CS512- Computer Games Design

Assignment #3

30-Nov-2022

Student #1 Name: Sateesh Katnam (002343783)

Student #2 Name: Gabriel Satwik Manikonda (002341846)

Content:

#1 Main Menu:

The Main menu of the game has three options

i) Double Player: Two users can play the game where the first player will be given as white and the other player will be given black.

ii) Player Vs AI: In this case the player can select the level of difficulty of AI by using a slider. The player will be given white and AI will play for black using MiniMax algorithm.

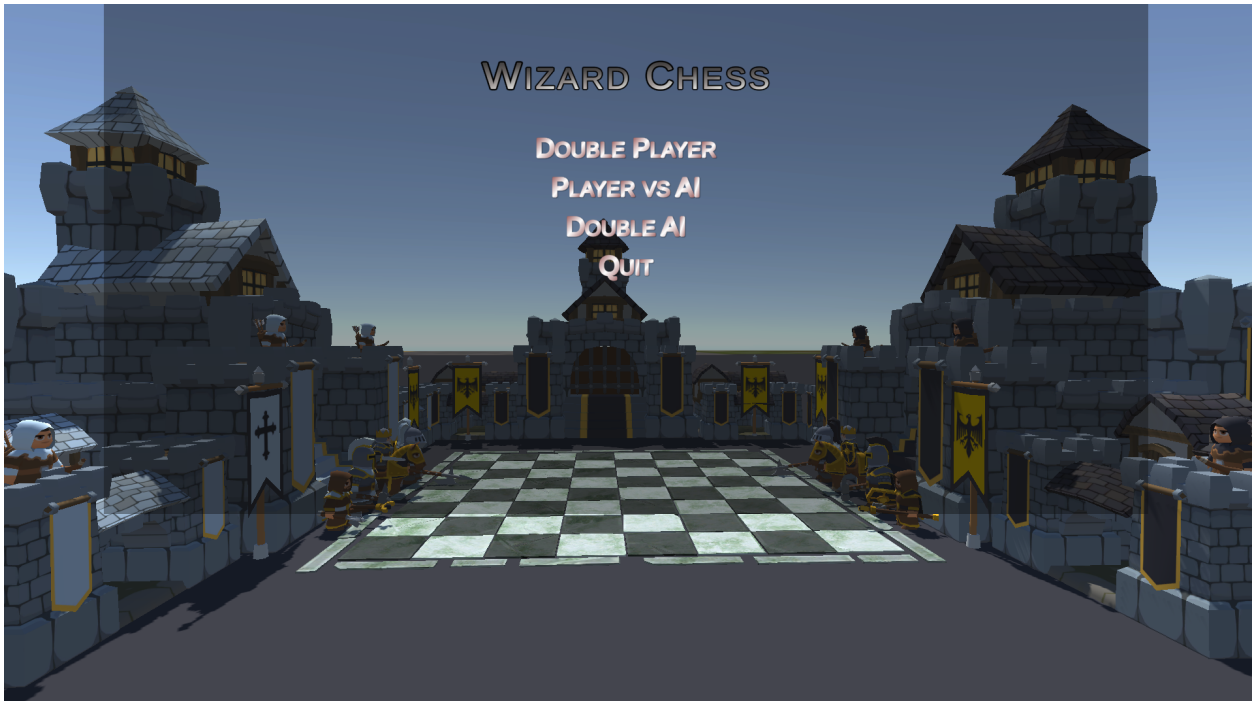
iii) AI VS AI: In this case both the players will be AI and use MiniMax algorithm to choose the moves.

