|  |
| --- |
| Wumpus World: Logic and Strategy in an Uncertain Labyrinth |
| A CAPSTONE PROJECT  Submitted By |
|  |
| A V ESHWAR  192211123  K CHARAN VENKATA KRISHNA  192211133 |
| In Partial Fulfillment for the completion of the course |
| CSA0911  PROGRAMMING IN JAVA FOR ARCHITECTURAL NEURAL PLATFORM  AUG 2024 |
|  |
| SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES  CHENNAI - 602105  TAMIL NADU, INDIA |



# **BONAFIDE CERTIFICATE**

This is to certify that the project report entitled **Wumpus World: Logic and Strategy in an Uncertain Labyrinth** submitted by A V ESHWAR - 192211123, K CHARAN VENKATA KRISHNA - 192211133, to Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai, is a record of bonafide work carried out by him/her under my guidance. The project fulfills the requirements as per the regulations of this institution and in my appraisal meets the required standards for submission.

GANESH RAMACHANDRAN

**COURSE FACULTY**

*Department of Deep Learning.*

*Saveetha School of Engineering,*

*SIMATS, Chennai - 602105*

# **ACKNOWLEDGEMENT**

This project work would not have been possible without the contribution of many people. It gives me immense pleasure to express my profound gratitude to our Honorable Chancellor **Dr. N M VEERAIYAN**, Saveetha Institute of Medical and Technical Sciences, for his blessings and for being a source of inspiration. I sincerely thank our Director of Academics **Dr. DEEPAK NALLASWAMY,** SIMATS, for his visionary thoughts and support. I am indebted to extend my gratitude to our Director **Dr. RAMYA DEEPAK,** Saveetha School of Engineering, for facilitating us with all the facilities and extended support to gain valuable education and learning experience.

I register my special thanks to **Dr. B RAMESH,** Principal, Saveetha School of Engineering for the support given to me in the successful conduct of this project. I wish to express my sincere gratitude to my Course faculty **GANESH RAMACHANDRAN**, for his inspiring guidance, personal involvement and constant encouragement during the entire course of this work.

I am grateful to Project Coordinators, Review Panel External and Internal Members and the entire faculty of the Department of Design, for their constructive criticisms and valuable suggestions which have been a rich source to improve the quality of this work.

**INDEX**

[BONAFIDE CERTIFICATE](#_jamwo2607bmz) 1

[ACKNOWLEDGEMENT](#_8yiu0rn5tdsd) 2

[1. ABSTRACT](#_u3bnbdop1w2) 4

[2. INTRODUCTION](#_ytjahc6y3luo) 5

[3. ARCHITECTURE DIAGRAM](#_wgnqfbsa5xqn) 6

[4. FLOWCHART](#_uylcc3c0vobr) 8

[5. UML DIAGRAM](#_p5otxqr3lwgf) 10

[6. CLASS DIAGRAM](#_mwjh6gk15gp3) 10

[7. CODE IMPLEMENTATION](#_ack6ofphfxze) 10

[7.1 JAVA CODE](#_es8tj0pw01co) 13

[7.2 HTML CODE](#_lpv4bxmmf4f6) 16

[7.3 CSS CODE](#_p66bhylntj5) 16

[8. OUTPUT SCREENSHOT](#_c61jzdgd0maj) 21

[9. CONCLUSION](#_3admx128b1cd) 24

[10. REFERENCES](#_weq4at85bunu) 25

# **1. ABSTRACT**

This capstone project explores the implementation and analysis of the Wumpus World, a classic problem in artificial intelligence that challenges logical reasoning and decision-making under uncertainty.

The project involves developing an intelligent agent capable of navigating a perilous labyrinth inhabited by a deadly Wumpus, hidden pits, and the allure of gold.

Using Java, the system architecture includes a robust backend for processing the agent's logic, a dynamic frontend for visualizing the world, and algorithms for knowledge representation and inference.

