Relazione del progetto di **Ingegneria della Conoscenza** - Estensione di *Dimensions Break*

Gruppo di lavoro:

- *Corvaglia Carlo,* 718859, c.corvaglia7@studenti.uniba.it
- *Gallo Angela,* 715989, a.gallo51@studenti.uniba.it

Repository:
https://github.com/h-angel22/ICON_Project/tree/main

Video dimostrativo delle funzionalità:
https://youtu.be/dxzGpVI0eQo

INDICE

INTRODUZIONE

Nella seguente documentazione verrà descritta la relazione tecnica del progetto
di Ingegneria della Conoscenza Anno Accademico 2022-2023.
Il progetto svolto consiste nella espansione di un videogioco realizzato da noi in precedenza.
L'espansione consiste in 3 fondamentali elementi:

- Generatore di livelli (inteso come risolutore di CSP)
- Un oggetto presente in ogni livello generato che indica la strada al giocatore (implementando un algoritmo di pathfinding)
- Un comportamento più intelligente del boss di fine livello (mediante albero di decisione appreso da un dataset fornito da noi)

L'idea è nata da un interessamento generale da parte del gruppo nel migliorare il videogioco.
Per questo progetto si è utilizzato il linguaggio Python (versione 3.10), Godot e VisualStudio Code come IDE.

Tutti i file sorgente realizzati per il progetto si trovano nella cartella *project/ICON_src/.* Al suo interno si trova anche la cartella python in cui si trovano gli script che implementano le funzioni descritte sopra.

PASSI PROPEDEUTICI

Siamo partiti definendo formalmente la struttura di un livello come una tripla comprendente il grafo delle stanze, la stanza iniziale e la stanza del boss (queste ultime appartenenti all'insieme delle stanze).

$G = (R, D)$
$L = (G, s \in R, b \in R)$

Questo ci torna utile per poter implementare un pathfinding e per definire i vincoli per il CSP.
Abbiamo poi integrato nella classe che gestisce il comportamento del boss una serie di parametri che è in grado di leggere per percepire lo stato della stanza e quindi del combattimento, parametri anche usati negli esempi dell'apprendimento automatico.

CONSTRAINT SATISFACTION PROBLEM

Sommario

In intelligenza artificiale, un CSP (Constraint Satisfaction Problem) è un problema in cui si deve trovare un'assegnazione di valori a un insieme di variabili in modo che un insieme di vincoli sia

soddisfatto. I vincoli sono relazioni tra due o più variabili che definiscono le combinazioni di valori consentite per le variabili.

Lo scopo del CSP nel nostro caso è la generazione di un livello.

Come detto in precedenza, il livello è strutturato come un grafo, per cui il nostro CSP utilizza come variabili gli archi che collegano il grafo. Il dominio di una variabile è composto da tutti i possibili collegamenti che si possono avere tra le stanze selezionate (scelte casualmente da una lista, a cui si aggiungono tre stanze speciali, quali, quella iniziale, quella del boss, e quella del tesoro).

I vincoli sono stati definiti in modo da ottenere un livello funzionante in tutti i suoi aspetti.

Strumenti utilizzati

Abbiamo utilizzato per la risoluzione del CSP i moduli messi a disposizione da “AIPython: Python Code for AIFCA”. Il risolutore di CSP viene richiamato dal gioco in Godot che fa partire il processo python (chiamando il file mainCSP.py) e ne prendi i risultati dallo standard output.

```
func _generate_level():
	var level_str = [""]
	while level_str[0] == "":
		var exit_code = OS.execute("python3", [py_path], true, level_str)
		print(level_str)
	var l = Level.new()
	l.initialize(level_str[0])
	ResourceSaver.save("user://lvl.tres", l)
	levels.append("user://lvl.tres")
	emit_signal("loaded")
	call_deferred("_load_level",loadlvl)
```

Decisioni di Progetto

L’output è fornito come stringa che associa alla path del file della stanza i collegamenti alle stanze adiacenti, da cui è possibile formarne un grafo.

