Bishop's University

Computer Games Design

Final Project
Instructed By: Prof. Russell Butler

Submitted By:
Jagjot Singh

1) Game features implemented:

We tried to implement the following features in the our project:

A) Chess rules.

B) MiniMax With Alpha - Beta pruning.

C) Piece selection and movement.

D) Detailed attack collisions and piece destruction.

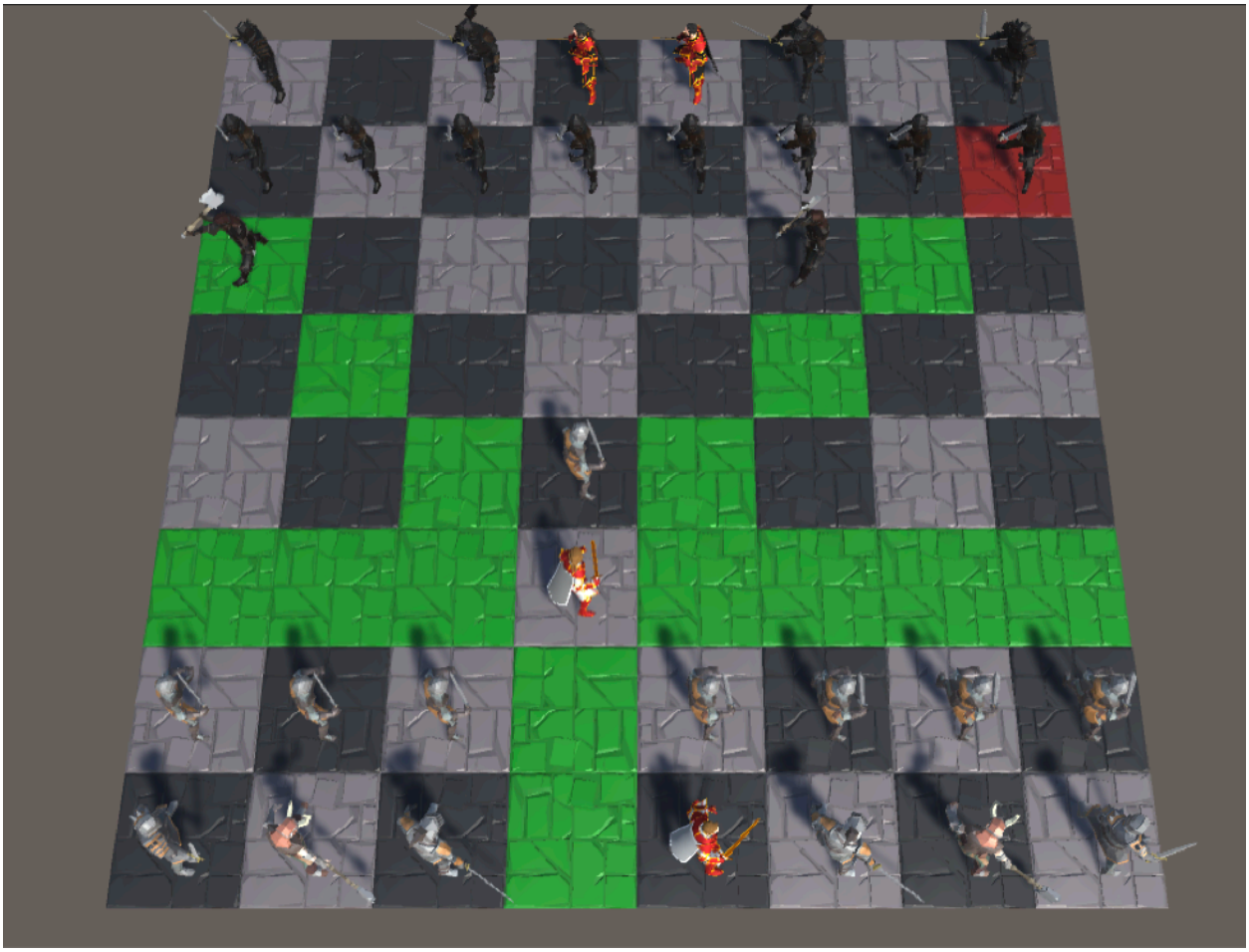
E) Random player(At difficulty lvl 1) and MiniMax player.

F) Difficulty slider.

G) Sound effects for hit and death.

