UNIVERSITÀ DEGLI STUDI
DI PERUGIA

Tesina di

Programmazione di Interfacce Grafiche e Dispositivi Mobili

Corso di Laurea in Ingegneria Informatica ed Elettronica A.A. 2018-2019

Dipartimento di Ingegneria

Docente

Prof. Luca Grilli

JPlumber

Applicazione Desktop Java FX

Studenti

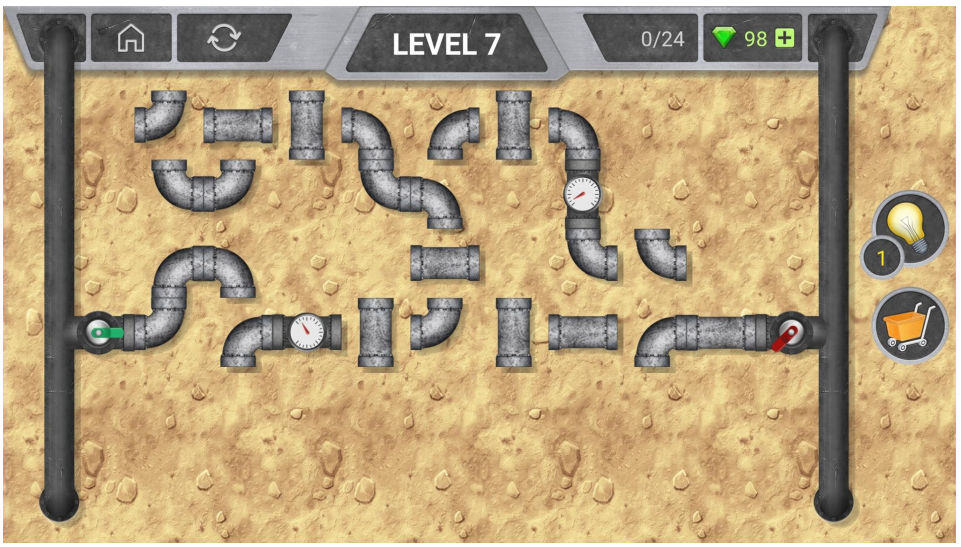
303078	Francesco	Aristei	francesco.aristei@studenti.unipg.it
301850	Ledio	Sheshori	ledio.sheshori@studenti.unipg.it

1. Descrizione del Problema

L'obiettivo di questo lavoro è lo sviluppo di un'applicazione desktop denominata JPlumber, la quale consiste nell'emulazione dell'app Pipe Line disponibile su Play Store.
L'applicazione sarà implementata utilizzando la tecnologia Java FX per ottenere un'animazione fluida.
Il codice sarà testato per la piattaforma Windows.
Di seguito verrà spiegato il funzionamento del gioco Pipe Line e una descrizione della versione da noi implementata (JPlumber).

1.1 Pipe Line

Pipe Line è un gioco rompicapo in cui l'utente, in un numero finito di mosse, deve riuscire a collegare una serie di tubi al fine di creare un percorso continuo per l'acqua dalla sorgente ad una destinazione data.
L'utente, vedrà di fronte a sé una schermata con una sorgente (o più), una destinazione e dei tubi di varie forme(a "I", a "L", a "T") disposti su una griglia invisibile; il giocatore, quindi, ruotando i tubi, deve individuare la configurazione giusta per far scorrere l' acqua.



2

Il gioco è composto da livelli: solo dopo aver passato con successo il livello n-1-esimo si può accedere al livello n-esimo.
Dopo aver completato un livello, l'utente viene valutato da 1 a 3 stelle in base al numero di mosse effettuate; chiaramente, meno mosse si impiegano, più stelle si ottengono.

1.2 L'applicazione JPlumber

Il nostro progetto consiste in una versione semplificata di Pipe Line composta da 4 livelli.
L'applicazione JPlumber fornisce un'interfaccia di gioco in cui l'utente può semplicemente cliccare al di sopra di un tubo per farlo ruotare in senso orario di 90°; questa operazione verrà ripetuta più volte (numero limitato di mosse in relazione al livello) dall'utente per riuscire ad ottenere il percorso corretto da sorgente a destinazione.
Per percorso corretto si intende che l'acqua deve scorrere da inizio a fine seguendo una linea continua senza che ci siano interruzioni di cammino e senza che l'acqua possa fuoriuscire.

