

**Progettazione object oriented
di un'interfaccia grafica JavaFX
per il simulatore Alchemist**

Tesi in Programmazione ad Oggetti

Relatore:
Prof. Mirko Viroli

Correlatore:
Ing. Danilo Pianini

Presentata da:
Niccolò Maltoni

Sommario

Lo scopo di questa tesi verte intorno allo studio del simulatore Alchemist e al fine di progettare un'interfaccia 2D potenziata per l'ambiente grafico relativo alla simulazione. La nuova interfaccia permette di interagire con la simulazione a tempo di esecuzione e di vedere chiaramente rappresentate informazioni su di essa; in particolare, è supportata una struttura modulare di effetti che per rendere ancora più facilmente osservabili determinate entità del sistema ed eventuali loro proprietà. Si è scelto di mantenere un'interfaccia il più possibile *user-friendly*, mantenendo un design più simile ai simulatori a scopo videoludico per favorire l'utilizzo da parte di utenti inesperti.

La seguente trattazione è strutturata su tre capitoli: nel capitolo 1 viene introdotto il contesto nel quale il lavoro descritto nella tesi ha preso parte, introducendo il simulatore Alchemist, la sua interfaccia grafica classica e il framework JavaFX; nel capitolo 2 si espone l'intero contributo fornito al progetto, analizzando singolarmente le fasi di analisi dei requisiti, design e progettazione e in ultimo implementazione della nuova interfaccia; infine, il capitolo 3 analizza i risultati ottenuti, interpretandoli anche in ottica di miglioramenti futuri.

Indice

Sommario	i
1 Introduzione	1
1.1 Alchemist	1
1.1.1 Introduzione ad Alchemist	1
1.1.2 Modello computazionale di Alchemist	2
1.1.3 Interfaccia utente classica	4
Esperienza utente	4
Swing	4
Gli effetti e l'interfaccia Effect	4
1.2 JavaFX	4
1.2.1 Introduzione a JavaFX	4
1.2.2 Il framework JavaFX	4
1.2.3 Struttura di una Applicazione JavaFX	4
1.2.4 Vantaggi di JavaFX su Swing	4
1.3 Interfaccia JavaFX per Alchemist: motivazioni	4
2 Contributo	5
2.1 Analisi dei requisiti	6
2.1.1 Requisiti funzionali	6
2.1.2 Requisiti non funzionali	6
2.2 Fonti d'ispirazione	6
2.2.1 Simulatori a scopo videoludico	6
Universe Sandbox	6
Universe Sandbox 2	6
SimCity	6

<i>INDICE</i>	iii
2.2.2 Material Design	6
2.3 Design dell'interfaccia	6
2.4 Progettazione	6
2.4.1 La barra inferiore	6
2.4.2 La struttura a drawer	6
2.4.3 L'architettura degli effetti	6
2.5 Dettagli implementativi	6
3 Conclusioni	7
3.1 Risultati	7
3.2 Lavori futuri	7
Bibliografia	8
Ringraziamenti	9

Capitolo 1

Introduzione

1.1 Alchemist

Alchemist [1, 7] è un meta-simulatore estendibile completamente *open-source* che esegue su Java Virtual Machine (JVM), nato all'interno dell'Università di Bologna e distribuito su licenza GNU GPLv3+ con *linking exception*; il codice è reperibile su GitHub¹, dove chiunque fosse interessato può collaborare sviluppando nuove estensioni, migliorando funzionalità esistenti e risolvendo possibili bug.

1.1.1 Introduzione ad Alchemist

In generale, una *simulazione* [3] è una riproduzione del modo di operare di un sistema o un processo del mondo reale nel tempo. L'imitazione del processo del mondo reale è detta *modello*; esso risulta essere una riproduzione più o meno semplificata del mondo reale, che viene aggiornata ad ogni passo di esecuzione della simulazione.