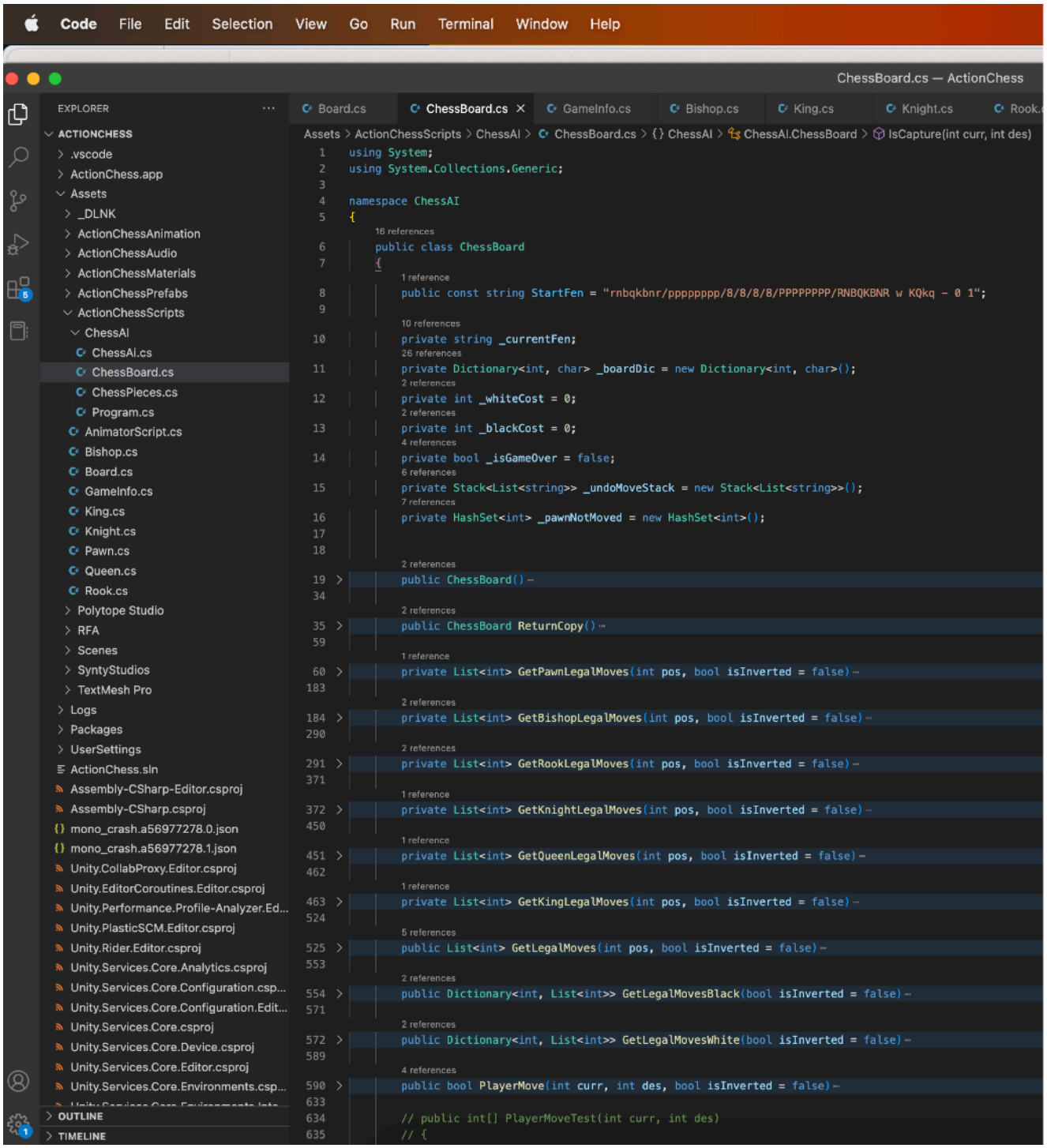
H) Multiplayer with Unity NetCode and Relay.

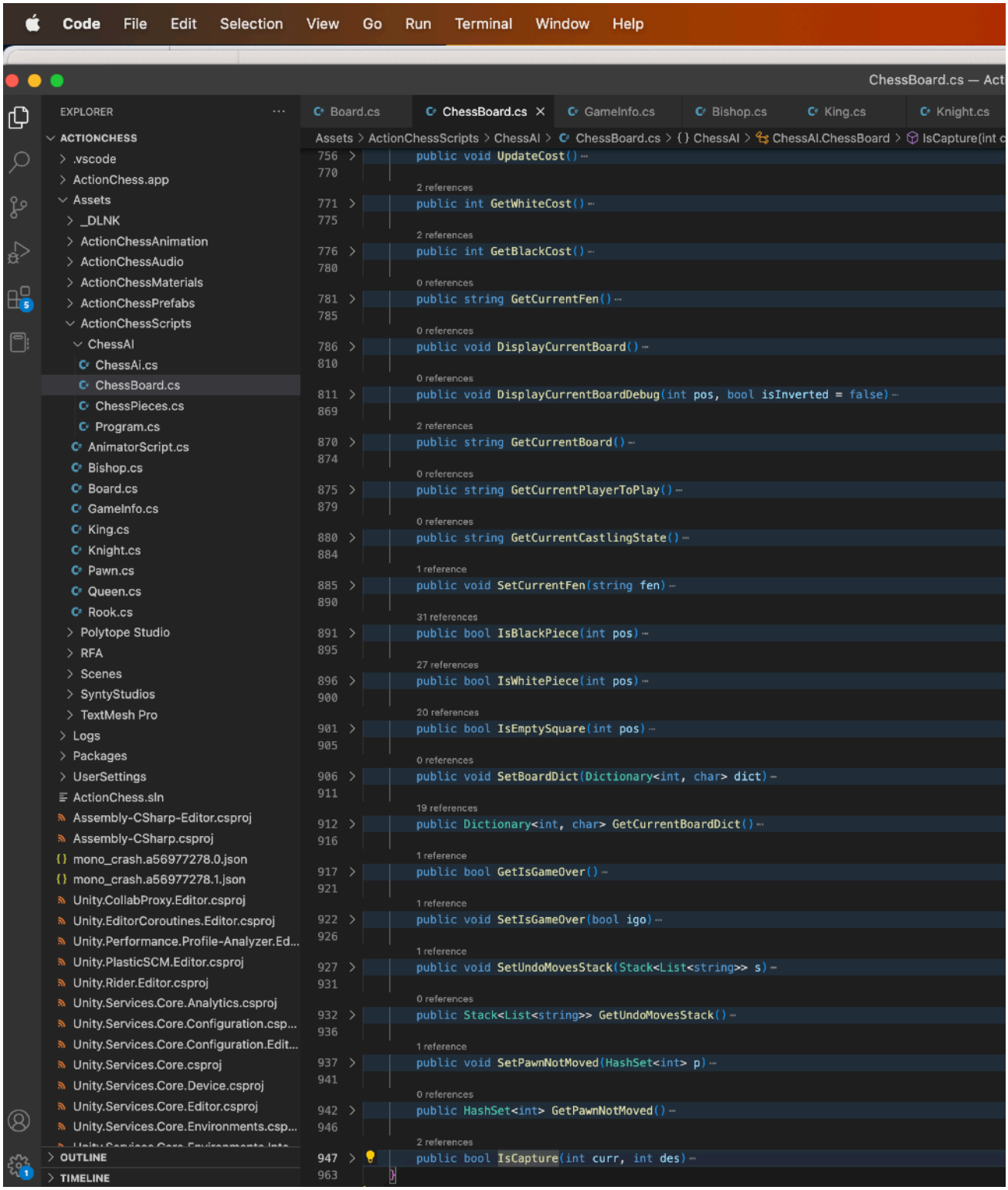
2) Game Screenshots:



