Making Game Using AI

By

Sahil Makvana
17bce053

Darshan Maradiya
17BCE057

DEPARTMENT OF COMPUTER ENGINEERING
Ahmedabad 382481

Making Game Using AI

Minor Project

Submitted in fulfilment of the requirements
For the degree of

Bachelor of Technology in Computer Science

By

Sahil Makvana
17bce053

Darshan Maradiya
17BCE057

Guided By
DR. K P AGRAWAL
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING
Ahmedabad 382481

CERTIFICATE

This is to certify that the project entitled “MAKING GAME USING AI” submitted by SAHIL MAKVANA (17BCE053) and DARSHAN MARADIYA (17BCE057), towards the partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Engineering of Nirma University is the record of work carried out by him/her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination.

Dr. K P Agrawal
Associate Professor
Department of CSE,
Institute of Technology,
Nirma University,
Ahmedabad

Dr. Madhuri Bhavsar
HOD
Dept. of CSE
Institute of Technology,
Nirma University,
Ahmedabad

ACKNOWLEDGEMENT

We would like to express my deepest appreciation to all those who provided me the possibility to complete this Minor Project Report. We acknowledge with thanks, the support rendered by Dr. K P Agrawal, under whose aegis we were able to complete the task in a given period. We also appreciate the constructive suggestions given by our friends to enhance the content of the report.

At the home front, we are extremely grateful to our family members for the support and encouragement, I got from them in completing the report

ABSTRACT

AI performs a major role in games nowadays in various ways. Chess which is one of the most popular and oldest board games, with two opponents playing on a chess board, generally designed in different colors of black and white. White begins, players alternate based on uniform rules, and each player forces the main opponent's chessman King to confirm him. The techniques and algorithms are also developed so that the machine can also play games like chess cleverly. This report discusses a few such techniques and algorithms and their implementation in Unity 3D Game Engine.

CONTENTS

Contents

CHAPTER:1 INTRODUCTION

Artificial intelligence is simply defined as intelligence performed by computers. AI separates itself from the learning styles of animals and humans. Specifically, artificial intelligence begins with the programming of behaviour. In game artificial intelligence (AI) is often needed for the interaction of the user, usually as a force against the player. There are some scenarios where the AI is there to help, and others where it's both fighting and helping you. In the domain of AI, there are many traditional and also machine learning algorithms. We worked on chess. We used unity 3D for making chess with AI implemented in this project. Unity is an extremely streamlined and impressive game engine, and they always strive to make game development more straightforward and seamless.

CHAPTER: 2 LITERATURE SURVEY

2.1 What *is* Game AI?

Game AI focuses primarily on the work that an entity needs to perform in its current situation. In traditional AI literature, this is called a "smart agent," and agents are generally game characters, but also vehicles, robots, or, in some cases, abstracts of the entire group. entity. After all, it's about the environment observing, making decisions accordingly, and acting on it. This is sometimes referred to as the sense / thought / action cycle.

- Sense: The agent detects - or is told about - things in their environment that may influence their behavior (e.g. threats nearby, points of interest to investigate)
- Think: The agent makes a decision about what to do in response (e.g. considers whether it is safe enough to make a higher winning valued move)
- Act: The agent performs actions to put the previous decision into motion (e.g. starts moving along a path towards the enemy or towards the item, etc)

In real-world AI problems, especially the ones making the information at any given instance, they are typically heavily focused on the 'sense' part of this cycle. This is generally done through some sort of machine learning, which is especially suitable for capturing a lot of noisy real data and extracting and understanding semantic information.

Games are unique in that they generally do not require a complex system to extract this information. As a result, the "sense" part of the cycle is often much simpler, and the complexity comes from the application of "thinking" and "act".

2.2 Constraints of Game AI development

Gaming AI generally has some limitations, including:

- In general, these are not "pre-trained" like machine learning algorithms. Creating a neural network under development and learning the best way is not practical. This game is generally more fun and challenging than "optimal".
- Agents can be educated to look for the best approach for a person, but designers don't really want to. It is done in real time. This means that the algorithm cannot monopolize the use of the processor for a long time to make decisions in this context.
- Most games are only between 16ms and 33ms to perform all the processing for the next graphic frame, so even a short decision of 10ms is too long.
- At least part of the system is not hard-coded, and in the case of databases, ideally it allows non-encoders to make adjustments more quickly.

With this in mind, we can look at a very simple AI approach that efficiently handles the entire sensory / thinking / act cycle.