The intelligent agent employs strategies such as propositional logic, probabilistic reasoning, and heuristic search to make informed decisions about movement, sensing, and action within the Wumpus World. Key features of the implementation include real-time environment perception, risk assessment, and goal-oriented planning.

The project demonstrates the practical application of AI concepts in solving complex problems, showcasing how logical deduction and strategic thinking can be programmed into autonomous agents.

Through this Wumpus World simulation, the project aims to illustrate the challenges and solutions in navigating uncertain environments, providing insights into the design of intelligent systems for real-world applications where complete information is not always available.

# **2. INTRODUCTION**

The Wumpus World is a classic artificial intelligence problem that challenges the fundamental aspects of logical reasoning and decision-making under uncertainty. This capstone project aims to develop a comprehensive simulation of the Wumpus World, demonstrating the application of advanced AI concepts in a complex, unpredictable environment.

The motivation behind this project stems from the increasing need for intelligent systems capable of operating in uncertain and potentially hazardous conditions. The Wumpus World serves as an excellent metaphor for real-world scenarios where agents must make critical decisions based on limited information, balancing risk and reward.

The project is structured around the creation of an intelligent agent tasked with navigating the treacherous Wumpus World labyrinth. This world is fraught with dangers: a gold-guarding Wumpus that can be smelled two squares away, hidden pits that create breezes in adjacent squares, and the tantalizing prospect of gold.

The architecture of the simulation is built using Java, leveraging its robust features for implementing complex AI algorithms. The backend handles the core logic of the Wumpus World, including environment generation, agent decision-making processes, and game state management. The frontend, developed using Java's GUI libraries, provides a visual representation of the Wumpus World, allowing users to observe the agent's decision-making process in real-time.

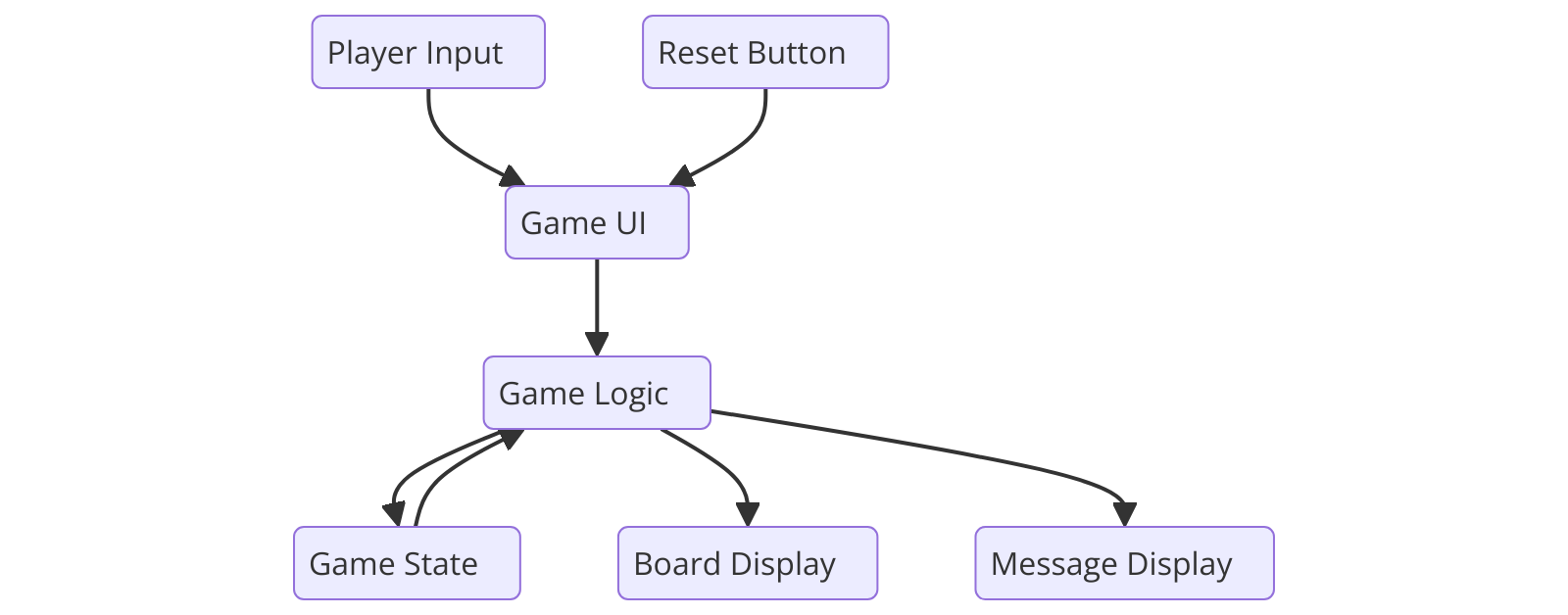
The intelligent agent's decision-making process involves several key components. First, it must perceive its environment through sensors, detecting smells, breezes, and glitter from the gold. Then, it must use this information to update its knowledge base, employing propositional logic to infer the likely locations of dangers and the gold. Finally, it must use this knowledge to plan its next move, balancing the need for exploration with the imperative to avoid danger.