Screenshot highlighting legal moves and pieces in danger

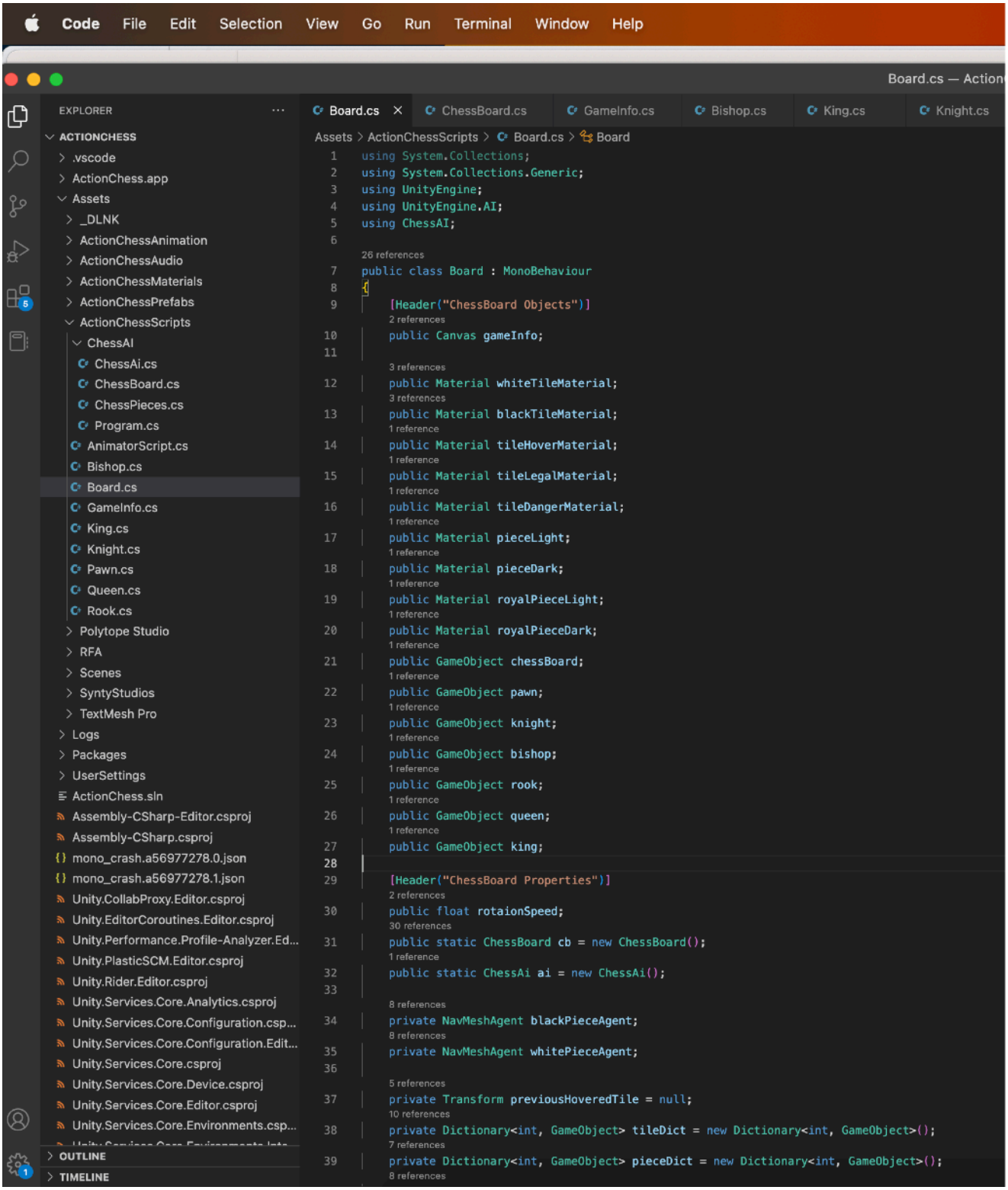
3) Chess base source code:

