



CdL in Intelligenza Artificiale – A.A. 2025/26
Esame di Programmazione con Laboratorio
Progetto - Prova pratica n. 4
Appello del 19 gennaio 2025

Utilizzando il linguaggio di programmazione **Python**, realizzare un'applicazione con interfaccia grafica per giocare a “**Percorso Evolutivo**”.

Il gioco è per **un solo giocatore** e si svolge su una **griglia 20×20** composta da celle quadrate. All'inizio di ogni partita la griglia viene generata automaticamente e contiene le seguenti tipologie di celle: una **posizione di partenza (P)**, unica, in cui si trova inizialmente il giocatore; una **posizione obiettivo (O)**, unica, che il giocatore deve raggiungere; **muri (X)**, che rappresentano ostacoli non attraversabili; **risorse (R)**, che forniscono punti bonus quando raccolte; **trappole (T)**, che penalizzano il punteggio quando attraversate; le restanti celle sono **celle vuote (.)**.

Prima di iniziare una nuova partita l'utente deve poter scegliere un **livello di difficoltà** tramite una voce di menu o un controllo nell'interfaccia. Il livello modifica la generazione della griglia variando le percentuali di muri, trappole e risorse secondo la seguente tabella, mantenendo invariata la dimensione della griglia: livello **Facile**: muri 15%, risorse 12%, trappole 3%; livello **Medio**: muri 20%, risorse 10%, trappole 5%; livello **Difficile**: muri 25%, risorse 8%, trappole 7%. Le percentuali sono indicative e possono essere arrotondate al numero intero di celle più vicino. Il livello scelto deve essere visualizzato nell'interfaccia tramite una label.

All'avvio dell'applicazione deve essere richiesto l'inserimento del **nome del giocatore**; se il nome non viene inserito, non deve essere possibile iniziare la partita. L'applicazione deve prevedere una **barra dei menu** dalla quale sia possibile avviare una **nuova partita** e visualizzare la **classifica**. L'opzione “Nuova partita” deve reimpostare completamente lo stato del gioco, inclusi griglia, posizione del giocatore, punteggio, contatori, azioni speciali disponibili, livello selezionato e timer.

Il giocatore può muoversi sulla griglia **solo nelle quattro direzioni cardinali** (alto, basso, sinistra, destra), tramite appositi **bottoni di movimento** presenti nell'interfaccia. Ogni mossa incrementa il contatore delle mosse effettuate. Il giocatore non può attraversare celle contenenti muri (X). Se il giocatore attraversa una cella contenente una **risorsa (R)**, la risorsa viene raccolta, la cella diventa vuota e il punteggio viene incrementato. Se il giocatore attraversa una **trappola (T)**, il punteggio viene penalizzato; la trappola resta attiva e può penalizzare anche attraversamenti successivi.

Durante l'intera partita il giocatore ha a disposizione **due azioni speciali**, ciascuna utilizzabile **una sola volta**. La prima azione speciale consente la **rimozione di un muro (X)** adiacente alla posizione corrente del giocatore; l'azione deve essere attivata tramite **doppio clic** sulla cella contenente il muro da rimuovere. La seconda azione speciale consente la **trasformazione di una trappola (T) adiacente** in una risorsa (R), sempre tramite **doppio clic** sulla cella corrispondente. Dopo l'utilizzo, ciascuna azione speciale non può più essere utilizzata per il resto della partita; l'interfaccia deve indicare chiaramente, tramite opportune label o icone, se un'azione speciale è ancora disponibile.

La griglia di gioco **evolve nel tempo**. Dopo ogni **cinque mosse del giocatore**, viene eseguito un metodo **step()** che modifica lo stato della griglia secondo le seguenti regole, applicate nell'ordine

via Archirafi, 34 - 90123 Palermo – tel 0039 091/238 91012-238 91025 – fax 0039 091/238 60710

sito web: <http://www.unipa.it/dipartimenti/dimatematiceinformatica>

e-mail: dipartimento.matematicainformatica@unipa.it – PEC: dipartimento.matematicainformatica@cert.unipa.it



indicato. Due risorse non raccolte vengono trasformate in celle vuote; se è presente una sola risorsa, ne viene trasformata una sola. Successivamente, una cella vuota casuale viene trasformata in risorsa. Poi, due celle vuote casuali vengono trasformate in trappole; se è presente una sola cella vuota, ne viene trasformata una sola. Infine, una trappola casuale viene trasformata in cella vuota. L'evoluzione **non può mai modificare** la posizione iniziale (P), l'obiettivo (O), la cella occupata dal giocatore né le **quattro celle adiacenti** ad essa, che costituiscono una *zona sicura*. I muri non vengono mai creati o rimossi automaticamente dall'evoluzione.