L'utente può disabilitare l'audio o scegliere il livello desiderato nel menù principale; invece, durante la partita, si possono vedere il numero di mosse rimanenti, resettare il livello oppure tornare al menù principale.

2. Specifica dei requisiti

L’applicazione JPlumber che si intende realizzare dovrà soddisfare i seguenti requisiti:

1. Presenza dell’audio al tocco di pulsanti, sottofondo musicale e motivetto in caso di vittoria o sconfitta. Inoltre, possibilità di disabilitare sia il sottofondo che il suono al tocco dei tasti dal menù principale.
2. Difficoltà crescente man mano che si supera ogni livello.
3. Possibilità di salvare la partita e ricominciare dall’ultimo livello raggiunto.
4. Possibilità di modificare componenti grafiche del gioco.
5. L’utente per ruotare un tubo dovrà semplicemente cliccare su di esso.
6. Possibilità di definire nuove forme di tubi.
7. Garantire la fluidità del gioco, in particolare nella rotazione dei tubi e nello scorrere dell’acqua in quest’ultimi.
8. L’interfaccia sarà comprensibile e lineare, ci saranno direttamente tutti i controlli necessari per l’utente.

9. Assenza di ridondanti finestre di configurazione e segnalazione (vittorie, sconfitte ed esaurimento rotazioni).
10. L’utente, in ogni livello, potrà effettuare solo un numero limitato di rotazioni in relazione alla complessità del livello stesso.
11. La geometria dei tubi sarà curva e non spigolosa.
12. L’utente può ricominciare la partita dal livello raggiunto nel caso in cui si accorgesse di non poter più trovare il percorso per l’ acqua.

Requisiti opzionali:

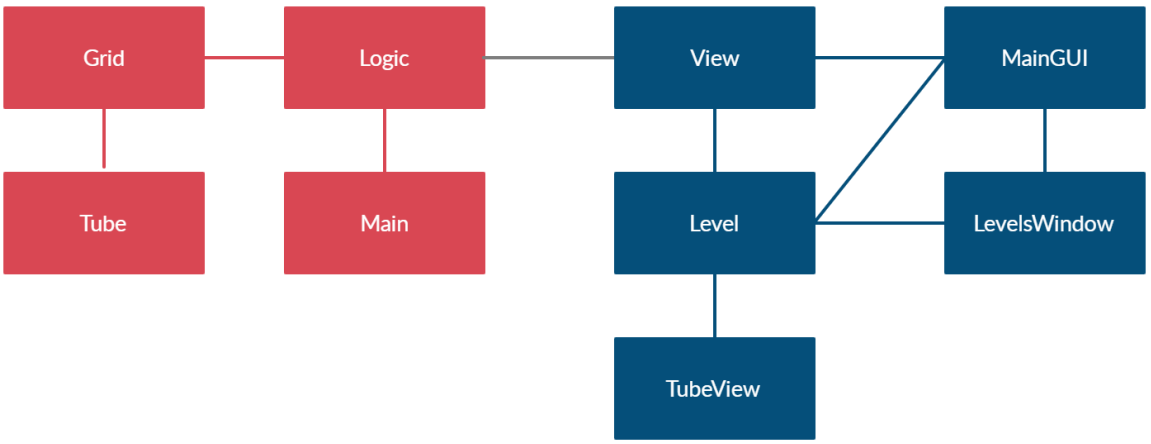
13. L’utente può visionare la soluzione del problema in qualsiasi momento.
14. Valutazione da 1 a 3 stelle del giocatore in base al numero di rotazioni effettuate per la risoluzione del livello.
15. Sequenzialità dei livelli: l’utente non può scegliere direttamente un livello a piacere, prima deve essere sbloccato. Il livello si considera sbloccato quando è stato risolto quello precedente.

3. Progetto

Viene ora descritta la struttura dell'applicazione realizzata, illustrandone prima l'architettura software per poi scendere nel dettaglio dei blocchi funzionali che la compongono.

3.1 Architettura del software

Per la realizzazione di JPlumber si è scelto di basarsi sul pattern di programmazione Logic-View. Il Logic si occupa di gestire i dati dell'applicazione: lo stato dei tubi nella griglia e verificare la validità del percorso. Il View costruisce l' interfaccia grafica e gestisce l'input dell'utente, quest'ultimo avviene mediante il mouse.



