Artificial intelligence  
  
  
  
WUMPUS  
  
  
Dr.Asaeei

Arman Tayebi   
  
  
تمرین:

تحلیل "WUMPUS" :

1. مفاهیم مرتبط با الگوریتم‌های جستجو:

- بازی "Hunt the Wumpus" نشان‌دهنده مفاهیمی همچون جستجو در گراف، الگوریتم‌های DFS (جستجوی عمق اول)، BFS (جستجوی سطح اول) و الگوریتم‌های هوش مصنوعی مانند A\* است. این الگوریتم‌ها در جستجوی هدف، کاوش محیط، و یافتن بهترین مسیر به کار می‌روند.

2. مفاهیم مرتبط با تصمیم‌گیری:

- در بازی "Hunt the Wumpus"، بازیکن نیازمند تصمیم‌گیری در مواجهه با شرایط ناقص و محیط پیچیده است. این شامل تصمیم‌گیری در مورد حرکت به یک اتاق خاص، اکتشاف گنجینه یا جلوگیری از ورود به اتاق حاوی Wumpus یا گودال است.

3. برنامه‌ریزی حرکت:

- برنامه‌ریزی حرکت در بازی نشان‌دهنده مفهومی اساسی در هوش مصنوعی است. بازیکن باید مسیر حرکت خود را در محیط مشخص کند تا به سرعت به هدف برسد و از خطرات جلوگیری کند.

4. آموزش مفاهیم مهم هوش مصنوعی:

- بازی "Hunt the Wumpus" یک محیط آموزشی جذاب برای آشنایی با مفاهیم مهم هوش مصنوعی مانند جستجو، تصمیم‌گیری، ریسک و پاداش است. این بازی به بازیکنان کمک می‌کند تا مفاهیم اصلی هوش مصنوعی را در یک سیاق تعاملی فهمیده و به کار بگیرند.

5. تعامل با محیط پویا و ناقص:

- بازیکن در مواجهه با محیط پویا و ناقص با مشکلاتی مانند نقاط کور، اطلاعات ناقص در محیط، و شوک‌های غیرمنتظره روبرو می‌شود. این نشان می‌دهد چگونه هوش مصنوعی باید با شرایط ناپایدار و متغیر مواجه شود.

6. شکار و ریسک:

- این بازی به بازیکنان نشان می‌دهد که چگونه تصمیم‌گیری در مواجه با ریسک‌های مختلف و انتخاب مسیر‌هایی که به شکار هدف می‌انجامد، بسیار مهم است.

به طور کلی، "Hunt the Wumpus" یک بازی آموزشی جذاب است که مفاهیم کلیدی هوش مصنوعی را به صورت عملی و تعاملی آموزش می‌دهد. این بازی به مسائل مرتبط با جستجو، تصمیم‌گیری هوشمندانه، و مدیریت ریسک در محیط‌های پویا می‌پردازد و می‌تواند در فهم بهتر دانشجویان در درس هوش مصنوعی موثر باشد.

کد ساده ای از WUMPUS :

import random

class WumpusGame:

def \_\_init\_\_(self, size):

self.size = size

self.rooms = [i for i in range(1, size + 1)]

self.wumpus\_location = random.choice(self.rooms)

self.pit\_locations = random.sample(self.rooms, 2)

self.current\_room = random.choice(self.rooms)

self.arrows = 1

def print\_instructions(self):

print("Welcome to Hunt the Wumpus!")

print("You are in a cave with rooms numbered 1 to", self.size)

print("Try to find and shoot the Wumpus without falling into pits.")

print("Commands: 'shoot', 'move', 'quit'")

def print\_current\_status(self):

print("You are in room", self.current\_room)

print("You have", self.arrows, "arrows left.")

def check\_for\_wumpus(self):

if self.current\_room == self.wumpus\_location:

print("You found the Wumpus! Congratulations, you win!")

return True

return False

def check\_for\_pit(self):

if self.current\_room in self.pit\_locations:

print("You fell into a pit! Game over.")

return True

return False

def play(self):

self.print\_instructions()

while True:

self.print\_current\_status()

action = input("Enter your command: ").lower()

if action == "quit":

print("Thanks for playing! Goodbye.")

break

elif action == "move":

new\_room = int(input("Enter the room number to move: "))

if new\_room in self.rooms:

self.current\_room = new\_room

if self.check\_for\_wumpus() or self.check\_for\_pit():

break

else:

print("Invalid room number. Try again.")

elif action == "shoot":

if self.arrows > 0:

target\_room = int(input("Enter the room number to shoot:"))

self.arrows -= 1

if target\_room == self.wumpus\_location:

print("Congratulations! You shot the Wumpus and won!")

break

else:

print("Missed! The Wumpus is still lurking.")

if random.randint(1, 4) == 1:

print("The Wumpus moved to a new room!")

self.wumpus\_location = random.choice(self.rooms)

else:

print("You're out of arrows! Game over.")

break

else:

print("Invalid command. Try again.")

if \_\_name\_\_ == "\_\_main\_\_":

game\_size = 5 # You can change the size of the cave as needed

wumpus\_game = WumpusGame(game\_size)

wumpus\_game.play()