One of the key challenges in developing this simulation was implementing an effective knowledge representation and reasoning system. The agent must maintain a model of the world that accurately reflects its current understanding, updating this model as new information is gathered. This requires a sophisticated system of logical inference, capable of drawing conclusions from both direct observations and indirect clues.

Another significant challenge was designing decision-making algorithms that could operate effectively in an uncertain environment. The agent must often make decisions based on incomplete information, weighing the potential risks and rewards of different actions. This necessitated the implementation of probabilistic reasoning techniques and strategic planning algorithms.

In conclusion, this capstone project demonstrates the application of advanced AI concepts in solving complex problems under uncertainty. The Wumpus World simulation provides a platform for exploring logical reasoning, knowledge representation, and strategic decision-making in a challenging and dynamic environment. By leveraging Java for both the backend logic and frontend visualization, the project showcases the integration of AI algorithms with interactive simulation

# **3. ARCHITECTURE DIAGRAM**



|  |
| --- |
| Figure 3.1 : Architecture Diagram  The architecture diagram of the Wumpus World simulation illustrates the various components and their interactions within the system. This modular architecture ensures scalability, maintainability, and efficient performance in simulating the complex, uncertain environment of the Wumpus World.  ● Overall Structure: The architecture diagram represents the high-level design of the Wumpus World simulation.  ● Components: It consists of several interconnected modules that handle different aspects of the game and AI agent's decision-making process.  ● Game UI: This component handles the user interface, allowing players to interact with the game and visualize the Wumpus World environment.  ● Player Input: Represents the interface for receiving player commands or actions.  ● Reset Button: Allows for resetting the game state to its initial configuration.  ● Game Logic: The core component that implements the rules of Wumpus World, processes player actions, and manages the AI agent's decision-making algorithms.  ● Game State: Maintains the current state of the Wumpus World, including the positions of the agent, Wumpus, pits, and gold.  ● Board Display: Responsible for rendering the visual representation of the Wumpus World grid.  ● Message Display: Provides feedback and information to the player about game events, agent perceptions, and decision-making processes.  This diagram delineates the modular structure of the Wumpus World simulation, emphasizing the separation of concerns to ensure maintainability and scalability. The Game UI serves as the entry point, handling player interactions and game visualization. It communicates with the Game Logic, which is the core component responsible for implementing the Wumpus World rules and the AI agent's decision-making processes.  The Game Logic interacts with multiple sub-components: Game State to keep track of the world's current configuration, Board Display to update the visual representation of the game, and Message Display to provide feedback and insights into the agent's reasoning process.  This architecture allows for the complex interplay between the player's actions, the AI agent's decision-making, and the uncertain environment of the Wumpus World. It facilitates the implementation of advanced AI concepts such as logical reasoning, knowledge representation, and strategic planning within the context of an interactive simulation. |
|  |