La base di conoscenza è fornita dal file rooms.txt con l’elenco delle possibili stanze da scegliere associate a 4 numeri. Questi 4 numeri rappresentano le possibili direzioni da cui ci si può muovere in una stanza (sinistra, destra, su e giù). In questo file i numeri assumo solo i valori di -1 e -2. Una direzione segnata da -2 indica una direzione non ammessa da quella stanza. Il -1 indica una direzione

```
CheckpointRoom.tscn -1 -1 -1 -1
CompassRoom.tscn -1 -1 -1 -1
NewBossRoom.tscn -2 -2 -1 -1
NewRoom1.tscn -1 -2 -2 -1
NewRoom10.tscn -1 -1 -2 -1
NewRoom2.tscn -1 -1 -1 -1
NewRoom3.tscn -2 -1 -1 -1
NewRoom4.tscn -1 -1 -2 -1
```

consentita ma non assegnata. L'output viene emesso con la stessa sintassi.

Abbiamo definito l'arco come una quadrupla di valori di cui 2 devono rimanere sempre null. Gli altri 2 valori assegnati indicano le due stanze che vengono collegate dall'arco. Quindi, se l'arco assume valore del tipo (x, y, null, null) abbiamo un collegamento che dalla porta di destra della stanza x si arriva alla stanza y dalla porta di sinistra. Nel caso (null, null, x, y) la porta di basso della stanza x conduce alla porta in alto della stanza y. IL risolutore di CSP lavora sull'assegnamento di queste quadruple agli archi.

```
CheckpointRoom.tscn -1 -1 1 -1
CompassRoom.tscn 5 6 4 0
NewBossRoom.tscn -2 -2 -1 5
NewRoom7.tscn -1 4 -1 -1
NewRoom2.tscn 3 -1 5 1
NewRoom3.tscn -2 1 2 4
NewRoom4.tscn 1 -1 -2 -1
```

L'algoritmo risolutivo scelto da noi per la risoluzione del CSP è lo Stochastic Local Search (SLS). Il motivo principale risiede nelle sue performance nel nostro caso specifico.

Se il risolutore non dovesse trovare una soluzione entro 200000 passaggi ricomincia dall'inizio ma con un insieme di stanze da collegare differente. Questo finché non trova una configurazione accettabile. Il numero di stanze di cui si compone il livello è 7 ed è stato scelto in quanto maggior numero di stanze che mantenesse un esecuzione in tempi near real time.

Valutazione

Le performance dell'algoritmo SLS sono state valutate secondo le metriche del tempo di esecuzione e della diversificazione dell'output. Entrambe queste metriche vengono beneficiate dalla tecnica di backtraking utilizzata nell'algoritmo. Il vantaggio sta infatti nel partire da un assegnazione totale per poi modificare con passaggi successivi della ricerca locale, cosa che permette di trovare una soluzione con un numero di controlli ridotto rispetto ad altri algoritmi testati. Altro beneficio dato dalla ricerca locale è che ogni livello ha una disposizione di stanze completamente diverse tra loro. Con altri algoritmi, come per esempio il DFS la tendenza era quella di trovare una soluzione che sviluppava il livello da sinistra verso destra e dall'alto verso il basso.

PATHFINDING

Sommario

Il pathfinding è il processo di trovare un percorso tra due nodi in un grafo. Il percorso può essere definito come una sequenza di nodi che collega il nodo di partenza al nodo di destinazione.

Nel nostro progetto, abbiamo assegnato alle varie stanze un costo, che dipende dallo stato di completamento: se una stanza è già stata attraversata e non ha nemici al suo interno il costo sarà minore rispetto a quello di una stanza ancora da affrontare. Il livello era già rappresentato come un grafo collegato per valori.