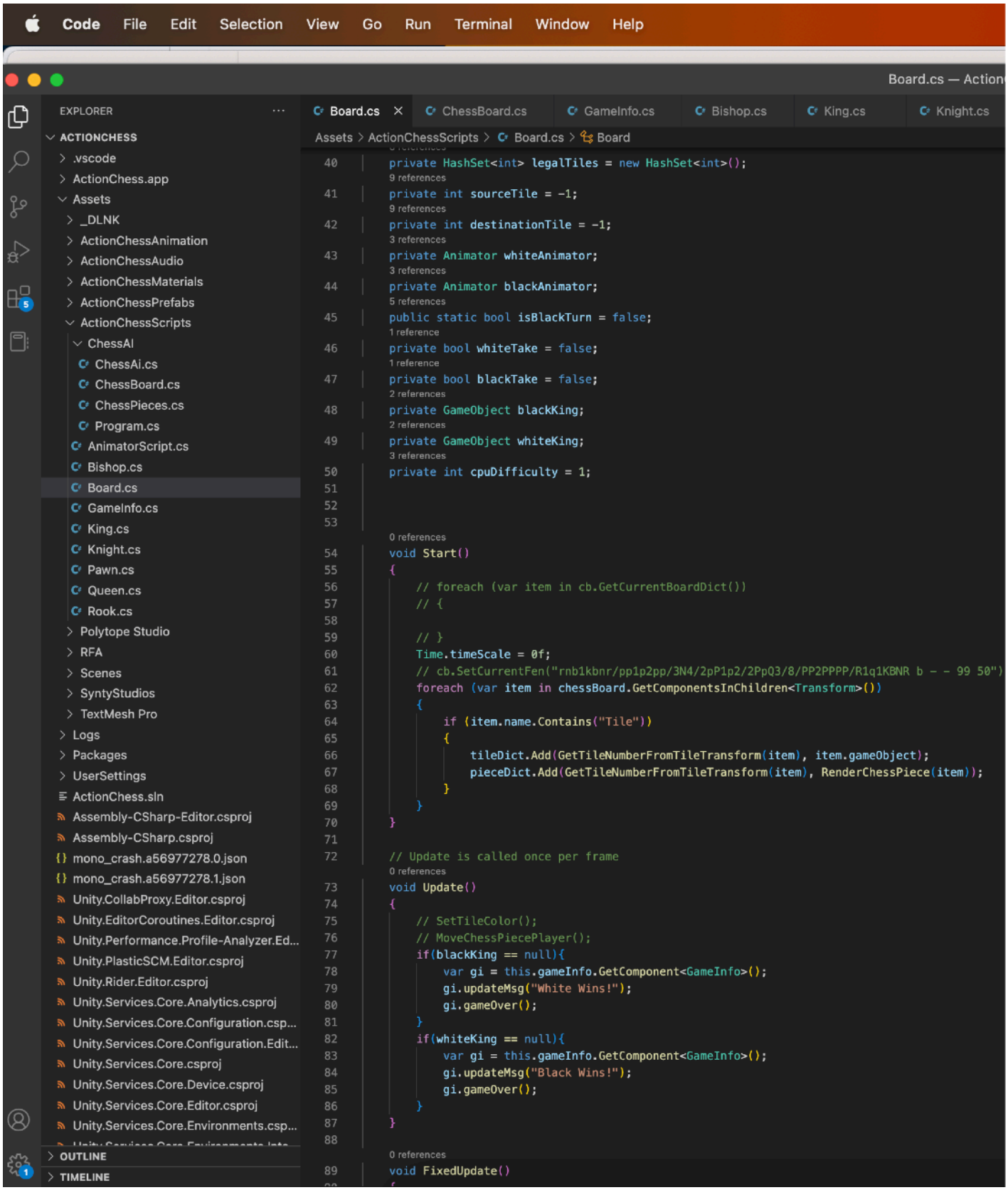


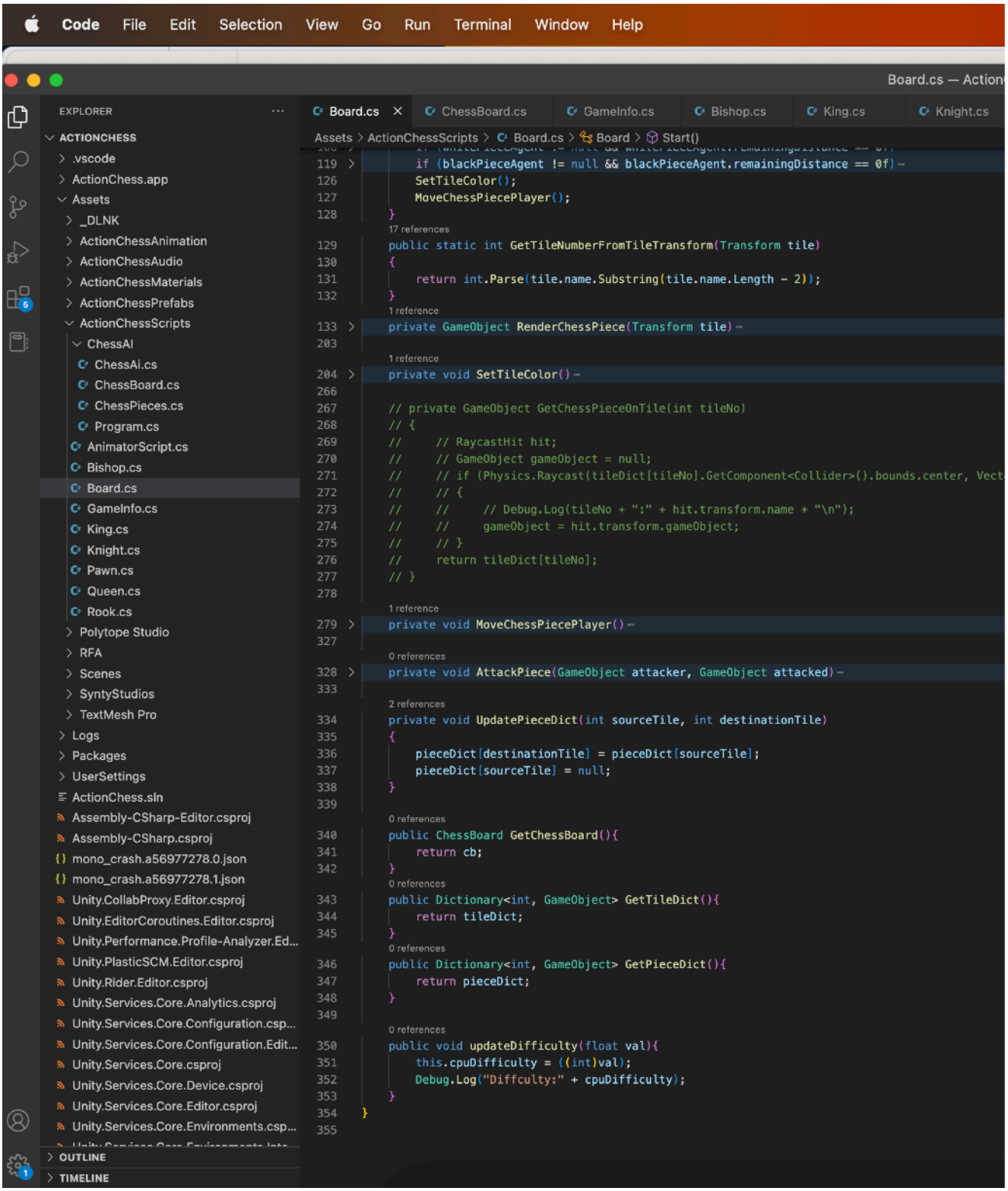


Above is the screenshot of base chess, As the code is nearly 1000 lines