Architettura dell'applicazione

La comunicazione fra i package avviene solo attraverso le classi Logic e View in modo da garantire disaccoppiamento tra i due moduli.

3.2. Descrizione dei moduli

3.2.1. Logic

Nel Logic è contenuta la memoria dell'applicazione, cioè la disposizione dei tubi durante tutto il periodo di gioco. Tale modulo serve inoltre per gestire la logica, nel senso che ad ogni rotazione di un tubo si verifica la validità del percorso da sorgente a destinazione. Le classi principali che compongono questo modulo sono:

- **Tube:** Classe astratta che rappresenta un generico tubo, verrà poi estesa dalle classi: TubeI, TubeT, TubeL cioè i tubi effettivamente utilizzati nei livelli e Source, Destination, Empty Cell ad indicare sorgente, destinazione e cella vuota. È costituita da una matrice di interi di 3 righe e 3 colonne, che assume il valore 1 dove è presente il tubo, 0 altrimenti. A ciascun tubo è associato uno stato che indica la sua attuale rotazione.
- **Grid:** Classe che modella logicamente la griglia di gioco. È costituita da una matrice quadrata di oggetti Tube. È estesa dalle classi Grid1, Grid2, Grid3 e Grid4 che conterranno la disposizione dei tubi nel corrispondente livello. Tale classe contiene tutti i metodi necessari a rilevare la presenza di un percorso corretto da sorgente a destinazione, eventualmente comunicandola al View tramite la classe Logic.
- **Main:** Classe necessaria per l'avvio dell'applicazione.
- **Logic:** È la classe principale del modulo Logic, si occupa della comunicazione con il View, variando lo stato della applicazione a seguito di richieste da parte del View.

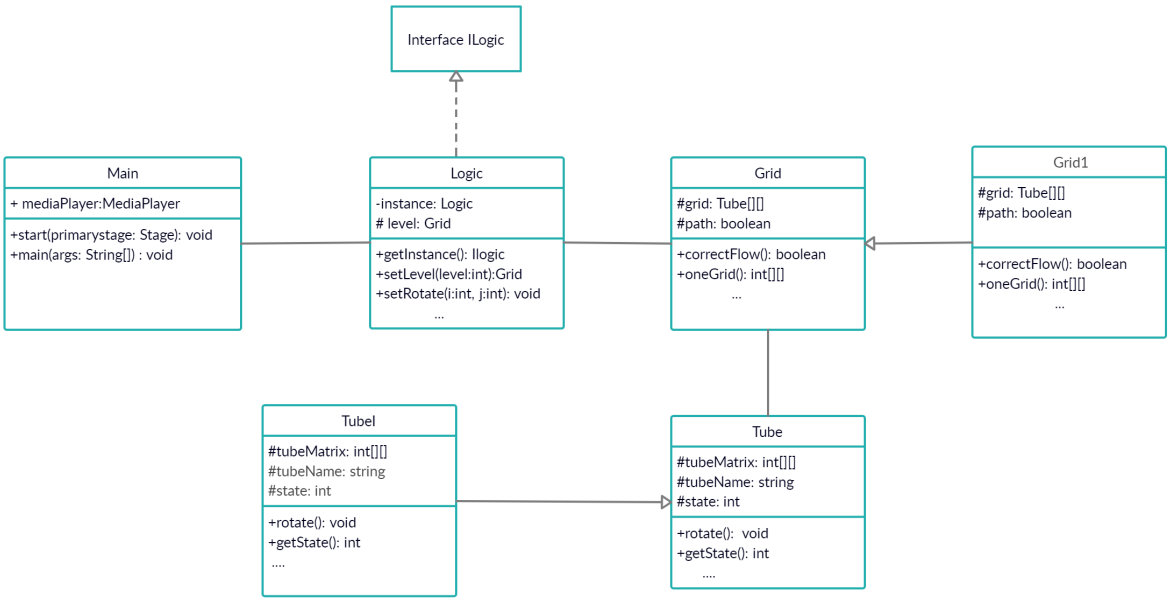


Diagramma UML delle classi del package Logic

Nota: Le rimanenti classi che estendono Tube e Grid sono state omesse.

