutto efficiente:

## Calcolo del suggerimento

void Partita::constructWords(unordered\_map<char, int> &choice, string &curWord, vector<string > &words)

{

for(auto it = choice.begin(); it != choice.end(); it++){

if(it->second <= 0) // se il contatore delle occorrenze del carattere è minore o uguale a 0 allora uso l'istruzione continue

continue; // mi permette di saltare direttamente al prossimo ciclo, senza uscirne (a meno che non sia vera la condizione di uscita)

curWord += it->first; // alla mia parola corrente aggiungo il carattere a cui sto puntando

it->second--; // decremento il contatore delle occorrenze in quanto ho utilizzato il carattere

if(myDictionary.contains(curWord)) // controllo che il dizioario contenga effettivamente la parola corrente

words.push\_back(curWord); // e nel caso lo aggiungo al vettore delle mie parole con il relativo punteggio

if (!(myDictionary.noSuccessor(curWord))) // se da quella parola non ci sono più successori è inutile continuare a controllare le combinazioni

constructWords(choice, curWord, words); // chiamata ricorsiva per poter controllare ogni possibile combinazione con le mie lettere

choice[curWord.back()]++; // incremento il contatore delle occorrenze dell'ultimo carattere utilizzato, per riportare la situazione all'inizio

curWord.pop\_back(); // tolgo l'ultimo carattere dalla parola corrente, per riportare la situazione all'inizio

}

}

Questa funzione, appoggiandosi su quelle che permettono di calcolare il punteggio e tutte le parole possibili dato un rack iniziale, calcola quale è la migliore parola da inserire in orizzontale o in verticale.

Concettualmente esamino tutto il campo prima per righe, poi per colonne, e per ognuna di esse controllo quali lettere sono presenti. Il mio rack di lettere sarà quindi composto da quelle prelevate dal campo e da quelle in possesso del giocatore.  
A questo punto calcolo grazie alla funzione, ogni possibile combinazione valida.  
Successivamente, per ogni parola possibile trovata, ne verifico la validità ed eventualmente il punteggio su ogni cella della riga o colonna.  
Una parola è ritenuta valida se:

* La posizione delle lettere già presenti sul tabellone rimane invariata
* La parola si intrecci almeno con una lettera già presente sul tabellone
* La parola composta utilizza almeno una lettera dalla mano del giocatore
* Le lettere posizionate sul tabellone arrivino solo dalla mano del giocatore

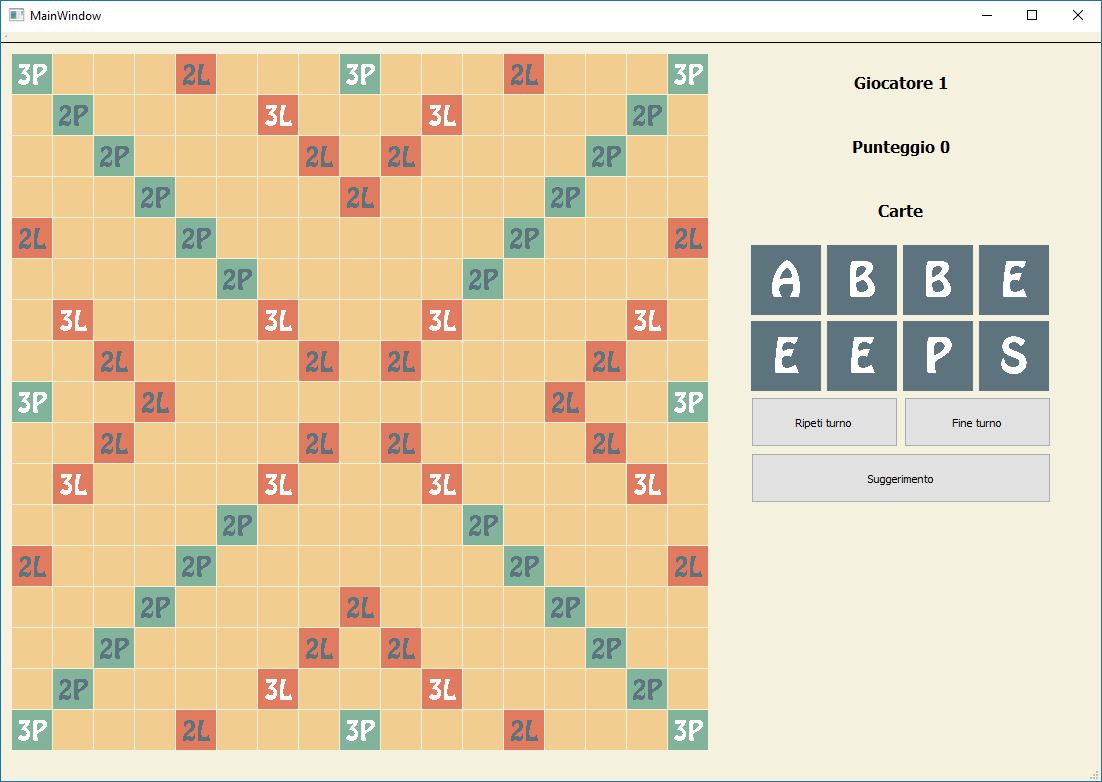
Alla fine viene tenuta la parola valida con il miglior punteggio.