of code, I've decided to post the screenshot of only methods used.

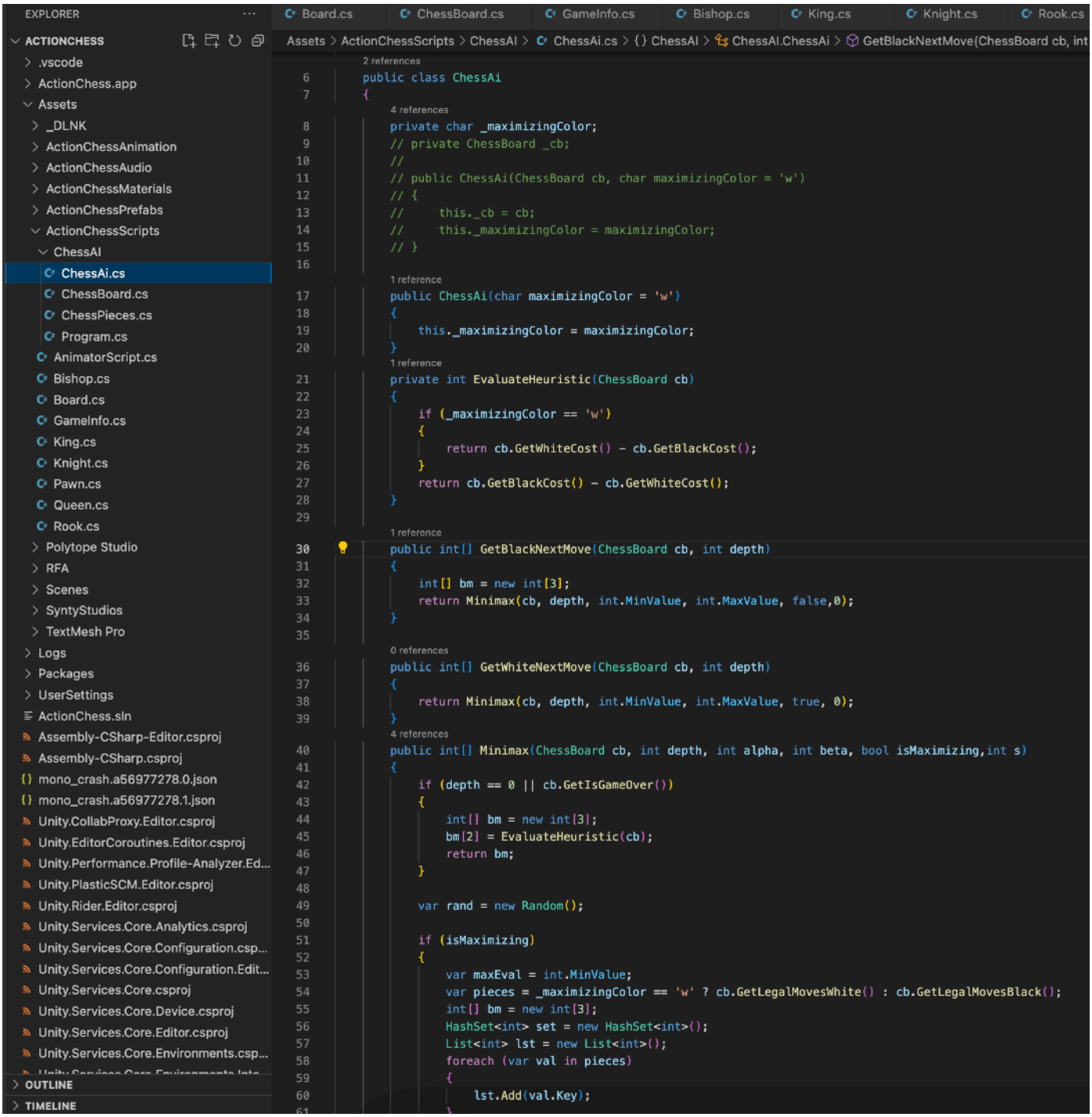
4) Source code for chessboard Unity C# Script:

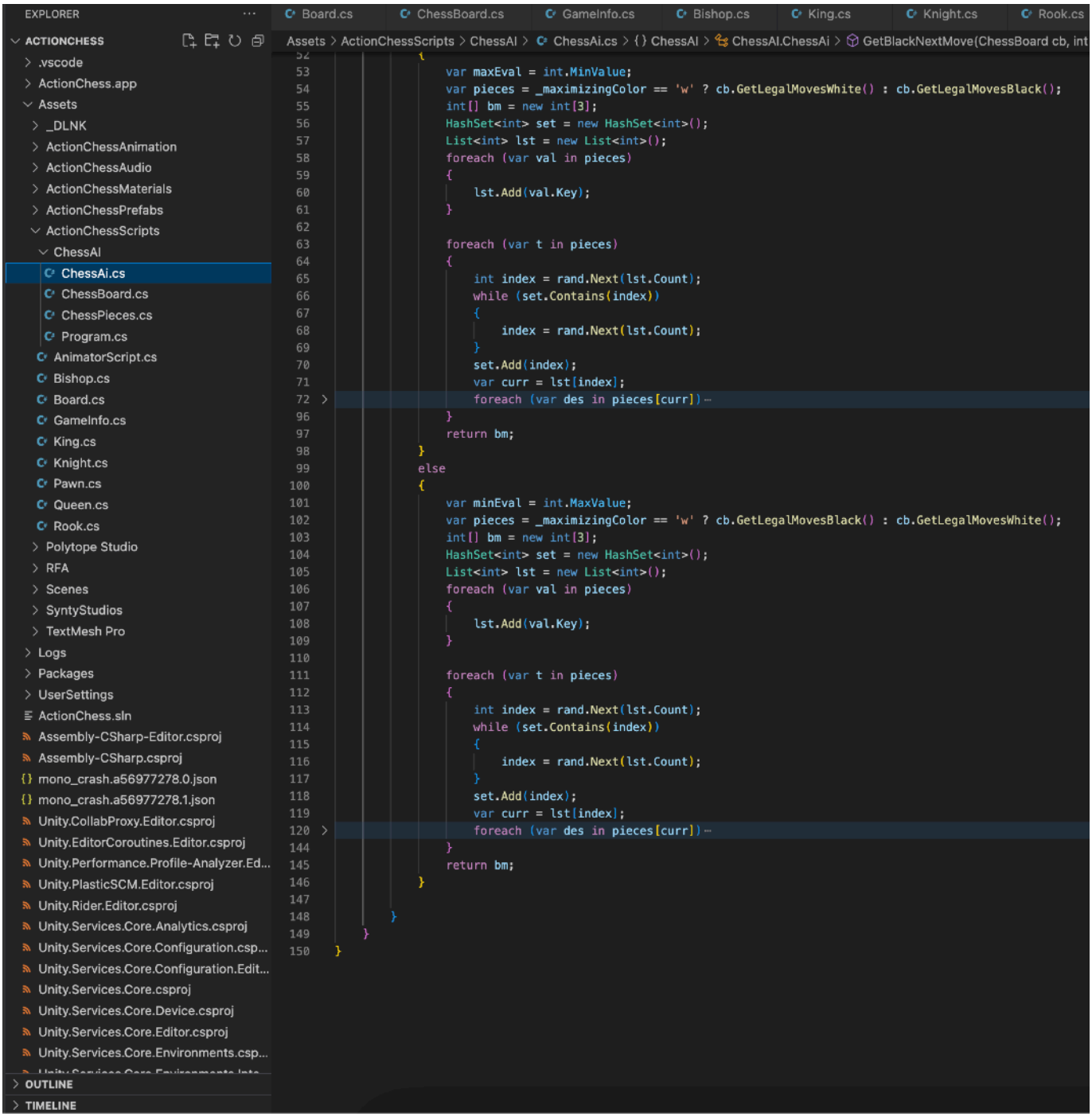



Code File Edit Selection View Go Run Terminal Window Help
Board.cs — Action
EXPLORER
ACTIONCHESS
.vscode
ActionChess.app
Assets
_DLNK
ActionChessAnimation
ActionChessAudio
ActionChessMaterials
ActionChessPrefabs
ActionChessScripts
ChessAI
ChessAi.cs
ChessBoard.cs
ChessPieces.cs
Program.cs
AnimatorScript.cs
Bishop.cs
Board.cs
GameInfo.cs
King.cs
Knight.cs
Pawn.cs
Queen.cs
Rook.cs
Polytope Studio
RFA
Scenes
SyntyStudios
TextMesh Pro
Logs
Packages
UserSettings
ActionChess.sln
Assembly-CSharp-Editor.csproj
Assembly-CSharp.csproj
mono_crash.a56977278.0.json
mono_crash.a56977278.1.json
Unity.CollabProxy.Editor.csproj
Unity.EditorCoroutines.Editor.csproj
Unity.Performance.Profile-Analyzer.Ed...
Unity.PlasticSCM.Editor.csproj
Unity.Rider.Editor.csproj
Unity.Services.Core.Analytics.csproj
Unity.Services.Core.Configuration.csp...
Unity.Services.Core.Configuration.Edit...
Unity.Services.Core.csproj
Unity.Services.Core.Device.csproj
Unity.Services.Core.Editor.csproj
Unity.Services.Core.Environments.csp...
OUTLINE
TIMELINE
Board.cs
ChessBoard.cs
GameInfo.cs
Bishop.cs
King.cs
Knight.cs
Assets > ActionChessScripts > Board.cs > Board
40 private HashSet<int> legalTiles = new HashSet<int>();
9 references
41 private int sourceTile = -1;
9 references
42 private int destinationTile = -1;
3 references
43 private Animator whiteAnimator;
3 references
44 private Animator blackAnimator;
5 references
45 public static bool isBlackTurn = false;
1 reference
46 private bool whiteTake = false;
1 reference
47 private bool blackTake = false;
2 references
48 private GameObject blackKing;
2 references
49 private GameObject whiteKing;
3 references
50 private int cpuDifficulty = 1;
51
52
53
0 references
54 void Start()
55 {
56 // foreach (var item in cb.GetCurrentBoardDict())
57 // {
58
59 // }
60 Time.timeScale = 0f;
61 // cb.SetCurrentFen("rnb1kbnr/pp1p2pp/3N4/2pP1p2/2PpQ3/8/PP2PPPP/R1q1KBNR b - - 99 50")
62 foreach (var item in chessBoard.GetComponentsInChildren<Transform>())
63 {
64 if (item.name.Contains("Tile"))
65 {
66 tileDict.Add(GetTileNumberFromTileTransform(item), item.gameObject);
67 pieceDict.Add(GetTileNumberFromTileTransform(item), RenderChessPiece(item));
68 }
69 }
70 }
71
72 // Update is called once per frame
0 references
73 void Update()
74 {
75 // SetTileColor();
76 // MoveChessPiecePlayer();
77 if(blackKing == null){
78 var gi = this.gameInfo.GetComponent<GameInfo>();
79 gi.updateMsg("White Wins!");
80 gi.gameOver();
81 }
82 if(whiteKing == null){
83 var gi = this.gameInfo.GetComponent<GameInfo>();
