地图生成系统

重点部分：

1. 地图地面：如何生成整张大地图
2. 数据：如何保存和更新地图数据
3. UI：如何生成和更新小地图

核心脚本：

1. MapManager.cs
2. MapChunkController.cs
3. MapGrid.cs
4. MapGenerate.cs
5. MapData.cs
6. MapConfig.cs

地图管理器 MapManager

说明：主要功能有地图地面生成、地图对象生成、AI对象生成、地图数据存储管理

初始化时生成地图：先找到数据 → 初始化地图内容（碰撞体、AI导航网格、根据存档恢复地图对象、AI对象、更新可见地图块） → 初始化UI

1. 找到地图存档：找到玩家设定的**地图初始化参数存档MapInitData**和**地图数据存档MapData**
2. 确定地图对象和AI对象生成配置：根据地图/AI对象生成配置生成一个临时字典，字典Key为顶点类型, Value为配置Id
3. 初始化地图生成器：生成地图数据（GenerateMapData）
    a. 生成网格顶点数据：设置地图真实长宽以及cell的大小，默认mapSize=20（chunk的个数），mapChunk=5（cell个数），cellSize=2（unity标准格子的个数）
    b. 使用玩家设定的随机种子生成柏林噪声图
    c. 确定各个顶点的类型以及计算周围网格贴图的索引数字
    d. 生成地面Mesh
    e. 设定地图物品生成随机种子
    f. 计算地图物品配置总权重和计算AI对象配置总权重

4. 初始化地面碰撞体网格，<生成整个地面的mesh>
5. <烘焙导航网格>：后续AI巡逻需要使用到
6. 根据存档情况判断是否需要加载之前的地图（生成地图块数据）
    a. 旧存档加载：
        i. 根据存档数据设置地图块加载最大数量，初始化加载进度条
        ii. 遍历所有的MapChunkIndexList，根据地图块索引<获取单个地图块数据>，根据下标和数据<生成地图块管理器>（管理器中数据和一些显示的对象，例如树木、AI等）
    b. 新存档创建：
        i. <游戏初始化时获得地图块的初始数量>
        ii. 根据玩家所在地图块计算可视范围内所有地图块索引并<生成地图块管理器>
7. 更新目前可见的地图块
8. 更新和关闭小地图UI（刷新一次）

生成地图块管理器

1. 传入地图块索引和地图块数据（可能为空）
2. 判断地图块索引是否合法
3. 使用（地图生成器）<在指定位置生成地图块控制器>MapChunkController并返回

游戏初始化时获得地图块的初始数量

1. 获得玩家在地图上的地图块索引
2. 根据玩家可视范围确定显示地图块索引，例如可视距离为1的话，那玩家可视范围则为 x → (-1, 1); y → (-1, 1) 。判断玩家可视范围是否超过地图边界，对于在边界内的地图块，需要+=到初始化数量上

地图生成器：MapGenerate

在指定位置生成地图块控制器`MapChunkController GenerateMapChunk`

1. 实例化GameObject：生成地图块GameObject并挂载MapChunkController、MeshFilter组件

```
// 生成地图块物体
GameObject mapChunkObj = new GameObject("Chunk_" + chunkIndex.ToString());
// 挂载脚本和组件
MapChunkController mapChunk = mapChunkObj.AddComponent<MapChunkController>();
mapChunkObj.AddComponent<MeshFilter>().mesh = mapChunkMesh;
```

2. 生成地图块贴图：使用协程分帧执行提高效率

a. 渲染当前地图贴图，详情见<分帧生成地图块贴图**>，**返回texture和isAllForest标记

b. 添加MeshRenderer组件

i. 如果当前地图块全是森林则使用共享材质，则在添加完MeshRenderer组件后设置shareMaterial为地图配置中的默认material `mapChunkObj.AddComponent<MeshRenderer>().sharedMaterial = mapConfig.mapMaterial;`

ii. 如果当前地图块中存在沼泽则创建一个Material（材质为沼泽材质），同时设置mainTexture设置为<分帧生成地图块贴图**>**中返回的带有沼泽的纹理。然后添加MeshRenderer组件，将material设置为刚刚的material

