Corso di laurea in Informatica - Dipartimento di Informatica Università di Pisa
Progetto - II appello 2019/2020

Attacco la Kamchatka!

Con un omaggio a Andrea Pazienza e al suo Pertini partigiano

Introduzione comune per i corsi di IS e BD

RisiKo! è un gioco da tavolo a turni, in cui da 2 a 6 giocatori combattono per l'occupazione di 42 territori in una mappa politica, cercando di raggiungere una missione segreta che generalmente richiede il controllo dei territori nella mappa o nell'annientamento di un giocatore avversario. La vittoria sarà del primo giocatore che raggiunge il suo obiettivo segreto.

RisiKo! ha diversi limiti di giocabilità noti. Una di queste limitazioni è che la configurazione della mappa è rigida, nel senso che sia il numero di territori che i confini tra territori (che determina come gli attacchi tra territori possono essere eseguiti) sono fissi per tutte le istanze di gioco. In particolare, il numero di territori fisso rende alcune istanze di gioco molto lunghe, specialmente quando è coinvolto un piccolo numero di giocatori.
In questo progetto, svilupperai una versione online e configurabile di RisiKo!, che supera le limitazioni sopra menzionate.

Descrizione del progetto

Attacco la Kamchatka! è una versione per computer altamente configurabile di RisiKo!.
RisiKo! è un gioco da tavolo, basato su turni, giocata su un tabellone raffigurante una mappa politica della Terra, divisa in vari territori, raggruppati in continenti. Lo scopo del gioco è quello di raggiungere un obiettivo, assegnato a ciascun giocatore all'inizio del gioco e tenuto segreto dagli altri giocatori. I giocatori controllano eserciti, che sono schierati in territori occupati (e raggruppati in continenti) e possono essere usati per catturare territori da altri giocatori. I risultati degli attacchi tra territori sono determinati lanciando dadi.

L'obiettivo di questo progetto è sviluppare una applicazione in cui gli utenti giocano a RisiKo!. Gli utenti useranno un'app per giocare, mentre lo stato del gioco viene gestito in maniera centralizzata. Il gioco deve implementare tutte le funzionalità di base della versione standard di RisiKo! (per una descrizione dettagliata: https://it.wikipedia.org/wiki/RisiKo! versione base italiana, spiegata anche in http://www.regoledelgioco.com/giochi-da-tavolo-di-ruolo-strategia/risiko/), comprese le regole per ottenere e piazzare eserciti, attaccare territori e fortificare, oltre a supportare le carte di RisiKo!. Inoltre, il gioco deve supportare maggiori livelli di configurazione, attraverso la possibilità di scegliere la mappa su cui si desidera giocare e diversi livelli di complessità.

Più precisamente, il sistema deve:

- consentire agli utenti di definire e salvare nuove configurazioni di mappe, modificando la mappa ufficiale del gioco, riducendo il numero di territori (rimuovendo i confini) e, se utile, il numero dei continenti;
- garantire la presenza della Kamchatka e della Čita in ogni configurazione;
- consentire agli utenti di caricare configurazioni di mappe salvate;
- costruire automaticamente delle mappe di diversa granularità, collassando i territori vicini in base al numero di giocatori e al tempo disponibile per il gioco;
- permettere a un utente di dichiarare la propria disponibilità a giocare in un dato momento;
- gestire una partita;
- permettere di organizzare tornei, caratterizzati da partite su mappa standard e pari numero di giocatori;

Scopo del progetto

Il progetto deve implementare le funzionalità sopra descritte, ovvero tutte le regole dello standard RisiKo! oltre alle caratteristiche di configurabilità. Opzionalmente, un singolo obiettivo per il gioco può essere implementato al posto delle carte obiettivo (questo obiettivo comune per tutti i partecipanti può essere, ad esempio, conquistare i due terzi del mondo). Il gioco deve avere almeno una mappa precaricata (può essere la mappa standard di RisiKo!), con i tre livelli di complessità (facile, medio, difficile), che causano il collasso nei territori.

Progetto di Ingegneria del Software

Attacco la Kamchatka!

