Work In Progress 2020

MAY 4

COMPANY NAME

Authored by: Your Name

Logo
Name

Contents

Still a work in progress....................3

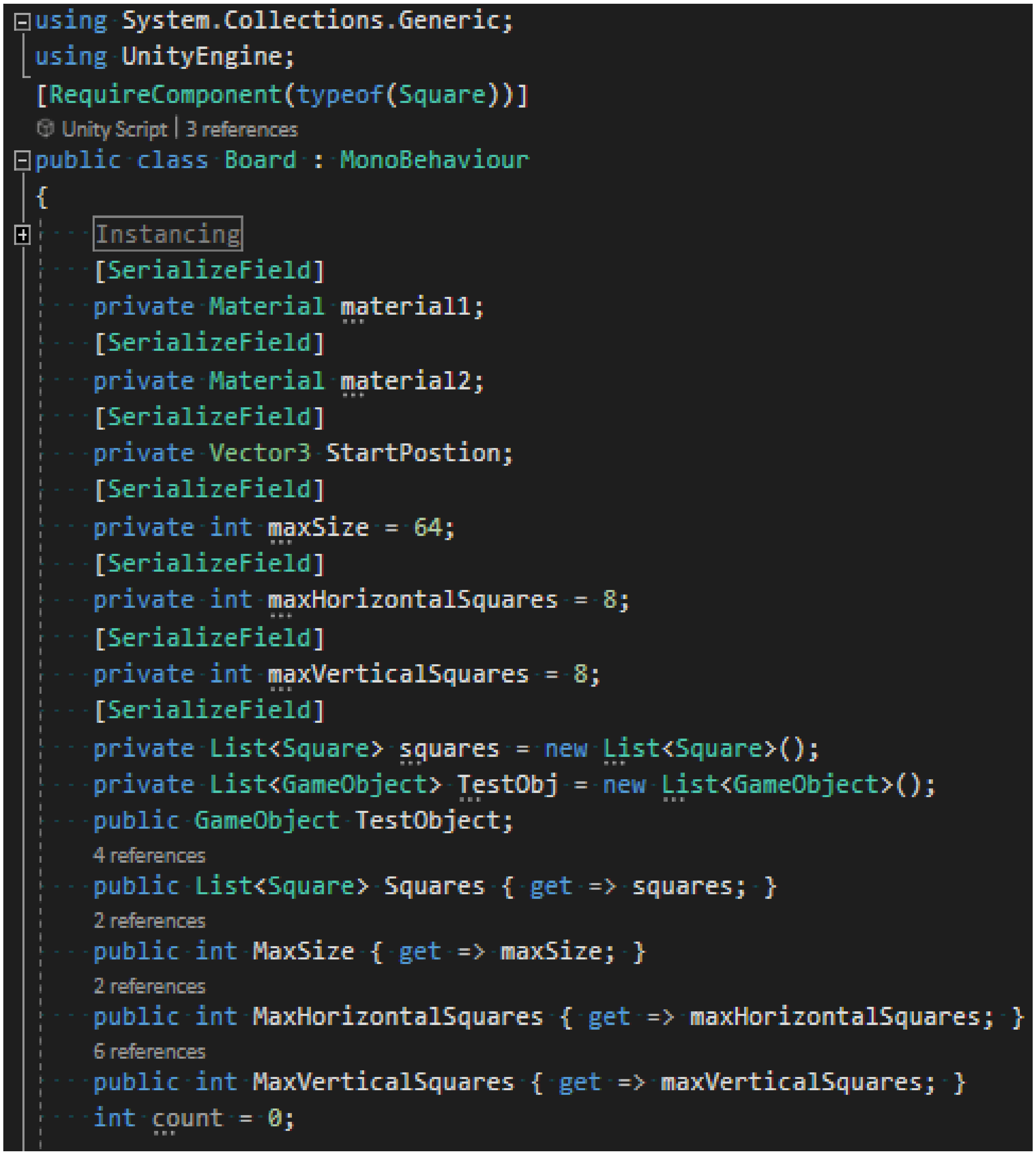
Above Code Generates this board here below from a single tile....................8

Player Controller....................9

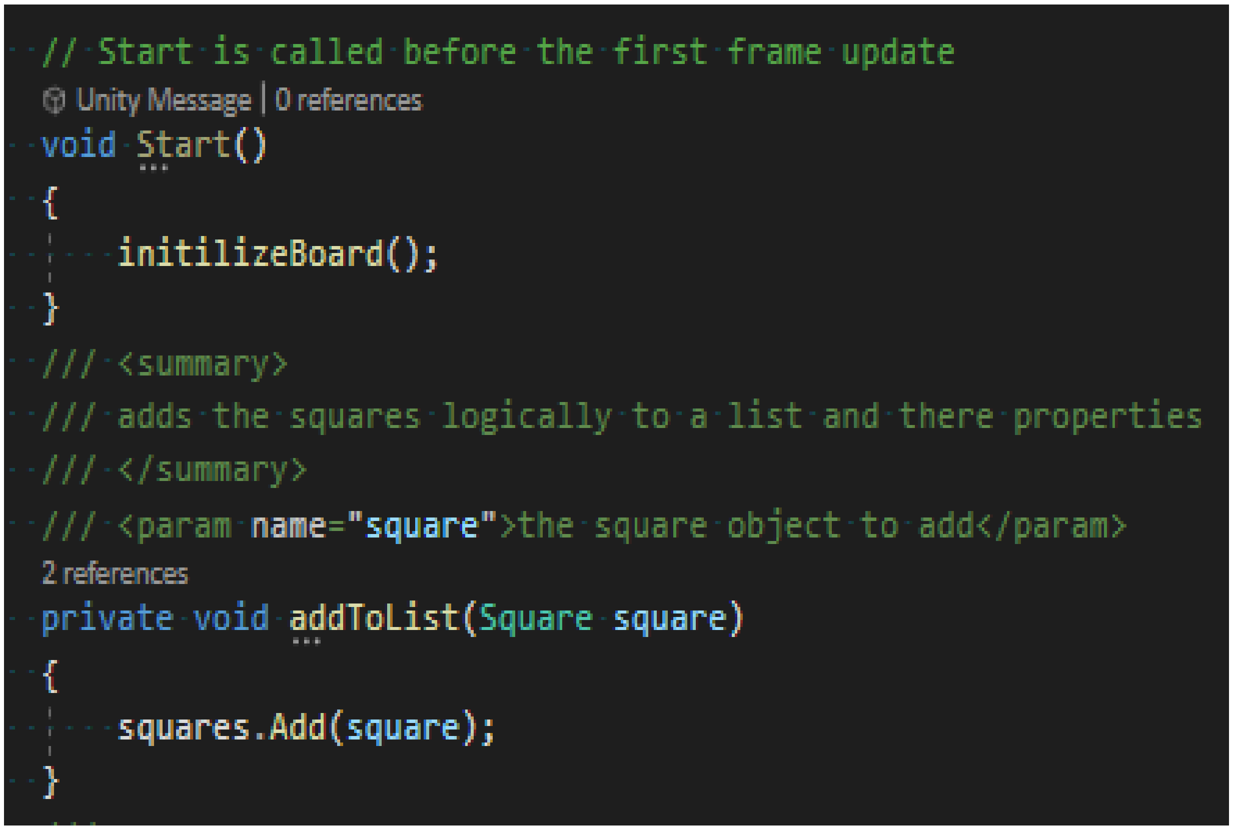
The next method I need to improve I don't like it....................12