3. 设置MapChunkController的position和parent
4. 初始化地图块数据

a. 如果没有存档数据则<生成地图块数据>然后<添加并保存单个地图块数据>

b. 如果有存档数据则<恢复地图块顶点数据**>**

分帧生成地图块贴图

1. 计算当前地图块的偏移量, 找到这个地图块每块具体的格子位置并统计出所有格子是否都为森林
2. 如果都为森林则不对地面进行渲染（地面默认是森林）
3. 如果包含沼泽则需要遍历MapChunk中包含的所有格子单独进行渲染

a. 约定好贴图都是矩形, 计算整个地图块texture的宽高（cell数 * 单个贴图的宽/高）

```
int textureCellSize = mapConfig.forestTexture.width;
int textureSize = mapConfig.mapChunkSize * textureCellSize;
mapTexture = new Texture2D(textureSize, textureSize, TextureFormat.RGB24, false);
```

b. 遍历所有格子并绘制格子中的每个像素点，从左下到右上依次绘制每个Cell中的像素，通过Texture2D中的SetPixel在指定位置设置像素点。需要注意沼泽贴图中像素点是透明的情况也需要绘制forestTexture同位置的像素颜色

```
Color color = mapConfig.forestTexture.GetPixel(x1, z1);
mapTexture.SetPixel(x1 + pixelOffsetX, z1 + pixelOffsetZ, color);
```

c. 最后设置一下Texture2D的filterMode和wrapMode然后Apply修改后的Texture2D

恢复地图块顶点数据

由于地图块中其他数据已经通过存档管理器存入了mapChunkData中，仅顶点数据无法序列化因而未存储到mapChunkData中，所以需要单独恢复

1. 根据传入的地图索引得到世界坐标
2. 遍历当前地图块中所有cell的顶点类型，添加到mapChunkData中

在地图块刷新时生成地图对象

地图块控制器 MapChunkController

注意：地图块控制器是在地图块GameObject对象实例化的时候挂载到其身上的，因此控制器只用来存储当前地图块数据、动态生成和回收地图上的物体、控制当前地图块上的物体显示/隐藏等功能，并不提供初始化地图块GameObject的方法

初始化

1. 根据传入的地图块索引、地图块数据初始化控制器中的地图数据，例如地图/AI对象实例和地图/AI对象数据。根据mapObjectDataDict生成后续可能需要销毁的地图对象字典，例如地面上的武器当到达销毁天数时需要在地图上移除该对象
2. 添加<地图块刷新事件>

实例化地图对象

1. 根据传入的地图对象数据拿到对应id并获取到配置信息
2. 从对象池中实例化一个地图对象并得到MapObjectBase组件

移除一个地图对象

1. 数据层面：根据传入的id从地图对象数据字典（MapObjectDataDict）中移除对应数据，并将out出的数据放入对象池
2. 显示层面：1. 根据传入的id从地图对象字典（MapObjectDict）中移除对象，并将out出的MapObjectBase放入对象池；2. 在小地图UI上直接<移除地图对象icon>

