Luca Bonelli e Gaspare Riccardo Lo Bue

Relazione su “P2P Chess”

Progetto per l’esame di Tecnologie Internet

Il progetto è una web application.

Il file principale è index.html, che specifica la struttura della pagina. Importa inoltre i file javascript e css necessari.

Per quanto riguarda lo stile dell’applicazione, sono state usate librerie W3.CSS, che implementano classi di stile applicabili agli elementi html, e il file index.css.

I file Javascript sono due: chessGame.js gestisce gli aspetti della partita di scacchi e agisce sull’html.

Invece pieces.js definisce classi e funzioni utili per la logica scacchistica.

Comunicazione P2P

Una volta finito il caricamento della pagina, viene eseguito chessGame.js.

Viene inizializzato l’oggetto Peer: si instaura una connessione con un server PeerJS che assegna al Peer un ID univoco.

Da questo punto in poi il peer rimane in ascolto di connessioni da parte di altri peer tramite un gestore di eventi: peer.on('connection', function (c) {…});

L’oggetto c passato come argomento alla funzione è di tipo Connection e identifica una connessione tra due peer.

In questa applicazione, ricevere una connessione equivale a ricevere una richiesta di gioco, quindi la funzione che gestisce la ricezione di connessione renderà visibile il menù per scegliere se accettare o rifiutare la partita.

Se si vuole invece inviare una richiesta di gioco, si può cliccare sul Button di conferma e verrà creata la connessione (oggetto Connection) con il peer il cui ID è stato scritto nella TextBox.

Una volta creato o ottenuto l’oggetto Connection, esso viene utilizzato per l’invio e la ricezione di dati.

I dati scambiati in questa applicazione possono essere di tipo String (usati per la gestione delle fasi della partita: hanno valore “start”, “resign” o ”refuse”), oppure oggetti.

Il secondo tipo è utilizzato alla fine del turno di un giocatore, quando viene inviata all’altro peer la mossa effettuata.

La mossa è descritta da un oggetto che contiene le vecchie e le nuove coordinate del pezzo mosso (le coordinate sono relative alle caselle della scacchiera).

Se durante la partita un peer si disconnette, l’altro peer vince la partita.

Scacchiera

La scacchiera dal punto di vista grafico è un canvas che viene colorato tramite Javascript.

Quando viene cliccato l'interno del canvas, viene riconosciuta la casella cliccata attraverso le funzioni getMousePos e isPointInsideRect.

Se si clicca su un proprio pezzo nel proprio turno, vengono evidenziate le caselle in cui esso può muovere dalla funzione showPossibleMoves.

Se si clicca su una di queste caselle evidenziate, si muove il pezzo e si invia la mossa all'altro peer.

L'ultima mossa effettuata viene evidenziata di verde dalla funzione drawMove.

Logica degli scacchi

La posizione dei pezzi nella scacchiera viene rappresentata da una matrice 8x8 di oggetti della classe Square o delle classi che ereditano da essa (definite nel file pieces.js).

Ognuna di queste classi (King, Queen, Rook…) ha attributi per identificare la posizione e il colore del pezzo, e metodi che restituiscono le caselle in cui esso può muovere (possibleMoves) o che minaccia (canAttack).

Il metodo possibleMoves viene richiamato quando si clicca sul pezzo durante il proprio turno. Un'altra funzione (showPossibleMoves, nel file chessGame) si occupa di mostrare graficamente queste mosse.

Funzionamento del metodo per determinare le mosse possibili:

1. Clicco su un pezzo, viene richiamato il metodo possibleMoves di esso.
2. Si trovano tutte le possibili mosse che il pezzo può fare in base alle sue regole di movimento e alla posizione degli altri pezzi nella scacchiera. Si ignora però se queste mosse possano mettere in pericolo il proprio re.
3. Ognuna di queste mosse, prima di essere dichiarata valida, viene quindi passata alla funzione checkMove.
4. checkMove simula la mossa in una scacchiera temporanea e richiama il metodo canAttack di ogni pezzo nemico per controllare se uno di essi può "mangiare il re". Se così fosse, dichiara che la mossa in realtà non è possibile.
5. Se la mossa non è possibile, viene scartata dal metodo possibleMoves.

All'inizio e alla fine di ogni turno viene anche controllata la presenza di uno scacco matto o di uno stallo utilizzando queste funzioni.

Scacco matto = nessuna possibleMoves di alcun pezzo e re in scacco nella posizione attuale.

Stallo = nessuna possibleMoves di alcun pezzo e re non in scacco nella posizione attuale.