This above is too much. Best thing that came to mind at that time sadly. 12

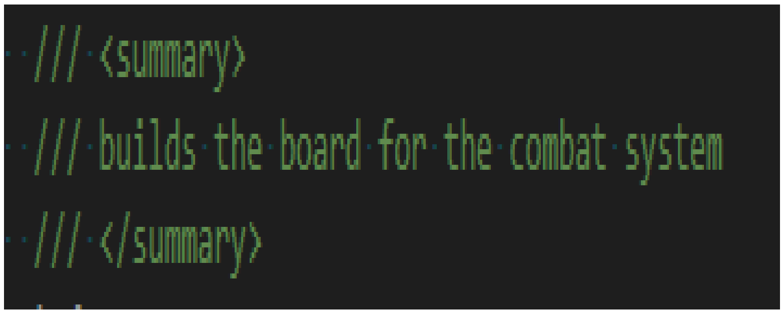
Still a work in progress



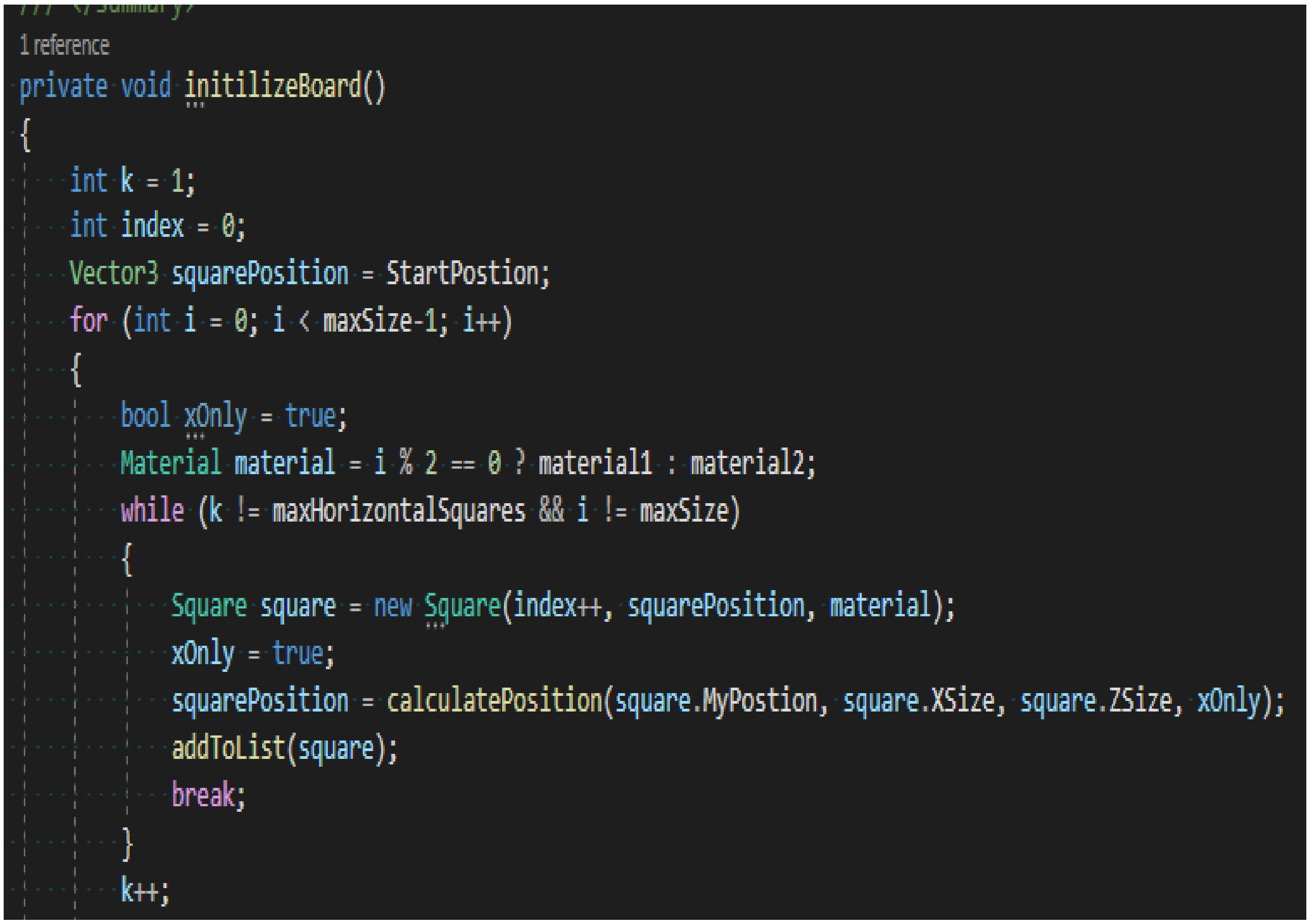
```
using System.Collections.Generic;
using UnityEngine;
[RequireComponent(typeof(Square))]
Unity Script | 3 references
public class Board : MonoBehaviour
{
    Instancing
    [SerializeField]
    private Material material1;
    [SerializeField]
    private Material material2;
    [SerializeField]
    private Vector3 StartPostion;
    [SerializeField]
    private int maxSize = 64;
    [SerializeField]
    private int maxHorizontalSquares = 8;
    [SerializeField]
    private int maxVerticalSquares = 8;
    [SerializeField]
    private List<Square> squares = new List<Square>();
    private List<GameObject> TestObj = new List<GameObject>();
    public GameObject TestObject;
    4 references
    public List<Square> Squares { get => squares; }
    2 references
    public int MaxSize { get => maxSize; }
    2 references
    public int MaxHorizontalSquares { get => maxHorizontalSquares; }
    6 references
    public int MaxVerticalSquares { get => maxVerticalSquares; }
    int count = 0;
```



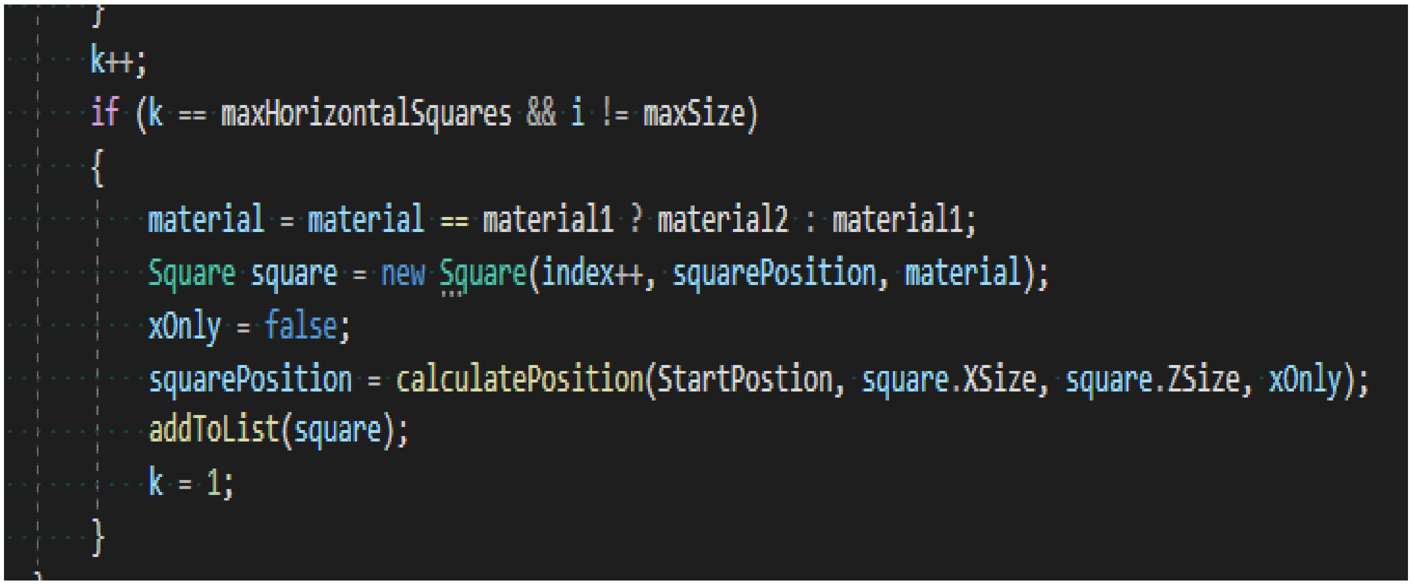
```
// Start is called before the first frame update
Unity Message | 0 references
void Start()
{
    initilizeBoard();
}
/// <summary>
/// adds the squares logically to a list and there properties
/// </summary>
/// <param name="square">the square object to add</param>
2 references
private void addToList(Square square)
{
    squares.Add(square);
}
```