Alchemist rientra nell'archetipo dei simulatori ad eventi discreti (DES) [2, 4]: gli eventi sono strettamente ordinati e vengono eseguiti uno alla volta, mentre il tempo viene fatto avanzare parallelamente ad ogni passo (detto *tick*). L'idea dietro al progetto è quello di riuscire ad avere un framework di simulazione il più possibile generico, in grado di simulare sistemi di tipologia e complessità diverse, mantenendo le prestazioni dei simulatori non generici (come ad esempio quelli impiegati in ambito chimico [5]).

¹<https://github.com/AlchemistSimulator/Alchemist>

Per perseguire questo obiettivo, la progettazione dell'algoritmo è partita dal lavoro di Gillespie del 1977 [6], al quale sono state aggiunte diverse modifiche per adattarlo al rinnovato metamodello.

1.1.2 Modello computazionale di Alchemist

Il modello (visibile in Figura 1.1.2 nella pagina seguente) che costituisce l'architettura base di Alchemist è, come detto, ispirato ad algoritmi tipici della simulazione a scopo di ricerca chimica e, dunque, ne riprende la nomenclatura, seppur con alcune libertà atte ad ottenere una maggiore flessibilità. Le entità su cui lavora sono le seguenti:

Molecule Una *Molecola* rappresenta il nome dato ad un particolare dato all'interno di un *Nodo*, del quale ne astrae parte dello stato.

Un parallelismo con la programmazione imperativa vedrebbe la *Molecola* come un'astrazione del nome di una variabile.

Concentration La *Concentrazione* di una *Molecola* è il valore associato alla proprietà rappresentata dalla *Molecola*.

Mantenendo il parallelismo con la programmazione imperativa, la *Concentrazione* rappresenterebbe il valore della variabile.

Node Il *Nodo* è un contenitore di *Molecole* e *Reazioni* che risiede all'interno di un *Ambiente* e che astrae una singola entità.

Environment L'*Ambiente* è l'astrazione che rappresenta lo spazio nella simulazione ed è l'entità che contiene i nodi.

Esso è in grado di fornire informazioni in merito alla posizione dei *Nodi* nello spazio, alla distanza tra loro e al loro vicinato; opzionalmente, l'*Ambiente* può offrire il supporto allo spostamento dei *Nodi*.

Linking rule La *Regola di collegamento* è la funzione dello stato corrente dell'*Ambiente* che associa ad ogni *Nodo* un *Vicinato*.

Vicinato Un *Vicinato* è un'entità costituita da un *Nodo* detto "centro" e da un insieme di altri *Nodi* (i "vicini").

L'astrazione dovrebbe avere un'accezione il più possibile generale e flessibile, in modo da poter modellare qualsiasi tipo di legame di vicinato, non solo spaziale.