2.3 Introduction to Chess

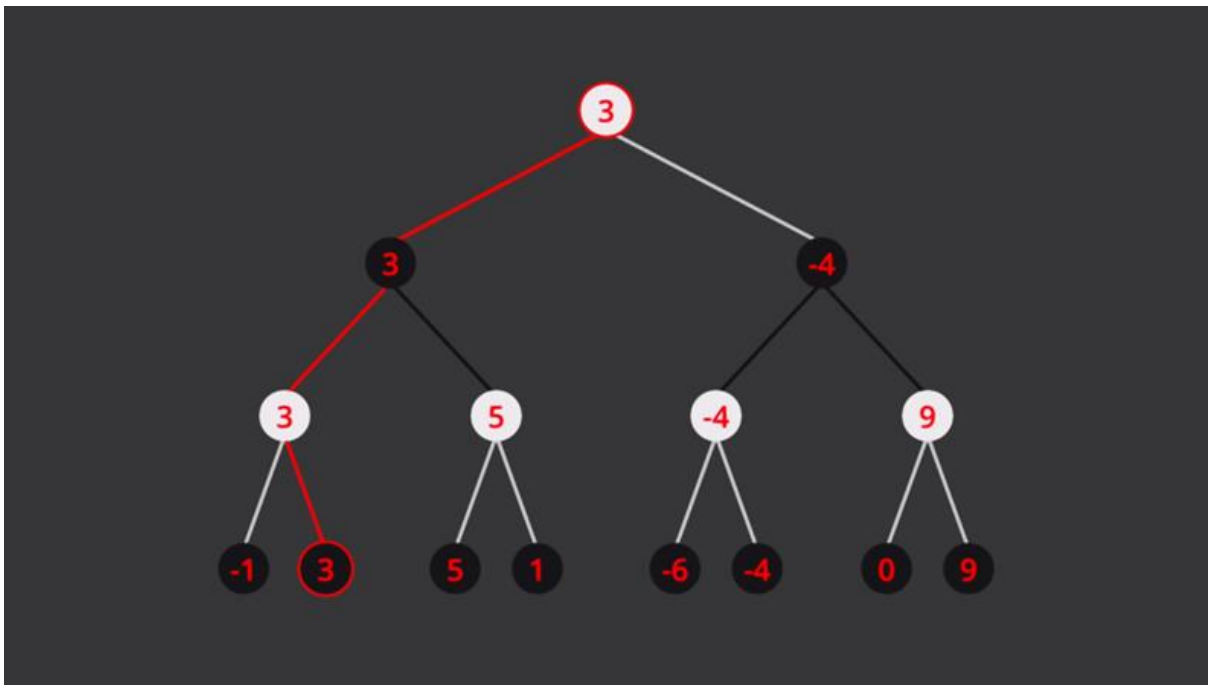
Chess is a game with all the details. This means that all game-related details are known to both players and there are no secrets or opportunities. The two players can see all the positions of the chessboard, and the movements do not proceed at the same time, but in the correct order, so they can know each other's movements. Chess is therefore a fully pointed game, a complete environment for testing artificial intelligence algorithms. Most researchers once concluded that the greatest asset of a computer is computational speed. The computer uses a variant of the brute force routine that is almost entirely dependent on the speed at which the computer can process the card. Chess has a small board. It is made of 64 cells and up to 32 pieces. Although there is an impressive number of legal chess positions, Researchers relied on the fact that computers can process more accurately and faster than humans. Therefore, you can "think" faster than humans can handle. In the old days, Of course, computers at the time had very low memory and computing power, so chess programs were very much about at best. But computers have become faster, and chess programs have become more powerful and smarter. The combined research of many good programmers has made it the most important part of the Minimax game tree chess program. Almost all chess programs use the Minimax tree in any way, the most popular artificial intelligence algorithm in the history of board game coding.

CHAPTER: 3 DECISION-MAKING ALGORITHMS

3.1 MINI_MAX ALGORITHM

At the heart of chess technology is the minimum-maximum local search of the game room. This technique tries to minimize your opponent's points and maximize your points. Check all possible movements at each depth (or called "ply" in computer tree terminology) and use static map scoring routines to determine leaf points. These points are distributed throughout the tree and you can choose the best movement at all depths. The larger the tree, the better decisions you can make (because the algorithm can see more progress). The branching factor is typically 25-40 per node (35 on average).

Following shows the example of such a search tree of the minimax algorithm. We can see that the algorithm finds a path with the highest possibility of winning the game (with heuristic value 3).



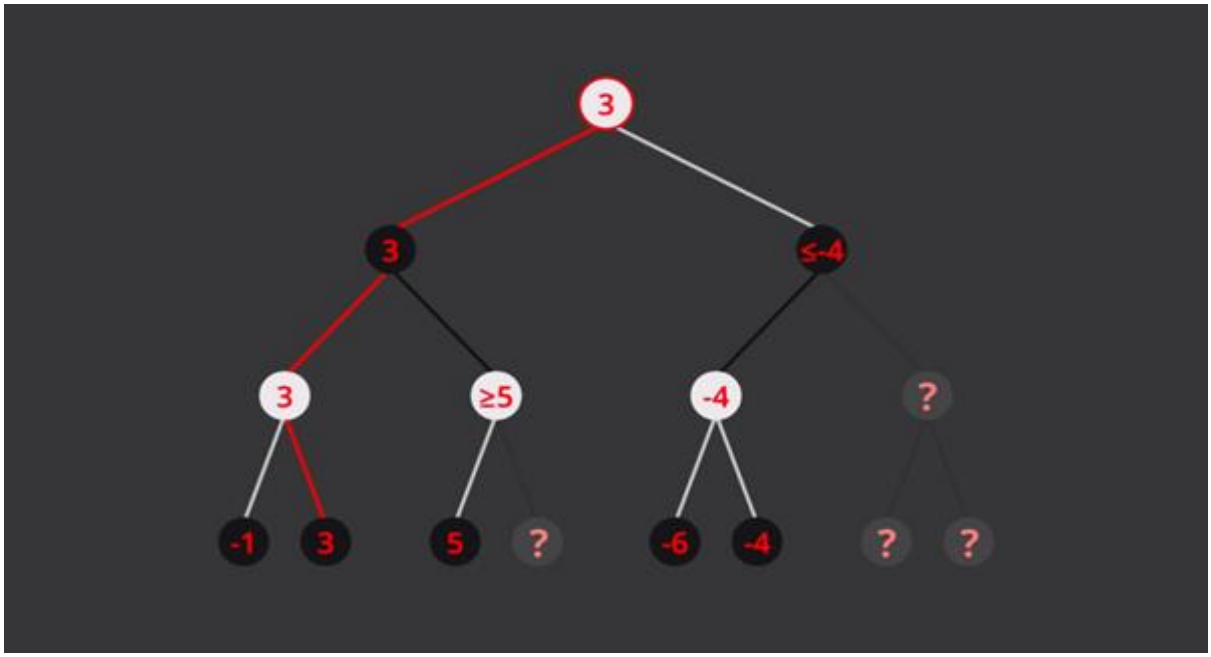
Following is the basic algorithm for minimax

```
function minimax(position, depth, maximizingPlayer)
    if depth == 0 or game over in position
        return static evaluation of position

    if maximizingPlayer
        maxEval = -infinity
        for each child of position
            eval = minimax(child, depth - 1, false)
            maxEval = max(maxEval, eval)
        return maxEval

    else
        minEval = +infinity
        for each child of position
            eval = minimax(child, depth - 1, true)
            minEval = min(minEval, eval)
        return minEval
```