Strumenti utilizzati

Abbiamo utilizzato per la ricerca del percorso i moduli messi a disposizione da “AIPython: Python Code for AIFCA”. Sempre attraverso la chiamata di sistema di Godot abbiamo avviato il file mainSearch.py a cui viene passato anche l’array degli archi e dei rispettivi costi. Lo script python risolve il pathfinding attraverso l’algoritmo A* e passa attraverso lo standard output il risultato.

```
func _path_finder() -> Array:
    var path: Array
    var path_str = []
    OS.execute("python3", [py_path, _archs_str(), current_room], true, path_str)
    path = path_str[0].split(" --> ")
    print(path)
    return path
```

Decisioni di Progetto

Il percorso, quindi, viene aggiornato nel caso in cui dovesse cambiare lo stato delle stanze.

Lo scopo del pathfinding è il raggiungimento della stanza del boss, da qualunque stanza, a condizione che il giocatore possieda l’oggetto “bussola”, che si trova nella stanza del tesoro. La chiamata per calcolare il pathfinding viene effettuata nuovamente se il giocatore dovesse scegliere di ignorare le indicazioni.

L’input al pathfinder è dato sotto forma di archi rappresentati da 3 numeri (le due stanze collegate ed il loro peso). Questo ritorna semplicemente l’array con l’ordine delle stanze da seguire.

```
Input path finder:0 4 2 - 1 4 2 - 1 5 2 - 2 4 2 - 3 6 2 - 3 5 2 - 3 4 2 - 4 0 2 - 4 2 2 - 4 3 2 - 4 1 2 - 5 3 2 - 5 1 2 - 6 3 2
Output path finder:[1, 4, 2
]
Input path finder:0 4 2 - 1 4 2 - 1 5 2 - 2 4 2 - 3 6 2 - 3 5 2 - 3 4 2 - 4 0 2 - 4 2 2 - 4 3 2 - 4 1 2 - 5 3 2 - 5 1 2 - 6 3 2
Output path finder:[4, 2
]
Input path finder:0 4 2 - 1 4 2 - 1 5 2 - 2 4 2 - 3 6 2 - 3 5 2 - 3 4 2 - 4 0 2 - 4 2 2 - 4 3 2 - 4 1 2 - 5 3 2 - 5 1 2 - 6 3 2
Output path finder:[2
]
```

ALBERO DECISIONALE APPRESO

Sommario

Un albero decisionale appreso è un modello di apprendimento automatico che rappresenta una serie di regole decisionali che possono essere utilizzate per classificare o predire un valore target.

Il processo di apprendimento di un albero decisionale inizia con un insieme di dati di training, che consiste di esempi di input e output. Nel nostro caso questi esempi sono stati generati mediante il log su console della lettura dei dati con il nostro output desiderato aggiunto manualmente,

