

# PROGETTO PROVA FINALE (INGEGNERIA DEL SOFTWARE)

Anno Accademico 2011/2012 - Revisione documento del 15/05/2012

## Parte 1 - Introduzione

Il progetto consiste nello sviluppo di una versione semplificata del gioco da tavolo [Carcassonne](#).

Il prodotto finale dovrà includere:

- diagrammi UML che mostrino come è stato progettato il software;
- implementazione funzionante del gioco conforme alle specifiche presenti in questo documento;
- codice sorgente dell'implementazione;
- codice sorgente dei test di unità.

Saranno oggetto di valutazione:

- la qualità della progettazione con particolare riferimento a un uso appropriato di interfacce, ereditarietà, composizione tra classi e uso dei design pattern visti a lezione;
- il livello di autonomia e impegno nello svolgimento del progetto;
- la stabilità dell'implementazione e la sua conformità alle specifiche date;
- la qualità e la leggibilità del codice scritto, con particolare riferimento a un uso appropriato dei nomi di variabili/metodi/classi/package, all'inserimento di commenti e documentazione JavaDoc (i nomi e la documentazione è preferibile che siano in inglese), all'evitare ripetizioni di codice o metodi di eccessiva lunghezza;
- la qualità e la copertura dei casi di test: il nome e i commenti di ogni test dovranno chiaramente specificare le funzionalità testate e i componenti coinvolti.

**Data di consegna finale e valutazione (non prorogabile): 19 Giugno 2012 (durante il laboratorio).**

Nella parte 2 di questo documento saranno presentate le regole del gioco da tavolo semplificate. Nella parte 3, le specifiche di implementazione di base, necessarie per avere la possibilità di ottenere una valutazione sufficiente. Nella parte 4, le specifiche di implementazione standard, che, in aggiunta a quelle di base, permettono di ottenere una valutazione massima pari a 30 e lode. Infine, nella parte 5 saranno presentate le specifiche di alcune estensioni facoltative che permetteranno di ottenere un ulteriore incremento di punteggio a discrezione dei docenti.

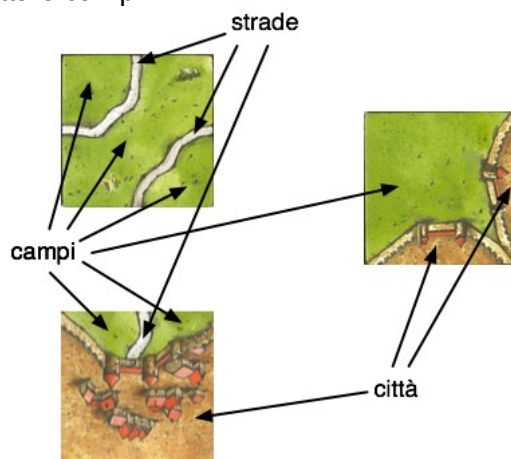
## Parte 2 - Regole del gioco

Carcassonne è un gioco a turni da 2 a 5 giocatori in cui ogni giocatore dovrà contribuire alla costruzione di un paesaggio ispirato alla cittadina medievale francese di Carcassonne. Durante il suo turno ogni giocatore ha la possibilità di aggiungere al paesaggio strade, città e campi allo scopo di guadagnare punti vittoria. I punti vittoria serviranno per determinare il vincitore a fine partita.

Queste regole propongono una versione semplificata della terza edizione del gioco originale. Per chi ha già conoscenza del gioco originale le uniche differenze consistono nell'impossibilità di posizionare segnalini nei campi, nell'assenza di tessere scudo e nell'assenza di tessere monastero.

### 2.1 Materiale a disposizione

- Una pila di tessere territorio quadrate raffiguranti degli elementi che possono essere porzioni di strade, città e campi.



- 7 segnalini di controllo per ogni giocatore: 7 segnalini rossi, 7 segnalini blu, 7 segnalini verdi, 7 segnalini gialli e 7 segnalini neri.



### 2.2 Scopo del gioco

Ogni giocatore posiziona delle tessere territorio e dei segnalini di controllo con lo scopo di guadagnare punti vittoria. Chi, alla fine, avrà totalizzato più punti vittoria risulterà vincitore.

### 2.3 Preparazione del gioco



Cercare nella pila delle tessere la seguente tessera: e posizionarla scoperta al centro dell'area di gioco. Mischiare le restanti tessere e tenerle coperte.

Ogni giocatore avrà a disposizione davanti a sé i 7 segnalini di controllo per utilizzarli durante la partita.

Selezionare il giocatore che inizierà per primo a giocare.

## 2.4 Svolgimento del gioco

Il gioco procede a turni, a partire dal primo giocatore, si procederà in senso orario. A ogni turno, il giocatore attivo compie le seguenti azioni in questo preciso ordine:

1. pesca una tessera territorio coperta e la posiziona nell'area di gioco;
2. opzionalmente, posiziona uno dei suoi segnalini di controllo sulla tessera appena messa in gioco;
3. se il posizionamento della tessera ha completato strade o città, sono conteggiati i punti vittoria corrispondenti e i segnalini di controllo posizionati su di esse saranno nuovamente disponibili.

### Azione 1: Posizionamento delle tessere

Pescare una tessera, mostrarla a tutti i giocatori, quindi metterla nell'area del gioco attenendosi alle seguenti regole:

- la nuova tessera deve avere almeno un lato in contatto con il lato di una delle tessere già in gioco;
- la nuova tessera va posizionata in modo che città, strade e campi siano contigui a quelli della/e tessera/e con cui è in contatto (ad esempio una strada non può finire in un campo, oppure una città non può attaccarsi ad un campo senza chiudere le mura).

Se per un qualche motivo la tessera pescata non avesse possibili posizionamenti legali questa è eliminata dalla partita e ne sarà pescata un'altra.

### Azione 2: Posizionamento dei segnalini di controllo

Quando un giocatore ha posizionato una tessera può, se lo desidera, posizionare uno dei suoi segnalini di controllo attenendosi alle seguenti regole:

- si può posizionare un solo segnalino per turno;
- il segnalino posizionato deve essere uno di quelli disponibili (quindi non può essere recuperato da quelli già posizionati sulle tessere in gioco);
- può essere posizionato esclusivamente sulla tessera che è stata appena messa in gioco;
- è necessario scegliere su quale parte specifica della tessera posizionarlo (cioè va posizionato sopra la porzione di strada o città che deve essere controllata);
- non si può posizionare il segnalino su una strada/città se altre porzioni della stessa, appartenenti a tessere precedentemente posizionate, contengono già un segnalino (è comunque possibile che, in seguito al posizionamento di una tessera, due città/strade inizialmente separate e dotate di segnalino si uniscano a formarne una sola).