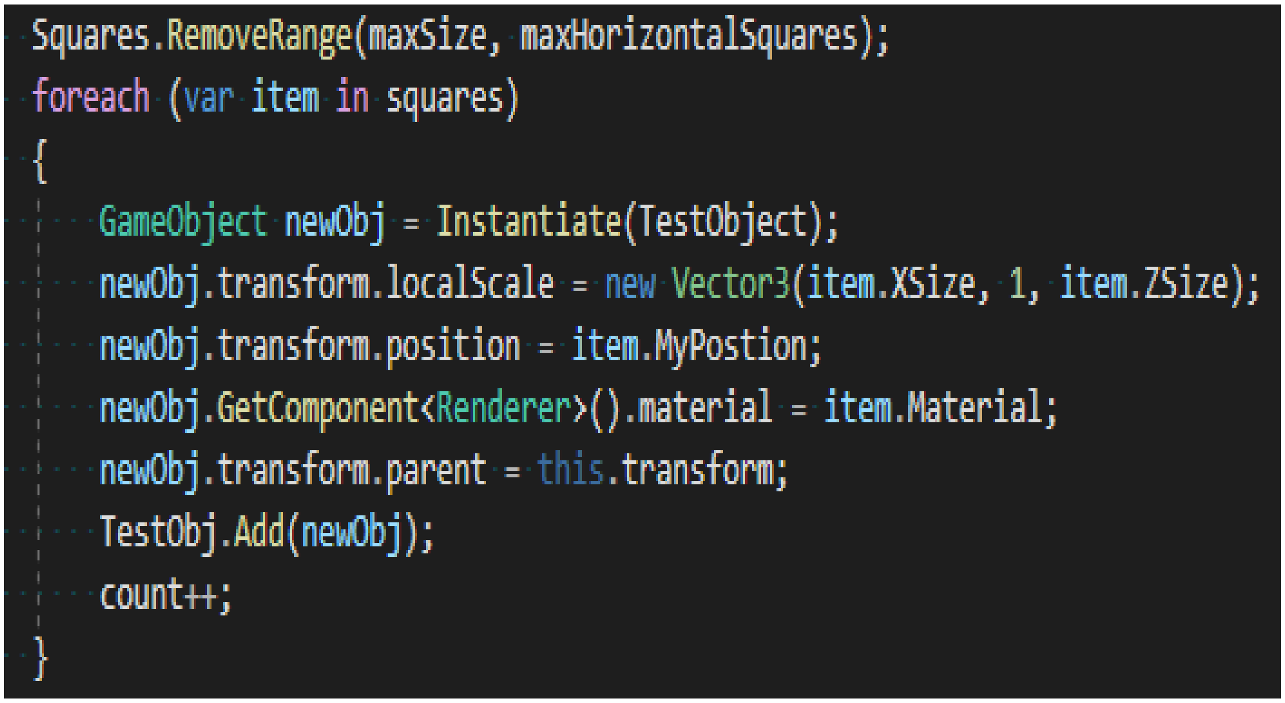
```
/// <summary>
/// builds the board for the combat system
/// </summary>
```



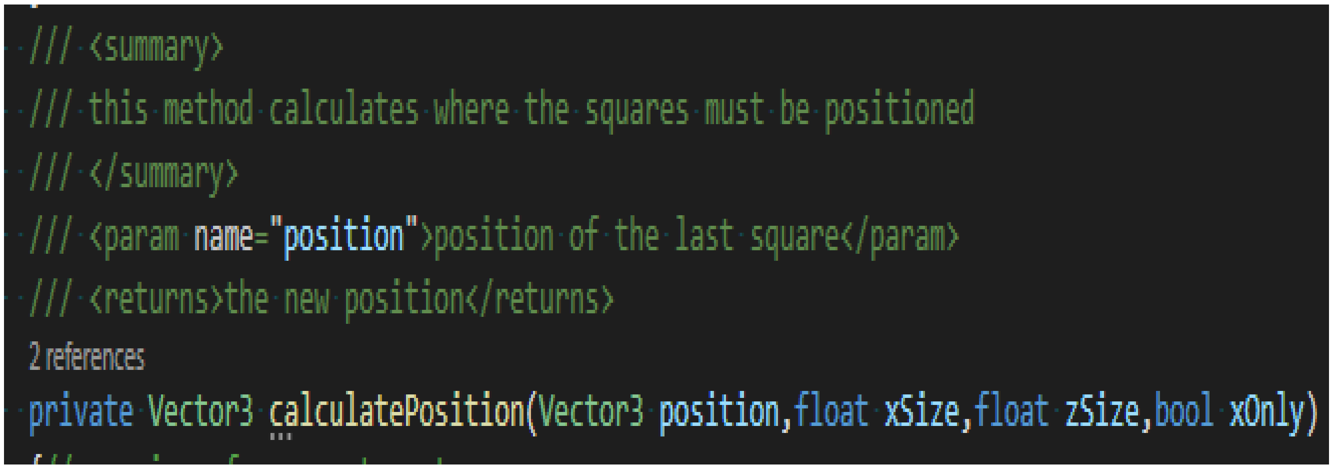
```
/// </summary>
1 reference
private void initilizeBoard()
{
    int k = 1;
    int index = 0;
    Vector3 squarePosition = StartPostion;
    for (int i = 0; i < maxSize-1; i++)
    {
        bool xOnly = true;
        Material material = i % 2 == 0 ? material1 : material2;
        while (k != maxHorizontalSquares && i != maxSize)
        {
            Square square = new Square(index++, squarePosition, material);
            xOnly = true;
            squarePosition = calculatePosition(square.MyPostion, square.XSize, square.ZSize, xOnly);
            addToList(square);
            break;
        }
        k++;
```



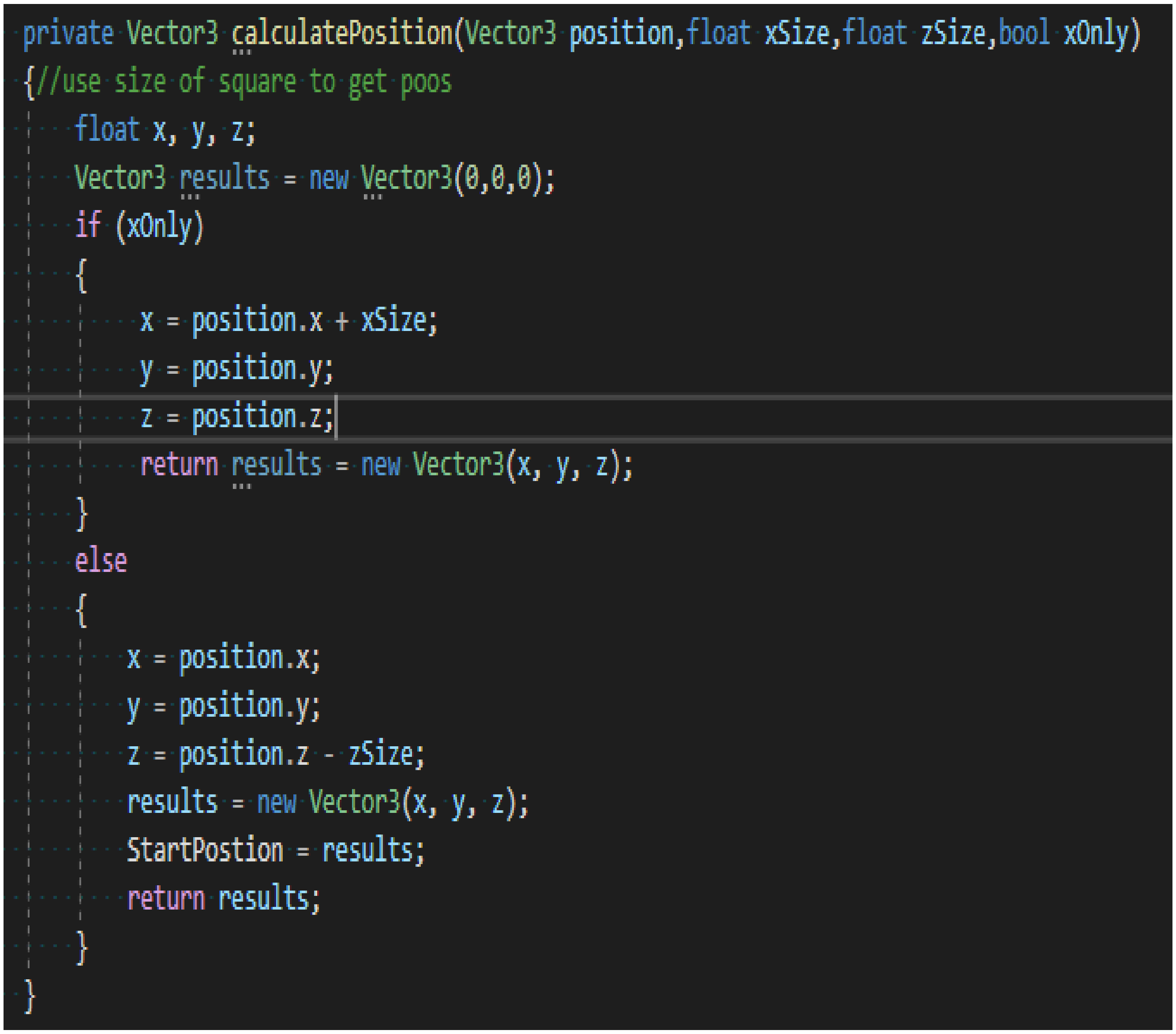
```
        }
        k++;
        if (k == maxHorizontalSquares && i != maxSize)
        {
            material = material == material1 ? material2 : material1;
            Square square = new Square(index++, squarePosition, material);
            xOnly = false;
            squarePosition = calculatePosition(StartPostion, square.XSize, square.ZSize, xOnly);
            addToList(square);
            k = 1;
        }
```



