Alessio La Greca 581578

Progetto per il corso Sviluppo di Applicazioni Mobili

Docente: Professor Vincenzo Gervasi

Capitolo 1: Introduzione

Air Androckey è un'app pensata per simulare una partita ad air hockey tra due amici, dove per vincere bisogna aver fatto almeno cinque punti e due punti in più dell'avversario. Per iniziare una partita i due giocatori dovranno scegliere chi fungerà da client e chi da server. A questo punto l'utente client manda una richiesta di gioco al server. Il meccanismo di comunicazione utilizzato è il Bluetooth, pertanto se i due dispositivi risultavano già accoppiati la richiesta di inizio partita verrà visualizzata immediatamente dal server. Altrimenti verrà richiesto ai dispositivi di accoppiarsi. Una volta accoppiati, il server deciderà se accettare o meno la richiesta, e in caso affermativo la partita avrà inizio. I due giocatori possono muovere usando il touch il proprio disco di gioco, e l'obiettivo è quello di mandare il dischetto nero nella porta avversaria, rappresentata dal lato inferiore dello schermo avversario.

Capitolo 2: Framework utilizzati

2.1: Bluetooth

I dispositivi dei due partecipanti sfrutteranno il bluetooth come meccanismo di comunicazione durante lo svolgimento della partita. La scelta del bluetooth come mezzo di comunicazione deriva dal fatto che i due giocatori devono essere vicini al momento della partita, idealmente ponendo la parte superiore del proprio device attaccata a quella del device dell'amico. Ciò implica che per poter giocare, i device dei due partecipanti devono prima essere accoppiati. Vediamo come si fa dunque a instaurare una partita.

Partendo dalla main activity, un giocatore sceglierà di aprire una partita (fungendo quindi da server) cliccando sul pulsante centrale. Questo farà sì che il device si renda discoverable ad altri dispositivi per la durata di 15 secondi, in attesa di una connessione. L'altro giocatore cliccherà sul primo pulsante così da arrivare ad un'activity che gli permetterà di unirsi ad una partita. Per poterlo fare, l'utente deve selezionare dalla listview il nome del dispositivo che sta fungendo da server. Se però i due device non si sono accoppiati in precedenza, tale nome non comparirà nella listview. Per poterli accoppiare in caso di necessità, il client non deve far altro che cliccare sul pulsante posto sulla parte inferiore dello schermo, che manderà una richiesta di accoppiamento a tutti i dispositivi vicini. Il server, essendosi reso discoverable, riceverà tale richiesta, e ai due utenti verrà richiesto con un dialog di sistema se accoppiarsi o meno. Se l'accoppiamento ha successo, o se i due device erano già accoppiati, il server avrà la possibilità di scegliere con due pulsanti se iniziare una partita con l'altro oppure no, visualizzando il nome del device client sullo schermo.

Durante lo svolgimento della partita il bluetooth entra in gioco prima all'inizio del match, per sincronizzare i due device, e poi tutte le volte in cui il dischetto nero tocca la parte superiore dello schermo. Quando ciò avviene, il dispositivo su cui si trovava manda dei messaggi all'altro, comunicandogli:

- La distanza del dischetto dalla parete sinistra fratto la larghezza dello schermo
- La velocità orizzontale del dischetto fratto la larghezza dello schermo
- La velocità verticale del dischetto fratto la larghezza dello schermo

Con questi tre dati, l'altro device è in grado di creare un nuovo dischetto nero in cima allo schermo, con la giusta velocità e posizione.

Il meccanismo del bluetooth viene usato anche in altri due casi:

- Quando il giocatore fa un punto, avvertendo l'altro dispositivo di aggiornare la sua variabile che tiene traccia del punteggio dell'avversario
- Alla fine della partita, quando ai due giocatori viene chiesto se fare un altro game oppure no

La gestione del bluetooth è a carico di un Bound Service interno all'applicazione, che conserva sia i socket necessari, sia gli stream di dati di input e output, fornendo con dei metodi la possibilità alle altre parti della app di effettuare delle operazioni di lettura e lettura su questi socket. In particolare, quando un utente entra nell'activity ClientResearchActivity o ServerResearchActivity, tale activity si lega al service per effettuare le operazioni bluetooth di collegamento. Se poi la partita ha inizio, queste activity fanno l'unbind dal service, e quelle che le seguono (ClientConnectedActivity e ServerConnectedActivity) fanno il bind. Una volta che la partita è terminata, anche queste fanno l'unbind, e tutte le risorse usate vengono liberate.

2.2: Touch

Per far sì che i giocatori possano muovere il proprio disco durante la partita ho scelto di usare il meccanismo di touch. In particolare, i due dischi principali si sposteranno nella posizione dello schermo toccata dall'utente, a patto che il disco sia già presente al suo interno. Trascinando il dito è poi possibile muovere il disco all'interno dello schermo. Quando questo tocca il dischetto nero, a quest'ultimo viene applicata una velocità basata sulla velocità che già aveva sommata alla velocità del disco principale, data da un VelocityTracker. I comandi touch vengono disattivati in alcune occasioni:

- All'inizio di una partita, quando c'è il counter iniziale
- Quando un giocatore fa punto

Anche i pulsanti etichettati "sì" e "no" che compaiono al termine di una partita sfruttano il touch per ricevere l'input dell'utente.

