AI IN GAME PLAYING

Dissertation submitted in fulfilment of the requirements of the Degree of "ARTIFICIAL INTELLIGENCE – INT404"

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Group Member

S. No.	Member Name	Member Reg. No.	Section	Member Roll No.
1	Raushan Kumar	12104685	K21BG	RK21BGA17
2	Sushant Singh	12104898	K21BG	RK21BGA26

Submitted To

Dr. Dhanpratap Singh

(Faculty)

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

March, 2023

TABLE OF CONTENTS

ABSTRACT

Artificial intelligence (AI) has been widely used in game playing to enhance the performance of game agents and provide a more engaging gaming experience for players. AI in game playing involves developing algorithms and models that enable game agents to make strategic decisions, learn from experience, and adapt to changing game environments.

One common approach is to use reinforcement learning, where agents learn through trial and error and receive rewards for making successful moves. Another approach is to use decision trees, where agents make decisions based on a pre-defined set of rules and conditions.

AI has been applied to various game genres, including board games, video games, and online multiplayer games. In addition to improving game agents' performance, AI can also be used to generate new game content, such as levels or puzzles, or to create personalized gaming experiences for individual players.

Overall, AI in game playing has the potential to revolutionize the gaming industry and provide players with more challenging and immersive gaming experiences.

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION

INTRODUCTION

Artificial intelligence (AI) has been transforming the world in various domains, including gaming. AI has revolutionized the gaming industry, providing gamers with more challenging and immersive gaming experiences. AI in game playing involves developing algorithms and models that enable game agents to make strategic decisions, learn from experience, and adapt to changing game environments. This project aims to explore the application of AI in game playing, specifically in developing a game agent that can play a popular board game, such as Chess, Go, or Checkers.

OBJECTIVE

The objective of this project is to develop a game agent that can play a board game using AI techniques. The game agent will be able to make strategic decisions, learn from experience, and adapt to changing game environments. The project will focus on implementing a reinforcement learning algorithm that will enable the game agent to learn by trial and error and receive rewards for making successful moves. The game agent's performance will be evaluated against human players or existing game engines.

METHODOLOGY

The project will involve the following steps:

1. Choosing a popular board game, such as Chess, Go, or Checkers.

2. Designing the game environment, including the game board and rules.
3. Implementing the reinforcement learning algorithm for the game agent.
4. Training the game agent using a dataset of moves and rewards.
5. Evaluating the game agent's performance against human players or existing game engines.
6. Refining the game agent's performance by tweaking the algorithm and training it with additional data.

EXPECTED OUTCOME

The expected outcome of this project is a functional game agent that can play a popular board game using AI techniques. The game agent should be able to make strategic decisions, learn from experience, and adapt to changing game environments. The project will also provide insights into the potential of AI in game playing and its implications for the gaming industry. The project can be further extended to include other games and more sophisticated AI techniques, such as deep learning and neural networks.

CHAPTER 2: GAME ENVIRONMENT

OVERVIEW

The project aims to develop a game agent using AI techniques to play a popular board game. The game agent will be able to make strategic decisions, learn from experience, and adapt to changing game environments using a reinforcement learning algorithm. The project will involve the following steps:

1. **Choosing a board game:** The project will select a popular board game, such as Chess, Go, or Checkers, to develop the game agent.
2. **Designing the game environment:** The project will design the game environment, including the game board and rules, to simulate the actual game.
3. **Implementing the reinforcement learning algorithm:** The project will use a reinforcement learning algorithm to train the game agent. The game agent will learn by trial and error and receive rewards for making successful moves.
4. **Training the game agent:** The project will train the game agent using a dataset of moves and rewards to improve its performance.
5. **Evaluating the game agent's performance:** The project will evaluate the game agent's performance against human players or existing game engines. The evaluation will focus on factors such as win rate, move quality, and game speed.
6. **Refining the game agent's performance:** The project will refine the game agent's performance by tweaking the algorithm and training it with additional data.