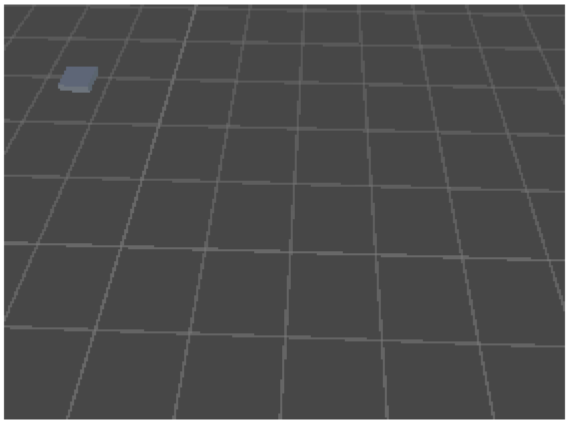
```
Squares.RemoveRange(maxSize, maxHorizontalSquares);
foreach (var item in squares)
{
    GameObject newObj = Instantiate(TestObject);
    newObj.transform.localScale = new Vector3(item.XSize, 1, item.ZSize);
    newObj.transform.position = item.MyPostion;
    newObj.GetComponent<Renderer>().material = item.Material;
    newObj.transform.parent = this.transform;
    TestObj.Add(newObj);
    count++;
}
```



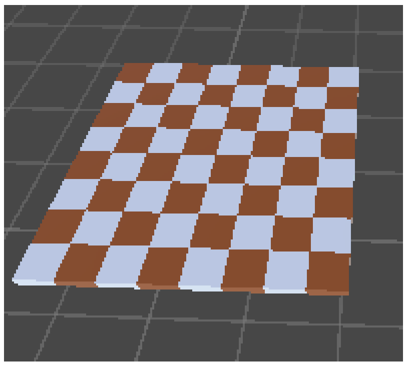
```
/// <summary>
/// this method calculates where the squares must be positioned
/// </summary>
/// <param name="position">position of the last square</param>
/// <returns>the new position</returns>
2 references
private Vector3 calculatePosition(Vector3 position,float xSize,float zSize,bool xOnly)
```



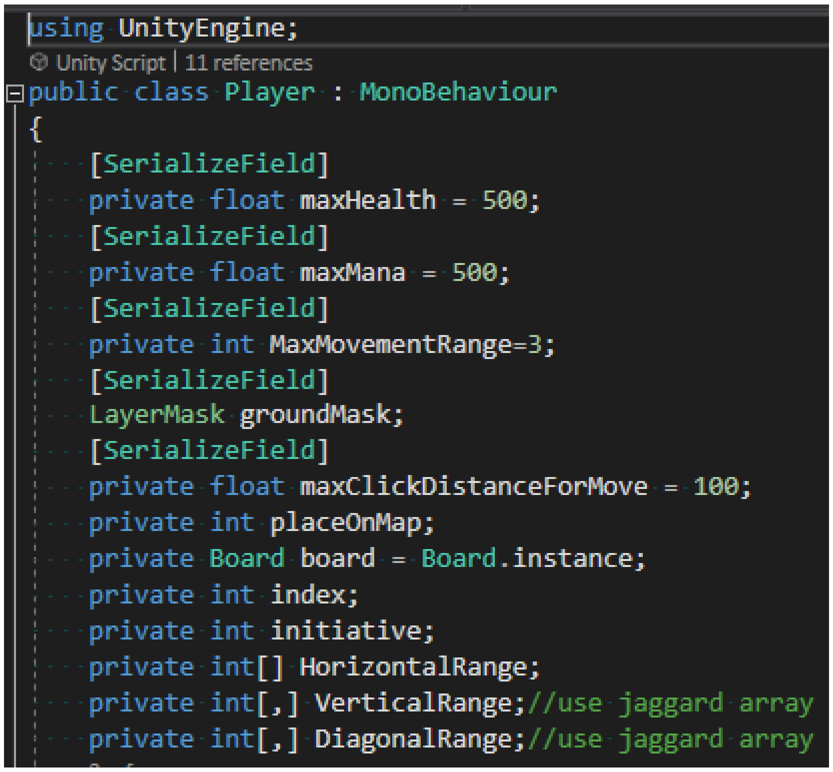
```
private Vector3 calculatePosition(Vector3 position,float xSize,float zSize,bool xOnly)
{//use size of square to get poos
    float x, y, z;
    Vector3 results = new Vector3(0,0,0);
    if (xOnly)
    {
        x = position.x + xSize;
        y = position.y;
        z = position.z;
        return results = new Vector3(x, y, z);
    }
    else
    {
        x = position.x;
        y = position.y;
        z = position.z - zSize;
        results = new Vector3(x, y, z);
        StartPostion = results;
        return results;
    }
}
```



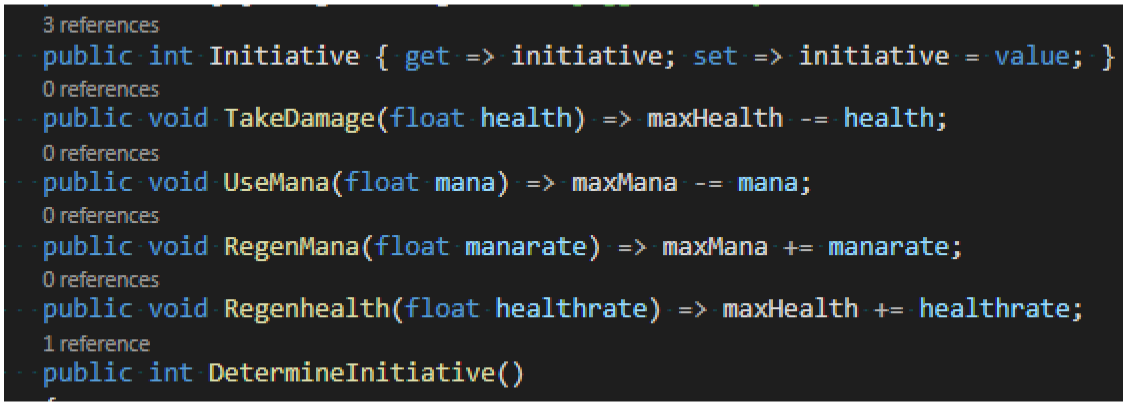
Above Code Generates this board here below from a single tile