# **4. FLOWCHART**

|  |
| --- |
| Figure 4.1 : Flowchart  The flowchart for "Wumpus World: Logic and Strategy in an Uncertain Labyrinth" depicts the step-by-step process of the game flow and decision-making.  ● Initialize Game: The process begins with the game setup, where the Wumpus World environment is created and the player (or AI agent) is positioned.  ● Move Player: The player or AI agent makes a move within the Wumpus World grid.  ● Check Player Position: After each move, the system evaluates the current position of the player.  ● Decision Points: Based on the player's position, four possible scenarios are considered:   * Wumpus: The player encounters the Wumpus. * Pit: The player falls into a pit. * Gold: The player finds the gold. * Safe: The current position is safe.   ● Outcomes:   * Wumpus Encounter: If the player meets the Wumpus, it leads to a Game Over scenario. * Fall into Pit: If the player falls into a pit, it also results in Game Over. * Gold Found: If the player finds the gold, it triggers a You Win condition, successfully completing the game. * Check Surroundings: If the position is safe, the player checks the surrounding cells for clues (e.g., breeze near pits, stench near Wumpus).   ● Display Message: After each move or event, the system provides feedback to the player about their current situation, sensory inputs, or game status.  ● Game Loop: If the game hasn't ended (by winning or losing), the process loops back to the Move Player step, allowing for the next action.  This flowchart visually represents the core gameplay loop of Wumpus World, highlighting the decision-making process and the various outcomes possible in this uncertain environment. It demonstrates how the player or AI agent must constantly evaluate their position, gather information about their surroundings, and make strategic decisions to navigate the dangerous labyrinth in search of gold while avoiding the Wumpus and pits. The iterative nature of the flowchart emphasizes the ongoing challenge and the need for logical reasoning at each step of the game. |
|  |

|  |
| --- |
|  |

# **6. CLASS DIAGRAM**

|  |
| --- |
| **Figure 6.1 : Class Diagram**  The class diagram for "Wumpus World: Logic and Strategy in an Uncertain Labyrinth" specifies the structure of the system by showing the main classes, their attributes, methods, and the relationships among them.  GameState: This class tracks the core game data, including:   * grid: Array (represents the game board) * playerPosition: int * wumpusPosition: int * goldPosition: int * pitPositions: Array * gameOver: boolean   These attributes store the current state of the Wumpus World, including the positions of all elements and whether the game has ended.  Game Logic: This class contains the core game mechanics and AI logic:   * initializeGame() * movePlayer() * checkSurroundings() * endGame() * revealBoard() * getAdjacentCells()   These methods handle the initialization of the game, player movement, environment checks, game termination, board revelation, and retrieval of adjacent cell information for decision-making.  Game UI: This class manages the user interface elements: Attributes:   * board: Element * message: Element * resetButton: Element   Methods:   * renderBoard() * updateMessage()   These components and methods handle the visual representation of the game board, displaying messages to the player, and providing a reset functionality.  The relationships between these classes are crucial for the game's functionality:   * Game Logic inherits from or uses GameState to access and modify the game's current state. * Game UI inherits from or uses Game Logic to execute game actions and update the display based on the current state.   This structure ensures a clear separation of concerns:   * GameState maintains the data model of the Wumpus World. * Game Logic implements the rules and AI strategies for navigating the world. * Game UI handles the presentation and user interaction aspects.   This design facilitates the implementation of complex AI behaviors, allows for easy state management, and provides a clear interface for user interaction, making it well-suited for simulating the uncertain and challenging environment of the Wumpus World. |
|  |

# **7. CODE IMPLEMENTATION**

## **7.1 JAVA CODE**

const board = document.getElementById('game-board');

const message = document.getElementById('message');

const resetButton = document.getElementById('reset-button');

let gameState;

function initializeGame() {

gameState = {

grid: Array(16).fill(null),

playerPosition: 0,

wumpusPosition: Math.floor(Math.random() \* 15) + 1,

goldPosition: Math.floor(Math.random() \* 14) + 2,

pitPositions: [],

gameOver: false

};