3.2 ALPHA-BETA PRUNING

This common cleaning routine is used to significantly reduce the time of minimax search space. Basically, you can track the worst and best behaviors of each player and use these behaviors to completely avoid the branches that can give you worse results. This cut provides the same exact movement as using the mini-max (that is, no loss of accuracy). Ideally, you can double the depth of your search tree without increasing search time. Sorting can be performed by checking every possible move point to a single expected position. The most intuitive way would be to rank the highest and lowest, but not always the best. Most movements are small wins and losses. Therefore, the sorting is based on the absolute value of the movement point (the smallest one is displayed first). The minimum-maximum average branching factor in chess is about 35, but with pruning and alpha-beta sorting, the program gets about 25 branching factors.

Following shows the example of an alpha-beta pruning algorithm where the unnecessary branches are avoided from searching.



Following is the general algorithm for alpha-beta pruning.

```
function minimax(position, depth, alpha, beta, maximizingPlayer)
   if depth == 0 or game over in position
      return static evaluation of position

   if maximizingPlayer
      maxEval = -infinity
      for each child of position
         eval = minimax(child, depth - 1, alpha, beta, false)
         maxEval = max(maxEval, eval)
      return maxEval

   else
      minEval = +infinity
      for each child of position
         eval = minimax(child, depth - 1, alpha, beta, true)
         minEval = min(minEval, eval)
      return minEval
```