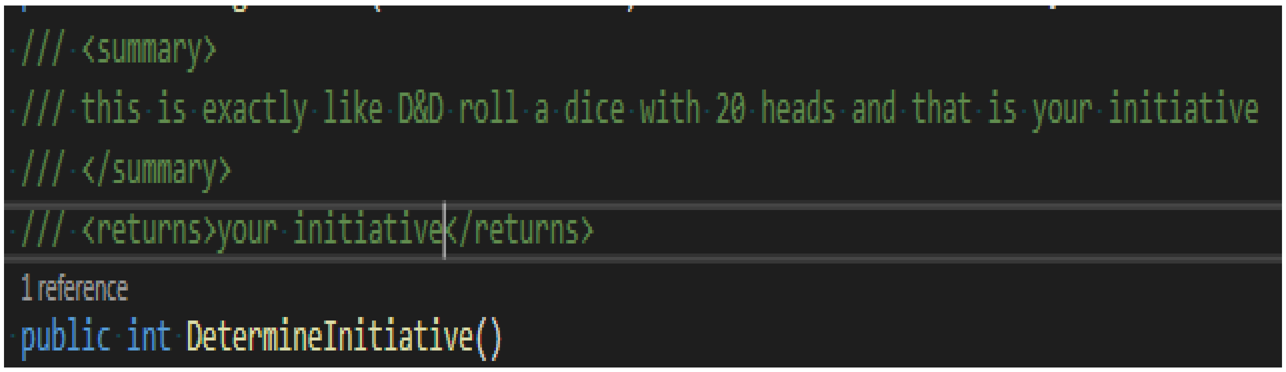
Player Controller



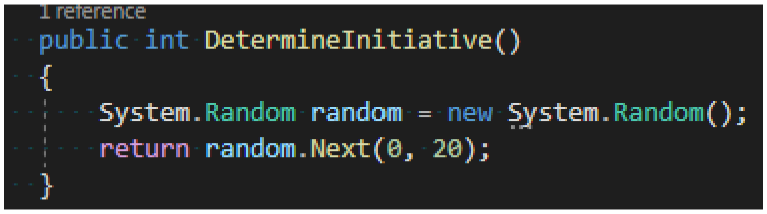
```
using UnityEngine;
Unity Script | 11 references
public class Player : MonoBehaviour
{
    [SerializeField]
    private float maxHealth = 500;
    [SerializeField]
    private float maxMana = 500;
    [SerializeField]
    private int MaxMovementRange=3;
    [SerializeField]
    LayerMask groundMask;
    [SerializeField]
    private float maxClickDistanceForMove = 100;
    private int placeOnMap;
    private Board board = Board.instance;
    private int index;
    private int initiative;
    private int[] HorizontalRange;
    private int[,] VerticalRange;//use jaggard array
    private int[,] DiagonalRange;//use jaggard array
```



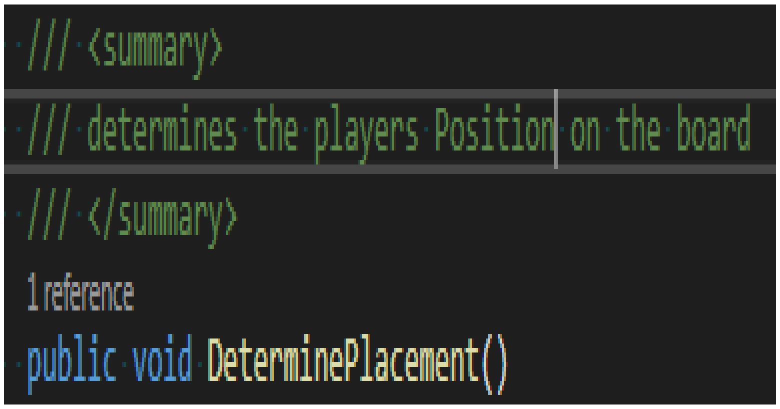
```
    3 references
    public int Initiative { get => initiative; set => initiative = value; }
    0 references
    public void TakeDamage(float health) => maxHealth -= health;
    0 references
    public void UseMana(float mana) => maxMana -= mana;
    0 references
    public void RegenMana(float manarate) => maxMana += manarate;
    0 references
    public void Regenhealth(float healthrate) => maxHealth += healthrate;
    1 reference
    public int DetermineInitiative()
```



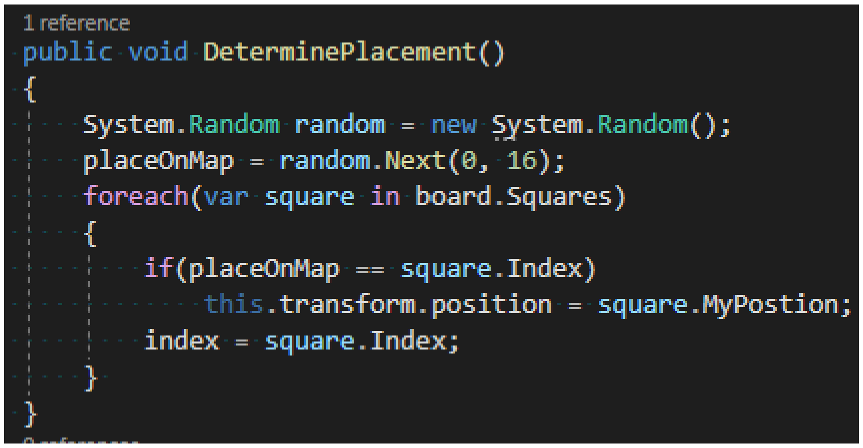
```
/// <summary>
/// this is exactly like D&D roll a dice with 20 heads and that is your initiative
/// </summary>
/// <returns>your initiative</returns>
1 reference
public int DetermineInitiative()
```



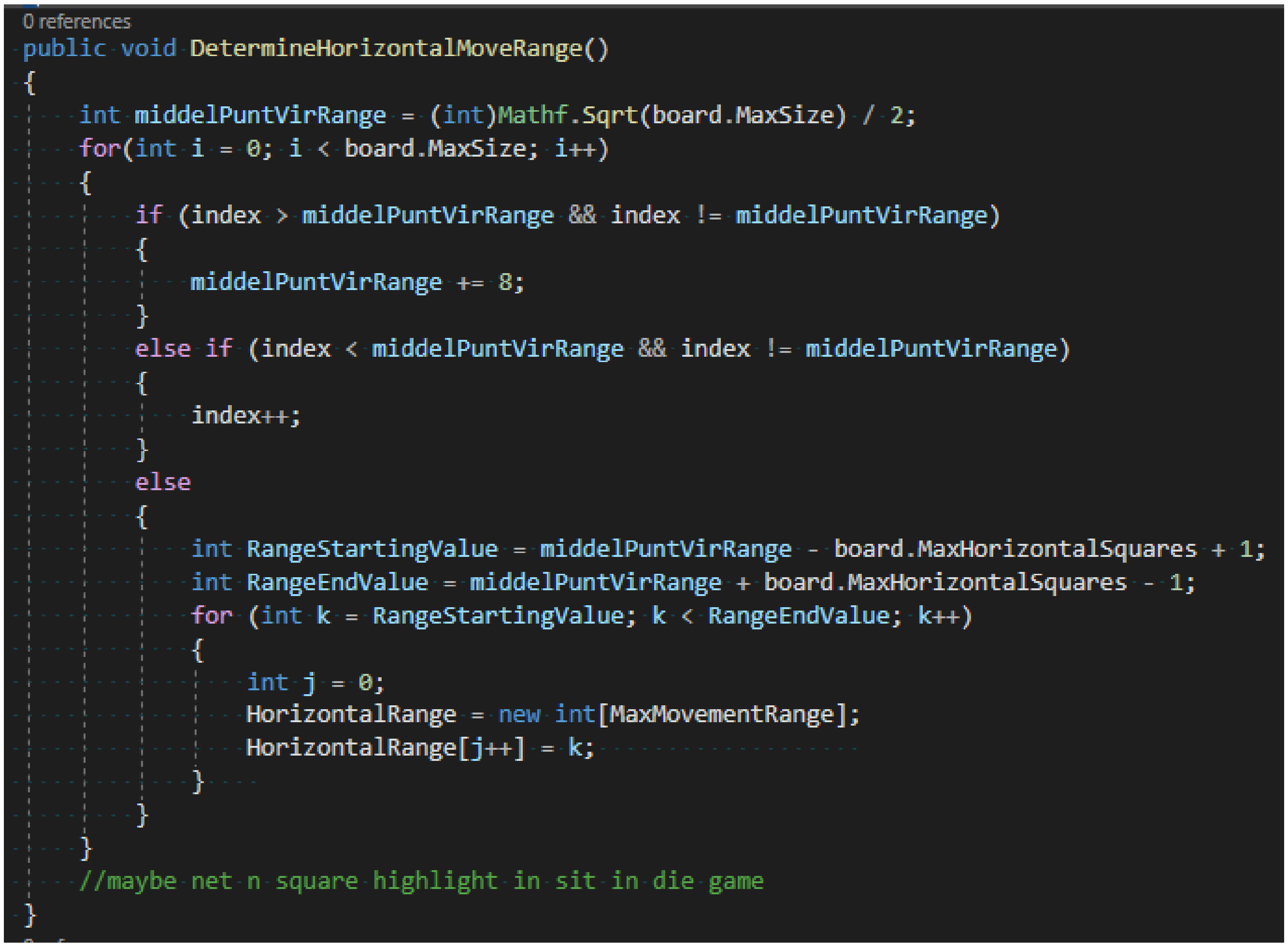
```
1 reference
public int DetermineInitiative()
{
    System.Random random = new System.Random();
    return random.Next(0, 20);
}
```