添加一个地图对象

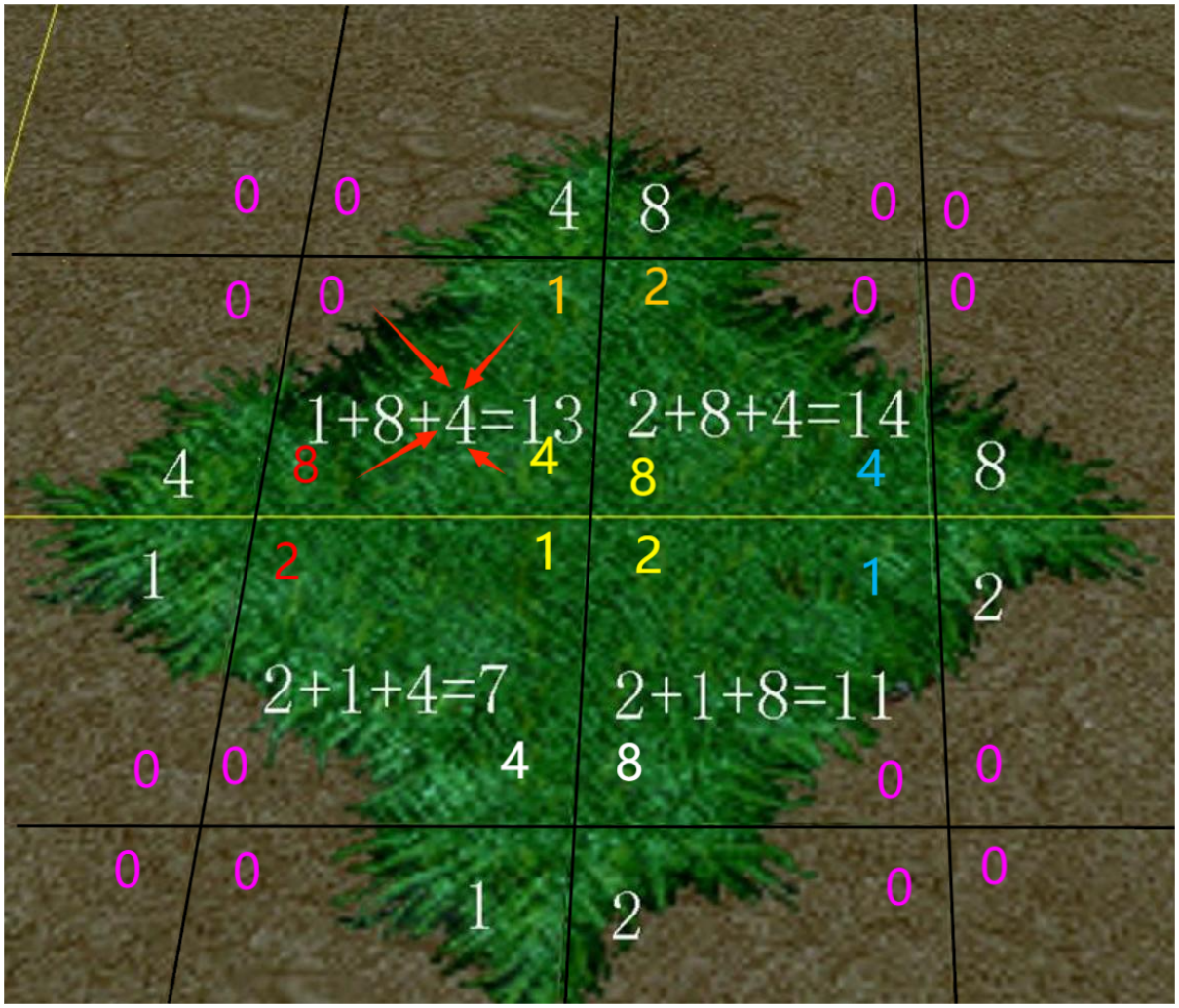
1. 数据层面：根据传入的数据，直接在地图对象数据字典中添加该数据，如果该地图对象数据存在销毁天数则将该地图对象加入待销毁地图字典中
2. 显示层面：如果当前地图块已显示（isActive==true）则需要<实例化地图对象>

地图块刷新事件：移除满足天数的地图物体，更新

1. 移除地图块上的物体：
    a. 遍历所有可能需要销毁的地图对象，更新剩余时间
    b. 遍历所有剩余时间为0的地图对象，直接<移除一个地图对象>
2. 在地图块上刷新物体：
    a. <在地图块刷新时生成地图对象>，<添加一个地图对象>到地图块控制器中