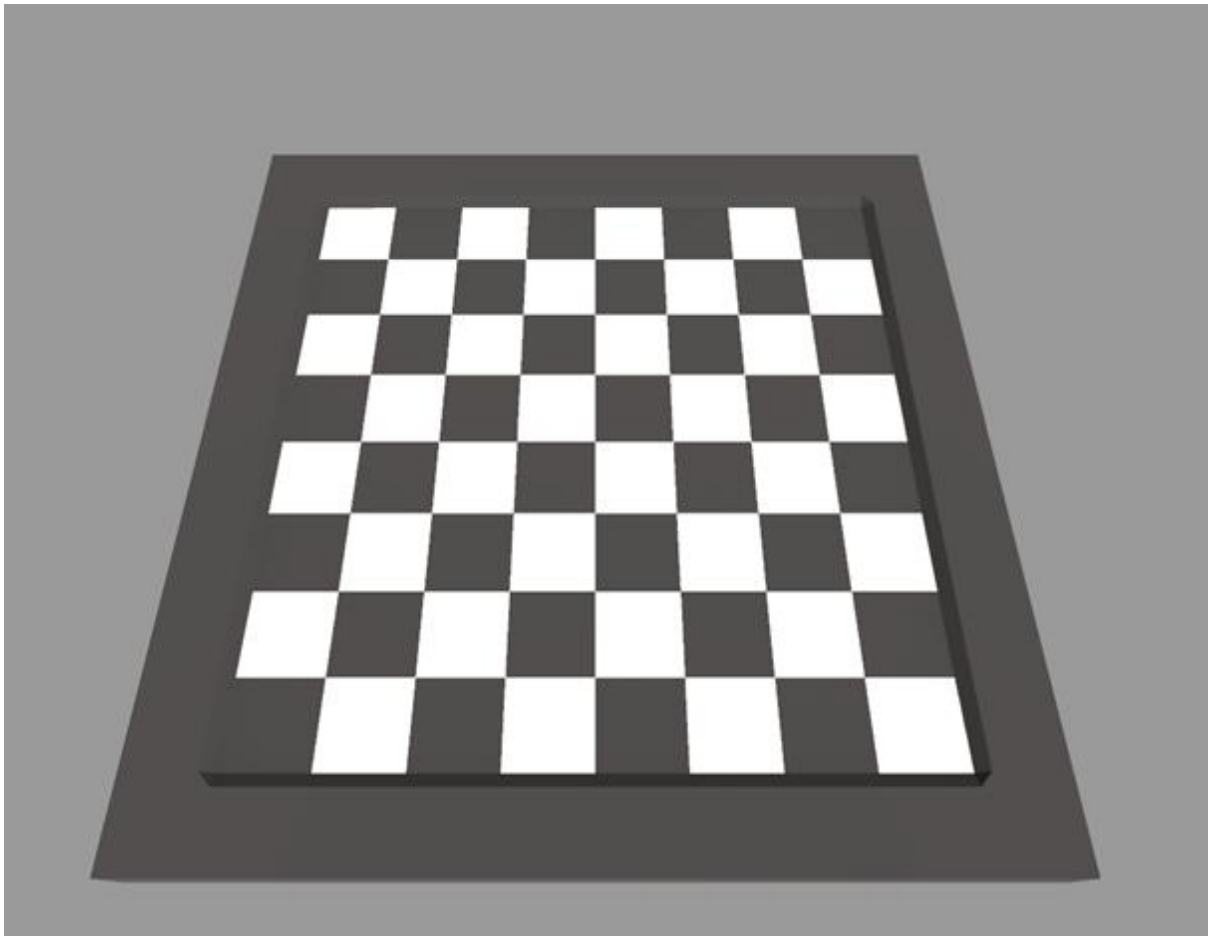
CHAPTER: 4 BUILDING 3D CHESS USING AI IN UNITY

4.1 Used Software and Language

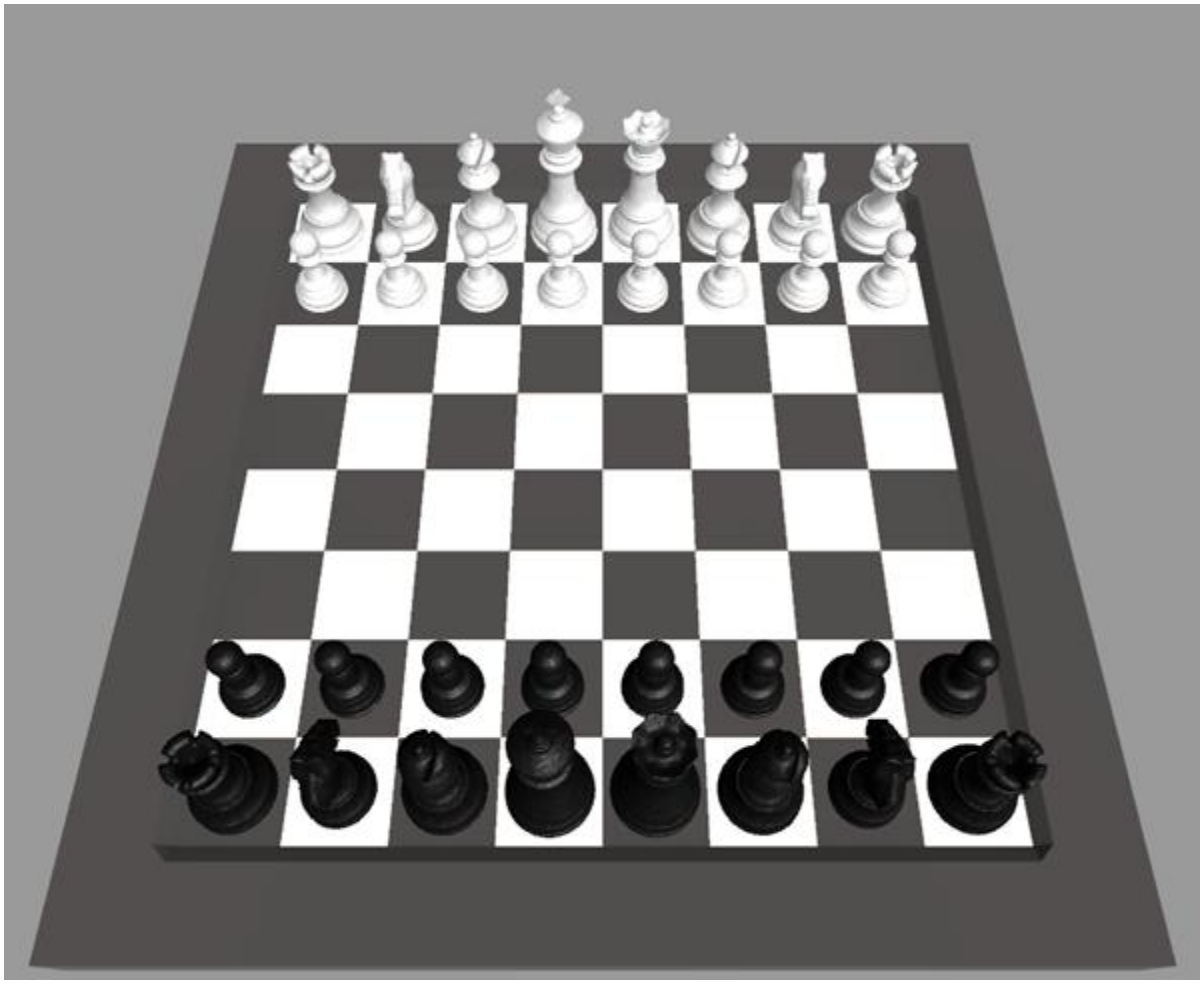
Software: Unity 3D Game Engine
Scripting Language: C#

4.2 Objects of the Game Environment

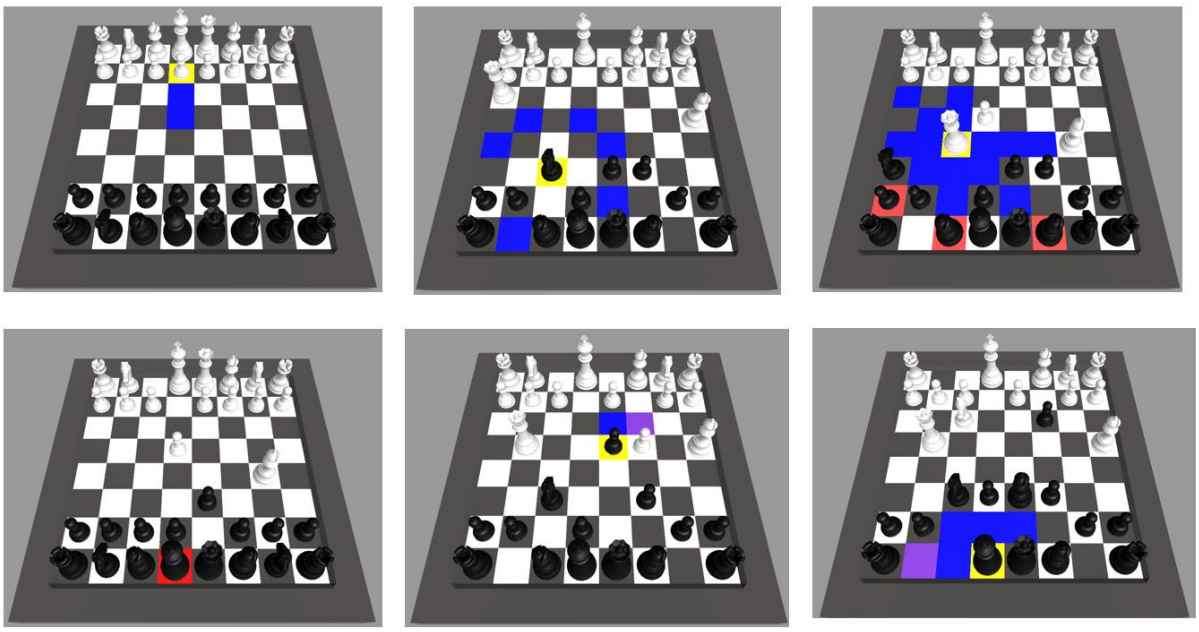
Necessary Scene Objects are created, which includes the following:
ChessBoard