Dopo ogni `step()` deve essere verificato se il giocatore dispone ancora di **almeno una mossa valida** e se esiste un percorso dal giocatore all'obiettivo considerando attraversabili tutte le celle tranne i muri. Se l'obiettivo diventa irraggiungibile a causa dell'evoluzione, la partita termina.

Il punteggio viene calcolato come segue: **+10 punti** per ogni risorsa raccolta; **-5 punti** per ogni trappola attraversata; **-1 punto** per ogni mossa effettuata; **+20 punti** per aver raggiunto la posizione obiettivo (O).

Il gioco termina in uno dei seguenti casi: il giocatore raggiunge l'obiettivo; il giocatore esaurisce il numero massimo di **30 mosse**; l'evoluzione della griglia rende impossibile raggiungere l'obiettivo. Al termine del gioco deve essere mostrato un **resoconto finale** che includa il punteggio totale, il numero di risorse raccolte, il numero di trappole attraversate e il numero totale di mosse effettuate, insieme al livello di difficoltà selezionato.

L'interfaccia grafica deve essere realizzata utilizzando **breezypythongui** e deve mostrare chiaramente la griglia di gioco. Le celle devono essere rappresentate con colori distinti: **verde** per l'obiettivo (O), **blu** per la posizione iniziale (P), **nero** per i muri (X), **rosso** per le trappole (T), **giallo** per le risorse (R) e **bianco** per le celle vuote. Devono essere presenti un **bottone START** per avviare la partita, quattro bottoni per il movimento del giocatore, e opportune **label** che mostrino in tempo reale il punteggio, il numero di mosse effettuate, lo stato delle azioni speciali, il livello selezionato e lo stato della partita. Deve inoltre essere presente una **label timer** che mostri il **tempo trascorso in secondi**, aggiornato in tempo reale a partire dall'inizio della partita.

Al termine della partita, il punteggio deve essere salvato nel file `classifica.txt`. Il salvataggio deve funzionare anche nel caso in cui il file non sia presente. La classifica deve essere consultabile tramite un'apposita voce di menu e deve mostrare **al massimo i 10 migliori risultati**. La classifica deve essere mantenuta ordinata in **ordine decrescente di punteggio**; in caso di parità di punteggio, deve precedere il giocatore che ha effettuato meno mosse; in caso di ulteriore parità, deve essere utilizzato l'ordine alfabetico del nome del giocatore. In classifica deve essere salvato anche il **livello** della partita; a parità di punteggio e mosse, deve precedere una partita di livello più difficile (Difficile prima di Medio prima di Facile).

Devono essere definite opportune classi per la gestione del gioco, includendo almeno le classi `Cella`, `Griglia`, e `Gioco`, individuando per ciascuna attributi di istanza e metodi significativi. In particolare, la classe `Griglia` deve contenere il metodo `step()` per l'evoluzione della griglia e un metodo per verificare l'esistenza di mosse valide e di un percorso verso l'obiettivo. La corretta progettazione orientata agli oggetti è parte integrante della valutazione della prova.

L'intero codice, opportunamente commentato, deve essere indispensabilmente prodotto su Github seguendo le indicazioni date a lezione e sintetizzate nel file in allegato, entro e non oltre le ore 20:00 del 12/01/2026.

Si richiede, inoltre, di allegare oltre alla traccia del progetto, una **breve relazione** che includa:



1. **La strategia adottata per la soluzione del progetto.**
2. **La descrizione delle classi utilizzate per definire la soluzione.**
3. **La descrizione dei metodi più rilevanti**, con particolare attenzione alla complessità di tempo, che deve essere indicata e motivata.
4. **Il ruolo di ciascun componente del gruppo**, specificando se tutti hanno collaborato su tutte le parti o come sono stati suddivisi i compiti. La relazione deve evidenziare una **equa distribuzione del lavoro**, che influirà sulla valutazione finale del progetto.