生成通用地图块数据`GenerateMapData()`

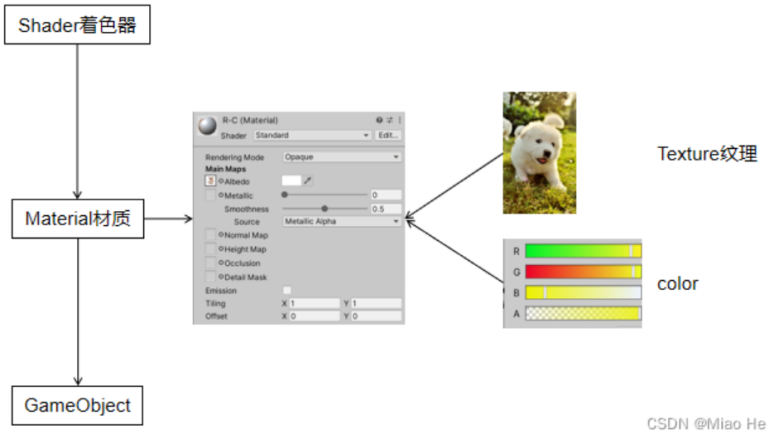
1. 统一生成整张地图网格/顶点数据
2. 使用随机种子生成随机柏林噪声图（Unity.Random.InitState）
3. 确定各个顶点的类型以及计算周围网格贴图的索引数字
    a. 通过魔兽争霸地形贴图拼接算法生成更合理的地形
    https://blog.csdn.net/twopointfive/article/details/104255705
    b.



4. 根据顶点类型实例化森林和沼泽材质
    a. 需要注意Material / Texture / Shader的关系

纹理、着色器与材质的关系：

Shader着色器的属性值配置可以在Material实现，而Material材质在Mesh Renderer中设置，且Mesh Renderer在GameObject物体中。三者关系如下图：



地图块生成流程

数据：

1. 需要生成的**数据**

```
MapChunkData mapChunkData = new MapChunkData();
mapChunkData.mapObjectDataDict = new Serialization_Dict<ulong, MapObjectData>();
mapChunkData.AIDataDict = new Serialization_Dict<ulong, MapObjectData>();
mapChunkData.forestVertexList = new List<MapVertex>();
mapChunkData.marshVertexList = new List<MapVertex>();
```

2. 顶点数据：可以直接根据世界坐标从地图数据中获取
3. MapChunk中包含的地图对象数据`mapObjectDataDict`
    a. 遍历MapChunk中包含的Cell
        i. MapCell顶点类型 == 空 不生成地图对象
        ii. MapCell顶点类型 != 空 则根据MapCell顶点类型和地图对象随机生成权重去生成对应结果，其中MapObject Id由MapData统一管理，每次创建新对象Id + 1

```
private MapObjectData GenerateMapObjectData(int mapObjectConfigId, Vector3 position, int destoryDay) {
    MapObjectData mapObjectData = PoolManager.Instance.GetObject<MapObjectData>();
    mapObjectData.id = mapData.currentId;
    mapData.currentId += 1;
    mapObjectData.configId = mapObjectConfigId;
    mapObjectData.position = position;
    mapObjectData.destoryDay = destoryDay;
```

```
    return mapObjectData;
}
```

4. MapChunk中包含的AI对象数据`AIDataDict`
    a. 首先判断当前MapChunk中包含的顶点类型数量是否超过阈值，例如沼泽顶点数量少的话意味着沼泽很小不适合在该地形中放置AI物体

地图顶点/网格数据：MapGrid

注意：MapGrid是构建整个地图的坐标关系，MapChunkIndex，MapCellIndex，WorldPosition是三种不同的坐标，他们之间有关联，目的是通过真实的世界坐标能够找到具体的chunk、cell即可