Nel caso in cui un giocatore non avesse più segnalini di controllo disponibili, continuerà a giocare le tessere normalmente, ma non potrà posizionarvi sopra alcunché finché non avrà recuperato dei segnalini di controllo.

### Azione 3: Conteggio dei punti vittoria

Ogni qual volta delle strade o delle città vengono completate in seguito al posizionamento di una tessera, esse forniscono immediatamente dei punti vittoria al giocatore che le controlla. Un giocatore controlla una strada o una città quando possiede la maggioranza di segnalini di controllo in quella strada o città. **Nel caso in cui più giocatori si contendano una strada o città con lo stesso numero di segnalini, i punti vittoria sono attribuiti per intero a tutti i giocatori contendenti.**

Dopo l'attribuzione dei punti vittoria relativi al completamento di una strada o di una città, i segnalini che vi erano posizionati sopra saranno nuovamente disponibili.

### *Punteggio strada*

Una strada si considera completa quando ambedue le estremità terminano in un incrocio, una città o se la strada compie un cerchio completo su di essa. Non c'è limite al numero di tratti di cui la strada è composta.

Il giocatore che controlla una strada completa ottiene il relativo punteggio pari ad 1 punto per ogni tessera di strada (se la strada dovesse passare due volte nella stessa tessera questa varrebbe solo 1 punto). Nel caso in cui alla fine del gioco la strada restasse incompleta, questa farebbe guadagnare al giocatore che la controlla comunque 1 punto per ogni tessera.

Esempi di strade complete:



Il giocatore rosso guadagna  
4 punti



Il giocatore rosso e il giocatore blu  
guadagnano 3 punti

### *Punteggio città*

Una città si considera completa quando è interamente circondata da mura senza lo spazio per ulteriori tessere all'interno delle mura. Non c'è alcun limite al numero di tessere di cui è composta la città. Il giocatore che controlla la città completa totalizza un punteggio di 2 punti per ogni tessera che compone la città. Nel caso in cui alla fine del gioco la città restasse incompleta, questa farebbe guadagnare al giocatore che la controlla comunque 1 punto per ogni tessera.

Esempi di città complete:



Il giocatore  
rosso  
guadagna  
4 punti



Il giocatore blu guadagna  $2 \times 7 = 14$  punti

#### *Recupero dei segnalini di controllo*

Ogni qual volta venga completata una strada o una città, dopo aver assegnato eventuali punti vittoria, i segnalini di controllo posizionati sull'elemento completato verranno nuovamente resi disponibili ai rispettivi giocatori.

**È quindi possibile per un giocatore posizionare una tessera che completa qualcosa, metterci sopra un segnalino, totalizzare i punti ed avere il segnalino disponibile, tutto nello stesso turno.**

## **2.5 Fine del gioco**

Non appena viene posizionata l'ultima tessera il gioco termina e si procede con il conteggio dei punti relativi alle strade e città non completate, i quali si sommeranno ai punti guadagnati durante la partita.

## Parte 3 - Specifiche implementative di base

In questa sezione vengono presentate le specifiche implementative corrispondenti ai requisiti minimi che il progetto deve soddisfare per ottenere una valutazione sufficiente.

### 3.1 Inizializzazione della partita

L'inizializzazione della partita deve essere composta dalle seguenti fasi.

1. Determinazione del numero di giocatori: l'applicazione richiede il numero di giocatori totali.
2. Acquisizione della pila di tessere disponibili dal file **carcassonne.txt** (recuperabile dalla stessa pagina da cui è stato scaricato questo documento).
3. Posizionamento della prima tessera presente nella pila nell'area di gioco.
4. Mescolamento delle restanti tessere.
5. Inizio del primo turno di gioco secondo l'ordine fisso: rosso, blu, verde, giallo, nero.

### Rappresentazione tessere

Il file **carcassonne.txt** contiene su ogni riga una rappresentazione testuale di ogni tessera di Carcassonne.

Data una tessera, la sua rappresentazione testuale si ricava nel seguente modo:

- Ogni tessera ha quattro lati: nord (**N**), sud (**S**), ovest (**W**), est (**E**).
- Su ogni lato si può trovare una porzione di strada (**S**), una porzione di città (**C**) oppure nessuno dei due (**N**).
- Ogni tessera può avere una o più connessioni tra strade o città sui diversi lati. In particolare può avere una connessione da nord a sud (**NS**), una da nord a est (**NE**), una da nord a ovest (**NW**), una da ovest a est (**WE**), una da sud a est (**SE**) oppure una da sud a ovest (**SW**).
- La stringa che rappresenta la tessera si ricava dunque così:  
"N=S|C|N S=S|C|N W=S|C|N E=S|C|N NS=0|1 NE=0|1 NW=0|1 WE=0|1  
SE=0|1 SW=0|1"

Dove i primi quattro blocchi mettono in relazione i quattro lati con il loro contenuto, mentre i restanti sei blocchi specificano quali lati siano connessi tra loro (**0** significa non connessi, **1** significa connessi).

Esempi di corrispondenza tra rappresentazione grafica e testuale di alcune tessere:



N=S S=S W=S E=N NS=0 NE=0 NW=0 WE=0 SE=0 SW=0



N=C S=C W=N E=N NS=1 NE=0 NW=0 WE=0 SE=0 SW=0



N=C S=S W=C E=C NS=0 NE=1 NW=1 WE=1 SE=0 SW=0

N=S S=S W=N E=N NS=1 NE=0 NW=0 WE=0 SE=0 SW=0

N=N S=C W=N E=C NS=0 NE=0 NW=0 WE=0 SE=0 SW=0

N=N S=C W=S E=S NS=0 NE=0 NW=0 WE=1 SE=0 SW=0

N=C S=S W=C E=S NS=0 NE=0 NW=1 WE=0 SE=1 SW=0

### Rappresentazione dello stato della partita

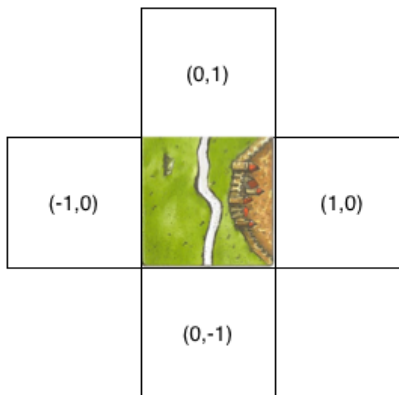
Lo stato della partita è composto dalle seguenti informazioni

1. Turno corrente, cioè il colore del giocatore che deve giocare il suo turno e la tessera che deve posizionare;
2. Pila con le tessere residue non ancora pescate;
3. Segnalini di controllo disponibili per ogni giocatore;
4. Punti accumulati da ogni giocatore;
5. Area di gioco.

