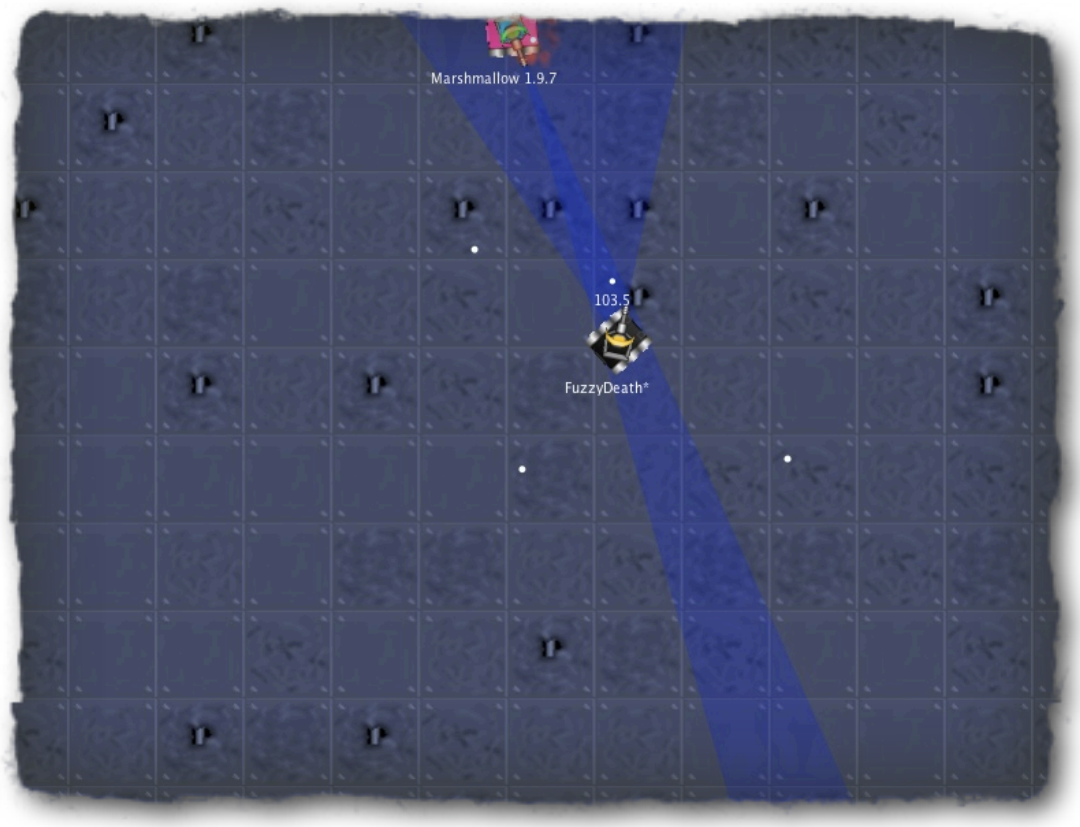
Progetto - Sistemi Intelligenti 2013

Implementazione di un sistema fuzzy per Robocode

FuzzyDeath



Prof. Alberto Borghese – A. A. 2012/2013

Francesco Giorgio – 826606
Andrea Toscano – 826177

Panoramica

Obiettivo del nostro progetto è stato quello di riprodurre e implementare un sistema fuzzy che fosse di supporto a un bot Robocode. Suddetto sistema sfrutta le informazioni sulla velocità del nemico e sulla distanza tra i due robot per decidere con che potenza sparare ed è così composto:

Variabili Crisp d'ingresso

Velocità del bot nemico

- **Range di valori: [-8, 8] px**

Distanza tra i due robot

- **Range di valori: [0, eucl(h, w)] px**
  Dove:
  - eucl(x, y) è una funzione che calcola la distanza euclidea bi-dimensionale dei punti di coordinate (0, h) e (w, 0).
  - h è l'altezza del campo di battaglia.
  - w è la larghezza del campo di battaglia.