2.3: MediaPlayer

Per implementare effetti sonori all'interno dell'applicazione è stato utilizzato il MediaPlayer di Android. Quando una partita ha inizio, all'interno del GameThread, vengono allocati cinque mediaplayer diversi:

- Uno per i suoni del counter iniziale
- Uno per l'effetto sonoro legato al disco comandato dall'utente
- Uno per l'effetto sonoro sprigionato dal dischetto nero quando sbatte contro una parete
- Uno per la colonna sonora di gioco
- Uno per l'effetto sonoro riprodotto quando un giocatore fa un punto

Essendo il gioco pensato per essere giocato in locale, ovvero con i due giocatori fisicamente vicini tra loro, a partita iniziata solo uno dei due dispositivi riprodurrà gli effetti sonori di inizio partita e la colonna sonora. Gli altri suoni saranno invece contestuali al luogo in cui si verificano gli urti e al giocatore che ha fatto punto.
Se un particolare suono deve essere riprodotto ma è già in esecuzione, allora viene interrotto e riprodotto da capo (ciò si verifica soprattutto nel caso dell'urto tra parete e dischetto nero, che prenderemo come esempio). Questa scelta deriva dai seguenti fatti:

1. Per riprodurre più istanze del medesimo suono in contemporanea avrei dovuto creare ulteriori MediaPlayer, complicandone la gestione e consumando più risorse
2. Realisticamente parlando, non sono molti i casi in cui il dischetto nero tocca le pareti ad una frequenza tale da richiedere che il suo suono, tra l'altro piuttosto breve, venga riprodotto più volte nella stessa finestra temporale. Quello che spesso accade è che il suono viene riprodotto, termina e il dischetto si trova ancora piuttosto lontano dal prossimo rimbalzo.

3. Riprodurre lo stesso effetto sonoro più volte in poco tempo sarebbe potuto risultare sgradevole all'orecchio dell'utente.

La colonna sonora del gioco viene riprodotta a un volume più basso rispetto a quello degli effetti sonori, e viene messa in pausa quando un giocatore fa punto. Inoltre, quando un giocatore supera i quattro punti e si avvicina quindi alla vittoria, la musica cambia.
Al termine dell'activity di gioco su tutti i MediaPlayer viene chiamato il metodo .release(), così da liberare tutte le risorse usate.

Capitolo 3: Thread usati

3.1: MyConnectionThread

Durante la fase di sviluppo iniziale del progetto mi sono reso conto che la gestione della connessione bluetooth poteva dare problemi in caso di mutithreading. In particolare avevo la necessità di assicurarmi che solo un thread alla volta, almeno in fase di instaurazione della connessione, potesse accedere ai socket e agli stream presenti nel service. Pertanto, per il thread che si occupa dell'instaurazione della connessione, appunto MyConnectionThread, ho deciso di utilizzare il pattern singleton. In questo modo, tutte le volte che un utente vuole attivare un nuovo thread di connessione dovrà prima distruggere quello precedente, se esiste ed è in esecuzione, e liberare le risorse di connessione attualmente usate.
Il thread si comporta da client o da server a seconda dell'activity che lo ha lanciato. Se si comporta da server, come prima cosa, apre un nuovo ServerSocket, poi fa l'accept e poi comincia un meccanismo di 3-way hand shake atto a sincronizzare i dispositivi. Il server, in particolare:

1. Cerca di leggere dallo stream il nome del device client che vuole connettersi a lui
2. Stampa a schermo il nome di tale device chiedendo all'utente se vuole accettare la connessione oppure no. L'utente, premendo uno dei pulsanti, effettua una write sullo stream inviando un messaggio all'altro dispositivo
3. Se l'utente ha confermato di voler giocare con l'altro giocatore, il thread aspetta un ulteriore messaggio da parte dell'altro device, e solo una volta ricevuto la partita ha inizio. Questo serve a sincronizzare i due dispositivi. Se invece l'utente rifiuta, non c'è nessuna lettura, e l'activity e il thread terminano.

Nel caso in cui invece il thread debba comportarsi da client, come prima cosa apre il socket verso il server, effettua la connect e inizia a sua volta il 3-way hand shake, dove:

1. Comunica al dispositivo server il proprio nome con una write
2. Aspetta la risposta mediante una read
3. Se la risposta è affermativa, la partita ha inizio. Altrimenti, il thread termina e le risorse del bluetooth vengono liberate.

In ogni caso, quando la partita ha inizio, questo thread termina, ma le risorse relative al bluetooth rimangono all'interno del service.

3.2: RenderThread

Questo thread viene istanziato all'inizio di una partita, con lo scopo di aggiornare l'interfaccia utente durante il gioco. In particolare si occupa di:

- Creare il background di gioco
- Creare il disco che il giocatore può muovere e spostarlo quando l'utente ci interagisce
- Muovere il dischetto nero sulla base della sua velocità attuale
- Aggiornare il counter del punteggio del giocatore quando questo fa un punto
- Mostrare al centro dello schermo un testo quando il giocatore fa goal o lo subisce
- Dare all'utente la possibilità di rigiocare mediante dei pulsanti alla fine di una partita

Il thread è stato scritto dal Professor Vincenzo Gervasi durante una demo di un' app android nel Corso di perfezionamento in Game Design dell'Università di Pisa.

3.3: GameThread

Quest'ultimo thread gestisce la maggior parti inerenti al gioco vero e proprio. In particolare:

- Crea e gesisce i mediaplayer durante la partita
- Aspetta che l'altro device gli comunichi qualche tipo di informazione, che può essere la posizione e velocità del disco oppure il fatto che l'avversario ha fatto punto.
- Libera tutte le risorse associate ai MediaPlayer quando la partita termina
- Aggiorna la lista di elementi che il RenderThread deve aggiornare durante la sua esecuzione
- Gestisce il touch alla fine della partita quando i due giocatori devono scegliere se giocare ancora o no