L'area di gioco è rappresentata da una griglia le cui righe e colonne sono identificate da un numero progressivo. Ogni cella di questa griglia può essere vuota, oppure può ospitare una tessera. **A inizio partita la tessera iniziale viene posizionata sempre nella cella (0,0)**, cioè la cella corrispondente all'intersezione tra la riga identificata con il numero 0 e la colonna identificata con il numero 0. Le colonne a sinistra rispetto alla prima tessera sono identificate da numeri negativi, quelle a destra da numeri positivi. Lo stesso vale per le righe: le righe superiori sono identificate da numeri positivi, quelle inferiori da numeri negativi.



Area di gioco iniziale:



Il primo giocatore potrà posizionare la sua tessera nelle posizioni (0,1), (-1,0), (1,0), (0,-1) a seconda della compatibilità dei lati.

Man mano che nuove tessere sono posizionate è necessario aumentare le dimensioni iniziali della griglia dell'area di gioco aggiungendo nuove celle vuote ai lati alle tessere posizionate.

### 3.2 Svolgimento di un turno

Un turno si svolge nel seguente modo










1. Aggiornamento visualizzazione dello stato della partita.
2. Possibilità per il giocatore corrente di ruotare di 90 gradi in senso orario la tessera da posizionare un numero arbitrario di volte.
3. Posizionamento della tessera alle coordinate specificate dal giocatore.
4. Possibilità per il giocatore corrente di posizionare un segnalino di controllo su uno dei lati della tessera che contenga una strada o una città.
5. Attribuzione del punteggio e restituzione segnalini in caso di chiusura della città o della strada.
6. Controllo delle condizioni di vittoria: calcolo dei punteggi finali in caso di fine partita oppure inizio del turno del giocatore successivo se esistono ancora tessere da pescare.

#### Visualizzazione stato della partita

Per visualizzare lo stato della partita deve essere mostrato su schermo il colore del giocatore corrente, i segnalini disponibili, la tessera che deve essere posizionata, le tessere posizionate precedentemente e le celle vuote adiacenti ad esse.

La visualizzazione dello stato dovrà avvenire come testo semplice in due dimensioni, convertendo gli elementi grafici del gioco in questo modo:



	(-1,2)	(0,2)	(1,2)	
(-2,1)				(2,1)
(-2,0)				(2,0)
(-2,-1)				(1,-1)
	(-1,-2)	(0,-2)		

```

+-----+-----+-----+
|      |      |      |      |
|      |      |      |      |
|      |      |      |      |
| (-1,2) | (0,2) | (1,2) |      |
|      |      |      |      |
+-----+-----+-----+
|      |      |      |      |
| #     | C1(B) | S1(G) | C1    | #
|      |      |      |      |
| (-2,1) | #S1     | C1.C1 |      | # (2,1)
|      |      |      |      |
|      | #       |      |      | #
|      | S1     | S1     | C2    | #
+-----+-----+-----+
|      |      |      |      |
|      | #       |      |      | #
| (-2,0) | #C1     |      | C1(R).C1 | C1# (2,0)
|      |      |      |      | #
|      | #       | S1     | S1     | #
+-----+-----+-----+
|      |      |      |      |
|      | #       | S1     |      | #
|      |      |      |      |
| (-2,-1) | #C1     | S1.S1 | S2(N)# (1,-1)
|      |      |      |      |
|      | #       |      |      | #
|      |      | C1(V) | S2     | #
+-----+-----+-----+
|      |      |      |      |
|      |      |      |      |
|      |      |      |      |
| (-1,-2) | (0,-2) |      |      |
|      |      |      |      |
+-----+-----+-----+

```

Le celle vuote adiacenti ai lati delle tessere posizionate precedentemente invece saranno rappresentate semplicemente con le loro coordinate.

Il giocatore potrà digitare il comando *"ruota"* per ruotare di 90 gradi in senso orario la tessera da posizionare. La rotazione consiste nella sostituzione della tessera corrente con una nuova tessera uguale alla precedente, ma con i lati e le connessioni scambiati nel seguente modo: N=x → E=x, E=x → S=x, S=x → W=x, W=x → N=x, NS=x → WE=x, NE=x → SE=x, NW=x → NE=x, WE=x → NS=x, SE=x → SW=x, SW=x → NW=x.

Il giocatore potrà digitare il comando "x,y" per posizionare la tessera corrente nella posizione di coordinate (x,y). Se l'azione non è consentita sarà visualizzato un messaggio di errore e il giocatore avrà nuovamente la possibilità di posizionare nuovamente la tessera, altrimenti la tessera è posizionata e la visualizzazione dello stato della partita sarà nuovamente aggiornata.

## Posizionamento segnalino

Dopo aver posizionato la tessera, se il giocatore ha ancora segnalini di controllo disponibili e ne vuole posizionare uno, deve digitare il nome della parte della tessera appena posizionata su cui vuole mettere il segnalino. I nomi disponibili sono quelli contenuti nella tessera, ad esempio se la tessera ha una città e due strade non collegate tra loro i nomi disponibili saranno **C1**, **S1**, **S2**. Se la mossa non è consentita sarà richiesto nuovamente l'inserimento del comando. Se la mossa è consentita il segnalino sarà posizionato e la visualizzazione dello stato della partita sarà nuovamente aggiornata.

Se il giocatore non vuole posizionare alcun segnalino dovrà digitare il comando "*pass*".

#### **Attribuzione del punteggio**

In questa fase si controlla se la tessera posizionata comporta l'attribuzione di punteggio a qualche giocatore. In caso affermativo il punteggio associato al giocatore viene incrementato come da regolamento e tutti i segnalini corrispondenti all'elemento completato vengono restituiti.

#### **Controllo condizioni di vittoria**

In questa fase si controlla se la pila di tessere disponibili è vuota. Se è vuota si sommano i punti relativi agli elementi di gioco incompleti e si determina il vincitore, altrimenti inizia il turno il giocatore successivo.