  Praticamente eucl(h, w) rappresenta la lunghezza della diagonale del campo di battaglia, ovvero la distanza massima a cui possono trovarsi due bot.

Variabile Crisp d'uscita

Potenza di fuoco

- **Range di valori: [0.1 , 3.0]**

Classi delle variabili

Velocità del bot nemico – 4 classi

- **Bassa**
- **Medio Bassa**
- **Media**
- **Alta**

Distanza tra i due robot – 4 classi

- **Bassa**
- **Medio Bassa**
- **Media**
- **Alta**

Potenza di fuoco – 4 classi

- **Bassa**
- **Medio Bassa**
- **Media**
- **Alta**

FAM

Distanza	**Velocità nemico**			
	Bassa	*Medio Bassa*	*Media*	*Alta*
Bassa	Alta	Media	Media	Alta
Medio Bassa	Alta	Media	Media	Media
Media	Media	Media	Medio Bassa	Medio Bassa
Alta	Medio Bassa	Medio Bassa	Bassa	Bassa

Implementazione del sistema fuzzy

Il sistema è stato progettato per essere altamente scalabile, vista la possibilità d'introdurre dinamicamente nuove variabili in input o in output, oltre che nuove regole per la FAM. Questo permetterebbe anche di modificare le attuali componenti del sistema attraverso un eventuale modulo di Reinforcement Learning.

Al fine di ottenere tale risultato, sono stati adottati i princìpi della programmazione ad oggetti, di astrazione e di polimorfismo per inclusione.

Il sistema viene attivato automaticamente all'inizio della battaglia e ogni qualvolta viene invocato l'evento *OnScannedRobot,* viene effettuata una computazione. Non vi è stato bisogno di fornire una rappresentazione per le variabili crisp d'input e di output, in quanto vengono già rappresentate dalle API di Robocode.

Linguaggio di programmazione adottato: Java.

Ambiente di sviluppo: Eclipse.

Variabili

FuzzyVariable

La classe FuzzyVariable è una classe astratta usata per rappresentare una variabile fuzzy. È identificata da un nome e da un insieme di classi che rappresentano i Fuzzy Set.

InputVariable

Estende FuzzyVariable e rappresenta una variabile di input dopo la fuzzyficazione. Possiede un metodo che, dato il valore crisp rilevato, calcola il grado di fit per ogni classe.

OutputVariable

Estende FuzzyVariable e rappresenta una variabile di output prima della defuzzyficazione. Possiede un metodo che, data la somma di tutti i gradi di fit attivati per la variabile, restituisce il valore crisp.

Classi (Fuzzy Set)

Class
È una classe astratta che rappresenta un Fuzzy Set. Ogni Fuzzy Set è caratterizzato da 4 punti:

- "a" e "d" rappresentano i valori all'esterno dei quali la variabile assumerebbe valore di fit pari a zero.
- "b" e "c" sono i valori all'interno dei quali la variabile assumerebbe valore di fit pari a uno.

In questo modo, assegnando i giusti valori a tali parametri, le aree dei Fuzzy Set possono essere rappresentate da trapezi o da triangoli, a piacimento. In questo progetto i Fuzzy Set delle variabili d'input sono rappresentati da trapezi, quelli delle variabili d'output da triangoli.

InputClass
Estende Class e rappresenta uno dei Fuzzy Set di cui è composta una variabile fuzzy d'input. Possiede un metodo che, dato il valore crisp rilevato, calcola il grado di fit per per se stesso.

OutputClass
Estende Class e rappresenta uno dei Fuzzy Set di cui è composta una variabile fuzzy d'output. Quando viene inizializzata una variabile di questo tipo, viene calcolato automaticamente il valore del centroide. Possiede un campo per memorizzare tutti i gradi di fit registrati ogni qualvolta una regola della FAM attiva la suddetta classe, oltre a un metodo che restituisce la somma pesata delle varie fitness con il centroide.