// Place pits

while (gameState.pitPositions.length < 3) {

const pitPosition = Math.floor(Math.random() \* 13) + 3;

if (!gameState.pitPositions.includes(pitPosition) &&

pitPosition !== gameState.wumpusPosition &&

pitPosition !== gameState.goldPosition) {

gameState.pitPositions.push(pitPosition);

}

}

renderBoard();

updateMessage("You're in the Wumpus world. Find the gold!");

}

function renderBoard() {

board.innerHTML = '';

for (let i = 0; i < 16; i++) {

const cell = document.createElement('div');

cell.className = 'cell';

cell.dataset.index = i;

cell.addEventListener('click', () => movePlayer(i));

if (i === gameState.playerPosition) {

cell.textContent = '🧑';

cell.classList.add('revealed');

} else if (gameState.grid[i] !== null) {

cell.textContent = gameState.grid[i];

cell.classList.add('revealed');

}

board.appendChild(cell);

}

}

function movePlayer(newPosition) {

if (gameState.gameOver || newPosition === gameState.playerPosition) return;

const oldPosition = gameState.playerPosition;

gameState.playerPosition = newPosition;

if (newPosition === gameState.wumpusPosition) {

endGame("Oh no! You've encountered the Wumpus. Game over!");

} else if (gameState.pitPositions.includes(newPosition)) {

endGame("You've fallen into a pit. Game over!");

} else if (newPosition === gameState.goldPosition) {

endGame("Congratulations! You've found the gold. You win!");

} else {

checkSurroundings();

}

gameState.grid[oldPosition] = null;

renderBoard();

}

function checkSurroundings() {

const adjacentCells = getAdjacentCells(gameState.playerPosition);

let message = "";

if (adjacentCells.includes(gameState.wumpusPosition)) {

message += "You smell something terrible nearby. ";

}

if (adjacentCells.some(cell => gameState.pitPositions.includes(cell))) {

message += "You feel a breeze. ";

}

if (adjacentCells.includes(gameState.goldPosition)) {

message += "You see a glimmer nearby. ";

}

updateMessage(message || "It's quiet here.");

}

function getAdjacentCells(position) {

const row = Math.floor(position / 4);

const col = position % 4;

const adjacent = [];

if (row > 0) adjacent.push(position - 4);

if (row < 3) adjacent.push(position + 4);

if (col > 0) adjacent.push(position - 1);

if (col < 3) adjacent.push(position + 1);

return adjacent;

}

function endGame(msg) {

gameState.gameOver = true;

updateMessage(msg);

revealBoard();

}

function revealBoard() {

gameState.grid[gameState.wumpusPosition] = '👹';

gameState.grid[gameState.goldPosition] = '💰';

gameState.pitPositions.forEach(pit => gameState.grid[pit] = '🕳️');

renderBoard();

}

function updateMessage(msg) {

message.textContent = msg;

}

resetButton.addEventListener('click', initializeGame);

initializeGame();

## **7.2 HTML CODE**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Wumpus World Game</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<div class="game-container">

<h1>Wumpus World</h1>

<div id="game-board"></div>

<div id="message"></div>

<button id="reset-button">Reset Game</button>

</div>

<script src="script.js"></script>

</body>

</html>

## **7.3 CSS CODE**

body {

font-family: 'Arial', sans-serif;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

background: linear-gradient(135deg, #6e45e2, #88d3ce);

color: #333;

}

.game-container {

background-color: rgba(255, 255, 255, 0.9);

border-radius: 15px;

padding: 25px;

box-shadow: 0 10px 30px rgba(0, 0, 0, 0.2);

max-width: 500px;

width: 100%;

}

h1 {

text-align: center;

color: #4a4a4a;

font-size: 2.5em;

margin-bottom: 20px;

text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.1);

}

#game-board {

display: grid;

grid-template-columns: repeat(4, 1fr);

gap: 12px;

margin-bottom: 25px;

background-color: #f0f0f0;

padding: 15px;

border-radius: 10px;

box-shadow: inset 0 0 10px rgba(0, 0, 0, 0.1);