All the Black and White Chessmen

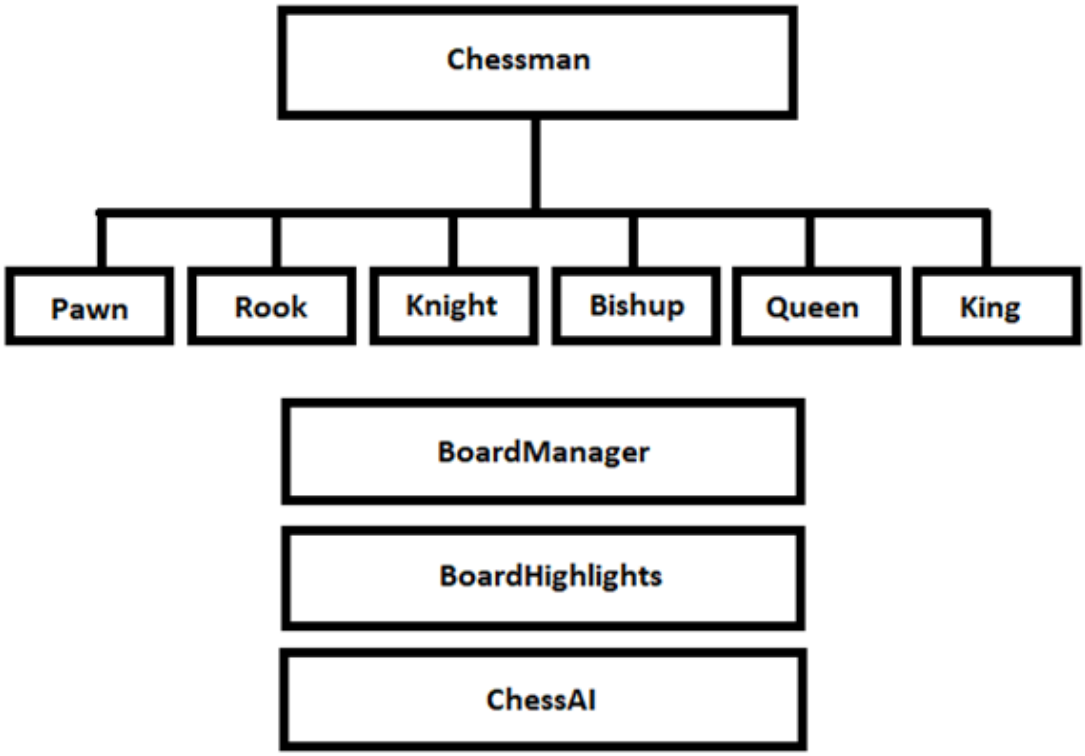


Yellow tile to indicate selected Chessman
Blue tile to indicate empty tile where the selected chessman is allowed to move
Light Red tile to indicate tile where the selected chessman can move and capture the opponent's chessman
Purple tile to indicate EnPassant move or Castling move
Red tile to indicate one of the King is in Check



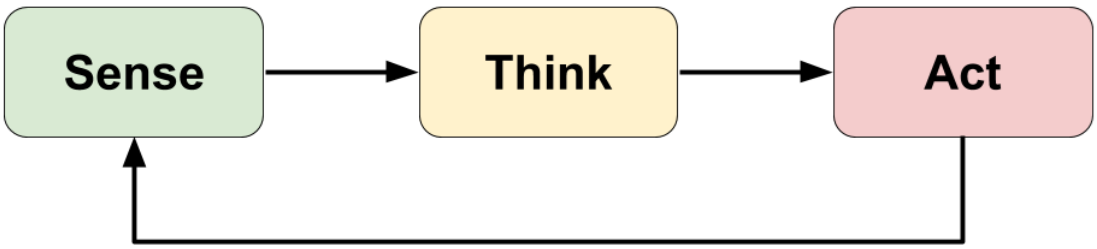
4.3 Class Hierarchy

Following are the classes used to make the game of chess. BoardManger.cs script is the heart of the game which manages the entire game. Chess AI performs a major role in playing from the computer side, which is the main motive of the project.



CHAPTER: 5 CHESS AI

General AI Agent runs on the following Cycle as discussed earlier.



The implementation of Chess AI also shows a similar cycle.

