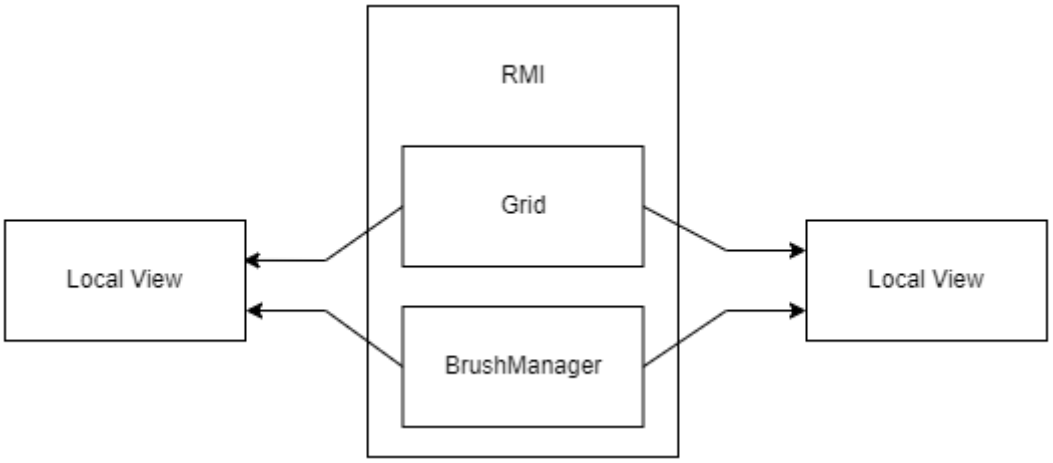
[Distributed Programming with Distributed Objects]

Descrizione problema

Il punto 3 del terzo assignment chiedeva di realizzare la stessa pixel art distribuita del punto 2 ma usando java RMI.

Introduzione

A differenza del caso precedente, questa volta è stato necessario cambiare pesantemente il codice fornito per poterlo adattare alle specifiche imposte dalla tecnologia. Questo in particolare è visibile nei numerosi try-catch che sono stati aggiunti e dall'introduzione delle due interfacce (estendenti Remote) per poter registrare nel server il brushManager e la griglia. Lo schema risultante è il risultate:



In pratica sono state spostate le strutture dati nel server, mentre permane in locale la view. Ogni volta che quest'ultima dovrà essere aggiornata, si faranno richieste agli oggetti remoti per ottenere i dati necessari al refresh.

Client-Server

Nello sviluppo del progetto è stato ripresa la struttura proposta negli esempi del laboratorio: si ha quindi un server che registra griglia e brushManager in RMI, e un client che, invece, avvia la pixelArt vera e propria richiedendo i riferimenti remoti.
Ogni client avrà un proprio brush in locale di cui esisteranno fondamentalmente due copie, una locale e una remota. Ci sarà poi un thread separato (che fa da refresher) che ogni 10 millesimi di secondo sincronizza le due copie e aggiorna la View.
In questo modo solo l'evento di click sulla griglia sarà immediatamente trasmesso all'oggetto remoto, mentre il cambio colore del brush e il movimento del mouse saranno gestiti localmente (questo dovrebbe migliorare le performance).

Differenze rispetto la versione locale

Il funzionamento della versione RMI è praticamente sovrapponibile a quello locale. Le uniche differenze sono quelle elencate in precedenza e lo spostamento del metodo draw dal brushManager al VisualizzerPanel: il brush manager ritorna la mappa di tutti i brush, e su questa si applica il foreach per poterli renderizzare uno a uno.
Come nel caso ad attori, anche in questo progetto si è deciso di far apparire più trasparenti i brush remoti: questa volta non si può usare un campo nella classe brush (visto che, a differenza del caso ad attori, qui la mappa sarà condivisa da tutti), ma si usa l'id del brush stesso.

Id

Il brushManager ha un contatore interno che verrà usato per la definizione degli id (incrementale). Ogni volta che un client richiede al brushManager di aggiungere il proprio brush, sarà restituito l'id assegnatoli che verrà passato prima alla view e poi al VisualizzerPanel. In particolare:

- La view lo userà per la rimozione del brush locale alla sua chiusura;
- Il visualizzerPanel lo sfrutterà, invece, come discriminante per renderizzare diversamente i vari brush.

Concorrenza e Performance

Leggendo qualche documentazione si legge che RMI applica di suo dei sistemi di protezione thrade-safe, permettendo di evitare di inserire lock per l'accesso agli oggetti remoti.
Per quanto riguarda le performance, queste sono di gran lunga inferiori a quelle della versione ad attori con vistosi lag anche nei pc più potenti. Si è provato a migliorare la situazione, ma si deve concludere che i rallentamenti sono dovuti alla tecnologia stessa.