84 gi.updateMsg("Black Wins!");
85 gi.gameOver();
86 }
87 }
88
0 references
89 void FixedUpdate()


Code File Edit Selection View Go Run Terminal Window Help
Board.cs — Action
EXPLORER
ACTIONCHESS
.vscode
ActionChess.app
Assets
_DLNK
ActionChessAnimation
ActionChessAudio
ActionChessMaterials
ActionChessPrefabs
ActionChessScripts
ChessAI
ChessAi.cs
ChessBoard.cs
ChessPieces.cs
Program.cs
AnimatorScript.cs
Bishop.cs
Board.cs
GameInfo.cs
King.cs
Knight.cs
Pawn.cs
Queen.cs
Rook.cs
Polytope Studio
RFA
Scenes
SyntyStudios
TextMesh Pro
Logs
Packages
UserSettings
ActionChess.sln
Assembly-CSharp-Editor.csproj
Assembly-CSharp.csproj
mono_crash.a56977278.0.json
mono_crash.a56977278.1.json
Unity.CollabProxy.Editor.csproj
Unity.EditorCoroutines.Editor.csproj
Unity.Performance.Profile-Analyzer.Ed...
Unity.PlasticSCM.Editor.csproj
Unity.Rider.Editor.csproj
Unity.Services.Core.Analytics.csproj
Unity.Services.Core.Configuration.csp...
Unity.Services.Core.Configuration.Edit...
Unity.Services.Core.csproj
Unity.Services.Core.Device.csproj
Unity.Services.Core.Editor.csproj
Unity.Services.Core.Environments.csp...
OUTLINE
TIMELINE
Board.cs
ChessBoard.cs
GameInfo.cs
Bishop.cs
King.cs
Knight.cs
Assets > ActionChessScripts > Board.cs > Board > Start()
if (blackPieceAgent != null && blackPieceAgent.remainingDistance == 0f)
SetTileColor();
MoveChessPiecePlayer();
17 references
public static int GetTileNumberFromTileTransform(Transform tile)
return int.Parse(tile.name.Substring(tile.name.Length - 2));
1 reference
private GameObject RenderChessPiece(Transform tile)
1 reference
private void SetTileColor()
// private GameObject GetChessPieceOnTile(int tileNo)
// {
// // RaycastHit hit;
// // GameObject gameObject = null;
// // if (Physics.Raycast(tileDict[tileNo].GetComponent<Collider>().bounds.center, Vect
// // {
// // // Debug.Log(tileNo + ":" + hit.transform.name + "\n");
// // gameObject = hit.transform.gameObject;
// // }
// return tileDict[tileNo];
// }
1 reference
private void MoveChessPiecePlayer()
0 references
private void AttackPiece(GameObject attacker, GameObject attacked)
2 references
private void UpdatePieceDict(int sourceTile, int destinationTile)
pieceDict[destinationTile] = pieceDict[sourceTile];
pieceDict[sourceTile] = null;
0 references
public ChessBoard GetChessBoard(){
return cb;
0 references
public Dictionary<int, GameObject> GetTileDict(){
return tileDict;
0 references
public Dictionary<int, GameObject> GetPieceDict(){
return pieceDict;
0 references
public void updateDifficulty(float val){
this.cpuDifficulty = ((int)val);
Debug.Log("Diffculty:" + cpuDifficulty);

