Liar's Dice
Sistemi Distribuiti AA 2015/2016

Davide Aguiari, Fabio Proietti

Universit di Bologna, Dipartimento di Scienze dell'Informazione
davide.aguiari@studio.unibo.it
fabio.proietti@studio.unibo.it

Abstract. Liar's Dice un gioco da tavolo nato nel sud America e basato su un meccanismo di scommesse e bluff. Dopo una breve introduzione, andremo a spiegare quali sono state le scelte progettuali utilizzate per implementare questo passatempo, incluse l'architettura della rete, le librerie implementative e gli aspetti legati alla usabilit e correttezza del sistema.

1 Introduzione

La nostra idea di progetto stata quella di sviluppare la versione multiplayer del famosissimo gioco da tavolo, con un approccio di rete totalmente distribuito (peer-to-peer) e attribuendo particolare attenzione all'affidabilit del sistema.
Liar's Dice [1] originario del sudamerica e importato in Europa dai *conquistadores* nel XV secolo. Il gioco conosciuto anche con i nomi di *Perudo*, *Dudo* o, grazie anche al film Pirati dei Caraibi, come *Pirate's dice*.[1]
Le successive sezioni tratteranno la logica del gioco, gli aspetti progettuali, le scelte implementative adottate, la valutazione dopo la fase di test, e per finire alcune considerazioni conclusive sul sistema e possibili miglioramenti futuri.

2 Logica del Sistema

Per rendere pi semplice la compresione progettuale del sistema, in questa sezione spiegheremo pi in dettaglio le regole del gioco.
Ogni partita si gioca con un numero di giocatori con un minimo di due ad un massimo di 8 (ma potrebbe essere superiore in base alla versione). Tutti i giocatori all'inizio di un nuovo round lanciano i dadi a loro disposizione (inizialmente sono 5) e li tengono comperti in modo che gli avversari non vedano i valori ottenuti. A questo punto colui che inizia il round, in base anche ai suoi dadi, deve effettuare una scommessa nel quale indicher quanti dadi dello stesso numero sono potenzialmente in gioco. A questo punto il giocatore successivo pu scegliere se credere alla precedente scommessa oppure dubitare. Nel primo caso esso tenuto ad effettuare un rilancio sulla base dell'ultima scommessa, definendo quindi o un

[1] https://www.youtube.com/watch?v=piGg5ZrmoQA

valore del dado maggiore rispetto al precedente oppure un numero di occorrenze superiore alla scommessa appena effettuata e passare di nuovo il turno. Nel caso in cui dubita (nella versione inglese il giocatore urla Liar!), tutti i partecipanti alla partita sveleranno i propri dadi per decretare chi ha vinto. Se la scommessa contiene un numero di occorrenze minore o uguale rispetto al valore del dado, il giocatore che ha precedentemente scommesso vince, e il suo "accusatore" perde un dado; altrimenti il vincitore sar chi ha dubitato e a perdere un dado lo scommettitore. Il round successivo quindi il giocatore perdente lancer un dado in meno rispetto agli altri. Se un giocatore rimane senza dadi avr automaticamente perso la partita e a vincere sar colui che riuscir ad eliminare tutti i partecipanti alla sfida.
La nostra versione inoltre contiene anche il dado jolly. Lanciando i dadi infatti, se il giocatore effettua una scommessa che non contenga il valore uno del dado, pu trasformare il valore delle facce che contengono 1, in un valore a suo piacimento. D'altro canto se l'oggetto della sua scommessa contiene dadi con valore 1, la possibilit di utilizzo del jolly decade.

3 Aspetti Implementativi

3.1 Design Pattern

La nostra implementazione si subito orientata sull'utilizzo del design pattern Model View Controller[2]. MVC si sposa perfettamente con le nostre esigenze: la parte che costituisce il Model infatti gestisce direttamente i dati, la logica e le regole del gioco; la View ha lo scopo di mostrare le informazioni elaborate (e quindi l'output grafico) all'utente, che tramite il Controller in grado di manipolare.

3.2 Implementazione della logica del gioco

La figura XXX mostra il digramma UML delle classi utilizzate per implementare la logica della nostra applicazione. Lo scopo del Model quindi quello di generare i giocatori, i dadi, verificare le scommesse e riorganizzare lo stato dello stesso in caso di guasti improvvisi.

3.3 L'interfaccia grafica

Per l'implementazione dell'interfaccia grafica ci siamo affidati alla libreria Open Source Slick2D [3]. E' formata da un insieme di tools e funzionalit basate sulle librerie grafiche OpenGL e LWJGL, offrendo quindi un supporto per includere animazioni, immagini, suoni, ecc. alla nostra grafica 2D. L'utilizzo di Slick2D passa attraverso l'implementazione di una classe principale che ha lo scopo di gestire tutte le caratteristiche principali della view come il frame rate, la grandezza della finestra, la modalit di rendering, il passaggio di stato ecc. Tutte le classi che invece definiscono gli stati del sistema, si basano su tre metodi principali:

– void init(): il metodo richiamato al lancio dell'interfaccia e serve principalmente per l'inizializzazione e il caricamento degli elementi;
– void update(): un metodo ciclico che aggiorna costantemente lo stato degli elementi inizializzati dal precedente metodo attraverso le informazioni generate dalla componente Controller;
– void render(): in base al refresh rate definito nella classe principale mostra gli elementi disegnati costantemente aggiornati.

4 Aspetti Progettuali

5 Valutazione e Conclusioni

References

1. https://en.wikipedia.org/wiki/Liar's_dice
2. https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
3. http://slick.ninjacave.com/