Wumpus World
AI Project

By
Abdullah Younis

For
CS-171
Introduction to Artificial Intelligence

Student Booklet

I. Table of Contents

II. Introduction

In this programming assignment, you are tasked with implementing a Knowledge-based, Wumpus World AI agent, which should be able to solve a Wumpus World game. You will have the choice of programming in Python, Java, or C++. A code base will be made available to you, and you are asked to edit one or more files from whichever shell you choose to use. Your agent should be able to read percepts and act rationally; your grade will be determined by your agent's performance measure. At the end of the quarter, your agent will compete against your classmates' agents in a tournament.

III. Wumpus World Game Mechanics

The Wumpus World is a cave consisting of rooms connected by passageways. Lurking somewhere in the cave is the terrible wumpus, a beast that eats anyone who enters its room. The wumpus can be shot by an agent, but the agent has only one arrow. Some rooms contain bottomless pits that will trap anyone who wanders into these rooms (except for the wumpus, which is too big to fall in). The only mitigating feature of this bleak environment is the possibility of finding a heap of gold. A concrete definition of the game is given by this PEAS description:

Performance Measure

The performance measure of an agent is an integer score calculated based on the following:

- Start at 0 points
- -1 point for each action taken
- -10 for using the arrow (additional to the -1 point)
- -1000 points for falling into a pit or being eaten by the wumpus
- +1000 for climbing out of the cave with the gold

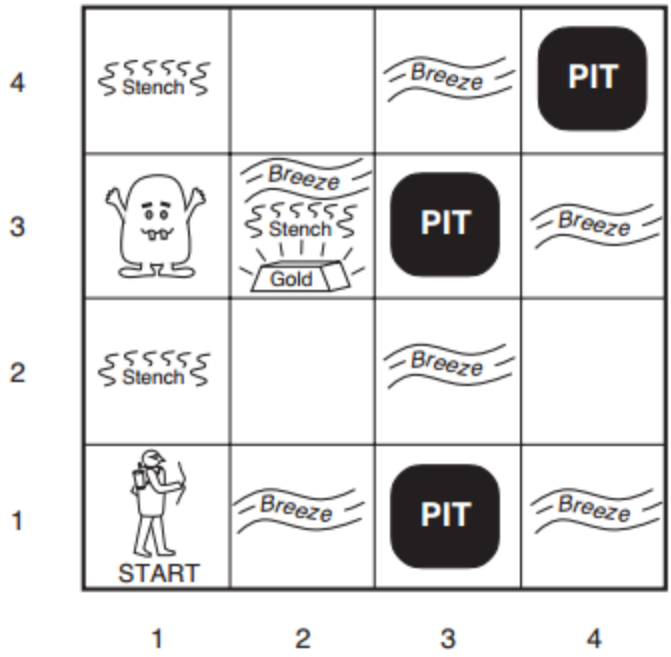
The game ends either when the agent dies, when the agent climbs out of the cave, or when the agent's score goes below -1000.

Environment

The environment can be classified as partially observable, deterministic, sequential, static, discrete, and single agent.

- An NxM grid of rooms, where $4 \leq N, M \leq 10$.
- The agent always starts in the bottom left square (1, 1), facing to the right.
- The locations of the gold and the wumpus are chosen randomly, with a uniform distribution, from the squares other than the start square.
- Each square other than the start can be a pit, with a 20% probability.
- The agent dies a miserable death if it enters a square containing a pit or a live wumpus.

An example 4x4 Wumpus World:



Actuators

- The agent can move **FORWARD**, **TURN_LEFT** by 90 degrees, or **TURN_RIGHT** by 90 degrees.
- The action **GRAB** can be used to pick up the gold if it is in the same square as the agent.
- The action **SHOOT** can be used to fire an arrow in a straight line in the direction the agent is facing. The arrow continues until it either hits and kills the wumpus or hits a wall. The agent has only one arrow, so only the first shoot action has any effect.
- The action **CLIMB** can be used to climb out of the cave, but only from square (1, 1).

Sensors

- In the square containing the wumpus and in the directly (not diagonally) adjacent squares, the agent will perceive a **STENCH**.
- In the squares directly adjacent to a pit, the agent will perceive a **BREEZE**.
- In the square where the gold is, the agent will perceive a **GLITTER.**
- When an agent walks into a wall, it will perceive a **BUMP**.
- When the wumpus is killed, it emits a woeful **SCREAM** that can be perceived anywhere in the cave. This percept will only be sensed on the turn immediately after the wumpus's death.

IV. Tasks to Complete

Setup Your Environment

In this section, you will find help setting up your coding environment. This project will take advantage of UCI's openlab; any other coding environment is not supported.

Install Required Applications

To connect to openlab, you will need to use SSH. SSH stands for Secure SHell. It is a program designed to allow users to log into another computer over a network, to execute commands on that computer and to move files to and from that computer. A Mac user can use the terminal application, whereas, a Windows user will need to install PuTTY. You can download PuTTY from here. Download the MSI installer for Windows, and run the installer for PuTTY.

Connect to Openlab

Connecting to openlab is as easy as SSHing into the openlab server. If you are on Windows and using PuTTY, type "openlab.ics.uci.edu" into the Host Name box; make sure the port is 22 and the SSH flag is ticked. Click open, and login using your ICS account info. If you are using a Mac, open the terminal found under Applications -> Utilities. Enter 'ssh ICSUSERNAME@openlab.ics.uci.edu' and login using your ICS account info.