### **3.3 Altri dettagli implementativi e suggerimenti**

Nelle specifiche di implementazione base non è richiesto l'uso del multithreading, della programmazione di rete con socket o RMI, di swing, di Android e della possibilità di gestire più partite contemporaneamente. La partita sarà gestita da un'interfaccia testuale e i vari giocatori che si alternano nei turni utilizzeranno la stessa tastiera. Tuttavia, per chi prosegue con l'implementazione dei requisiti standard è necessario che la progettazione dell'applicazione sia realizzata tenendo bene a mente il paradigma **Model-View-Controller (MVC)** e gli altri pattern spiegati nel corso di Ingegneria del Software in modo che sia semplice sostituire e separare la parte che contiene la logica del gioco dalla parte che contiene la logica di presentazione e di interazione con i giocatori. In particolare, in caso di implementazione dei requisiti standard, e quindi di aggiunta di una interfaccia grafica e della programmazione di rete, il gioco deve continuare ad essere giocabile senza rete e in modalità testuale.

### **3.4 Specifiche aggiuntive per progetto da tre persone**

Nel caso si svolga il progetto in un gruppo di tre persone è necessario considerare come parte di queste specifiche base anche le regole/specifiche aggiuntive descritte nell'Appendice A alla fine di questo documento.

## Parte 4 - Specifiche implementative standard

In questa sezione presentiamo i requisiti tecnici a cui dovrà rispondere il progetto qualora implementato nella sua interezza. Per un'implementazione completa viene richiesta una versione di Carcassonne che possa funzionare in rete secondo un paradigma client-server.

### Client

I client potranno essere sviluppati sia utilizzando JavaSE che utilizzando la tecnologia Android. Coloro i quali decideranno di sviluppare il client con tecnologia JavaSE dovranno utilizzare obbligatoriamente Swing per le interfacce grafiche e, invece, dovranno scegliere tra l'utilizzo dei Socket o di RMI per la comunicazione di rete. La scelta delle due tecnologie è completamente indifferente ai fini del progetto ma si consiglia di scegliere la tecnologia in sede di progettazione. Coloro i quali, invece, opteranno per Android dovranno utilizzare le risorse native messe a disposizione dall'SDK di Android sia per l'interfaccia grafica che per la comunicazione di rete.

### Server

Il server deve agire come gestore di partite perciò deve permettere di creare una partita, inizializzarla, giocarla e concluderla secondo le regole del gioco. Inoltre, deve essere in grado di gestire più partite contemporanee. Per implementarlo si deve utilizzare tecnologia JavaSE. È di capitale importanza che il server di gioco possa gestire client dei due diversi tipi nella stessa partita.

### 4.1 Modifiche alla specifica

Per espandere le specifiche di base verso quelle standard non è necessario cambiare la specifica delle regole del gioco, mentre, invece, dei cambiamenti sono indotti sulle specifiche implementative. Nello specifico, le specifiche fornite in precedenza vanno ampliate con le seguenti:

7 segnalini rossi, 7 segnalini blu, 7 segnalini verdi, 7 segnalini gialli e 7 segnalini neri

**Avvio Partita** - L'assunzione di base è che ogni client che voglia partecipare ad una partita conosca l'indirizzo (numerico o simbolico) del server. **Non** è, quindi, richiesto un servizio di discovery. Il primo giocatore si connette al server e diventa il giocatore rosso. Questo evento provoca l'avvio in automatico di una nuova partita aperta ad altri giocatori di cui il primo giocatore diventa il fondatore. Questi dovrà anche dare un nome univoco alla partita. Successivamente, tutti gli altri client che si connettono vengono aggiunti alla partita ancora aperta e viene loro assegnato un colore (secondo l'ordine rosso, blu, verde, giallo e nero) fino a quando non si verifica una delle due seguenti condizioni:

1. Tutti i colori disponibili sono assegnati;
2. Se dopo un certo timeout (configurabile) ci sono almeno 2 giocatori connessi alla partita.

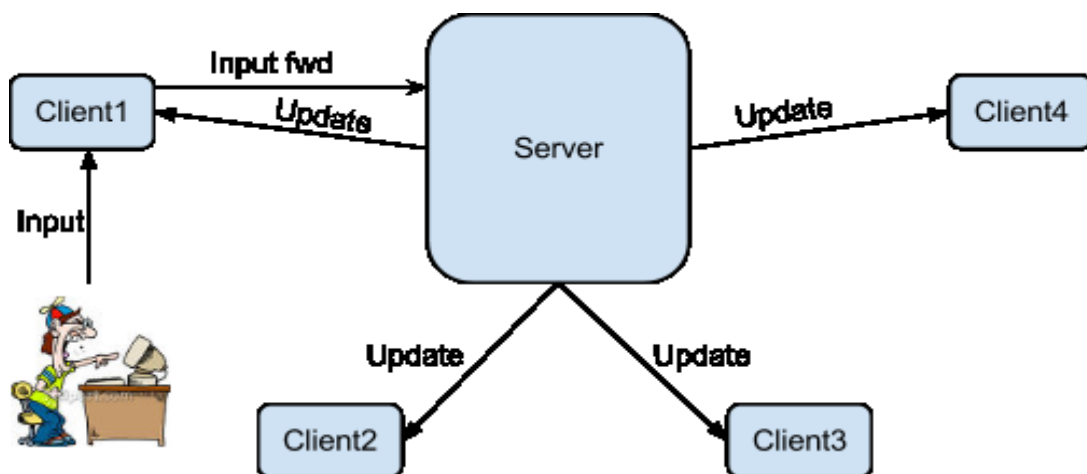
Al verificarsi di una delle due condizioni il server avvia automaticamente la partita e termina la fase di avvio. È importante precisare che un nuovo giocatore che contatta il server può o creare una nuova partita se non ne esistono di già avviate o entrare a far parte di una partita ancora in fase di avvio. Questo significa che non è possibile per un giocatore creare una nuova partita se ne esiste una ancora in fase di avvio.

**Gioco** - In questa fase il server consente ad i giocatori di svolgere i propri turni secondo le regole illustrate precedentemente fino all'esaurimento delle tessere. In più, ora, bisogna anche considerare la possibilità che i client vadano offline. Il server, perciò, deve risultare resiliente a questo genere di problemi. In particolare, ai giocatori è permesso di abbandonare temporaneamente la partita per via della perdita di connettività. Durante l'assenza di un giocatore la partita è sospesa per un certo tempo (definito a priori), trascorso il quale, il server elimina il giocatore assente. L'eliminazione comporta l'eliminazione dei segnalini del giocatore ma non l'eliminazione delle tessere inserite. Dopo l'eliminazione il gioco ricomincia senza il giocatore eliminato. Se, invece, il giocatore dovesse riconnettersi alla partita prima che scada il tempo prestabilito, allora la partita riprenderebbe da dove si era interrotta. Ovviamente durante la sospensione gli altri utenti devono essere informati di quanto sta accadendo e non potranno interagire col gioco.

