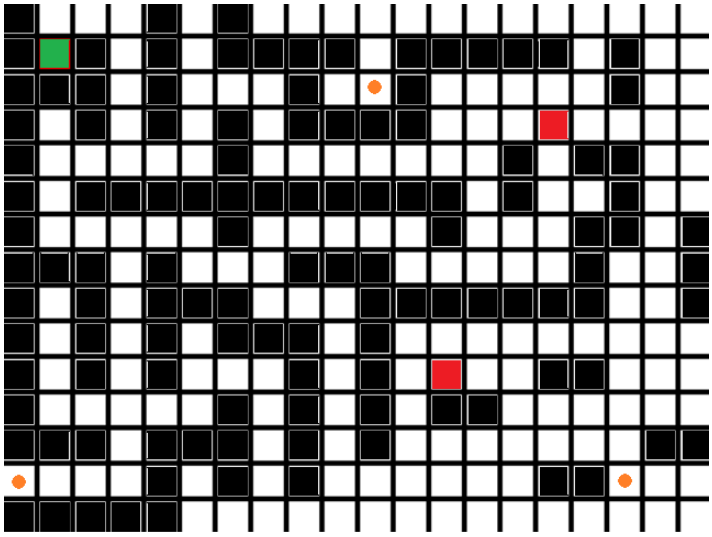
FAKE-MAN

1109671 Lerco Nicola,
1110494 Colucci Edoardo,
1111836 Bolognini Massimiliano,
1109395 Trapani Francesco

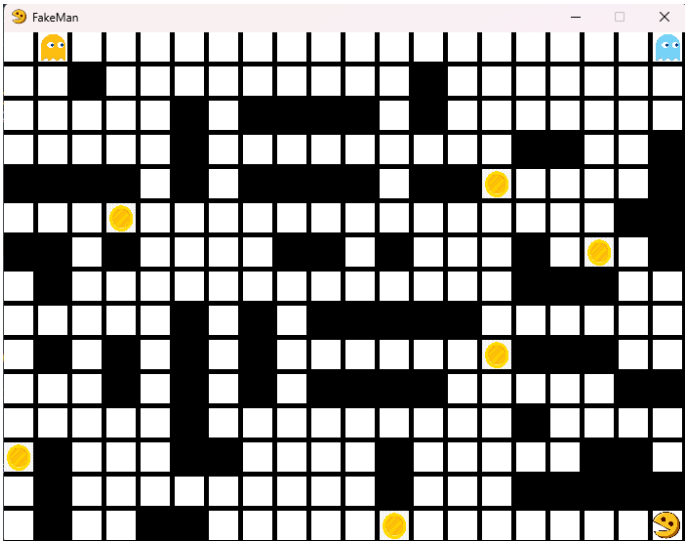
SPIEGAZIONE GENERALE

Fake-man è un gioco single player dove il giocatore impersonifica un personaggio di fantasia con l'biettivo di collezionare tutti i gettoni sparsi nella mappa, cercando di non farsi uccidere dai nemici nei panni di colorati fantasmini.

Il gioco è ispirato al celebre gioco di Pac Man, gioco del 1980. Questo nome è stato scelto perché appunto, imitazione (elaborata) del gioco originale.



PROTOTIPO



PROGETTO FINALE

COME FUNZIONA IL GIOCO:

- Il giocatore impersonifica FakeMan, il protagonista. Il main player può muoversi nei rispettivi quadratini bianchi attraverso spostamenti adiacenti usando le relative freccette della tastiera. I quadratini neri, oltre ai confini della mappa, invece, sono i punti dove il giocatore va a sbattere o che non può attraversare;
- L'avventura parte con il giocatore principale e due nemici, il conteggio dei nemici aumenta di 1 ogni 20 mosse del giocatore, bisogna stare attenti a quante mosse usi perché ci si potrebbe ritrovare davanti ad un'orda di nemici pronti ad acchiapparti!
- I nemici sono rappresentati da dei fantasmini colorati e sono guidati dall'intelligenza artificiale;

- L’obiettivo del giocatore è quello di raccogliere tutte le monete per completare il livello prima che il giocatore venga mangiato dal nemico, tutto questo accompagnato da musica stile Arcade Game così da riecheggiare lo stile anni 80’.

PROGETTAZIONE IN PYTHON:

- L’interfaccia grafica e la logica di movimento sono state gestite in python.
- Tutto quello che è grafica è stato generato attraverso la libreria pygame.
- L’interfacciamento tra python e prolog è stato gestito attraverso pyswip.
- Pyswip permette l’invocazione di algoritmi prolog che saranno spiegati successivamente.
- La parte riguardante l’audio è gestita interamente da pygame.

Il loop principale è gestito da un while true che permette al gioco di proseguire fino alla chiusura dell’applicazione. La funzione che permette l’interfacciamento prolog è:

- Aggiorna_posizione_nemico(): questa funzione contiene il core per il movimento appunto dei nemici. Essa invoca gli script prolog, passando la posizione corrente dei nemici, del player, dei coins e ritornando la prossima posizione che andrà ad aggiornare la mappa di disegno dei nemici.

FAKE-AI IN PROLOG

Il gioco prevede la divisione dei nemici in due ruoli principali: followers e campers.

- **Comportamento del follower**:
  I nemici followers si muovono in base alla loro percezione del giocatore e alla memoria dell'ultima posizione vista, utilizzando l'algoritmo A* per inseguire efficacemente.

  Se nel turno corrente un nemico **vede il giocatore**, aggiorna la sua memoria e utilizza A* per raggiungerlo.

  Se **non lo vede**, ma ha ancora in memoria un’ultima posizione vista, continua a muoversi verso di essa con A*.

  Se raggiunge la posizione memorizzata e il giocatore non è lì, cancella la memoria e torna a vagare casualmente.

- **Comportamento del camper**:
  il camper segue sempre le posizioni da raggiungere che si prefissa ricercando il percorso con l’algoritmo **A*.**

  Se nel turno corrente il camper non ha nessuna posizione target, allora prenderà come target la posizione del pallino più vicino ad esso.

  Una volta che la posizione target viene raggiunta, il camper si muoverà in una posizione randomica adiacente ed il target corrente verrà cancellato.

Quando il camper **vede il giocatore** (cioè il giocatore si trova in una posizione che ha la stessa X o Y di quella del camper e non ci sono muri tra il giocatore e quest'ultimo) allora cambierà la posizione del target alla posizione in cui ha visto il giocatore

MOTIVATIONS

- **Perché A*:**
    - È ottimo per permettere ai nemici di creare dei percorsi a basso costo, sacrificando però, un pizzico di imprevedibilità.
    - Avremmo potuto fare muovere il nemico con minimax, ma dopo avere visto che calcolare le funzioni di valutazione richiedeva troppo tempo e che considerare le mosse di molteplici nemici era troppo complesso.
    - Grazie ad A* l'algoritmo è più efficiente perché calcola un percorso alla volta risultando molto più agevole nella gestione di più nemici. Mentre min Max e alfabeta in questo falliscono perché dovrebbero gestire tutte le combinazioni dei vari nemici generati e le loro mosse per ogni turno rendendo il tutto impraticabile.