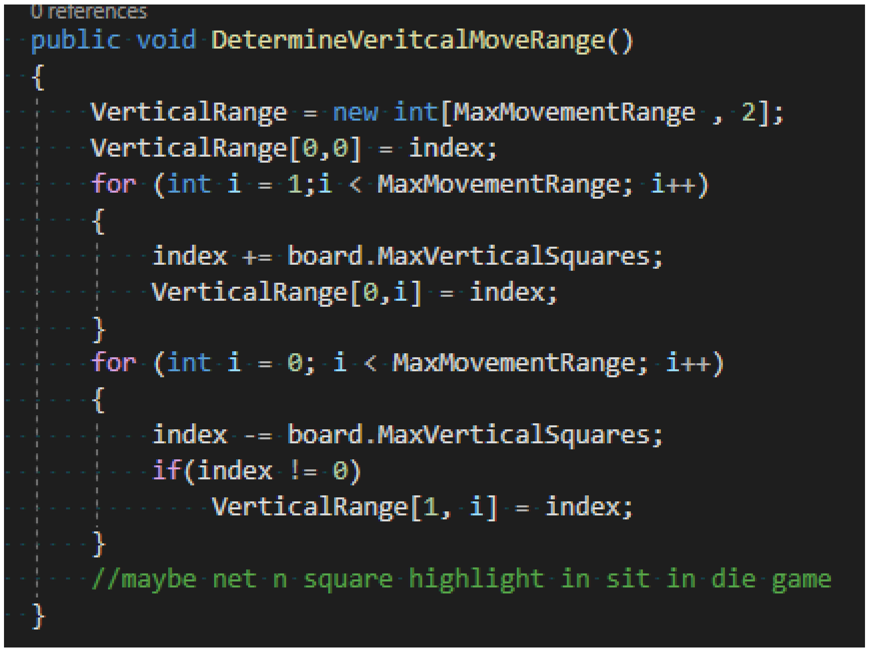
```
/// <summary>
/// determines the players Position on the board
/// </summary>
1 reference
public void DeterminePlacement()
```



```
1 reference
public void DeterminePlacement()
{
    System.Random random = new System.Random();
    placeOnMap = random.Next(0, 16);
    foreach(var square in board.Squares)
    {
        if(placeOnMap == square.Index)
            this.transform.position = square.MyPostion;
        index = square.Index;
    }
}
```

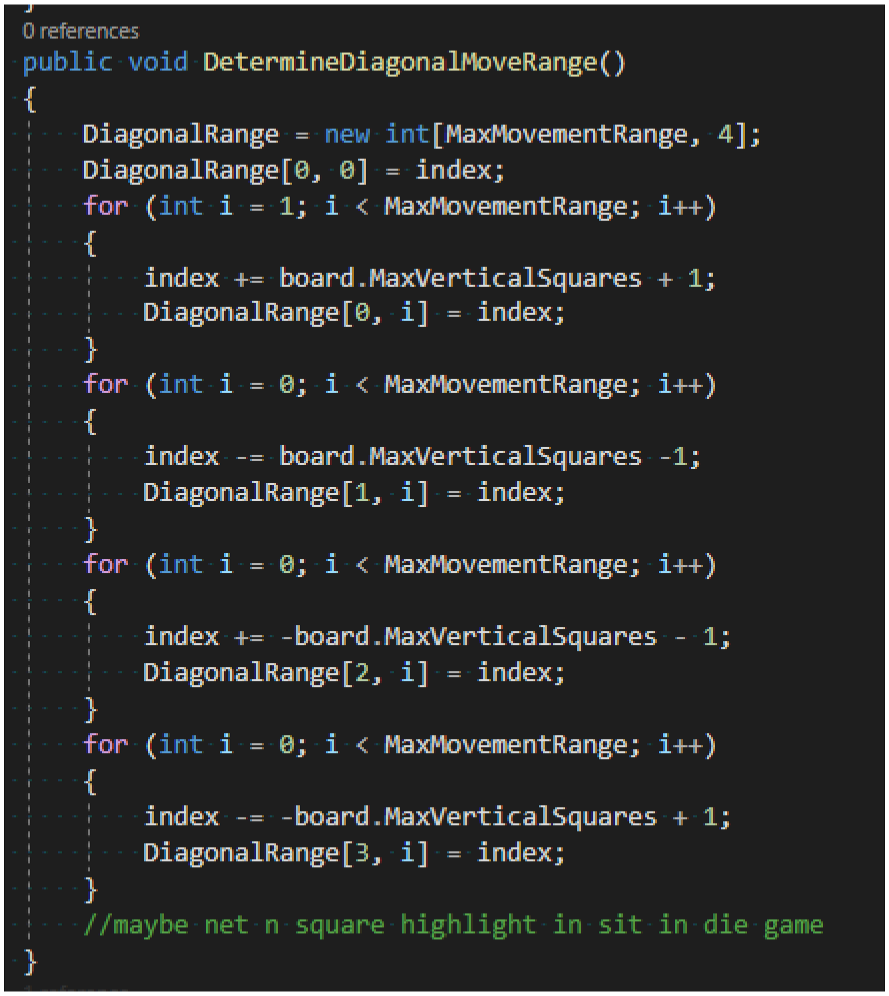


```
0 references
public void DetermineHorizontalMoveRange()
{
    int middelPuntVirRange = (int)Mathf.Sqrt(board.MaxSize) / 2;
    for(int i = 0; i < board.MaxSize; i++)
    {
        if (index > middelPuntVirRange && index != middelPuntVirRange)
        {
            middelPuntVirRange += 8;
        }
        else if (index < middelPuntVirRange && index != middelPuntVirRange)
        {
            index++;
        }
        else
        {
            int RangeStartingValue = middelPuntVirRange - board.MaxHorizontalSquares + 1;
            int RangeEndValue = middelPuntVirRange + board.MaxHorizontalSquares - 1;
            for (int k = RangeStartingValue; k < RangeEndValue; k++)
            {
                int j = 0;
                HorizontalRange = new int[MaxMovementRange];
                HorizontalRange[j++] = k;
            }
        }
    }
    //maybe net n square highlight in sit in die game
}
```



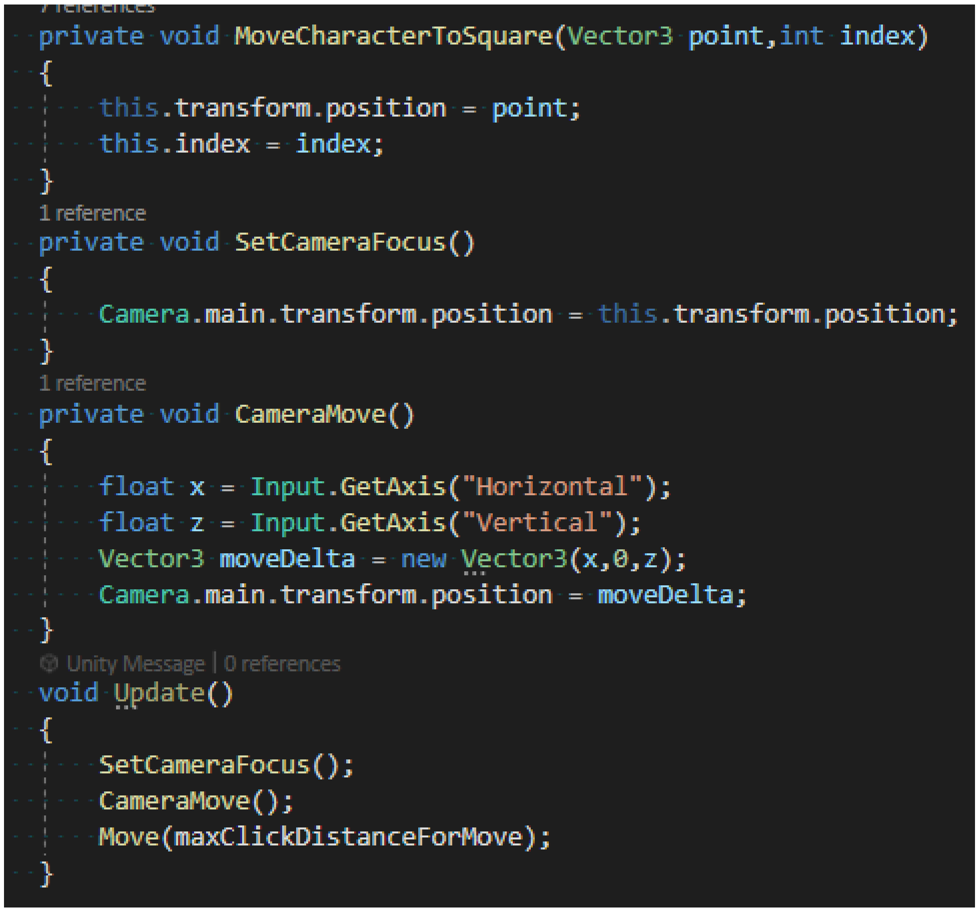
```
0 references
public void DetermineVeritcalMoveRange()
{
    VerticalRange = new int[MaxMovementRange , 2];
    VerticalRange[0,0] = index;
    for (int i = 1;i < MaxMovementRange; i++)
    {
        index += board.MaxVerticalSquares;
        VerticalRange[0,i] = index;
    }
    for (int i = 0; i < MaxMovementRange; i++)
    {
        index -= board.MaxVerticalSquares;
        if(index != 0)
            VerticalRange[1, i] = index;
    }
    //maybe net n square highlight in sit in die game
}
```