3.2.2. View

È il modulo che implementa l'interfaccia grafica e si occupa di comunicare al Logic l'input d'utente. La GUI sviluppata è personalizzabile, nel senso che è possibile scegliere tra due stili:

- Dark Theme
- Light Theme

Le classi principali che implementano il View sono:

- **MainGUI:** Classe che rappresenta la schermata iniziale, l'utente può scegliere di iniziare o riprendere la partita salvata, cambiare lo stile e silenziare la musica. Stabilisce le dimensioni della finestra di gioco.
- **LevelsWindow:** Rappresenta il menù in cui scegliere tra 4 possibili livelli, di difficoltà graduale.
- **TubeView:** È la rappresentazione grafica del tubo. Si rappresenta tramite una matrice quadrata 3x3 di oggetti Rectangle. Tramite il metodo PaintTube i rettangoli vengono colorati riproducendo un tubo a I a L o a T nello stato determinato dalla matrice di interi fornita dal Logic.
- **Level:** Classe astratta che rappresenta lo scheletro della schermata di gioco. È composta da una griglia in cui si dispongono i tubi in base al livello selezionato e da una sezione in cui l'utente può:
    - Ricominciare la partita dall'inizio
    - Vedere il numero di mosse rimanenti
    - Tornare alla schermata di selezione dei livelli
    - Abbandonare la partita decidendo se salvare o meno

  Ad ogni tubo della griglia si associa un ascoltatore di eventi per catturare l'input di utente e comunicare al View la rotazione del tubo.
  È inoltre il Level che tramite il View richiede al Logic di verificare la correttezza del percorso sorgente-destinazione tramite il metodo Victory.

Dal Level appaiono le schermate di vittoria e di sconfitta in base all'esito della partita.

- **View:** Si occupa della comunicazione con il Logic, in particolare fa da tramite tra quest'ultimo e il Level per aggiornare la disposizione dei tubi. Un metodo importante è setLevel, chiamato dal costruttore delle classi che estendono Level per settare il livello desiderato.

Nota: Per comodità si è deciso di far gestire il numero rimanente di mosse al View, anche se è un dato riguardante lo stato della applicazione.