Ogni nodo interno rappresenta una domanda sulla variabile di input. I rami che si diramano da un nodo interno rappresentano le possibili risposte alla domanda. Il nodo foglia rappresenta la decisione finale che viene presa per un dato input.

```
left_wall_dist right_wall_dist top_wall_dist bottom_wall_dist x y target_dist at_risk action
1905.375854, 173.191956, 330.178955, 720.363708, 0.757249, -0.653127, 581.379395, 1, dash_left
1905.375854, 173.191956, 330.178955, 720.363708, 0.757249, -0.653127, 581.379395, 1, dash_left
1905.375854, 173.191956, 330.178955, 720.363708, -0.127768, -0.991804, 847.926697, 1, dash_left
1905.375854, 173.191956, 330.178955, 720.363708, -0.68257, -0.73082, 1014.550659, 1, dash_left
1905.375854, 173.191956, 330.178955, 720.363708, -0.85639, -0.516329, 1164.522705, 1, dash_left
1905.375854, 173.191956, 330.178955, 720.363708, -0.911922, -0.410365, 1281.172363, 1, dash_left
1905.375854, 173.191956, 330.178955, 720.363708, -0.939616, -0.34223, 1389.492188, 1, dash_left
1905.375854, 173.191956, 330.178955, 720.363708, -0.87638, -0.481621, 1197.85083, 1, dash_left
1905.375854, 173.191956, 330.178955, 720.363708, -0.630922, -0.775846, 989.556152, 1, dash_left
1905.375854, 173.191956, 330.178955, 720.363708, 0.072274, -0.997385, 806.273438, 1, dash_left
197.777222, 1877.073486, 316.053101, 689.757812, -0.921893, -0.387445, 368.059235, 1, dash_right
197.777222, 1877.073486, 316.053101, 689.757812, -0.882891, -0.469577, 476.303375, 1, dash_right
197.777222, 1877.073486, 316.053101, 689.757812, -0.801978, -0.597354, 592.910522, 1, dash_right
197.777222, 1877.073486, 316.053101, 689.757812, -0.588615, -0.808414, 726.199036, 1, dash_right
197.777222, 1877.073486, 316.053101, 689.757812, -0.150345, -0.988634, 851.169861, 1, dash_right
197.777222, 1877.073486, 316.053101, 689.757812, 0.260984, -0.965343, 942.820068, 1, dash_right
197.777222, 1877.073486, 316.053101, 689.757812, 0.662648, -0.748931, 1076.134277, 1, dash_right
197.777222, 1877.073486, 316.053101, 689.757812, 0.755623, -0.655007, 1134.460815, 1, dash_right
197.777222, 1877.073486, 316.053101, 689.757812, 0.755623, -0.655007, 1134.460815, 1, dash_right
197.777222, 1877.073486, 316.053101, 689.757812, 0.900867, -0.434096, 1334.441895, 1, dash_right
1442.730713, 596.496216, 858.729919, 202.099976, -0.228244, 0.973604, 425.493835, 0, shoot_right
1442.730713, 596.496216, 858.729919, 202.099976, -0.100612, 0.994926, 455.089264, 0, shoot_right
1442.730713, 596.496216, 858.729919, 202.099976, 0.286507, 0.958078, 562.179688, 0, shoot_right
1442.730713, 596.496216, 858.729919, 202.099976, 0.92795, 0.372706, 668.543152, 0, shoot_right
1442.730713, 596.496216, 858.729919, 202.099976, 0.852267, 0.523107, 832.080566, 0, shoot_right
1442.730713, 596.496216, 858.729919, 202.099976, 0.587014, 0.809577, 837.446045, 0, shoot_right
1442.730713, 596.496216, 858.729919, 202.099976, 0.988469, -0.151426, 1072.617188, 0, shoot_right
1442.730713, 596.496216, 858.729919, 202.099976, 0.946537, 0.322595, 1215.950562, 0, shoot_right
596.496216, 1442.730713, 858.729919, 202.099976, -0.940071, 0.34098, 337.925568, 0, shoot_left
596.496216, 1442.730713, 858.729919, 202.099976, -0.978441, 0.206529, 367.176636, 0, shoot_left
596.496216, 1442.730713, 858.729919, 202.099976, -0.962502, 0.271275, 348.288391, 0, shoot_left
596.496216, 1442.730713, 858.729919, 202.099976, -0.93387, 0.357612, 337.34964, 0, shoot_left
596.496216, 1442.730713, 858.729919, 202.099976, -0.900104, 0.435675, 346.689636, 0, shoot_left
596.496216, 1442.730713, 858.729919, 202.099976, -0.863107, 0.505021, 374.795319, 0, shoot_left
596.496216, 1442.730713, 858.729919, 202.099976, -0.829367, 0.558703, 281.280823, 0, shoot_left
596.496216, 1442.730713, 858.729919, 202.099976, -0.898901, 0.438152, 230.956833, 0, shoot_left
596.496216, 1442.730713, 858.729919, 202.099976, -0.95267, 0.304007, 286.269104, 0, shoot_left
596.496216, 1442.730713, 858.729919, 202.099976, -0.986111, 0.166086, 395.257721, 0, shoot_left
1442.730713, 596.496216, 858.729919, 202.099976, -0.780657, 0.624959, 272.643402, 0, shoot_right
```

Strumenti utilizzati

Anche in questo caso abbiamo utilizzato per la ricerca del percorso i moduli messi a disposizione da “AIPython: Python Code for AIFCA”. E anche in questo caso abbiamo avviato il file learnDT.py attraverso lo script in Godot che gli passa l’array della lettura dello stato della stanza. Ritorna quindi la stringa che rappresenta il dizionario degli output.