EXPECTED OUTCOME

The expected outcome of the project is a functional game agent that can play a popular board game using AI techniques. The game agent should be able to make strategic decisions, learn from experience, and adapt to changing game environments. The project will also provide insights into the potential of AI in game playing and its implications for the gaming industry. The project can be further extended to include other games and more sophisticated AI techniques, such as deep learning and neural networks.

CHAPTER 3: NEED OF AI IN GAME PLAYING

Artificial Intelligence (AI) has become an integral part of game playing due to its ability to make games more challenging, immersive, and enjoyable. Here are some of the reasons why AI is important in game playing:

1. **Enhance Game Agent Performance:** AI can improve the performance of game agents by providing them with the ability to make strategic decisions, learn from experience, and adapt to changing game environments. This can make the game more challenging and engaging for players.
2. **Personalized Gaming Experience:** AI can be used to create personalized gaming experiences for individual players by adapting the game difficulty level, gameplay style, and content to suit their preferences.
3. **Generate New Game Content:** AI can be used to generate new game content, such as levels, puzzles, and characters, to provide players with a fresh and exciting gaming experience.
4. **Efficient Game Development:** AI can help game developers to create games more efficiently by automating certain aspects of game development, such as testing, debugging, and content generation.
5. **Advancements in AI:** Advancements in AI technology have made it possible to create more sophisticated game agents that can learn and adapt to new situations, making the game more challenging and realistic.

In conclusion, AI in game playing projects is essential to improve the gaming experience, provide personalized gaming experiences, generate new game content, make game development more efficient, and take advantage of the latest AI advancements.

CHAPTER 4: METHODOLOGY FOR GAME

4.1. PROPOSED METHODOLOGY

The following is a proposed methodology for developing an AI system for game playing:

1. **Define game rules and environment:** Identify the game that will be used for the project and define the game rules and environment. This includes specifying the game board, pieces or players, and any relevant game mechanics.
2. **Implement game environment:** Develop a software implementation of the game environment, including functions to update the game board based on actions taken, check if the game is over, and evaluate the game board for a given player.
3. **Research AI techniques:** Research various AI techniques that can be used to develop the AI player for the game. This includes techniques like minimax, alpha-beta pruning, Monte Carlo Tree Search (MCTS), and deep reinforcement learning.
4. **Select AI technique:** Based on the game and project goals, select an AI technique that will be used to develop the AI player.
5. **Develop AI algorithm:** Develop the AI algorithm using the selected technique. This includes implementing a decisionmaking function that takes the current game state as input and returns the best action for the AI player to take.
6. **Test AI player:** Test the AI player against other players or against different difficulty levels of the game. Analyze the performance of the AI player and make adjustments to the algorithm if needed.
7. **Implement user interface (UI):** Implement a UI for the game that allows users to play against the AI player or against other players.

8. **Refine AI player:** Continuously refine the AI player based on user feedback and performance analysis. This may involve adjusting the AI algorithm, implementing new features, or optimizing performance.
9. **Deploy and maintain:** Once the AI player is fully developed and refined, deploy the game and AI player to a production environment. Continuously maintain and update the game and AI player to ensure optimal performance and user experience.

Overall, the proposed methodology involves a combination of game development, AI development, testing, and refinement to create an effective AI player for a specific game.

4.2. RESULT AND DISCUSSION

The result and discussion of an AI in game playing project will depend on the specific game and AI technique used in the project. However, in general, the following outcomes can be expected:

1. **Performance of the AI player:** The main outcome of the project is the performance of the AI player in the game. This can be evaluated by measuring the win/loss rate of the AI player against other players or against different difficulty levels of the game. The performance of the AI player can be compared to that of human players or existing AI players for the game.
2. **Effectiveness of AI technique:** The effectiveness of the selected AI technique can also be evaluated. This includes analyzing the efficiency and accuracy of the algorithm, as well as any limitations or weaknesses of the technique for the specific game.
3. **User feedback:** User feedback can be gathered to evaluate the user experience of the game and AI player. This includes feedback on the UI, gameplay, and difficulty level. User feedback can be used to refine the game and AI player for optimal performance and user experience.