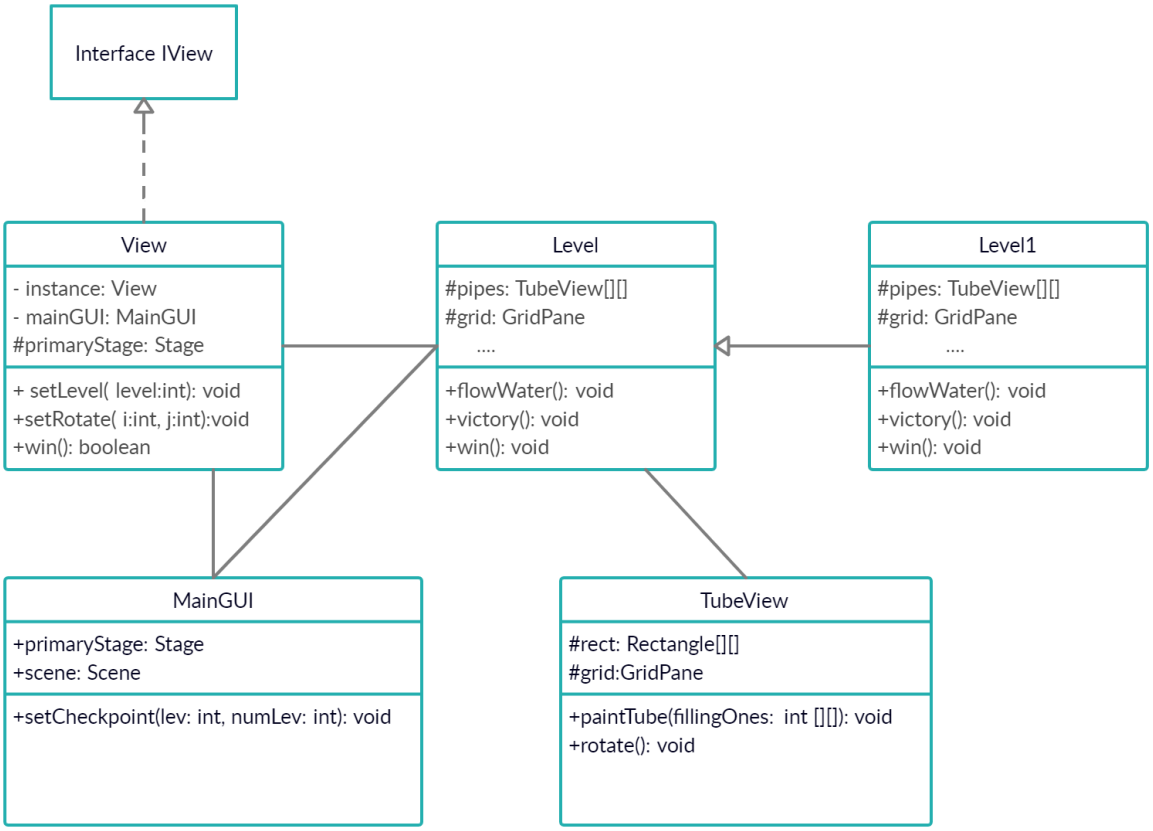


Diagramma UML delle classi del package View

Nota: Le rimanenti classi che estendono Level sono state omesse.

3.3. Problemi Riscontrati

I problemi principali riscontrati sono:

- Implementazione dello scorrimento dell'acqua
- Riconoscimento del percorso corretto da sorgente a destinazione

3.3.1. Implementazione dello scorrimento dell'acqua

Tale problema è stato gestito mediante il metodo waterFlow() richiamato non appena si trova il percorso corretto sorgente-destinazione. Esso è un metodo astratto in Level e viene sovrascritto nelle classi che implementano Level.
La classe principale utilizzata è FillTransition, appartenente al package *javafx.animation.Transition,* che consente di colorare un nodo e poter osservare il processo di colorazione.
L'idea principale è quella di mettere in un array di TubeView tutti i tubi che saranno percorsi dall'acqua; si scandiscono in ordine i tubi e, attraverso il metodo singleTube(), tutti i rettangoli del tubo (che possono essere verdi o neri a seconda dello stile scelto) vengono inseriti ognuno in un oggetto FillTransition, per poter creare un array di FillTransition.
Viene poi scandito tutto l'array di FillTransition con un for, colorando ogni rettangolo prima di bianco poi di celeste per creare l'effetto. La colorazione tra un rettangolo e il successivo è ritardata.
Se il percorso si dirama e si creano più sequenze, allora si utilizzano più array di FillTransition.
Nota: Tubeview è composta da una matrice 3x3 di rettangoli, dove quelli colorati indicano il tubo, la stringa "up" in singleTube indica che la scansione della matrice avviene dall'alto.

3.3.2. Riconoscimento del percorso corretto sorgente-destinazione

Il problema da risolvere è quello di riconoscere se il giocatore ha individuato il giusto percorso da sorgente a destinazione per decretare così il superamento del livello. La gestione di questa problematica è stata affrontata mediante un metodo ricorsivo: waterPath(). Tale metodo è avviato ogni volta che il giocatore, durante la partita, ruota un tubo. A partire dalla sorgente,

rappresentata da un oggetto della classe Source, che estende Tube, si va a controllare se ci siano intorno tubi con cui la sorgente è collegata. Se si riscontra un collegamento tra la sorgente e un tubo, si richiama ricorsivamente il metodo su questo tubo andando a guardare ancora eventuali corrispondenze tra il tubo in questione e quelli intorno ad esso. Le chiamate ricorsive proseguono fin tanto che si trovano tubi collegati. Il caso base è rappresentato da due possibili situazioni:

- Si è raggiunta la destinazione, in tal caso il giocatore ha trovato il percorso corretto e supera il livello.
- Un tubo su cui si sta eseguendo il metodo non trova tubi a cui è collegato, si esce dal metodo che verrà nuovamente eseguito quando il giocatore ruoterà un tubo.

4. Considerazioni finali

JPlumber è un'applicazione semplificata di Pipe Line, presenta varie opzioni per l'utente. Grazie al fatto che i vari livelli nel gioco sono estensioni della classe Level e presentano quindi la stessa struttura di base, è facile ampliare il numero di livelli con poche modifiche al codice già presente, senza grossi sforzi. Un ulteriore spunto di sviluppo può essere l'assegnazione di un numero di stelle, da 1 a 3, all'utente in base al numero di mosse impiegato per superare il livello.

5. Bibliografia

- Programmazione di interfacce grafiche e dispositivi mobili: Unistudium. https://www.unistudium.unipg.it/unistudium/course/view.php?id=18188
- Documentazione JavaFX. https://docs.oracle.com/javafx/2/api/index.html
- Documentazione CSS. https://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html#fontprop