Trascrizione due

presentazione

Buongiorno alla commissione, mi chiamo Alessandro Mezzogori, sono uno studente del corso di laurea in ingegneria informatica. L'elaborato che presento si concentra sulla proposta di un linguaggio di programmazione per la prototipizzazione dei giochi da tavolo.

GDL

al momento uno dei pochi modi per descrivere un gioco programmaticamente tramite un linguaggio specializzato si tratta é Gaem description language creato da Micheal Genesereth per il general gameplaying prioject. grazie alla sua specializzazione, dovuta alla concenzione di un qualsiasi gioco come una macchina a stati finiti e alla sua semplicitá sintattica, la rende ottima per allenare un agente artificiale per estrapolare le relazioni descritte. Purtroppo consegue difficile da utilizzare in giochi non ben definiti, ovvero nella fase di prototipizzazione e design.

Questo porta ad avere una mancanza di un linguaggio per tale scopo, TTPL é la mia proposta per tappare il buco, ha l'obbiettivi di fornire un unica interfaccia per la prototipizzazione dei giochi da tavolo, la cui deve avvenire in maniera flessibile, facilmente utilizzabile deve essere a supporto dello sviluppatore o designer per non rendere la codifica del gioco un impresa piú ardua. basarsi su un concetto di azione e reazione facilmente concepibile da un essere umano.

questo tre obbiettivi sono realizzati tramite tramite tre principali scelte di design del linguaggio, fare uso del paradigma ad oggetti e avere un archittettura ad eventi, input injection. il paradigma ad oggetti negli ultimi anni si é rilevato efficace per modellare le relazioni i comportamenti, in generale la struttura di giochi e videogiochi, a testimonianza i due game engine piú utilizzati al mondo, Unity e Unreal Engine, rispettivamente utilizzano csharp come linguaggio di scripting e c++ come linguaggio di programmazione entrambi nel paradigma ad oggetti.

L'architettura ad eventi rende possibile disaccoppiare il codice generante gli eventi dal quello consumante, gli eventi sono una struttura facilmente mappabile tra il mondo dei giochi e il mondo del codice poiché una qualsiasi azione, comando, cambio di turno ecc.. possono essere classificati come degli eventi
nel lignuaggio le azioni di una classe sono gli handler dell'evento, un azione puó essere associata a piú di un evento e ogni evento puó avere zero o piú handlers, da un singolo punto di partenza si creano N punti di esecuzione.

gli handler degli eventi implementato nel linguaggio si chiamano azioni, trovate all'interno delle classi si compongono di quattro componenti principali, divisi in due sotto gruppi principali non rigidi: componenti in pre-azione: sono delle funzioni che sono valutate prima che l'azione abbia eseguito l'effetto e servono per interrompere il flusso dell'azione se non rispetta certe condizioni.

requirements

i requisiti sono una funzione con tipo di ritorno booleano, il quale indica se l'azione é attiva oppure no

triggers

se l'azione é attiva sará valutato il trigger specifico all'evento che si sta gestendo il triggers anche'esse é una funzione booleano che indica se si deve continuare il flsuso dell'azione oppure interromperlo poiché non é un evento mirato a questo tipo di azione se tutti i check pre azioni passano si continua con i componenti intra azione

componente intra-azione

input

sono l'insieme di block, tag, filtri che permettono di recuperare i valori utilizzati all'intenro dell'effetto

effetto

che si occupa di eseguire le istruzioni legate all'azione in se, funzione senza tipo di ritorno.

Input injection

la terza scelta principale di design del linguaggio é di usare un sistema di iniezione dell'input diverso dal passaggio di parametri a funzioni, unendolo con il concetto di dependency injection e inversion of control tramite un container globale che tiene traccia di tutti gli oggetti riferimento creati

é il sistema che popola le variabili create dagli input con i valori richiesti

principalmente scelto poiche ripulisce il codice da molti passaggi di argomenti alle azioni / funzioni e inoltre velocizza il processo di sviluppo dato che non si deve passare attraverso altri oggetti o flussi di lavoro per recuperare gli input.

per concludere le tre scelte, unite alle parti secondarie fornite dal linguaggio come le funzioni e le strutture dati native trovo che fornisca un set di strumenti sufficiente per la prototipizzazione di un gioco, abilitando lo sviluppatore o designed nella progettazione digitale.