Decisioni di Progetto

Lo stato della stanza è rappresentato da:

- La distanza del boss dai 4 muri
- La distanza dal giocatore
- La direzione verso cui si trova il giocatore (rappresentata mediante x e y di un vettore direzionale normalizzato)
- La presenza di un proiettile in prossimità del boss (valore booleano rappresentato da 0 e 1)

L'albero decisionale agisce come classificatore che identifica la situazione della stanza classificandola in una delle 5 possibili azioni che il boss può compiere:

- jump
- shoot_left
- dhoot_right
- dash_left
- dash_right

Ogni volta che il boss avrà terminato di eseguire un azione farà una nuova lettura della stanza, passando l'array dello stato come argomento allo script che ritornerà le coppie di che associano le classi (azioni da svolgere) alle loro probabilità. Queste coppie vengono inserite in un dizionario e quella con il valore più alto viene svolta dal boss.

Valutazione

La metrica adatta a misurare le prestazioni dell'albero, nel nostro caso è l'accuracy. Questa è una metrica appropriata per la classificazione perché misura direttamente la capacità dell'albero decisionale di classificare i dati correttamente.

Split Choice	Leaf Choice	#leaves	accuracy	absolute loss	squared loss	log loss (bits)
accuracy	empirical dist	1	0.2551020	0.7966998	0.6352525	2.3082517
accuracy	bounded empirical	1	0.2551020	0.7966998	0.6352525	2.3082517
accuracy	Laplace	1	0.2551020	0.7967725	0.6353456	2.3083195
accuracy	mode	1	0.2551020	0.7448980	0.7448980	inf
accuracy	median	5	0.2346939	0.7653061	0.7653061	inf
absolute loss	empirical dist	16	0.6938776	0.4244081	0.2679142	inf
absolute loss	bounded empirical	16	0.6938776	0.4257346	0.2669173	1.2539446
absolute loss	Laplace	11	0.6938776	0.4930551	0.3021754	1.2589310
absolute loss	mode	10	0.6428571	0.3571429	0.3571429	inf
absolute loss	median	10	0.6632653	0.3367347	0.3367347	inf
squared loss	empirical dist	16	0.7142857	0.4186452	0.2598618	inf
squared loss	bounded empirical	16	0.7142857	0.4184412	0.2588496	1.2256703
squared loss	Laplace	12	0.6836735	0.4934622	0.3050486	1.2622262
squared loss	mode	10	0.6428571	0.3571429	0.3571429	inf
squared loss	median	10	0.6632653	0.3367347	0.3367347	inf
log loss (bits)	empirical dist	15	0.6938776	0.4128429	0.2652474	inf
log loss (bits)	bounded empirical	15	0.6938776	0.4136593	0.2640433	1.2513235
log loss (bits)	Laplace	13	0.7040816	0.5141623	0.3240527	1.3295934
log loss (bits)	mode	1	0.2551020	0.7448980	0.7448980	inf
log loss (bits)	median	1	0.2551020	0.7448980	0.7448980	inf

Le misurazioni sono state effettuate attraverso la funzione *testSDT()* messa a disposizione dallo strumento *AIPython.* Questi risultati ci indicano che l'accuratezza del nostro albero decisionale si aggira intorno al 70%. Per ottenere questo risultato sono stati utilizzati circa 300 esempi. Il funzionamento percepito in gioco è abbastanza soddisfacente il che si riscontra nei dati della misurazione.

Uno dei principali problemi di un albero decisionale è il rischio di overfitting, cioè il troppo adattamento ai dati di training e la perdita della capacità di generalizzazione. Un fenomeno di questo tipo probabilmente accade nei casi delle classi *dash_left* e *dash_right,* che tendono ad avere comportamenti a volte scorretti e vengo attivati soprattutto quando il boss si trova vicino alla posizione in cui molti dei casi di esempio di dash sono stati registrati.