IS: Consegna entro il 15 giugno 2020 ore 23:59

Regole di consegna del progetto per IS

Oltre alle informazioni pubblicate su didawiki (e replicate sul sito valutami):

1. Deve essere consegnato un archivio contenente:
    a. il file .vpp del progetto
    b. un file pdf che possiamo stampare per correggere più velocemente i progetti. Deve contenere: nome cognome e matricola dei membri del gruppo; le parti testuali del progetto; i diagrammi prodotti, rispettando l'ordine degli esercizi dati
2. Sia i file che l'archivio devono chiamarsi col cognome del referente del gruppo (chi fa la sottomissione del progetto).
3. Il progetto deve essere inviato per email alla propria docente entro la data fissata: roberta.gori@unipi.it laura.semini@unipi.it
4. Per correttezza nei confronti dei compagni di gruppo, chi sottomette mette in CC tutti i membri del gruppo.
5. La mail deve avere subject: **ProgettoIS2**
6. Tutte le mail (mittente e destinatari) devono essere istituzionali (@unipi.it / @studenti.unipi.it).

Scopo del progetto per quanto riguarda Ingegneria del software

In aggiunta ai requisiti descritti sopra, si assuma che ci sia una persona, giocatore o organizzatore che configura una partita e invita i giocatori a giocare. L'invito può essere personale (per esempio nei tornei) o inviato a tutti i giocatori che in quel momento sono online sulla piattaforma. La stessa persona avvia la partita.

Nel progetto:

1. Descrivere con un diagramma UML tutti i casi d'uso del sistema. Per uno di essi (non banale) dare la narrativa.
2. Dare una diagramma delle classi che descriva le carte e la mappa.
3. Dare un diagramma di attività che modelli un turno di gioco.
4. Dare un diagramma di macchina a stati che modelli l'evoluzione di una coppia di stati adiacenti quando sono teatro di un attacco.
5. Descrivere l'architettura della parte del sistema che realizza configurazione di una partita e gestione del gioco secondo la vista C&C, curando di specificare le interfacce delle componenti. Usare (almeno) gli stili publish-subscriber e MVC. Dare anche una vista ibrida assumendo di avere dei fat client.
6. Dare un diagramma di sequenza che descriva come l'architettura individuata realizza uno dei due casi d'uso del punto 5.
7. Dare un diagramma di struttura composita di una delle componenti individuate.

8. La seguente procedura calcola nel vettore risultato[] l'esito di un attacco. Si suppone che i vettori siano tutti lunghi 3 ma che i loro valori significativi siano terminati dal -1. I vettori dell'attaccante e del difensore contengono i risultati dei loro lanci (al più tre per ciascuno). Il vettore risultato contiene una sequenza di 1 e 2: 1 se l'attaccante vince il singolo scontro e 2 se lo perde.

```
void esitoAttacco(int attaccante[ ], int difensore [ ], int risultato[ ]) {

    int i=0, lun1=0, lun2=0, min=0;
    while (i<3  && attaccante[i] != -1)
            {lun1=lun1+1;i++;}
    i=0;
    while (i<3  && difensore[i] != -1)
            {lun2=lun2+1; i++;}
    \\ Sort (int a[], int n, int m) ordina a[] in modo decrescente, tra le posizioni n e m
    Sort(attaccante, 0, lun1-1);
    Sort(difensore, 0, lun2-1);
    if  (lun1>lun2)
            min=lun2;
    else    min=lun1;
    for(i=0; i< min; i++){
            if (attaccante[i]> difensore[i])
                    risultato[i]=1;
            else risultato[i]=2;
            }
    risultato[i]= -1;
    }
```

a. Scrivere un driver per testare il metodo esitoAttacco con un caso di test a piacere e che stampi se il test ha avuto successo o meno.
b. Disegnare il grafo di flusso del metodo esitoAttacco, usando un diagramma di attività (etichettare i nodi con codice e non con i numeri di linea)
c. Dare una batteria di casi di test che copra al 100% le decisioni e le condizioni
d. Dare la misura della copertura delle condizioni semplici della batteria individuata al punto c.