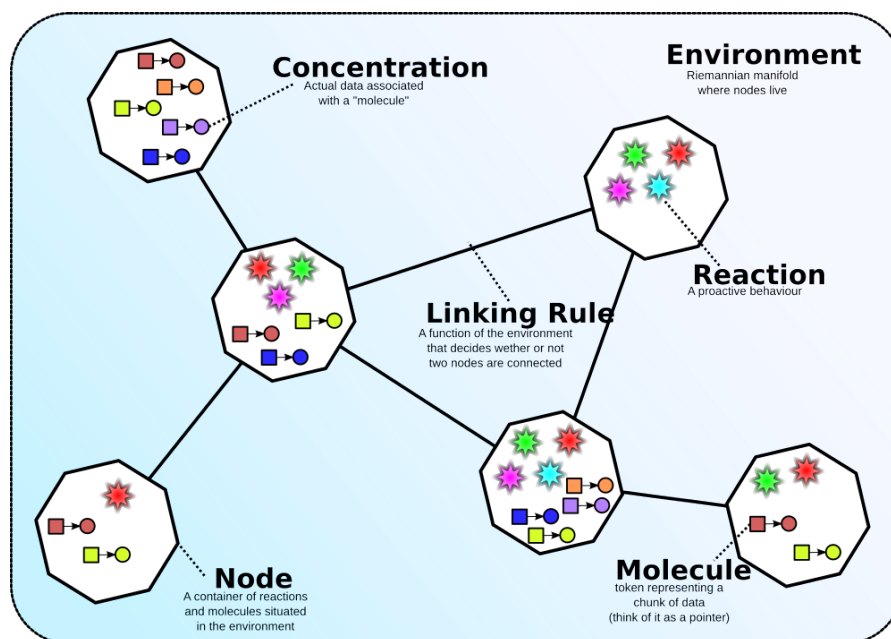


Figura 1.1: La figura, presa dal sito ufficiale [1], offre una rappresentazione grafica delle diverse entità. All'interno di un ambiente, che modella il sistema, si trovano i nodi connessi tra loro attraverso dei collegamenti; ogni nodo è composto da reazioni e molecole, ognuna delle quali ha associata una concentrazione.

Reaction Il concetto di *Reazione* è da considerarsi molto più elaborato di quello utilizzato in chimica: in questo caso, si può considerare come un insieme di *Condizioni* sullo stato del sistema, che qualora dovessero risultare vere innescerebbero l'esecuzione di un insieme di *Azioni*.

Una *Reazione* è dunque un qualsiasi evento che può cambiare lo stato dell'*Ambiente* e si compone di un insieme di condizioni, una o più azioni e una distribuzione temporale.

La frequenza di accadimento può dipendere da:

- Un tasso statico;
- Il valore di ciascuna *Condizione*;
- Una equazione che combina il tasso statico e il valore delle *Condizioni*, restituendo un "tasso istantaneo";
- Una distribuzione temporale.

Ogni *Nodo* è costituito da un insieme (anche vuoto) di *Reazioni*.

Condition Una *Condizione* è una funzione che associa un valore numerico e un valore booleano allo stato corrente di un *Ambiente*.

Action Un'*Azione* è una procedura che provoca una modifica allo stato dell'*Ambiente*.

1.1.3 Interfaccia utente classica

Esperienza utente

Swing

Gli effetti e l'interfaccia **Effect**

1.2 JavaFX

1.2.1 Introduzione a JavaFX

1.2.2 Il framework JavaFX

1.2.3 Struttura di una Applicazione JavaFX

1.2.4 Vantaggi di JavaFX su Swing

1.3 Interfaccia JavaFX per Alchemist: motivazioni

Capitolo 2

Contributo

2.1 Analisi dei requisiti

2.1.1 Requisiti funzionali

2.1.2 Requisiti non funzionali

2.2 Fonti d'ispirazione

2.2.1 Simulatori a scopo videoludico

Universe Sandbox

Universe Sandbox 2

SimCity

2.2.2 Material Design

2.3 Design dell'interfaccia

2.4 Progettazione

2.4.1 La barra inferiore

2.4.2 La struttura a drawer

2.4.3 L'architettura degli effetti

2.5 Dettagli implementativi

Capitolo 3

Conclusioni

3.1 Risultati

3.2 Lavori futuri

Bibliografia

- [1] *Alchemist*. URL: <http://alchemistsimulator.github.io/>.
- [2] E. Babulak e M. Wang. «Discrete event simulation: State of the art». In: *International Journal of Online Engineering (iJOE)* 4.2 (2007), pp. 60–63.
- [3] J. Banks et al. *Discrete-Event System Simulation: Pearson New International Edition*. Pearson Higher Ed, 2013.
- [4] G. S. Fishman. «Principles of discrete event simulation.[book review]». In: (1978).
- [5] D. T. Gillespie. «A general method for numerically simulating the stochastic time evolution of coupled chemical reactions». In: *Journal of computational physics* 22.4 (1976), pp. 403–434.
- [6] D. T. Gillespie. «Exact stochastic simulation of coupled chemical reactions». In: *The journal of physical chemistry* 81.25 (1977), pp. 2340–2361.
- [7] D. Pianini, S. Montagna e M. Viroli. «Chemical-oriented simulation of computational systems with ALCHEMIST». In: *Journal of Simulation* 7.3 (ago. 2013), pp. 202–215. ISSN: 1747-7786. DOI: 10.1057/jos.2012.27. URL: <https://doi.org/10.1057/jos.2012.27>.

Ringraziamenti