RobotSwarm

Una libreria Java che consente la simulazione di uno *sciame di robot* che si muovono nello spazio.

Descrizione

Il progetto realizza una libreria per simulare uno sciame di robot, che si muovono in uno spazio che può contenere degli ostacoli, ovvero delle figure circolari o rettangolari. L'ambiente di movimento ha dimensioni illimitate.

Nel progetto è inoltre presente una GUI che permette di osservare l'evoluzione della simulazione. L'utente può inserire file di configurazione sia per l'ambiente che per i comandi dei robot.

Esempi di file di configurazione si trovano tra le risorse dell'applicazione, sotto la cartella ***resources*** dell'applicazione (app/src/main/resources/it/unicam/cs/pa/app/files/example_*.txt).

Responsabilità

Ambiente

1. **Creazione di forme geometriche**: la creazione di forme geometriche è assegnata all'interfaccia ***ShapeFactory***, che crea delle ***Shape*** come ***Circle*** e ***Rectangle*** utilizzando gli argomenti passati all'interno di un array di elementi di tipo double;
2. **Gestione dell'ambiente**: la gestione dell'ambiente è affidata alla classe ***Environment***, che rappresenta l'ambiente in cui operano i robot, mantenendo traccia delle forme geometriche e dei robot presenti;
3. **Parsing del file di configurazione dell'ambiente**: il parsing del file di configurazione dell'ambiente è affidato alla classe ***EnvironmentParser***, che si occupa di leggere i dati in ingresso dal file di configurazione e creare le figure. Tutte le figure create vengo inserite in una lista che potrà poi essere richiamata per inserire le figure nell'ambiente. La classe utilizza, inoltre, un'interfaccia funzionale (***ShapeChecker***) per controllare i parametri inseriti per la creazione delle figure.

Ogni posizione di ogni elemento dell'ambiente (robot o shape) è gestita tramite la classe ***Position*** che individua, tramite delle coordinate **x** e **y**, un elemento all'interno dell'ambiente. La classe, inoltre, mette a disposizione metodi per la creazione di posizioni randomiche all'interno dell'ambiente, per la validazione dei parametri inseriti, per il calcolo della distanza di una posizione rispetto ad un'altra e per il calcolo della posizione media rispetto ad una lista di posizioni.

Robot e comandi

1. **Gestione dei robot**: l'interfaccia **Robot** rappresenta un robot nell'ambiente e fornisce funzionalità per gestire il suo movimento e il suo comportamento durante la simulazione. Le interfacce ***RobotLoopManagement*** e ***RobotMovementManagement*** definiscono le responsabilità relative alla gestione dei loop e dei movimenti dei robot, rispettivamente. La classe ***BasicRobot*** fornisce un'implementazione semplice di robot, gestendo la posizione, la

direzione e la velocità di movimento del robot nell'ambiente. Inoltre gestisce anche il movimento del robot nell'ambiente, calcolando la posizione finale del robot, e la funzione del robot di segnalare la propria condizione.
2. **Parsing del programma dei robot**: il parsing del file di configurazione del programma dei robot è assegnato alla classe ***CommandParser***, che si occupa di leggere i dati in ingresso dal file di configurazione e di creare le classi di comandi disponibili in ***RobotCommands***. L'insieme dei comandi creati viene inserito in una lista di comandi.
3. **Gestione dei comandi**: le interfacce ***Command***, ***LoopCommand*** e ***MovementCommand*** definiscono le responsabilità relative all'esecuzione dei comandi dei robot. In particolare l'interfaccia *LoopCommand* definisce e viene implementata da tutti i comandi che defiscono un loop nel programma (***Do forever***, ***Repeat***, ***Until***, ***Continue***, ***Done***). Mentre l'interfaccia *MovementCommand* definisce e viene implementata da tutti i comandi che implicano il movimento di un robot (***Move***, ***Move random***, ***Follow***, ***Stop***). I comandi ***Signal*** e ***Unsignal*** sono considerati dei comandi di movimento del robot. La gestione dei comandi nella simulazione è affidata ad ogni singola istanza di *Robot*, che registra gli indici dei comandi da eseguire e memorizza un contatore del numero di ripetizioni di eventuali loop. Tramite l'utilizzo della classe ***Deque***, la simulazione può gestire dei loop annidati.

Simulazione

1. **Gestione della simulazione**: la gestione della simulazione dello sciame è affidata alla classe ***SimulatorController*** che gestisce la logica della simulazione, coordinando l'interazione tra l'ambiente, i robot e i comandi eseguiti. Permette l'esecuzione di un passo della simulazione, eseguendo i comandi per ogni robot nell'ambiente. Durante ogni passo, per ogni robot viene selezionato il comando giusto in base all'indice del comando registrato dal robot in `currentCommandIndex`. Ad ogni passo viene notificata l'interfaccia di simulazione per aggiornare la visualizzazione.

L'interfaccia ***Simulator***, implementata dal controller della GUI, permette l'esecuzione della simulazione automatica impostando un tempo totale di esecuzione e un tempo di esecuzione di ogni passo.

GUI

La GUI, realizzata con JavaFX 21, permette l'inserimento dei file di configurazione per ambiente e per programma dei robot, visualizzando a schermo i dettagli inseriti di entrambi. Inoltre, permette di generare dei robot nell'ambiente posizionati randomicamente. Una volta impostato il tutto, può partire la simulazione: è possibile eseguire una simulazione automatica, inserendo i tempi di simulazione desiderati o una simulazione passo-passo tramite i rispettivi pulsanti. E' possibile, inoltre, muovere o zoomare la visuale dell'ambiente tramite gli appositi pulsanti.

Avvio simulazione e utilizzo esempi

Per avviare la simulazione è sufficiente eseguire da terminale i comandi:

- `gradle build`
- `gradle run`

Successivamente, premendo i pulsanti ***Load shape*** e ***Load program*** è possibile caricare i file di configurazione. Degli esempi di utilizzo si possono trovare nella la cartella ***resources*** dell'applicazione (app/src/main/resources/it/unicam/cs/pa/app/files/example_*.txt).