**Conteggio** - Si ricorda che il conteggio è un compito computazionalmente pesante e, nelle specifiche standard, il server deve essere in grado di gestire più partite contemporaneamente.

## 4.2 Architettura

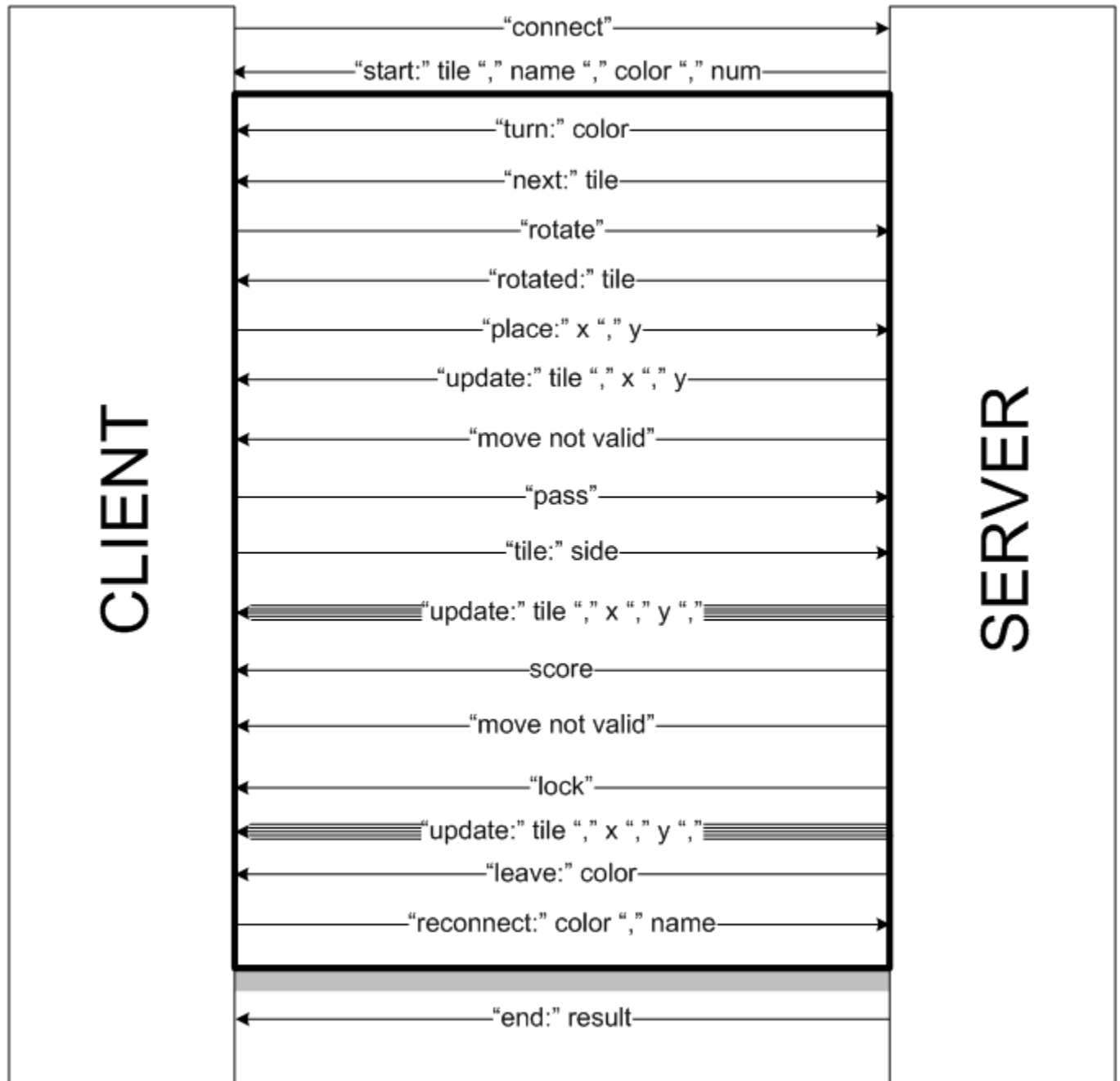
L'architettura software del progetto prevede, come abbiamo già detto, un server centrale ed un certo numero di client che si connettono per giocare. Tutta la logica di gioco è delegata al server mentre i client si limitano, da una parte, a comunicare al server le azioni intraprese dagli utenti e, dall'altra, a visualizzare le conseguenze delle mosse. Per esemplificare meglio, in figura, è rappresentata una partita con 4 client (giocatori).



Il Client1 riceve un qualche genere di input dall'essere umano che lo gestisce. Essendo un client, la richiesta non può essere gestita in locale perciò viene inoltrata al Server (Input fwd). Quest'ultimo processerà l'input, aggiornerà il proprio stato interno, e notificherà a tutti i client partecipanti l'aggiornamento di stato. Si raccomanda **molto caldamente** di utilizzare il pattern **MVC** (Model-View-Controller) per progettare l'intero sistema!

### 4.3 Protocollo di comunicazione

In questa sezione vogliamo fornire il protocollo di comunicazione tra i client e il server. Lo scopo di formalizzare tale protocollo è duplice: da una parte si vuole guidare l'implementazione identificando le operazioni di gioco, dall'altra si vogliono uniformare le implementazioni dei vari gruppi in modo da raggiungere l'interoperabilità. Come risulta evidente dalla figura, tutti i messaggi sono puramente testuali e questo protocollo è pensato per essere implementato direttamente su socket.