}

.cell {

aspect-ratio: 1;

background: linear-gradient(145deg, #e6e6e6, #ffffff);

display: flex;

justify-content: center;

align-items: center;

font-size: 28px;

cursor: pointer;

transition: all 0.3s ease;

border-radius: 10px;

box-shadow: 5px 5px 10px #d1d1d1, -5px -5px 10px #ffffff;

}

.cell:hover {

transform: translateY(-3px);

box-shadow: 7px 7px 15px #d1d1d1, -7px -7px 15px #ffffff;

}

.cell.revealed {

background: linear-gradient(145deg, #ffffff, #f0f0f0);

box-shadow: inset 3px 3px 7px #d1d1d1, inset -3px -3px 7px #ffffff;

}

#message {

text-align: center;

margin-bottom: 25px;

font-weight: bold;

font-size: 1.2em;

color: #4a4a4a;

padding: 15px;

background-color: #f8f8f8;

border-radius: 8px;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

}

#reset-button {

display: block;

width: 100%;

padding: 12px;

background: linear-gradient(145deg, #ff6b6b, #feca57);

color: white;

border: none;

border-radius: 8px;

cursor: pointer;

font-size: 18px;

font-weight: bold;

text-transform: uppercase;

transition: all 0.3s ease;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

}

#reset-button:hover {

background: linear-gradient(145deg, #feca57, #ff6b6b);

transform: translateY(-2px);

box-shadow: 0 6px 8px rgba(0, 0, 0, 0.15);

}

@media (max-width: 500px) {

.game-container {

padding: 15px;

}

.cell {

font-size: 22px;

}

h1 {

font-size: 2em;

}

#reset-button {

font-size: 16px;

}

}

# **8. OUTPUT SCREENSHOT**

|  |
| --- |
| Figure 8.1 : Game Interface |

|  |
| --- |
| Figure 8.2 : Encountered the WUMPUS ! |

|  |
| --- |
| Figure 8.3 : Terrible and Glimmer , Which means Both WUMPUS and GOLD are at adjacent . |

|  |
| --- |
| Figure 8.4 : Feels Breeze and Glimmer , Which means Pits and Gold are right adjacent to the player ! |

|  |
| --- |
| Figure 8.5 : Ohh no !! Fell into a Pit !! |

|  |
| --- |
| Figure 8.6 : Congratulations you found the GOLD !! |

# **9. CONCLUSION**

This capstone project successfully developed the Wumpus World game using Java, addressing the need for strategic decision-making in a simulated environment.

The game integrates real-time logic-based scenarios, ensuring players can test their problem-solving skills and decision accuracy.

By combining a robust Java backend with a user-friendly HTML and CSS frontend, the project demonstrates the effective use of enterprise-level technologies.

The challenges of maintaining game logic and designing an intuitive interface were met with innovative solutions, resulting in a functional and scalable application.

This project underscores the importance of software development skills in solving practical problems and contributing to the gaming industry.

# **10. REFERENCES**

• Russell, S., & Norvig, P. (2020). Artificial Intelligence: A Modern Approach (4th ed.). Pearson.

• Genesereth, M., & Nilsson, N. (1987). Logical Foundations of Artificial Intelligence. Morgan Kaufmann.

• Poole, D., Mackworth, A., & Goebel, R. (1998). Computational Intelligence: A Logical Approach. Oxford University Press.

• McCarthy, J., & Hayes, P. J. (1969). Some Philosophical Problems from the Standpoint of Artificial Intelligence. Stanford University.

• Green, C. (1969). Application of Theorem Proving to Problem Solving. IJCAI Proceedings.

• Shapiro, S. C. (1992). Artificial Intelligence. Addison-Wesley.

• Dechter, R. (2003). Constraint Processing. Morgan Kaufmann.

• Ghallab, M., Nau, D., & Traverso, P. (2004). Automated Planning: Theory and Practice. Elsevier.

• Laird, J. E. (2012). The Soar Cognitive Architecture. MIT Press.

• Newell, A., & Simon, H. A. (1972). Human Problem Solving. Prentice-Hall.