## Altro

Per gestire i Jolly nel modo più semplice possibile, faccio scegliere al giocatore che lettera deve sostituire il Jolly quando viene posizionato, attraverso l’apposita dialog. In questo modo, è possibile gestire il jolly come se fosse una normalissima tessera rappresentante una lettera, ad eccezione naturalmente della sostituzione.

# Situazioni di gioco

## Inizio di una partita



Questa è la schermata che si presenta all’inizio di una nuova partita. Sulla sinistra abbiamo tutto il campo da gioco, sulla destra invece tutte le informazioni del giocatore di turno, con le relative azioni che può compiere.

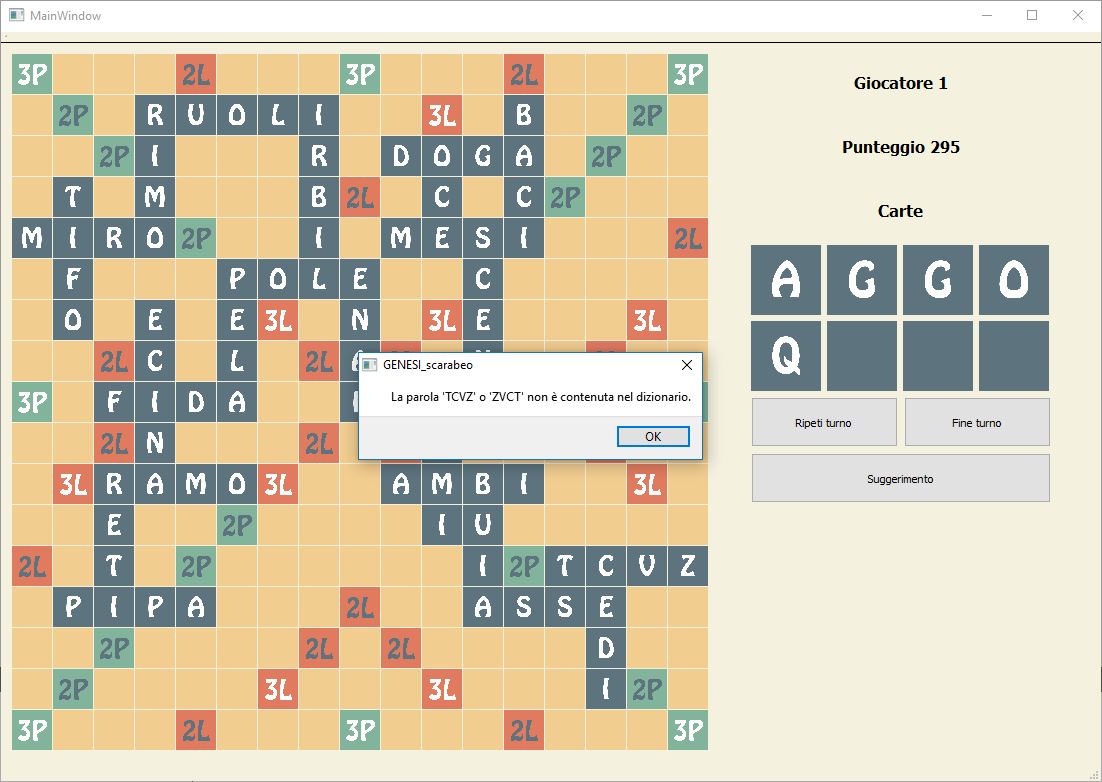
Per iniziare a posizionare le lettere è sufficiente cliccare sulla tessera del giocatore e successivamente sulla casella del tabellone.