In figura sono rappresentati i messaggi. Le stringhe tra virgolette vanno interpretate come tali mentre, invece, le altre sono nomi simbolici definiti come di seguito:

Simbolo	Rappresentazione
tile	Vedi sezione 3.1 <sup>1</sup>
name	Stringa Alfanumerica
color	red   blue   green   yellow   black
x	Stringa Numerica
y	Stringa Numerica
side	N   S   W   E
score	red=num, blue=num, green=num, yellow=num, black=num

Il primo messaggio che viene scambiato contiene la stringa “connect” ed è usato dal client per chiedere al server di connettersi ad una nuova partita. Quando la partita comincia per il verificarsi di una delle due condizioni, il server invia il messaggio “start:” tile”, “name”, “color”, “num” dove tile è la prima tessera da mettere sul tabellone, name è il nome della partita generato dal server, color è il colore assegnato dal server al giocatore specifico e, infine, num è il numero di giocatori che prenderanno parte alla partita. Notare che, essendo i colori assegnati sempre nello stesso ordine, dal numero di giocatori si può dedurre quali siano i colori coinvolti. Questi due messaggi terminano la fase di avvio del gioco. Facendo riferimento alla figura, si entra nell’area delimitata dal rettangolo che rappresenta un singolo turno di gioco. Il singolo turno inizia col messaggio “turn:” color che indica quale giocatore sarà attivo durante il turno che sta per iniziare, a tutti gli altri non è consentito interagire col gioco ma riceveranno tutti gli aggiornamenti dal server visto che tutti i messaggi spediti dal server sono spediti verso tutti i client. D’ora in poi parleremo solo del giocatore che sta attivamente interagendo col server, tutti gli altri riceveranno solo gli aggiornamenti da visualizzare ma non potranno rispondere. Col messaggio “next:” tile il server presenta al giocatore la tessera che deve essere posizionata sul tabellone. Il giocatore può ruotarla in senso orario utilizzando il messaggio “rotate”. Il server reagirà con il messaggio “rotated:” tile che contiene la tessera col nuovo orientamento. Una volta che la tessera sarà orientata correttamente il giocatore segnalerà al server dove posizionarla usando il messaggio “place:” x”, “y. In base alla validità o meno della mossa il server potrà rispondere con il messaggio “update:” tile”, “x”, “y che il client interpreterà

---

<sup>1</sup> Per rappresentare i segnalini posizionati su una strada o una città bisogna aggiungere nella rappresentazione della tessera fornita nella sezione 3.1, subito dopo l’indicazione di strada o città di uno dei quattro lati, l’iniziale del colore del segnalino separata da una virgola. Ad esempio, se ho una tessera rappresentata come:

“N=C S=S W=C E=C NS=0 NE=1 NW=1 WE=1 SE=0 SW=0” (città a nord, ovest, est, strada a sud)

e posiziono un segnalino Verde nella città che è tangente a Nord/Ovest/Est, potrò rappresentare la tessera indifferentemente in questi tre modi diversi:

“N=C+V S=S W=C E=C NS=0 NE=1 NW=1 WE=1 SE=0 SW=0”

“N=C S=S W=C+V E=C NS=0 NE=1 NW=1 WE=1 SE=0 SW=0”

“N=C S=S W=C E=C+V NS=0 NE=1 NW=1 WE=1 SE=0 SW=0”

aggiungendo la tessera nella posizione desiderata sull'interfaccia grafica. Nel caso in cui, invece, la mossa non sia valida il server utilizzerà il messaggio `"move not valid"`. In questo caso l'operazione di posizionamento dovrà essere ripetuta prima di procedere. Nel caso il giocatore non voglia mettere un segnalino sulla tessera appena posizionate, questi dovrà usare il messaggio `"pass"` per informarne il server. Altrimenti si usa il messaggio `"tile:" side` per indicare al server dove posizionare il segnalino. I possibili risultati sono esattamente quelli precedenti, cioè una serie di messaggi `"update:" tile", "x", "y` se la mossa è andata a buon fine e `"move not valid"` in caso contrario. È necessario che il server mandi una serie di aggiornamenti perché il posizionamento di una tessera potrebbe risultare nel completamento di una strada o una città e quindi tutti i segnalini coinvolti dovrebbero essere rimossi. Al termine di questi aggiornamenti il server spedisce a tutti il messaggio `"score"` che contiene i punteggi aggiornati alla fine del turno.

Come abbiamo fatto presente in precedenza, durante il gioco uno o più giocatori possono disconnettersi<sup>2</sup>. Non appena la disconnessione venga rilevata dal server, questi deve immediatamente fermare il gioco ignorando ogni messaggio in arrivo e notificando l'interruzione ai giocatori rimanenti tramite il messaggio `"lock"`. Qualora i giocatori disconnessi dovessero cercare di riconnettersi alla partita dovranno usare il messaggio `"reconnect:" color ", "` name. Tale riconnessione può avvenire solo entro un certo timeout (configurabile all'avvio dell'applicazione), scaduto il quale il server rimuove dal proprio stato interno tutti i segnalini del giocatore che ha abbandonato la partita e spedisce dei messaggi di aggiornamento ai giocatori ancora presenti. Una volta che la fase di aggiornamento ha avuto termine, il server manda il messaggio `"leave:" color` per notificare ai giocatori rimanenti il colore del giocatore che ha abbandonato la partita.

Quando il turno è terminato il ciclo ricomincia (messaggi all'interno del quadrato). Nel caso in cui, invece, le tessere siano esaurite il server procede al conteggio e spedisce a tutti i giocatori il messaggio `"end: " score` contenente i risultati finali della partita.

Nota Bene: Il protocollo presentato è pensato per essere implementato utilizzando il pattern MVC perciò se ne tenga conto in fase di design dell'applicazione.

---

<sup>2</sup> Si suppone che la disconnessione avvenga perché il client perde l'accesso di rete e non perché il client fallisce. Quindi il client continua a mantenere il proprio stato interno tra cui anche il proprio nome che gli servirà per riconnettersi al server di gioco.



## Parte 5 - Specifiche implementative avanzate

Di seguito sono proposte alcune funzionalità avanzate che concorrono alla valutazione. Attenzione: il loro contributo non è necessariamente additivo. Design e codice verranno comunque valutati in quanto tali e contribuiranno al giudizio globale. Si può scegliere la combinazione desiderata di funzionalità, compatibilmente con le scelte di progetto precedenti. Le funzionalità proposte in questa sezione sono avanzate. La loro realizzazione è intesa come un'opportunità aggiuntiva per chi volesse approfondire concetti inerenti le piattaforme o gli algoritmi di gioco che superano le richieste minime del progetto.

### 5.1: Estensioni generali

**Gestione degli utenti.** Realizzare un sistema di gestione degli utenti che supporti il login dei giocatori, conservi per ciascuno le statistiche di gioco (numero di vittorie, numero medio di mosse prima della vittoria) e produca una classifica ordinata prima per numero di vittorie e quindi per numero medio di mosse.

**Stateful server.** Implementare la possibilità di salvare lo stato del server su disco e di ricaricarlo all'avvio successivo. In particolare il server potrà inviare un comando PAUSE ai client per indicare il suo spegnimento e da lì in avanti non accetterà ulteriori comandi, salverà lo stato su disco e terminerà la sua esecuzione. Al riavvio dovrà ricaricare la partita dal punto in cui è stata interrotta.