5) MiniMax with Alpha- Beta pruning source code:




EXPLORER
ACTIONCHESS
.vscode
ActionChess.app
Assets
_DLNK
ActionChessAnimation
ActionChessAudio
ActionChessMaterials
ActionChessPrefabs
ActionChessScripts
ChessAI
ChessAi.cs
ChessBoard.cs
ChessPieces.cs
Program.cs
AnimatorScript.cs
Bishop.cs
Board.cs
GameInfo.cs
King.cs
Knight.cs
Pawn.cs
Queen.cs
Rook.cs
Polytope Studio
RFA
Scenes
SyntyStudios
TextMesh Pro
Logs
Packages
UserSettings
ActionChess.sln
Assembly-CSharp-Editor.csproj
Assembly-CSharp.csproj
mono_crash.a56977278.0.json
mono_crash.a56977278.1.json
Unity.CollabProxy.Editor.csproj
Unity.EditorCoroutines.Editor.csproj
Unity.Performance.Profile-Analyzer.Ed...
Unity.PlasticSCM.Editor.csproj
Unity.Rider.Editor.csproj
Unity.Services.Core.Analytics.csproj
Unity.Services.Core.Configuration.csp...
Unity.Services.Core.Configuration.Edit...
Unity.Services.Core.csproj
Unity.Services.Core.Device.csproj
Unity.Services.Core.Editor.csproj
Unity.Services.Core.Environments.csp...
OUTLINE
TIMELINE
Board.cs
ChessBoard.cs
GameInfo.cs
Bishop.cs
King.cs
Knight.cs
Rook.cs
Assets > ActionChessScripts > ChessAI > ChessAi.cs > { } ChessAI > ChessAI.ChessAi > GetBlackNextMove(ChessBoard cb, int
{
var maxEval = int.MinValue;
var pieces = _maximizingColor == 'w' ? cb.GetLegalMovesWhite() : cb.GetLegalMovesBlack();
int[] bm = new int[3];
HashSet<int> set = new HashSet<int>();
List<int> lst = new List<int>();
foreach (var val in pieces)
{
lst.Add(val.Key);
}
foreach (var t in pieces)
{
int index = rand.Next(lst.Count);
while (set.Contains(index))
{
index = rand.Next(lst.Count);
}
set.Add(index);
var curr = lst[index];
foreach (var des in pieces[curr])
}
return bm;
}
else
{
var minEval = int.MaxValue;
var pieces = _maximizingColor == 'w' ? cb.GetLegalMovesBlack() : cb.GetLegalMovesWhite();
int[] bm = new int[3];
HashSet<int> set = new HashSet<int>();
List<int> lst = new List<int>();
foreach (var val in pieces)
{
lst.Add(val.Key);
}
foreach (var t in pieces)
{
int index = rand.Next(lst.Count);
while (set.Contains(index))
{
index = rand.Next(lst.Count);
}
set.Add(index);
var curr = lst[index];
foreach (var des in pieces[curr])
}
return bm;
}
}
}
}

6) Player vs Player:

We have implemented the multiplayer functionality using Unity Netcode and Unity Relay service.

Players can request for the Unity Relay code and pass it to other player to join the multiplayer.

7) Source Code for Unity Relay Service:

```
    private string joincode;
// Start is called before the first frame update
0 references
private async void Start()
{
    await UnityServices.InitializeAsync();
    AuthenticationService.Instance.SignedIn += () =>
    {
        Debug.Log("Signed In " + AuthenticationService.Instance.PlayerId);
    };

    await AuthenticationService.Instance.SignInAnonymouslyAsync();

}

1 reference
private async void CreateRelay()
{
    try
    {
        Allocation allocation = await RelayService.Instance.CreateAllocationAsync(1);
        joincode = await RelayService.Instance.GetJoinCodeAsync(allocation.AllocationId);
        // Debug.Log("JoinCode: " + joincode);

        NetworkManager.Singleton.GetComponent<UnityTransport>().SetHostRelayData(
            allocation.RelayServer.IpV4,
            (ushort)allocation.RelayServer.Port,
            allocation.AllocationIdBytes,
            allocation.Key,
            allocation.ConnectionData
        );

        NetworkManager.Singleton.StartHost();

    }
    catch (RelayServiceException e)
    {
        Debug.Log(e);
    }

}

1 reference
private async void JoinRelay(string code)
{
    try
    {
        Debug.Log("Joing Relay with code: " + code);
        JoinAllocation joinAllocation = await RelayService.Instance.JoinAllocationAsync(code);

        NetworkManager.Singleton.GetComponent<UnityTransport>().SetClientRelayData(
            joinAllocation.RelayServer.IpV4,
            (ushort)joinAllocation.RelayServer.Port,
            joinAllocation.AllocationIdBytes,
            joinAllocation.Key,
            joinAllocation.ConnectionData,
            joinAllocation.HostConnectionData
        );
        NetworkManager.Singleton.StartClient();
    }
    catch (RelayServiceException e)
    {
        Debug.Log(e);
    }
}
```

8) Files submitted:

a) ActionChess.zip:

Windows version of the game inside ‘AC_Win’ folder.

b) AC_Mac.zip: MacOS version on the game.

c) Scripts.zip: Contains the C# scripts.