5.1 SENSE

In sensing the environment in the case of chess, the following thing is being sensed and cloned by Chess AI in the sensing part when it is Computer's turn

- [8 * 8] Chessmans Matrix which stores the reference of the objects of class Chessman
- References to WhiteKing, BlackKing, WhiteRooks, BlackRooks for Castling move
- EnPassant Matrix which store coordinates of the latest available EnPassant move position
- ActiveChessmans List which stores references to the chessman present currently on the board

Chess AI senses the above-mentioned things, saves separate references to them, and makes clones of each of the things so that it can manipulate it while thinking the next move.

```
// New State History Stack
History = new Stack< State>();

/* -------------------- Sense -------------------- */

ActualChessmansReference = BoardManager.Instance.Chessmans;
ActualWhiteKing = BoardManager.Instance.WhiteKing;
ActualBlackKing = BoardManager.Instance.BlackKing;
ActualWhiteRook1 = BoardManager.Instance.WhiteRook1;
ActualWhiteRook2 = BoardManager.Instance.WhiteRook2;
ActualBlackRook1 = BoardManager.Instance.BlackRook1;
ActualBlackRook2 = BoardManager.Instance.BlackRook2;
ActualEnPassant = BoardManager.Instance.EnPassant;

ActiveChessmans = new List<Chessman>();
Chessmans = new Chessman[8, 8];

for(int x=0; x<8; x++)
    for(int y=0; y<8; y++)
    {
        if(ActualChessmansReference[x, y] != null)
        {
            Chessman currChessman = ActualChessmansReference[x, y].Clone();
            ActiveChessmans.Add(currChessman);
            Chessmans[x, y] = currChessman;
        }
        else
        {
            Chessmans[x, y] = null;
        }
    }

Shuffle(ActiveChessmans);

EnPassant = new int[2]{ActualEnPassant[0], ActualEnPassant[0]};
```

5.2 THINK

In thinking Part,

- The Actual Environment is replaced by the clone Environment so that Chess AI will play several moves in the cloned environment at the backend and the Original Environment will not be affected.
- The function call to Think() which is simply choosing depth and calling the MiniMax() written for Chess AI and Determining move
- Restoring the Original Environment and Making the move

```
/* ---------------------- Think ---------------------- */

// Critical Part:
// We are about to change the heart of the Game Management which is taken care by BoardManager script
// And that is Chessmans[,] array located in BoardManager script
// We need to change it because in some functions which are declared in this script, which will think of NPC's next move,
// Are using some other class functions also. These other class functions are using BoardManager.Instance.Chessmans array
// As it is storing pointers to all the chessmans present on the board at some position.
// Hence already stored reference to the actual Chessmans[,] array and now assigning it the different clone we made
// (Same goes for all Kings and Rooks, EnPassant move array)
BoardManager.Instance.Chessmans = Chessmans;
BoardManager.Instance.WhiteKing = Chessmans[ActualWhiteKing.currentX, ActualWhiteKing.currentY];
BoardManager.Instance.BlackKing = Chessmans[ActualBlackKing.currentX, ActualBlackKing.currentY];
if(ActualWhiteRook1 != null) BoardManager.Instance.WhiteRook1 = Chessmans[ActualWhiteRook1.currentX, ActualWhiteRook1.currentY];
if(ActualWhiteRook2 != null) BoardManager.Instance.WhiteRook2 = Chessmans[ActualWhiteRook2.currentX, ActualWhiteRook2.currentY];
if(ActualBlackRook1 != null) BoardManager.Instance.BlackRook1 = Chessmans[ActualBlackRook1.currentX, ActualBlackRook1.currentY];
if(ActualBlackRook2 != null) BoardManager.Instance.BlackRook2 = Chessmans[ActualBlackRook2.currentX, ActualBlackRook2.currentY];
BoardManager.Instance.EnPassant = EnPassant;

// Think for which favourable move to make
Think();

// Restoring the Chessmans[,] array
BoardManager.Instance.Chessmans = ActualChessmansReference;
BoardManager.Instance.WhiteKing = ActualWhiteKing;
BoardManager.Instance.BlackKing = ActualBlackKing;
BoardManager.Instance.WhiteRook1 = ActualWhiteRook1;
BoardManager.Instance.WhiteRook2 = ActualWhiteRook2;
BoardManager.Instance.BlackRook1 = ActualBlackRook1;
BoardManager.Instance.BlackRook2 = ActualBlackRook2;
BoardManager.Instance.EnPassant = ActualEnPassant;
```

The Think() function implements the Minimax or Alphabeta Algorithm.
Following is the pseudocode of MiniMax() for Chess AI

5.3 ACT

In Act part,

- ChessAI sets the variable “SelectedChessman” and by calling the function MoveChessman() from BoardManger.cs, it is making its finally determined move.

```
/* ---------------------- Act ---------------------- */
// For most favourable move
// select chessman
Debug.Log(NPCSelectedChessman.GetType() + " to (" + moveX + ", " + moveY + ") " + winningValue + "\n"); // remove this line
BoardManager.Instance.SelectedChessman = BoardManager.Instance.Chessmans[NPCSelectedChessman.currentX, NPCSelectedChessman.currentY];
BoardManager.Instance.allowedMoves = BoardManager.Instance.SelectedChessman.PossibleMoves();

// Debug.Log("Moving");
// move chessman
BoardManager.Instance.MoveChessman(moveX, moveY);

// Stop the StopWatch
stopwatch.Stop();
totalTime += stopwatch.ElapsedMilliseconds;
totalRun++;

// Update average response time
averageResponseTime = totalTime / totalRun;
```