**Estensione.** Per i gruppi di due persone, implementare l'estensione descritta in Appendice A.

**Bot player.** Sviluppare un client in grado di giocare una partita autonomamente comunicando con il server tramite il protocollo di comunicazione testuale via socket. Si richiede in particolare di progettare un semplice reasoner in grado di decidere data la situazione la prossima mossa da fare (non è necessario che sia la mossa ottima, ma corretta secondo le regole del gioco e, possibilmente, sensata) e l'infrastruttura di comunicazione necessaria per interfacciarlo via socket secondo il protocollo definito precedentemente. Eventuali estensioni (e.g. possibilità di scegliere il livello di abilità) saranno valutate positivamente.

### 5.2: Estensioni per il progetto Swing

**Suggerimento mosse.** Il client supporta le decisioni dell'utente indicando le possibili mosse valide tra tutte quelle realizzabili con la tessera corrente.

**Screenshot.** Salvataggio di uno screenshot grafico del tavolo di gioco. Il client offrirà al suo utilizzatore la possibilità di salvare uno screenshot in formato JPEG del tavolo da gioco.

**Storia.** Indicare in un frame laterale la sequenza delle mosse effettuate dai giocatori in ordine di occorrenza.

### 5.3: Estensioni per il progetto Android

**Geolocalizzazione.** Ogni terminale comunicherà periodicamente al server la propria posizione geografica (localizzazione *coarse* o *fine*, indifferente) e otterrà le posizioni geografiche degli altri

giocatori. Il client Android, dopo aver ottenuto la posizione degli altri giocatori segnalerà al proprio utilizzatore tutti quelli posizionati nel raggio di 100m da lui. Estendere opportunamente il protocollo di comunicazione per supportare questa funzionalità.

**Condivisione.** Invitare i propri amici a giocare inviando loro un'email (l'utente dovrà poter inserire l'indirizzo del destinatario) che indichi l'indirizzo e la porta del server che ospita la partita.

**Energy awareness.** Il sistema dovrà indicare in sovraimpressione il livello attuale di carica della batteria in percentuale. In verde per le percentuali sopra il 30, in giallo per quelle tra 20 e 30, e in rosso per quelle dal 10 in giù. Dovrà inoltre terminare la partita quando il livello raggiungerà il 3%, avvisando con un popup l'utente.

# Appendice A - Regole aggiuntive

Le regole aggiuntive descritte in questa appendice permettono di trasformare la versione di Carcassonne semplificata descritta nella parte 2 di questo documento in una versione del gioco conforme alle regole ufficiali.

## A.1 Regole aggiuntive Campi

### Regole aggiuntive per il posizionamento delle tessere

Nella fase di posizionamento di una tessera un giocatore potrà posizionare il suo segnalino non solo su strade e città, ma anche in uno dei campi della tessera, purché il proseguimento di quel campo nelle tessere posizionate in precedenza non contenga già un segnalino. L'unica differenza rispetto al posizionamento di segnalini su strade e città è che il lato di una tessera potrebbe avere due differenti campi, come nelle seguenti due tessere:



Nella prima tessera il campo **K1** (quello in alto a destra rispetto alla strada) è presente in tutti e quattro i lati, mentre il campo **K2** è presente solo in una parte del lato ovest e sud. Nella seconda tessera sono presenti due campi **K1** e **K2** delimitati da una strada negli stessi lati. Per risolvere il problema di ambiguità un segnalino posizionato su un campo sarà associato a un angolo della tessera anziché a un lato (come normalmente avviene con strade e città). Cioè, nel comando “*posiziona*” il giocatore specificherà un angolo tra **NE** (nord-est), **NW** (nord-ovest), **SE** (sud-est), oppure **SW** (sud-ovest), anziché il lato.

Un'altra differenza è che un segnalino posizionato in un campo non potrà mai più ridiventare disponibile, anche se il campo fosse completato.

### Regole aggiuntive per il calcolo del punteggio

Poiché i segnalini dei campi non possono mai ridiventare disponibili, il punteggio dei campi è calcolato solo a fine partita.

I punti dei campi sono pari a 3 per ogni città **completa** che toccano. I campi sono delimitati dalle strade, dalle città e dai bordi delle tessere.

Ad esempio:



Il giocatore rosso guadagna 6 punti perché possiede due campi che confinano con la città completa 1.

Il giocatore blu e verde guadagnano entrambi 3 punti perché hanno il numero maggiore di segnalini nello stesso campo che tocca solo la città completa 1.

Il giocatore nero guadagna 6 punti perché uno dei suoi segnalini possiede un campo che tocca le città complete 3 e 5, mentre l'altro segnalino nero non produce punteggio perché si trova in un campo la cui maggioranza dei segnalini è del giocatore giallo.

Il giocatore giallo guadagna 12 punti perché possiede la maggioranza in un campo che confina con le città complete 2, 3, 4 e 5.

## Rappresentazione dello stato della partita in presenza di campi

La rappresentazione testuale bidimensionale delle tessere con l'esplicitazione dei campi segue le seguenti regole:

- Se un lato di una tessera contiene una città, non ci sono campi in quel lato.
- Se un lato di una tessera contiene una strada, in quel lato ci sono due campi distinti.
- Se un lato di una tessera non ha né strade né città, in quel lato c'è un unico campo.
- I campi sono rappresentati dalla lettera **K** seguita dal numero progressivo del campo all'interno della tessera. La lettera **K** va messa negli angoli della tessera che sono tangenti ad un campo. Se due angoli diversi sono tangenti allo stesso campo, il numero progressivo dopo la **K** sarà uguale.
- Se un giocatore ha posizionato un segnalino in un campo, l'iniziale del colore del giocatore andrà inserita tra parentesi dopo l'identificatore del campo.

Esempi di rappresentazione testuale bidimensionale campi:



```
+.....+
.K1  S1  K2.
.      .
.S1      C1.
.      .
.K2(R) C1  .
+.....+
```



```
+.....+
.K1(V) C1  K1.
.      .
.S1      S1.
.      .
.K2      K2.
+.....+
```

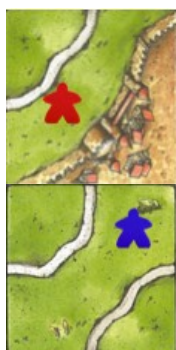


```
+.....+
.K1  S1  K2.
.      .
.S1      S2.
.      .
.K2(B) S2  K3.
+.....+
```

## Rappresentazione dello stato di una tessera contenente segnalini nei campi a livello di protocollo

La rappresentazione di una casella contenente segnalini nei campi può essere rappresentata nei messaggi scambiati tra client e server nello stesso modo con cui le caselle si rappresentano nelle specifiche di implementazione standard, con l'aggiunta dell'iniziale del giocatore di fianco ai parametri esistenti che rappresentano le connessioni tra angoli, separata da una virgola. Ad esempio se il giocatore rosso ha un segnalino nel campo dell'angolo a nord-est, il parametro **NE=x** diventa **NE=x,R**.

