~\OneDrive - St Paul's Catholic College\Documents\2D Strategy Game - Liberator\Assets\Scripts\FieldManager.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FieldManager : MonoBehaviour
{
    // we want to have an array that contains all rows in the scene, since all rows have row manager component, we can use this type declaring the array
    public RowManager[] allRows;
    // With a static method you do not need an instance of the class to call the method, just the class
    public static HexData [,] allHexesArray; // contains all hexes in the Battlefield
    int allRowsLength; // this is how many rows there actually is
    public Sprite availableAsTarget; // green frame
    public Sprite notAavailable; // empty, red frame
    public Sprite availableToMove; // white frame
    [SerializeField] SpriteRenderer MonsterPrefab; // monster prefab in a field within the FieldManager Script
    private int numOfMonsters = 15; // the amount of monsters spawned on the map

    void Awake() // awake will execute the code sooner than start. This is to make sure all hexes in the field are created before working with them
    {
        // We want to combine all rows in one array. We remember all objects with the row manager component are children of the object field manager component
        // GetComponentsInChildren returns all components of defined type in the game object or any of its children
        allRows = GetComponentsInChildren<RowManager>(); // gets all the rows
        allRowsLength = allRows.Length;

        for (int i = 0; i < allRowsLength; i++)
        {
            // gets all the hexes in the row of all 16 rows and puts them into the array allHexesInRow
            allRows[i].allHexesInRow = allRows[i].GetComponentsInChildren<HexData>();
        }
        CreateAllHexesArray(); // creating the battlefield array
    }

    private void Start()
    {
        AvailablePos Soldier = FindObjectOfType<AvailablePos>(); // getting the soldier
        IAdjacentFinder adjFinder = new PositionsForSoldier(); // getting the position for the soldier
        HexData startingHex = Soldier.GetComponentInParent<HexData>(); // getting the starting hex based on where the soldier is
        int stepsLimit = Controller.soldier.steps;
        startingHex.DefineMeAsStartingHex(); // all the characteristics of the starting hex assigned to it from the function in the HexData script
        Soldier.GetAvailablePositions(Soldier.GetComponentInParent<HexData>(), stepsLimit, adjFinder); // making all positions available
        SpawnMonsters(); // creating monsters randomly across the map
    }

    private void CreateAllHexesArray()
    {
        int heightOfArray = allRows.Length; // this gets the value of the variable allrows which is 16
```

```
48          int widthOfArray = allRows[0].allHexesInRow.Length; // this get the value of all the hexes in the first row, which is 19
49          allHexesArray = new HexData[widthOfArray, heightOfArray]; // creating the battlefield array
50
51      for (int i = 0; i < heightOfArray; i++) // loops through 19 times because heightofarray is 19
52      {
53          // this loop assings coordinate values to variables to each battlefield hex in every row starting from i
54          for (int j=0; j < widthOfArray; j++)
55          {
56              // widthofarray - j - 1 = 19 -i -1 | heightOfArray -i - 1 = 16 - i - 1 |
57              allHexesArray[j, i] = allRows[heightOfArray - i - 1].allHexesInRow[widthOfArray - j - 1]; // getting the first row i from the hexfield, then entering the allhexesinrow array, then getting all the hexes in the row
58              allHexesArray[j, i].verticalCoordinate = i+1; // vertical coordinate is simply equal to i + 1
59              allHexesArray[j, i].horizontalCoordinate = j + 1; // we start coordinate system by (1,1)
60              // first loop does: allHexesArray[0,0] = allRows[16 - 0 - 1].allHexesInRow[19 - 0 - 1]
61              // allHexesArray[0,0].verticalCoordinate = 0 + 1
62              // allHexesArray[0,0].horizontalCoordinate = 0 + 1
63          }
64      }
65      print(allHexesArray);
66    }
67     private void SpawnMonsters() // function that spawns monsters randomly across the map except on mountains and water
68     {
69      for (int i = 0; i < numOfMonsters; i++)
70      {
71          int horizontal = UnityEngine.Random.Range(0, 19);
72          int vertical = UnityEngine.Random.Range(0,16);
73
74          if (allHexesArray[horizontal, vertical].tag != "Water" && allHexesArray[horizontal, vertical].tag != "Mountain")
75          {
76              var monster = Instantiate(MonsterPrefab, allHexesArray[horizontal, vertical].transform.position, allHexesArray[horizontal, vertical].transform.rotation);
77              monster.transform.parent = allHexesArray[horizontal, vertical].transform;
78          }
79      }
80     }
81 }
82
```