主要功能：

1. 添加地图顶点（Vertex）：`Dictionary<Vector2Int, MapVertex> vertexDict`
    a. 其中Key是MapCell坐标体系中的具体坐标（该坐标应该可以通过MapChunk去索引到），Value中position真是世界坐标
    b. 顶点坐标中包含世界坐标position、顶点类型vertextType、当前Cell中地图对象Id mapObjectId
2. 添加地图网格（Cell）：`Dictionary<Vector2Int, MapCell> cellDict`
    a. 其中Key是MapCell坐标体系中的具体坐标（该坐标应该可以通过MapChunk去索引到），Value中position真是世界坐标左下角的一处坐标 `new MapCell() { position = new Vector3(x * cellSize - offset, 0, z * cellSize - offset) }` 这样做的是贴图算法具体执行逻辑导致的
    b. 地图网格中包含贴图位置position和贴图素材textureIndex
3. 计算格子贴图的索引数字：通过noiseMap的值设定Cell地图顶点类型

地图系统数据结构

玩家设定的地图初始化参数存档：

地图数据存档：按照地图上包含对象的层级可以分成 MapData→MapChunkData→MapObjectData

1. 地图数据`MapData`
2. 地图块数据`MapChunkData`
3. 地图上的物体数据`MapObjectData`

```
// 地图初始化数据
[Serializable]
public class MapInitData {
    public int mapSize;         // 地图大小(地图块数量)
    public int mapSeed;         // 地图随机种子
    public int spawnSeed;       // 地图对象随机种子
    public float marshLimit;    // 沼泽生成变量
```

```
}

// 地图数据
[Serializable]
public class MapData {
    // 当前地图对象id取值
    public ulong currentId = 1;             // 整个地图当前生成对象Id
    // 地图块索引列表: 已经生成过的所有地图块
    public List<Serialization_Vector2> MapChunkIndexList = new List<Serialization_Vector2>();   // 地图上包含的地图块索引
}

// 地图块对象数据
[Serializable]
public class MapObjectData {
    public ulong id;                                // 地图中地图对象id
    public int configId;                            // 地图物体配置id
    public int destoryDay;                          // 地图对象销毁天数, -1代表无效
    private Serialization_Vector3 sv_position;      // 坐标: sv_postion存档用, position外部调用用
    public Vector3 position {
        get => sv_position.ConverToVector3();
        set => sv_position = value.ConverToSVector3();
    }
}

// 地图块数据
[Serializable]
public class MapChunkData {
    // 当前地图上的所有地图对象
    public Serialization_Dict<ulong, MapObjectData> mapObjectDataDict;  // 地图对象字典, 使用Id检索地图对象
    public Serialization_Dict<ulong, MapObjectData> AIDataDict;         // AI对象字典, 使用Id检索AI对象
    // 记录当前地图块顶点数量
    [NonSerialized]
    public List<MapVertex> forestVertexList;            // 记录当前森林顶点数量
    [NonSerialized]
    public List<MapVertex> marshVertexList;             // 记录当前沼泽顶点数量
}
```

地图系统配置

地图配置：

```
public class MapConfig : ConfigBase {
    [LabelText("地图块包含网格数")]
    public int mapChunkSize;                // 地图块大小
    [LabelText("地图块网格大小")]
    public float cellSize;                  // 网格大小
    [LabelText("噪声图采样间隔大小")]
    public float noiseLacunarity;           // 噪声图采样间隔大小
    [LabelText("森林贴图")]
    public Texture2D forestTexture;         // 森林贴图
    [LabelText("沼泽贴图")]
    public Texture2D[] marshTextures;       // 沼泽贴图
    [LabelText("地图材质")]
    public Material mapMaterial;            // 地图材质
    [LabelText("玩家可视距离")]
    public int viewDistance;                // 玩家可视距离, 单位为ChunkSize
    [LabelText("地图块刷新概率")]
    public int mapChunkRefreshProbability;  // 每天早晨地图块刷新地图物品的概率

    [Header("AI")]
    [LabelText("地图块AI数量限制")]
    public int maxAIOnChunk;
    [LabelText("森林/沼泽地图块生成AI最小顶点数")]
```

```
    public int generateAIMinVertexCount;
}
```