The next method I need to improve I don't like it



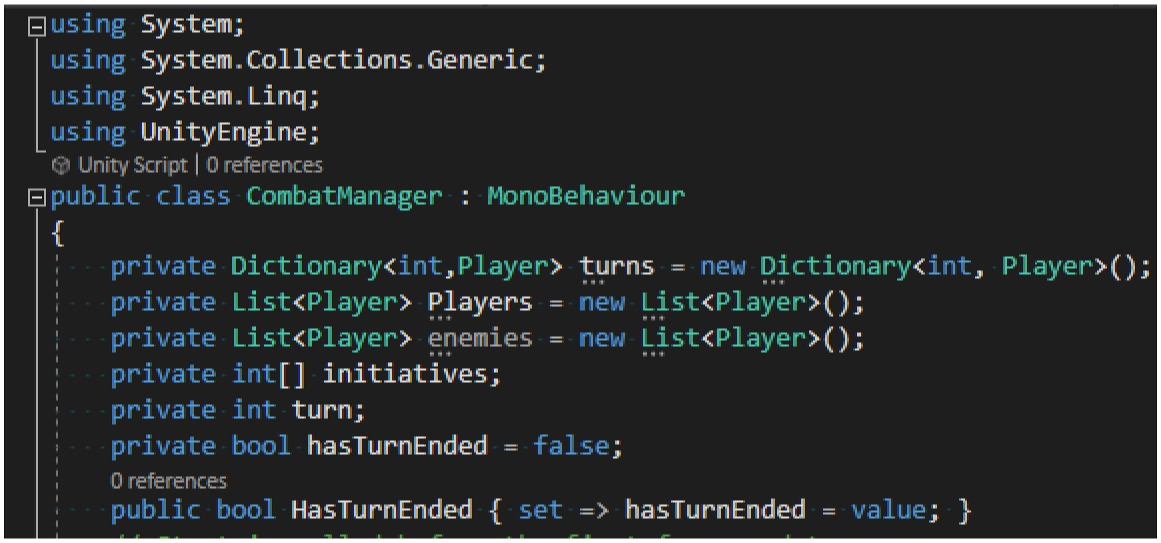
```
0 references
public void DetermineDiagonalMoveRange()
{
    DiagonalRange = new int[MaxMovementRange, 4];
    DiagonalRange[0, 0] = index;
    for (int i = 1; i < MaxMovementRange; i++)
    {
        index += board.MaxVerticalSquares + 1;
        DiagonalRange[0, i] = index;
    }
    for (int i = 0; i < MaxMovementRange; i++)
    {
        index -= board.MaxVerticalSquares -1;
        DiagonalRange[1, i] = index;
    }
    for (int i = 0; i < MaxMovementRange; i++)
    {
        index += -board.MaxVerticalSquares - 1;
        DiagonalRange[2, i] = index;
    }
    for (int i = 0; i < MaxMovementRange; i++)
    {
        index -= -board.MaxVerticalSquares + 1;
        DiagonalRange[3, i] = index;
    }
    //maybe net n square highlight in sit in die game
}
```

This above is too much. Best thing that came to mind at that time sadly.

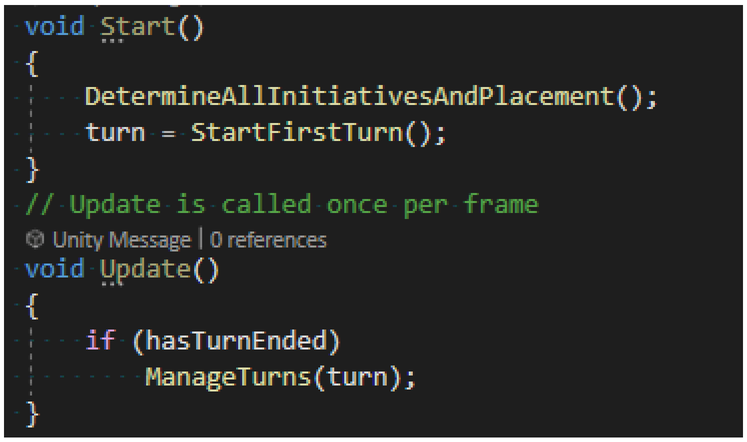


```
private void MoveCharacterToSquare(Vector3 point,int index)
{
    this.transform.position = point;
    this.index = index;
}
1 reference
private void SetCameraFocus()
{
    Camera.main.transform.position = this.transform.position;
}
1 reference
private void CameraMove()
{
    float x = Input.GetAxis("Horizontal");
    float z = Input.GetAxis("Vertical");
    Vector3 moveDelta = new Vector3(x,0,z);
    Camera.main.transform.position = moveDelta;
}
Unity Message | 0 references
void Update()
{
    SetCameraFocus();
    CameraMove();
    Move(maxClickDistanceForMove);
}
```

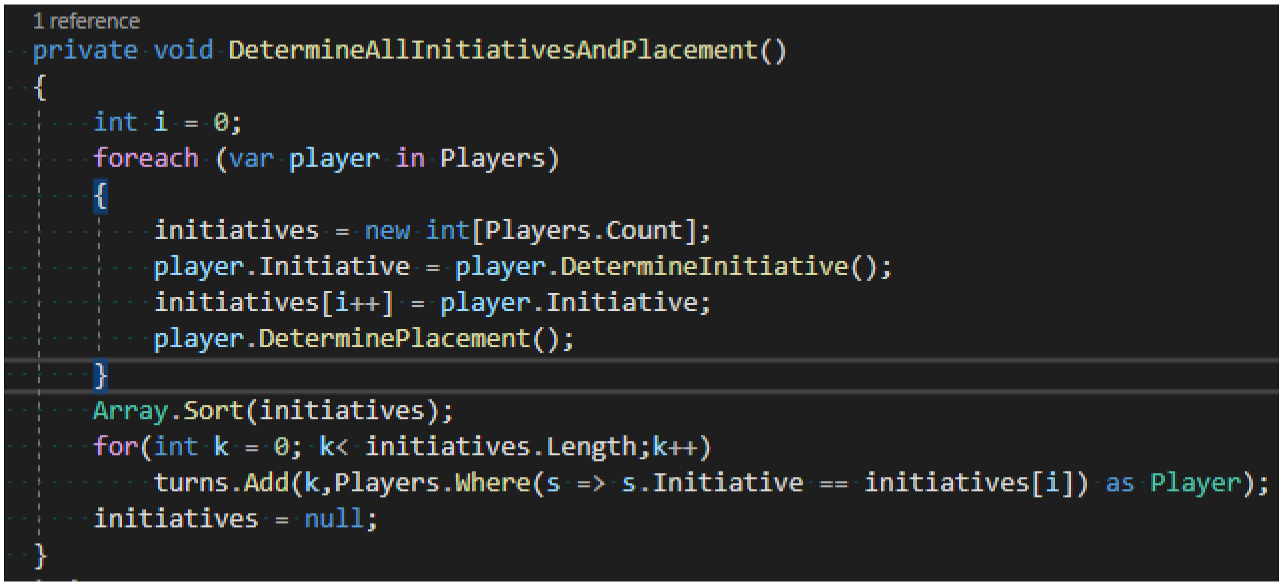
There are still a lot of work (other methods and such) that I need to implement here I