Following is the pseudocode of MiniMax() for Chess AI

Minimax.cs

```
int MiniMax(depth, isMax)
{
    // If max depth is reached or Game is Over
    if(depth == 0 || isGameOver())
    {
        // Static Evaluation Function
        int value = StaticEvaluationFunction();
        return value;
    }
    if(isMax)
    {
        MaxValue = System.Int32.MinValue;
        foreach(NPCchessman in ActiveChessmans)
        {
            foreach(position of allowedmoves(NPCchessman))
            {
                Move(NPCchessman, position, depth);

                int thisMoveValue = MiniMax(depth-1, !isMax);
                if(MaxValue < thisMoveValue)
                {
                    MaxValue = thisMoveValue;
                    SelectedChessman = NPCchessman;
                    SelectedMovePosition = position;
                }

                Undo(depth);
            }
        }
        return MaxValue;
    }
    else
    {
        MaxValue = System.Int32.MaxValue;
        foreach(PlayerChessman in ActiveChessmans)
        {
            foreach(position of allowedmoves(PlayerChessman))
            {
                Move(PlayerChessman, position, depth);

                int thisMoveValue = MiniMax(depth-1, !isMax);
                if(MinValue > thisMoveValue)
                {
                    MinValue = thisMoveValue;
                    SelectedChessman = PlayerChessman;
                    SelectedMovePosition = position;
                }

                Undo(depth);
            }
        }
        return MinValue;
    }
}
```

Following is the pseudocode of AlphaBeta() for Chess AI

AlhpaBeta.cs

```
{
    // If max depth is reached or Game is Over
    if(depth == 0 || isGameOver())
    {
        // Static Evaluation Function
        int value = StaticEvaluationFunction();
        return value;
    }
    if(isMax)
    {
        MaxValue = System.Int32.MinValue;
        foreach(NPCchessman in ActiveChessmans)
        {
            foreach(position of allowedmoves(NPCchessman))
            {
                Move(NPCchessman, position, depth);

                int thisMoveValue = MiniMax(depth-1, !isMax);
                if(MaxValue < thisMoveValue)
                {
                    MaxValue = thisMoveValue;
                    SelectedChessman = NPCchessman;
                    SelectedMovePosition = position;
                }
                Undo(depth);
                if(thisMoveValue > alpha) alpha = thisMoveValue;
                if(beta <= alpha) break;
            }
        }
        return MaxValue;
    }
    else
    {
        MaxValue = System.Int32.MaxValue;
        foreach(PlayerChessman in ActiveChessmans)
        {
            foreach(position of allowedmoves(PlayerChessman))
            {
                Move(PlayerChessman, position, depth);

                int thisMoveValue = MiniMax(depth-1, !isMax);
                if(MinValue > thisMoveValue)
                {
                    MinValue = thisMoveValue;
                }
                Undo(depth);
                if(thisMoveValue < beta) beta = thisMoveValue;
                if(beta <= alpha) break;
            }
        }
        return MinValue;
    }
}
```

5.4 Static Evaluation Function

The static evaluation function used in the minimax and alpha-beta algorithm simply sums up the values decided for the pieces that are present on the board. Following are the values that are taken in use.

King	900
Queen	90
Rook	50
Bishup	30
Knight	30
Pawn	10

The following is the code for Static Evaluation Function.

```
// This function is simply calculating the summation of Chessman values
private int StaticEvaluationFunction()
{
    int TotalScore = 0;
    int curr = 0;
    foreach(Chessman chessman in ActiveChessmans)
    {
        if(chessman.GetType() == typeof(King))
            curr = 900;
        if(chessman.GetType() == typeof(Queen))
            curr = 90;
        if(chessman.GetType() == typeof(Rook))
            curr = 50;
        if(chessman.GetType() == typeof(Bishup))
            curr = 30;
        if(chessman.GetType() == typeof(Knight))
            curr = 30;
        if(chessman.GetType() == typeof(Pawn))
            curr = 10;

        if(chessman.isWhite)
            TotalScore -= curr;
        else
            TotalScore += curr;
    }
    return TotalScore;
}

// Checking for checkmate (Game Over)
private bool isGameOver()
{
    // To be implemented
    int currScore = StaticEvaluationFunction();
    if((currScore < -290 ) || (currScore > 290))
        return true;
    return false;
}
```

5.5 Performance :

The observed performance is as follows.
In the Minimax Algorithm, with a depth equal to 4 the average response time for the AI is 2.4 to 3.7 seconds. From depth equal to 5 the minimax algorithm stops working.
In the AlphaBeta Algorithm, with a depth equal to 4 the average response time for AI is 1.2 to 2.6 seconds. With depth equal to 5 the algorithm still works but gets significantly slower and is not working from 6 onwards.

CONCLUSION

However, the exploration of chess artificial intelligence is almost complete. But, the chess system did not lose its usefulness. There are many other artificial intelligence methods that can be used and tested in chess systems. A neural network, the interaction between genetic technology and computers. Chess is a controlled environment in which a computer is presented with state and goals, and the computer looks for ways to achieve those goals and makes decisions. Therefore, areas of decision-making techniques and other artificial intelligence can occasionally be found in chess programs. So while the chess system looks like it's been completely explored, it's a great literary work. Repeated multiple times by researchers and programmers, it continues to produce many treasures. Artificial intelligence offers great benefits.

REFERENCES

1. https://www.aihorizon.com/essays/chessai/intro.htm
2. https://www.researchgate.net/publication/220814778_The_Role_of_Chess_in_Artificial_Intelligence_Research
3. https://www.semanticscholar.org/paper/The-Role-of-Chess-in-Artificial-Intelligence-Levinson-Hsu/fac0dd639b47514c89efa0393c3b6c29a1d408e6
4. https://www.researchgate.net/publication/319390201_APPLYING_ALPHA-BETA_ALGORITHM_IN_A_CHESS_ENGINE
5. https://umm-csci.github.io/senior-seminar/seminars/spring2017/marckel.pdf