Rule
Rappresenta una singola regola della FAM. Possiede due campi per memorizzare le condizioni e le classi da attivare, oltre a un metodo che, dato l'insieme della variabili fuzzy (sia d'input che d'output) attualmente in uso, calcola con che grado di fitness le classi di output sono attivate (un numero compreso tra 0.0 e 1.0) e che memorizza tale valore.

Fuzzy System
È il fulcro del sistema, crea le variabili fuzzy, le regole e interagisce con le API Robocode per ottenere i rilevamenti, attivare le regole e indicare al bot con quanta potenza sparare.

Altri comportamenti

Anti-gravity movement

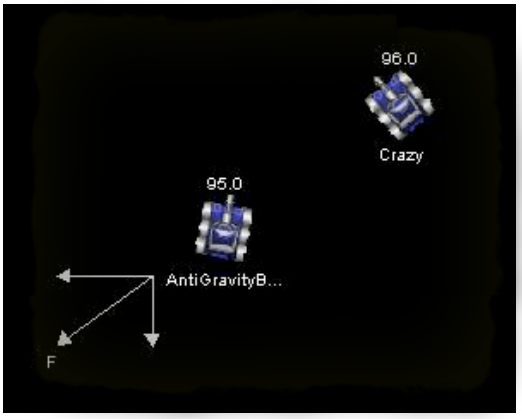
Il movimento del bot è regolat"/>o da alcune funzioni che implementano un sistema di attrazione/ repulsione verso un determinato punto. Il tutto si concretizza in un sistema di forze positive o negative che agiscono sul movimento su ognuno degli assi. (1)

Diamo alcune definizioni.

Punti di gravità: sono i punti nel campo di Robocode che il programmatore definisce come aree per le quali il robot dev'essere respinto o attratto

Forze: ogni punto ha associato un valore di forza. Più il valore è grande, più un robot tende a stare distante/vicino da esso.

Componenti delle forze: Ogni forza ha una componente che agisce in direzione x (orizzontale) ed una in direzione y (verticale). Un angolo di 45° ha componenti uguali in direzioni x e y. Un angolo di 90° agisce interamente nella direzione x e un angolo di 0° agisce interamente su y.

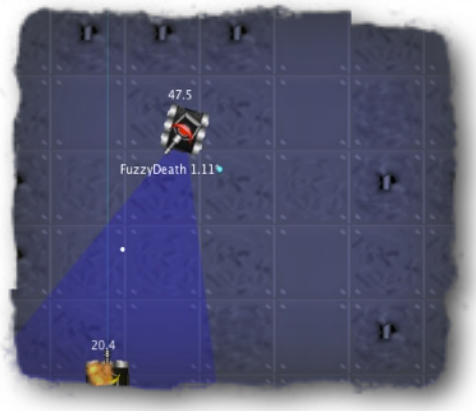


Risoluzione delle forze: questo è il processo di elaborazione della forza totale prodotta quando più forze agiscono fra di loro. Ad esempio, se vi è una forza di -200 che agisce in direzione x ed una di 300 nella medesima direzione, la forza complessiva prodotta è 100 in direzione x.

Nel nostro caso abbiamo definito la posizione corrente del bot nemico e le pareti come punti di gravità dai quali stare a distanza. Evitando un contatto ristretto col nemico si ha più possibilità di evitare i suoi proiettili, mentre stando lontani dalle pareti si evita di scontrarsi con esse e di conseguenza perdere energia.

Lock-on tracker

Dato che il bot combatterà in battaglie 1 vs. 1, si è deciso di fare in modo che lo scanner resti fisso sul primo nemico disponibile in modo da tenerlo sotto controllo e poter prevedere i suoi movimenti.



"/>1. **Owens, Alisdair.** Secrets from the Robocode masters: Anti-gravity movement. *ibm.com.* [Online] May 2002. http://www.ibm.com/developerworks/library/j-antigrav/.