Code:

```
 void Start()
    {
        doublePlayer.onClick.AddListener(delegate { PlayGame(1); });
        playerVsAI.onClick.AddListener(delegate { PlayGame(2); });
        playerVsAI1.onClick.AddListener(delegate { PlayGame(2); });
        doubleAI.onClick.AddListener(delegate { PlayGame(3); });
    }
    public void PlayGame(int mode)
    {
        difficultyLevel = mode!= 2 ? 0
:difficultyLevel>=1?difficultyLevel:1;
        BoardManager.difficultyLevel = difficultyLevel;
        BoardManager.playMode = mode;
        Debug.Log(mode +": "+ difficultyLevel);
        SceneManager.LoadScene(1);

    }
```

Screenshots



#1 Piece Control:

Each piece will have their own scripts to get the possible moves and the move of all the pieces will happen from one script. After the move the rotation of the pieces will remain same.

Code:

```
public virtual void  movePiece(Vector3 tileLocation)
    {
        NavMeshAgent agent = transform.GetComponent<NavMeshAgent>();
        Animator anim = transform.GetComponent<Animator>();
        anim.SetBool("isWalking", true);
        if (team == 0)
        {
            this.transform.rotation = Quaternion.Euler(0, 180, 0);
        }
        else
        {
            this.transform.rotation = Quaternion.Euler(0, 0, 0);
        }
        agent.destination = tileLocation;
}
```

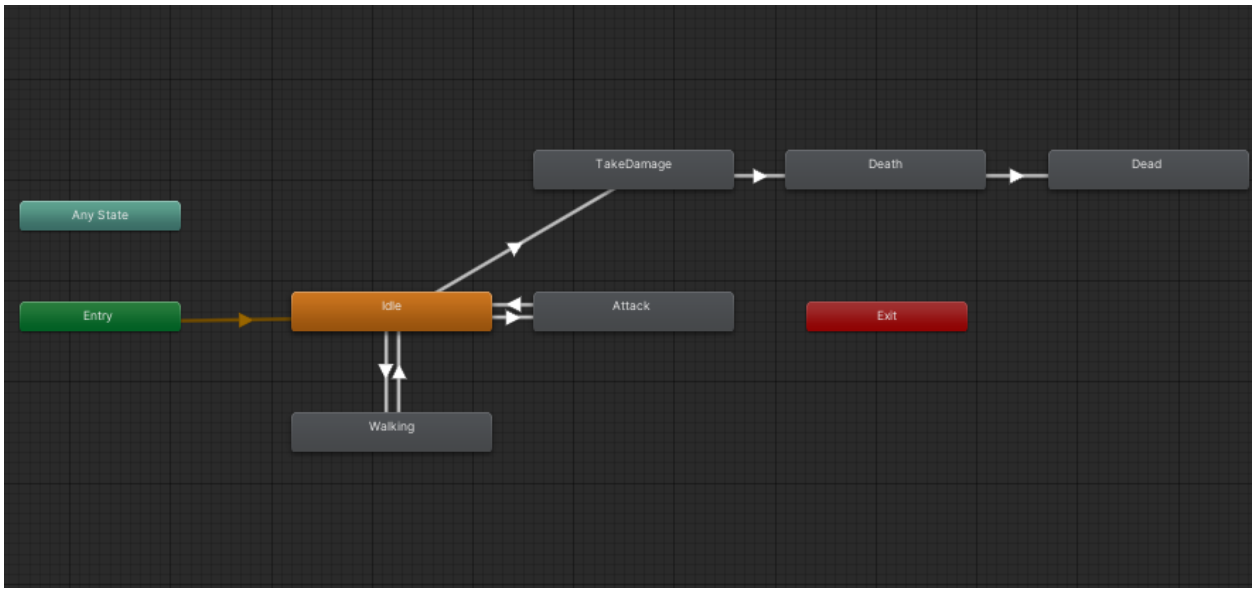
Black Pieces:

White Pieces:

#3 Animations:

There are a total 5 animations: Idle, Walking, Attack, Take damage and Death. The animations will be the same for all the pieces except the Knight. But the transitions will be the same for all.

Transition state Diagram:



The last Dead state is used just to remove the object from the scene.