Originality report

COURSE NAME
Minor

STUDENT NAME
MARADIYA DARSHAN DINESHKUMAR

FILE NAME
Minor Project Report

REPORT CREATED
Nov 28, 2020

Summary

Flagged passages	9	10%
Cited/quoted passages	1	3%

Web matches

gamedev.net	3	6%
coursehero.com	2	3%
gamedesigning.org	2	2%
uptownautospa.com	1	0.9%
britannica.com	1	0.7%
therivercommunity.org	1	0.6%

1 of 10 passages

Student passage FLAGGED

This is to certify that the project entitled “MAKING GAME USING AI” **submitted by** SAHIL MAKVANA (17BCE053) **and** DARSHAN MARADIYA (17BCE057), **towards the partial fulfillment of the requirements for the**…

Top web match

CERTIFICATE **This is to certify that the project** entitled “SCHOOL MANAGEMENT SYSTEM” **submitted by** VINYA DENGRE(17BCE134) **and** KARAN SHAH(17BCE109), **towards the partial fulfillment of the requirements**…

17BCE109_17BCE134_School_Management_System_MiniProject ... https://www.coursehero.com/file/62495954/17BCE109-17BCE134-School-Management-System-MiniProjectdocx/

2 of 10 passages

Student passage FLAGGED

We **would like to express my deepest appreciation to all those who provided me the possibility to complete this** Minor Project **Report**. We acknowledge with thanks, the support rendered by Dr.

Top web match

May 03 2013 sample of acknowledgement for report May 3 2013 Admin Acknowledgement Acknowledgement sample for assignment Acknowledgement sample for project Thesis acknowledgement sample I **would like to**…

Acknowledgement for group project report - Uptown Auto Spa https://uptownautospa.com/reflective-fabric/acknowledgement-for-group-project-report.html

---

3 of 10 passages

Student passage FLAGGED

At the home front, we are **extremely** grateful **to** our **family members for the support and encouragement, I got from them in completing the report**

Top web match

I also appreciate them and am thankful to all my friends and colleagues who helped me to further enhance my content of report. I am **extremely** thankful **to** my **family members for the support and**…

17BCE109_17BCE134_School_Management_System_MiniProject ... https://www.coursehero.com/file/62495954/17BCE109-17BCE134-School-Management-System-MiniProjectdocx/

---

4 of 10 passages

Student passage FLAGGED

Chess which is **one of the most popular** and oldest **board games**, with **two opponents** playing **on a** chess **board**, generally **designed** in different colors **of black**

Top web match

Chess, one of the oldest and **most popular board games**, played by **two opponents on a** checkered **board** with specially **designed** pieces **of** contrasting colours, commonly white and **black**.

Chess | game | Britannica https://www.britannica.com/topic/chess

---

5 of 10 passages

Student passage FLAGGED

…is simply defined as intelligence performed by computers. AI **separates itself from the learning styles of animals and humans**. **Specifically, artificial intelligence begins with the programming of**

Top web match

Artificial intelligence is simply defined as intelligence displayed by machines. It's its own category, as it **separates itself from the learning styles of animals and humans**. **Specifically, artificial**…

How AI in Unity is Revolutionizing Video Game Design https://www.gamedesigning.org/learn/unity-ai/

---

6 of 10 passages

Student passage FLAGGED

In game artificial intelligence (AI) **is often needed** for the interaction of **the user, usually as a force against the player**. There are some scenarios where the AI is there…

Top web match

Jan 11 2018 In video games Artificial Intelligence AI usually refers to how a non player character makes decisions. ... **In** any given **game artificial** intelligence AI **is often needed** to interact with …

How to make an ai that plays games - The River Community http://therivercommunity.org/stat-200/how-to-make-an-ai-that-plays-games.html

---

7 of 10 passages

Student passage FLAGGED

…for making chess with AI implemented in this project. **Unity is an extremely streamlined and impressive game engine, and they always strive to make game development more straightforward and seamless**.

Top web match

Unity is an extremely streamlined and impressive game engine, and they always strive to make game development more straightforward and seamless.

How AI in Unity is Revolutionizing Video Game Design https://www.gamedesigning.org/learn/unity-ai/

---

8 of 10 passages

Student passage CITED

Sense: The agent detects - or is told about - things in their environment that may influence their behavior (e.g. **threats nearby, points of interest to investigate**)**Think: The agent makes a decision**…

Top web match

Sense: The agent detects - or is told about - things in their environment that may influence their behaviour (e.g. **threats nearby**, items to collect, **points of interest to investigate**) **Think: The agent**…

The Total Beginner's Guide to Game AI - Artificial Intelligence ... https://www.gamedev.net/tutorials/programming/artificial-intelligence/the-total-beginners-guide-to-game-ai-r4942/

---

9 of 10 passages

Student passage FLAGGED

In real-world AI problems, especially the ones making the information **at** any given instance, **they are typically heavily focused on the 'sense' part of this cycle**. **This is** generally **done** through **some**…

Top web match

In real-world AI problems, especially the ones making the news **at** the moment, **they are typically heavily focused on the 'sense' part of this cycle**…**This is** usually **done** by **some sort of machine**…

The Total Beginner's Guide to Game AI - Artificial Intelligence
... https://www.gamedev.net/tutorials/programming/artificial-intelligence/the-total-beginners-guide-to-game-ai-r4942/

---

10 of 10 passages

Student passage FLAGGED

…a long time to make decisions in this context. **Most games** are **only between** 16ms **and** 33ms **to perform all the processing for the next** graphic **frame**, so even a short decision **of**

Top web match

It needs to run in 'real-time', which in this context means that the algorithm can't monopolise CPU usage for a long time in order to make the decision. Even taking just 10 milliseconds to make a…

The Total Beginner's Guide to Game AI - Artificial Intelligence
... https://www.gamedev.net/tutorials/programming/artificial-intelligence/the-total-beginners-guide-to-game-ai-r4942/
