













 Stench		 Breeze	 PIT
  Stench  Gold	 Breeze	 PIT	 Breeze
 Stench		 Breeze	
 START	 Breeze	 PIT	 Breeze

Wumpus World

Student Manual

For Introduction to Artificial Intelligence

Contributors: Abdullah Younis, Ling Jin

Contact: Ling Jin (lingj6@uci.edu)

Last update: Sep 5, 2018 by Ling Jin

Table Of Content

Table Of Content.....	1
1. Introduction	2
2. Team Formation	2
Submission	2
Deadline	3
3. Wumpus World Game Mechanics	3
Performance Measure	3
Environment	3
Actuators	4
Sensors	5
4. Task to Complete	5
Setup Your Environment	5
Install Required Applications	5
Connect to Openlab	5
Program Your AI	6
Compile Your AI	6
Test Your AI	6
Write Your Project Report	6
Submit Your Project	6
5. Understanding the Tournament	6
6. Grading	7
7. Academic Honesty	Error! Bookmark not defined.
Academic Honesty	Error! Bookmark not defined.
8. Appendix: Shell Manual	8
Name	8
Synopsis	8
Options	8
Operands	9
Examples	9
9. Note	10

1. 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. Remember, you should run all codes in openlab.

2. Team Formation

Please form projects teams (individual or pairs) and submit your team information.

You should be able to self-sign-up into groups of 1-2 on Canvas (see People::StudentGroups). You or your group should submit your team name and the names of your term members.

Submission

- Please use only letters, digits in your team name.
- Please do not use space, underscore or any other special characters in your team name.
- Please follow the **exact** format shown below or points will be deducted.
TeamName: <enter_team_name>
Member1: <enter member 1 name only>
Member2: <enter member 2 name only>

3. 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 gold.

The gaming 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 $N \times M$ grid of rooms, where $4 \leq N, M \leq 7$.
- 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 4x6 Wumpus World:

- The world file is in .txt format. The information of Wumpus world will be displayed as several lines:

```
4      6
3      2
0      3
4
0      2
2      2
2      3
3      1
~
~
```

meaning:

- First line: the size of world. (4-> column number; 6 -> row number)
 - Second line: the position of Wumpus. (3-> fourth col [start from 0 col]; 2-> third row [start from 0 row])
 - Third line: the position of gold.
 - Fourth line: the number of pitfalls.
 - Fifth line to last line: the position of all pitfalls.
- The corresponding 4x6 world (4 columns and 6 rows)

```
      .      .      .      .
      .      .      B.     .
GB.    B.    PB.    BS.
P.     B.    PBS.   WB.
B.     .     B.    PS.
@.     .     .     B.
```

- symbol meaning:
 - @: agent's current position
 - W: Wumpus
 - P: Pitfall
 - G: Gold
 - B: Breeze
 - S: Stench

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.

4. Task 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 `'sshICSUSERNAME@openlab.ics.uci.edu'` and login using your ICS account info.

Extra information about openlab:

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

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 Professor's instructions. Make sure your report is in **pdf** format and place it inside the `'doc'` folder. Template will be provided. **The template is in page 10.**

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. Please add these three folders to a zip file and submit it to Canvas.

5. 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 if it performs the same actions independent of the world it is running on.

6. Grading

- The scoring will depend on (1) team formation, (2) three progressively more difficult deadlines (Minimal, Draft, Final AI), (3) the final report, and (4) tournament bonus. All submissions lose 10% of the grade for each day or fraction thereof the submission is late. Late Final AIs are not eligible to participate in the tournament nor get a tournament bonus.
- Team formation: This will require students to complete the team formation with a certain format, and register their team with canvas. Students will get 100% if they follow the format and instructions exactly and correctly, and lose points for any error; the number of points lost is at the sole discretion of the Tournament Director.
- Minimal AI: The minimal AI requires students to compile and submit files correctly (5.0pts) and score ≥ -10 on average across 10,000 random caves (5pts). ->total pts: 10.
- Draft AI: The draft AI requires students to compile and submit files correctly (5.0pts) and score ≥ 100 on average across 10,000 random caves (30pts). ->total pts: 35.
- Final AI: The final AI requires students to compile and submit files correctly (5.0pts) and score ≥ 200 on average across 10,000 random caves (30pts). Furthermore, students also need to submit final report in the final AI folder. (15pts) -> total pts: 50.
- For each deadline, they get 100% if they achieve the stated goal of that deadline. If they achieve less than the stated goal, then they get a number of points equal to the percentage of the goal they achieved. For example, if the goal is for their AI to achieve 500 and it achieves only 350, then they get $70\% = (350/500) \cdot 100\%$ for that deadline.
- For the final report, if students write each section in clear, logical, technical prose, they get 100%. If not, they will lose points; the number of points lost is at the sole discretion of the Tournament Director.

- For the tournament, students' Final AI (or final submission for Sudoku) will compete against all other Final AIs in a tournament. They will receive a decile ranking, 1-10, depending on their performance: top 10% = 10, second 10% = 9, etc., lowest 10% = 1.

7. Appendix: Shell Manual

Name

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

```
Wumpus_World = python3 Main.pyc
```

If using python shell

```
java -jar Wumpus_World.jar
```

If using java shell

```
Wumpus_World
```

If using cpp shell

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.

`Wumpus_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.

8. Note

All the various CS-171 AI project shells were written by former CS-171 students who became

interested in AI and signed up for CS-199 in order to pursue their interest and write more

interesting AI project shells. Please let me know if this is of interest to you (CS-171 grade of A- or better required).

9. Common Mistakes

Below are the common mistakes that students often make when submitting.

- Leaving 'print' or 'cout' statements in the source codes. Please delete these because it will create problem for the grader in both time and script.
- Giving additional information or text in the Team Formation submission. Please only fill the required info.
- Not testing in openlab. This seldom happens but it does: it runs in your computer environment, but it doesn't run in openlab (and openlab is where the grading will take place), so please test it for consistency.
- Only submitting the source code(s)/ Not submitting a zip file. Please use the make/ make all command provided, as it will make a zip file with your team name (<team_name>.zip).

CS-171 Wumpus World Final AI Report

Team name _____

Member #1 (name/id) _____ Member #2 (name/id or N/A) _____

[Delete this line.] Space suggestions below are approximate estimates. Use more or less text in each, as desired.

[Delete this line.] You are obliged to use Times New Roman font, size 12.

Do not exceed one page total.

I. In about 1/2 page of text, describe what you did to make your Final AI agent "smart."

II. In about 1/4 page of text, describe problems you encountered and how you solved them.

III. In about 1/4 page of text, provide suggestions for improving this project.