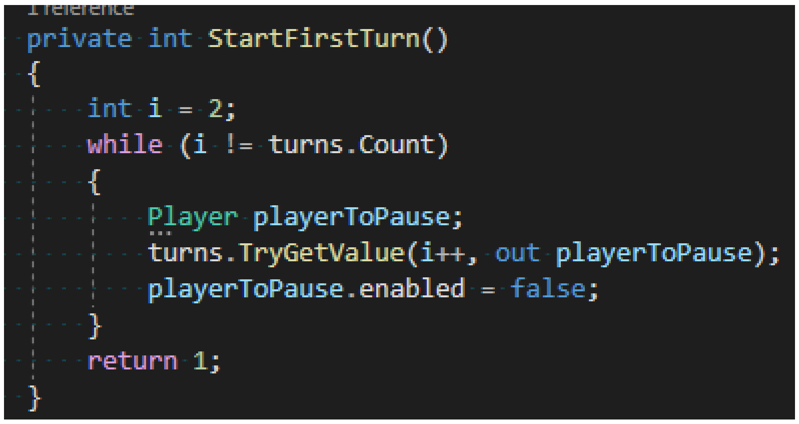
```
using System;
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
Unity Script | 0 references
public class CombatManager : MonoBehaviour
{
    private Dictionary<int,Player> turns = new Dictionary<int, Player>();
    private List<Player> Players = new List<Player>();
    private List<Player> enemies = new List<Player>();
    private int[] initiatives;
    private int turn;
    private bool hasTurnEnded = false;
    0 references
    public bool HasTurnEnded { set => hasTurnEnded = value; }
```



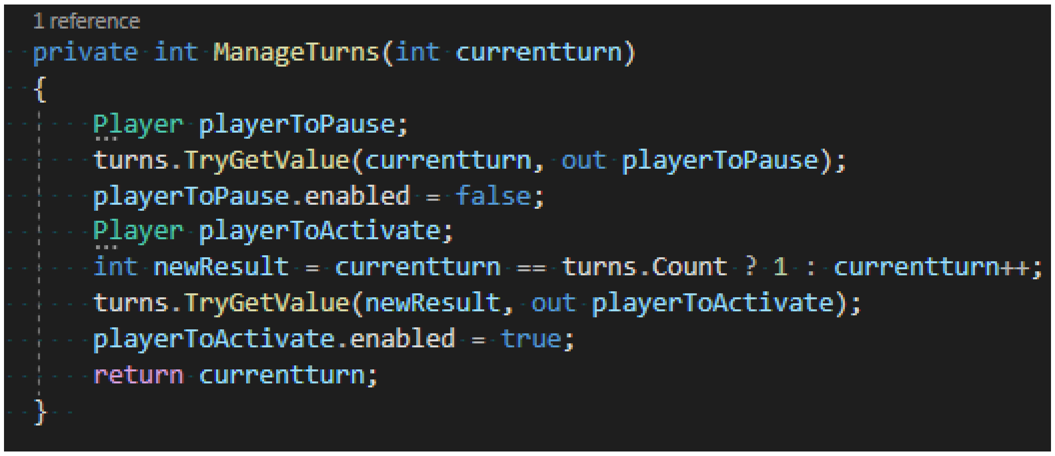
```
void Start()
{
    DetermineAllInitiativesAndPlacement();
    turn = StartFirstTurn();
}
// Update is called once per frame
Unity Message | 0 references
void Update()
{
    if (hasTurnEnded)
        ManageTurns(turn);
}
```



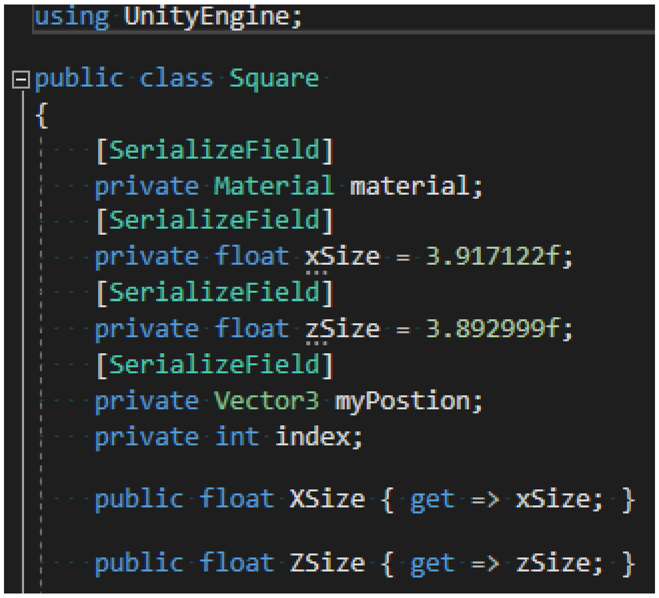
```
1 reference
private void DetermineAllInitiativesAndPlacement()
{
    int i = 0;
    foreach (var player in Players)
    {
        initiatives = new int[Players.Count];
        player.Initiative = player.DetermineInitiative();
        initiatives[i++] = player.Initiative;
        player.DeterminePlacement();
    }
    Array.Sort(initiatives);
    for(int k = 0; k< initiatives.Length;k++)
        turns.Add(k,Players.Where(s => s.Initiative == initiatives[i]) as Player);
    initiatives = null;
}
```



```
1 reference
private int StartFirstTurn()
{
    int i = 2;
    while (i != turns.Count)
    {
        Player playerToPause;
        turns.TryGetValue(i++, out playerToPause);
        playerToPause.enabled = false;
    }
    return 1;
}
```



```
1 reference
private int ManageTurns(int currentturn)
{
    Player playerToPause;
    turns.TryGetValue(currentturn, out playerToPause);
    playerToPause.enabled = false;
    Player playerToActivate;
    int newResult = currentturn == turns.Count ? 1 : currentturn++;
    turns.TryGetValue(newResult, out playerToActivate);
    playerToActivate.enabled = true;
    return currentturn;
}
```

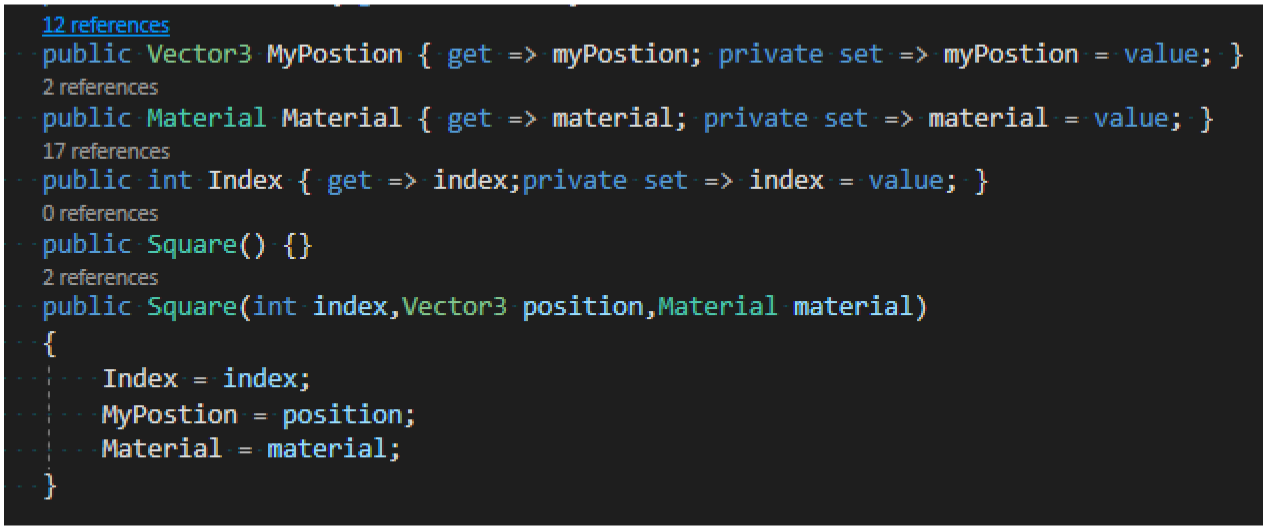


```
using UnityEngine;

public class Square
{
    [SerializeField]
    private Material material;
    [SerializeField]
    private float xSize = 3.917122f;
    [SerializeField]
    private float zSize = 3.892999f;
    [SerializeField]
    private Vector3 myPostion;
    private int index;

    public float XSize { get => xSize; }

    public float ZSize { get => zSize; }
```



```
12 references
public Vector3 MyPostion { get => myPostion; private set => myPostion = value; }
2 references
public Material Material { get => material; private set => material = value; }
17 references
public int Index { get => index;private set => index = value; }
0 references
public Square() {}
2 references
public Square(int index,Vector3 position,Material material)
{
    Index = index;
    MyPostion = position;
    Material = material;
}
```