Upload the shell to Openlab

To upload your shell to Openlab, you will need to use SCP, which stands for secure file copy. This program will allow you to copy files over a network. If you are using a mac, type this into the terminal:

```
scp -r /path/to/shell/folder/ ICSUSERNAME@openlab.ics.uci.edu:/path/
```

If you are using Windows, type this into a command prompt:

```
pscp -r /path/to/shell/folder/ ICSUSERNAME@openlab.ics.uci.edu:/path/
```

Extra Information about Openlab:

- http://www.ics.uci.edu/~lab/students/#unix
- https://www.ics.uci.edu/computing/linux/hosts.php

Extra Information about UNIX:

- https://cgi.math.princeton.edu/compudocwiki/index.php?title=Documentation_and_Information:Getting_started_with_Linux

Program Your AI

Once you have your environment setup, you can start to program your agent. In the 'src' folder of your shell you will find the source code of the project. You are only allowed to make changes to the MyAI class.

Compile Your AI

Compiling your program is easy as executing the command **make** from the shell's root directory (the directory with the makefile in it).

Test Your AI

To run your program after you have compiled it, navigate to the bin folder. You should find the compiled program inside. Refer to the Shell Manual Appendix for help running it. To generate large amounts of worlds to use with the folder option, refer to the World Generator. If you are using the Python Shell make sure you are using Python 3.5.2. On openlabs, run the command **module load python/3.5.2** to load Python 3.5.2.

Write Your Project Report

Write a report according to your Professor's instructions. Make sure your report is in **pdf** format and place it inside the 'doc' folder.

Submit Your Project

At this point you should have your most up-to-date source code in the 'src' folder, your report in **pdf** format in the 'doc' folder, and your compiled project in the 'bin' folder. Navigate to your shell's root directory and execute the command **make submission**. It will ask you for some information and create a zip file inside the folder. Submit this zip file to EEE or Canvas.

V. Understanding the Tournament

After you submit your project and the deadline passes, you will be entered into a tournament with your classmates. The tournament checks to make sure you followed all the instructions correctly, then runs your agent across 10000 worlds of variable sizes from 4x4 to 7x7. Every agent is run on the same 10000 worlds to ensure fairness. Your agent's average score and standard deviation is calculated and a scoreboard is constructed that will be made available. Your agent will be timed-out if it hangs for longer than 2 hours. After the scoreboard is constructed, scores and standard deviations are checked for any illegal submissions. These include two agents with the same score, or an agent who are not smart. An agent is not smart is performs the same actions independent of the world it is running on.

VI. Appendix: Shell Manual

Name

The command line name used to invoke this program will change depending on the shells:

$$\text{Wumpus\_World} = \begin{cases} \text{python Main.pyc} & \text{if using python shell} \\ \text{java } -\text{jar Wumpus\_World.jar} & \text{if using java shell} \\ \text{Wumpus\_World} & \text{if using cpp shell} \end{cases}$$

Synopsis

Wumpus_World [Options] [InputFile] [OutputFile]

Options

-m Use the ManualAI instead of MyAI. If both –m and –r specified, ManualAI will be turned off.

-r Use the RandomAI instead of MyAI.

-d Debug mode, which displays the game board after every move. Redundant with –m.

-h Displays help menu and quits program.

-v Verbose mode, which displays name of world files as they are loaded.

-f Treats the InputFile as a folder containing many worlds. The program will then construct a world for every valid world file found. This will trigger the program to display average score and standard deviation instead of a single score. The InputFile operand must be specified with this option.

Operands

InputFile A path to a valid Wumpus World file, or folder with –f. This operand is optional unless used with –f or OutputFile.

OutputFIle A path to a file where the results will be written. This is optional.

Examples

Wumpus_World	Constructs a random 4x4 world, runs the MyAI agent on the world, and prints output to console.
Wumpus_World -m	Constructs a random 4x4 world, runs the ManualAI agent on the world, and prints output to console.
Wumpus_World -d	Constructs a random 4x4 world, runs the MyAI agent on the world, and prints output to console. After every turn, the game pauses and prints the current game state to the console.
Wumpud_World -r	Constructs a random 4x4 world, runs the RandomAI agent on the world, and prints output to console.
Wumpus_World -h	Prints the help menu and terminates.
Wumpus_World -rd	Constructs a random 4x4 world, runs the RandomAI agent on the world, and prints output to console. After every turn, the game pauses and prints the current game state to the console.

Wumpus_World /path/to/world/file.txt	Constructs the world specified in the file, runs the MyAI agent on the world, and prints output to console.
Wumpus_World -f /path/to/world/files/	Constructs all the worlds specified in the folder, runs the MyAI agent on all the worlds, and prints output to console.
Wumpus_World -fv /path/to/world/files/	Constructs all the worlds specified in the folder, prints world names as they are loaded; runs the MyAI agent on all the worlds, and prints output to console.
Wumpus_World -rf /path/to/world/files/	Constructs all the worlds specified in the folder, runs the RandomAI agent on all the worlds, and prints output to console.
Wumpus_World /path/to/world/file.txt /path/to/output/file.txt	Constructs the world specified in the file, runs the MyAI agent on the world, and prints output to the output file.
Wumpus_World -rf /path/to/world/files/ /path/to/output/file.txt	Constructs all the worlds specified in the folder, runs the RandomAI agent on all the worlds, and prints output to the output file.

Notes

The Python shell uses Python version 3.5.2.

When using debug mode or ManualAI, the board will printed to the console. Each tile is represented as a full stop potentially followed by a series of characters. A '@' represents the agent, a 'P' represents a pit, a 'W' represents the wumpus, a 'G' represents the gold, a 'B' represents a breeze, and an 'S' represents a stench.