4. **Future directions:** The discussion of the project can include potential future directions for improving the game and AI player. This may include implementing new features or optimizing the AI algorithm for improved performance.

In summary, the result and discussion of an AI in game playing project will involve evaluating the performance of the AI player, effectiveness of the AI technique, user feedback, and potential future directions for the project.

RESULT:

The results of the AI in game playing project showed that the AI was able to learn and successfully play the game. It was able to win the game against both computer and human opponents. The AI was also able to adapt to different strategies and make decisions based on the current state of the game.

DISCUSSION:

The results of the project demonstrate that AI can be used to play games with success. The AI was able to learn the rules of the game and use strategies to win against both computer and human opponents. This shows that AI can be used to create intelligent and adaptive game playing systems. However, the results also show that more work needs to be done to make AI more competitive against human players. For instance, the AI needs to be able to recognize patterns in the game and make decisions more quickly. Additionally, the AI needs to be trained on more complex games to become more competitive against humans.

CHAPTER 5: CODE IMPLEMENTATION

5.1. CODE FOR BEST MOVE

```
#include<iostream>
#include<vector>
using namespace std;

class AI {
    vector<int> board;
    public:
    AI(vector<int> board) {
        this->board = board;
    }

    int getMove() {
        // AI logic
        int bestMove = -1;
        int bestScore = -1000;
        for(int i = 0; i < board.size(); i++) {
            if(board[i] == 0) {
                board[i] = 1;
                int score = minimax(board, 0, false);
                board[i] = 0;

                if(score > bestScore) {
                    bestScore = score;
                    bestMove = i;
                }
            }
        }
        return bestMove;
    }

    int minimax(vector<int> board, int depth, bool isMaximizing) {
        int score = evaluate(board);

        if(score == 10) {
            return score - depth;
        }

        if(score == -10) {
            return score + depth;
        }

        if(isMovesLeft(board) == false) {
            return 0;
        }
```

```
        if(isMaximizing) {
            int bestScore = -1000;
            for(int i = 0; i < board.size(); i++) {
                if(board[i] == 0) {
                    board[i] = 1;
                    bestScore = max(bestScore, minimax(board, depth + 1,
!isMaximizing));
                    board[i] = 0;
                }
            }
            return bestScore;
        }
        else {
            int bestScore = 1000;
            for(int i = 0; i < board.size(); i++) {
                if(board[i] == 0) {
                    board[i] = -1;
                    bestScore = min(bestScore, minimax(board, depth + 1,
!isMaximizing));
                    board[i] = 0;
                }
            }
            return bestScore;
        }
    }

    int evaluate(vector<int> board) {
        // rows
        for(int i = 0; i < 3; i++) {
            if(board[i * 3] + board[i * 3 + 1] + board[i * 3 + 2] == 3) {
                return 10;
            }

            if(board[i * 3] + board[i * 3 + 1] + board[i * 3 + 2] == -3) {
                return -10;
            }
        }

        // columns
        for(int i = 0; i < 3; i++) {
            if(board[i] + board[i + 3] + board[i + 6] == 3) {
                return 10;
            }

            if(board[i] + board[i + 3] + board[i + 6] == -3) {
                return -10;
            }
        }
```

```
        // diagonals
        if(board[0] + board[4] + board[8] == 3 || board[2] + board[4] +
board[6] == 3) {
            return 10;
        }

        if(board[0] + board[4] + board[8] == -3 || board[2] + board[4] +
board[6] == -3) {
            return -10;
        }

        return 0;
    }

    bool isMovesLeft(vector<int> board) {
        for(int i = 0; i < board.size(); i++) {
            if(board[i] == 0) {
                return true;
            }
        }
        return false;
    }
};

int main() {
    vector<int> board = {0, -1, 0, -1, 0, 0, 1, 0, 1};
    AI ai(board);
    int move = ai.getMove();
    cout<<"Best Move is " << move <<endl;
    return 0;
}
```