Esempi:



N=S S=C W=S E=C NS=0 NE=0 NW=1 WE=0 **ES=1+R** SW=0

N=S S=S W=S E=S NS=0 **NE=0+B** NW=1 WE=0 ES=1 SW=0

## A.2 Regole aggiuntive Monasteri

Alcune tessere aggiuntive presenti nel file **monasteri.txt** presentano una lettera aggiuntiva **M** alla fine di ogni riga. Tali tessere sono dette tessere monastero e hanno le seguenti caratteristiche aggiuntive.

### Regole aggiuntive per il posizionamento delle tessere

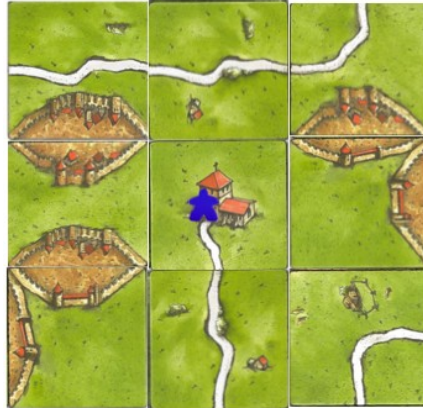
Nella fase di posizionamento di una tessera contenente un monastero un giocatore potrà posizionare il suo segnalino non solo su strade, città e campi, ma anche sul monastero rappresentato nella tessera. Per fare questo il giocatore ha la possibilità di usare **M** come parametro del comando "*posiziona*" del suo turno di gioco anziché specificare un lato o un angolo della tessera.

### Regole aggiuntive per il calcolo del punteggio

Un monastero si considera completato quando la sua tessera è completamente circondata da altre tessere. Il giocatore con il segnalino di controllo nel monastero totalizza 9 punti.

Nel caso in cui alla fine del gioco il monastero fosse incompleto, questo totalizza 1 punto per la sua tessera, più 1 punto per ogni tessera che lo circonda.

Esempio:



Il giocatore blu guadagna 9 punti

## Rappresentazione dello stato della partita in presenza di monasteri

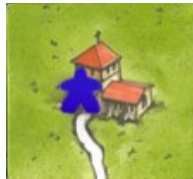
La rappresentazione testuale bidimensionale delle tessere con i monasteri segue le seguenti regole:

- Se nella tessera è presente un monastero viene aggiunta una **M** al centro della tessera.
- Se esiste un segnalino sul monastero di una tessera, viene aggiunto tra parentesi l'iniziale del colore del giocatore che ha posizionato il segnalino.
- Le tessere contenenti monasteri hanno sempre un unico campo.

Esempi di rappresentazione testuale bidimensionale monasteri:



```
+.....+
.K1      K1.
.        .
.   M   .
.        .
.K1  S1  K1.
+.....+
```



```
+.....+
.K1      K1.
.        .
.  M(B) .
.        .
.K1  S1  K1.
+.....+
```

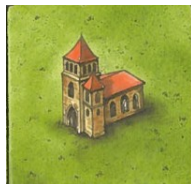


```
+.....+
.K1      K1.
.        .
.   M   .
.        .
.K1      K1.
+.....+
```

## Rappresentazione dello stato di una tessera contenente un monastero a livello di protocollo

La rappresentazione di una casella contenente un monastero può essere rappresentata nei messaggi scambiati tra client e server nello stesso modo con cui le caselle si rappresentano nelle specifiche di implementazione standard, con l'aggiunta del parametro aggiuntivo **M**. Se nel monastero è presente un segnalino, sarà aggiunto al parametro **M** l'iniziale del giocatore che ha messo il segnalino separata da una virgola.

Esempi:



N=N S=N W=N E=N NS=0 NE=0 NW=0 WE=0 ES=0 SW=0 **M**



N=N S=S W=N E=N NS=0 NE=0 NW=0 WE=0 ES=0 SW=0 **M+B**

### A.3 Regole aggiuntive Scudi

Alcune tessere aggiuntive presenti nel file **scudi.txt** presentano una lettera aggiuntiva **U** alla fine di ogni riga. Tali tessere sono dette tessere scudo e hanno la particolarità di avere una sola porzione di città e di seguire le regole di posizionamento dei segnalini della normali tessere. La caratteristica aggiuntiva delle tessere scudo rispetto alle tessere normali è che il punteggio determinato dalla porzione di città che contengono è raddoppiato. Cioè una porzione di città con lo scudo vale 4 punti se la città viene chiusa, oppure 2 punti se la città resta aperta fino a fine partita.

#### Rappresentazione dello stato della partita in presenza di scudi

La rappresentazione testuale bidimensionale delle tessere con gli scudi seguono queste regole:

- Se nella tessera è presente uno scudo viene aggiunta una **U** al centro.
- Una tessera che contiene uno scudo deve contenere una porzione di città.
- Una tessera che contiene uno scudo non può contenere un monastero.

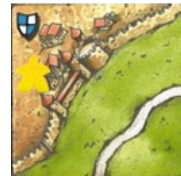
Esempi di rappresentazione testuale bidimensionale scudi:



```
+.....+
.K1          K1.
.
.C1    U    C1.
.
.K2          K2.
+.....+
```



```
+.....+
.      C1      .
.
.C1    U    C1.
.
.      C1      .
+.....+
```

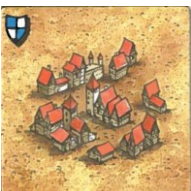


```
+.....+
.      C1    K1.
.
.C1(G) U    S1.
.
.K1    S1    K2.
+.....+
```

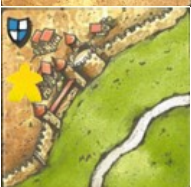
#### Rappresentazione dello stato di una tessera contenente scudi a livello di protocollo

La rappresentazione di una casella contenente scudi può essere rappresentata nei messaggi scambiati tra client e server nello stesso modo con cui le caselle si rappresentano nelle specifiche di implementazione standard, con l'aggiunta del parametro aggiuntivo **U**.

Esempi:



N=C S=C W=C E=C NS=1 NE=1 NW=1 WE=1 ES=1 SW=1 **U**



N=C S=S W=C,G E=S NS=0 NE=0 NW=1 WE=0 ES=1 SW=0 **U**