#4 Collisions:

Each piece will have a capsule collider and the weapon has a box collider, whenever the enemy piece comes into the range of the attacking piece it will trigger an attack the enemy it will take the damage and eventually die.

```
 void OnTriggerEnter(Collider other)
{

     if (other.gameObject.tag == "Enemy" &&
other.GetComponent<ChessPiece>().team!=
transform.GetComponent<ChessPiece>().team)
        {
            transform.GetComponent<Animator>().SetBool("isWalking",
false);
            transform.GetComponent<Animator>().SetBool("isAttacking",
true);

other.gameObject.GetComponent<Animator>().SetBool("isDead", true);
        }
    }
```

#5 AI

The AI is built using the Minimax Algorithm, it will pick the piece and get the valid moves which are defined for the pieces and calculate the best possible move for the current board.

```
private Move MiniMax(ref int[,] board, bool isMax, int depth, float
alpha, float beta)
    {
        Debug.Log("Mini Max");
        List<Move> moves = new List<Move>();
```

```
        int t = Move.GetPossibleMoves(board, isMax, ref moves,
this.board.pieces);
        Debug.Log("possible Move:" + t);
        if (t == 0)
        {
            if (depth == 0)
                this.board.drawn = true;

            return null;
        }
        else if (t == -1)
        {
            if (depth == 0)
            {
                this.board.CheckMate = true;
            }
            return null;
        }
        float minMax = 0.0f;
        int k = 0;
        float eval = 0.0f;
        if (isMax) eval = -Mathf.Infinity;
        else eval = Mathf.Infinity;
        for (int i = 0; i < moves.Count; i++)
        {
            int temp = BoardManager.ApplyMove(ref board, moves[i]);

            if (depth == treeHeight - 1)
            {
                eval = Evaluator.Evaluate(board);
            }
            else
            {
                Move bestMove = MiniMax(ref board, !isMax, depth + 1,
alpha, beta);

                if (bestMove == null)
                {
```

```
                    BoardManager.ReverseMove(ref board, moves[i],
temp);
                    continue;
                }

                int temp1 = BoardManager.ApplyMove(ref board,
bestMove);

                eval = Evaluator.Evaluate(board);

                BoardManager.ReverseMove(ref board, bestMove, temp1);
            }

            if (i == 0)
            {
                minMax = eval;
                BoardManager.ReverseMove(ref board, moves[i], temp);
                continue;
            }

            if (isMax)
            {
                if (minMax < eval)
                {
                    minMax = eval;
                    k = i;
                }

                if (alpha < eval)
                    alpha = eval;

            }
            else
            {
                if (minMax > eval)
                {
                    minMax = eval;
                    k = i;
                }
```

```
                if (beta > eval)
                    beta = eval;
            }
            BoardManager.ReverseMove(ref board, moves[i], temp);

            if (beta <= alpha)
                break;
        }


        if (t == -2)
        {
            if (depth == 0)
                this.drawMove = moves[k];
        }
        Debug.Log("Move start:" + moves[k].start[0] + ":" +
moves[k].start[1]);
        Debug.Log("Move end :"+ moves[k].end[0]+":"+
moves[k].end[1]);
        return moves[k];
    }
```

#6 Game End

The game will be ended once the board has a checkmate or the board is drawn.

Code Snippet:

```
if (isWhiteTurn)
        {
            Debug.Log("white turn");
           if (checkmateCheck(1))
            {
                WinBoardScript.defaultText = "White Wins !!!";
                SceneManager.LoadScene(2);
                return;
```

```
            }

            if (drawn)
            {
                WinBoardScript.defaultText = "Game Drawn !!!";
                SceneManager.LoadScene(2);
                return;
            }
            player1.Update();

        }
        else
        {
            if (checkmateCheck(0))
            {
                WinBoardScript.defaultText = "Black Wins !!!";
                SceneManager.LoadScene(2);
                return;
            }

            if (drawn)
            {
                WinBoardScript.defaultText = "Game Drawn !!!";
                Debug.Log("Game Drawn");
                SceneManager.LoadScene(2);
                return;
            }

            player2.Update();

        }
```

Screenshot:

#Game Scene Screenshots:


Back


Back


Back