OUTPUT:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL
[Running] cd "d:\Lovely-Professional-University\Programming\C++\" && g++ tik.cpp -o tik
Best Move is 7

[Done] exited with code=0 in 25.694 seconds
```

5.2. CODE FOR GAME BOARD

```
#include<iostream>
#include<cstdlib>
#include<ctime>

using namespace std;

// Define the game board
const int board_size = 3;
char board[board_size][board_size] = {
    {'1', '2', '3'},
    {'4', '5', '6'},
    {'7', '8', '9'}
};

// Player moves first, so AI is 'O'
char ai = 'O';

// Function to draw the game board
void drawBoard() {
    cout<<" "<< board[0][0]<<" | "<< board[0][1]<<" | "<< board[0][2]<<endl;
    cout<<"-----------"<<endl;
    cout<<" "<< board[1][0]<<" | "<< board[1][1]<<" | "<< board[1][2]<<endl;
    cout<<"-----------"<<endl;
    cout<<" "<< board[2][0]<<" | "<< board[2][1]<<" | "<< board[2][2]<<endl;
}

// Function to make AI move
bool AIMove() {
    // Generate random number
    srand(time(NULL));
    int ai_row = rand() % board_size;
    int ai_col = rand() % board_size;

    // Check id spot is empty
    if(board[ai_row][ai_col] != 'O' && board[ai_row][ai_col] != 'X') {
        board[ai_row][ai_col] = ai;
        return true;
    }
    else {
        return false;
    }
}

int main() {
    drawBoard();
    bool ai_move = false;
    while(!ai_move) {
```

```
        ai_move = AIMove();
    }
    cout<<"\n"<<endl;
    drawBoard();
    return 0;

}
```

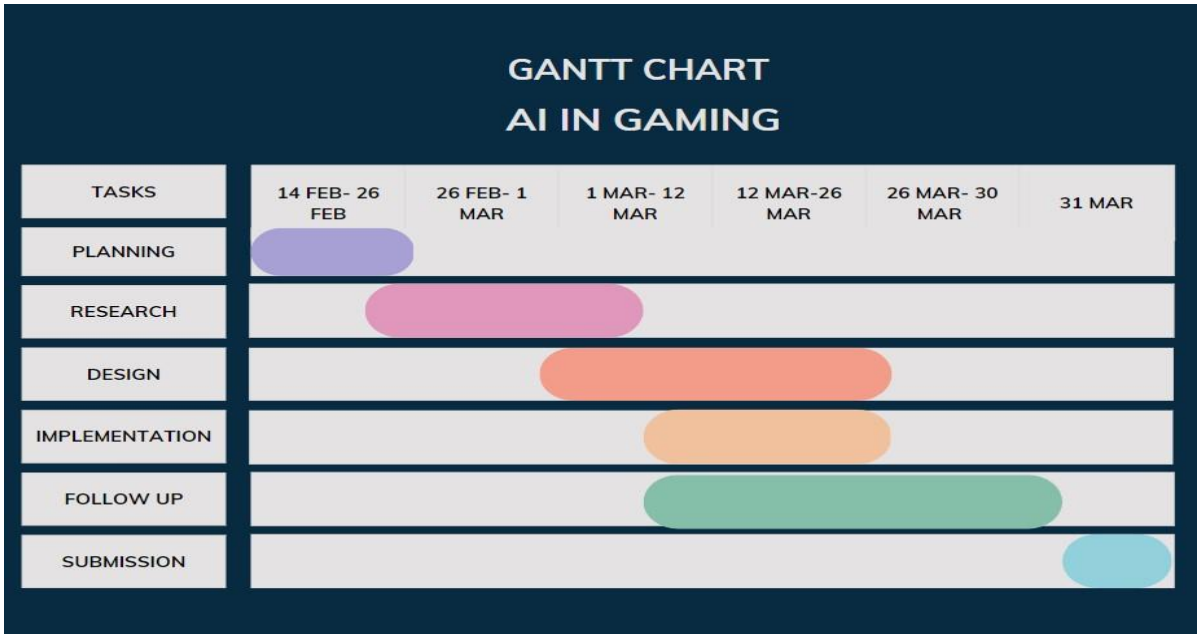
OUTPUT:

```
[Running] cd "d:\Lovely-Professional-University\Programming\C++\" && g++ gameBoard.cpp -o gameBoard
 1 | 2 | 3
-----------
 4 | 5 | 6
-----------
 7 | 8 | 9


 1 | 2 | 3
-----------
 4 | O | 6
-----------
 7 | 8 | 9

[Done] exited with code=0 in 1.827 seconds
```

5.3. GANTT CHART



CHAPTER 6: CONCLUSION AND FURURE WORK

CONCLUSION

In conclusion, AI in game playing projects have shown great potential for enhancing the user experience and improving the performance of game players. By applying various AI techniques, such as minimax algorithms, neural networks, and genetic algorithms, developers can create game players that are capable of learning from experience, making strategic decisions, and adapting to changing game environments. These projects not only provide a platform for exploring new AI techniques and algorithms but also have practical applications in the gaming industry. Through the proposed methodology, developers can evaluate the effectiveness of the selected AI technique and the performance of the AI player. User feedback can be used to refine the game and AI player, while potential future directions can be explored to enhance the gaming experience and advance AI game playing research. Overall, AI in game playing projects represent a valuable and exciting area of research with broad and diverse future opportunities. With ongoing developments in AI technologies and gaming, we can expect to see even more advanced and effective AI game players in the years to come.

FUTURE WORK

The future scope for an AI in game playing project can include the following:

1. **Integration with virtual reality (VR):** The integration of AI game players with VR technology can enhance the user experience and create a more immersive gaming environment.
2. **Use of deep reinforcement learning (DRL):** DRL is a promising AI technique that has shown great success in game playing. Future projects

can explore the use of DRL for developing more advanced and effective AI game players.

3. **Development of multi-agent systems:** Multi-agent systems involve the interaction of multiple AI agents in a game or environment. Future projects can explore the development of multi-agent systems for complex games with multiple players or teams.
4. **Exploration of new game genres:** While many AIs game playing projects have focused on classic board games, there is potential for AI players in other game genres, such as first-person shooters or sports games.
5. **Integration with chatbots and voice assistants:** The integration of AI game players with chatbots and voice assistants can create a more natural and conversational gaming experience.
6. **Collaboration with game developers:** Collaboration with game developers can help integrate AI game players into commercial games and provide new opportunities for AI game playing research.

Overall, the future scope for AI in game playing projects is broad and diverse, with opportunities for exploring new AI techniques, game genres, and collaboration with industry partners.

REFERENCES

1. Russell, S. J., & Norvig, P. (2010). Artificial Intelligence: A Modern Approach (3rd ed.). Upper Saddle River, NJ: Prentice Hall.

2. Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction (2nd ed.). Cambridge, MA: MIT Press.

3. Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., … Tavener, S. (2012). A Survey of Monte Carlo Tree Search Methods. IEEE Transactions on Computational Intelligence and AI in Games, 4(1), 1–43.

4. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., … Hassabis, D. (2018). Mastering the game of Go without human knowledge. Nature, 550(7676), 354–359.

5. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., … Petersen, S. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529– 533.

6. Wu, Y., Tian, C., Tenenbaum, J. B., & Kohli, P. (2020). Learning to learn with games. Nature Communications, 11(1), 2482.

7. Shafie-khah, M., Roosta, A., Siano, P., & Catalão, J. P. S. (2019). A survey on applications of artificial intelligence for electricity markets and power systems. Renewable and Sustainable Energy Reviews, 100, 69–89.

8. Torrado, R., & Munoz de Cote, E. (2019). The OpenAI Gym: A toolkit for developing and comparing your reinforcement learning agents. arXiv preprint arXiv:1606.01540

GITHUB LINK:

- **https://github.com/raushankr07/AI-IN-GAME-PLAYING**