## Richiesta di un suggerimento



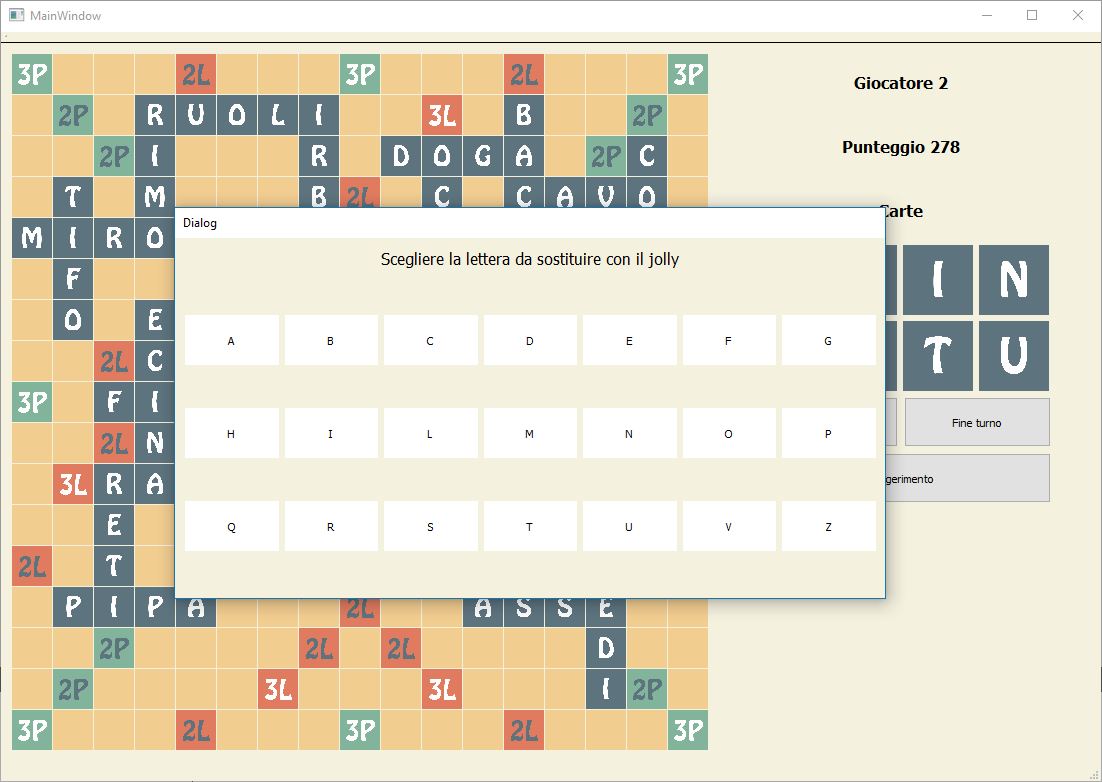
In questa schermata è stato richiesto un suggerimento tramite l’apposito bottone. Il giocatore di turno riceverà quindi, sotto forma di avviso, quale sarà la parola con il maggiore punteggio che si può ottenere in questa situazione.  
Viene comunicata la parola in orizzontale e verticale, con il relativo punteggio e le righe/colonne di partenza.

## Inserimento di parola non valida



In questa schermata, è stata inserita una parola non valida, di conseguenza il programma avvisa il giocatore, e dà la possibilità di ripetere il turno.

## Inserimento di un Jolly



In questa schermata è stato inserito un jolly, e di conseguenza il programma chiede quale parola deve sostituire il jolly. Per selezionare una lettera, è sufficiente cliccarla con il mouse.