地图UI窗口<游戏场景>

核心资源：

1. UI/UI_MapWindow
2. Prefabs/UI/UI_MapIcon.prefab
3. Prefabs/UI/UI_MapItem.prefab

核心脚本：

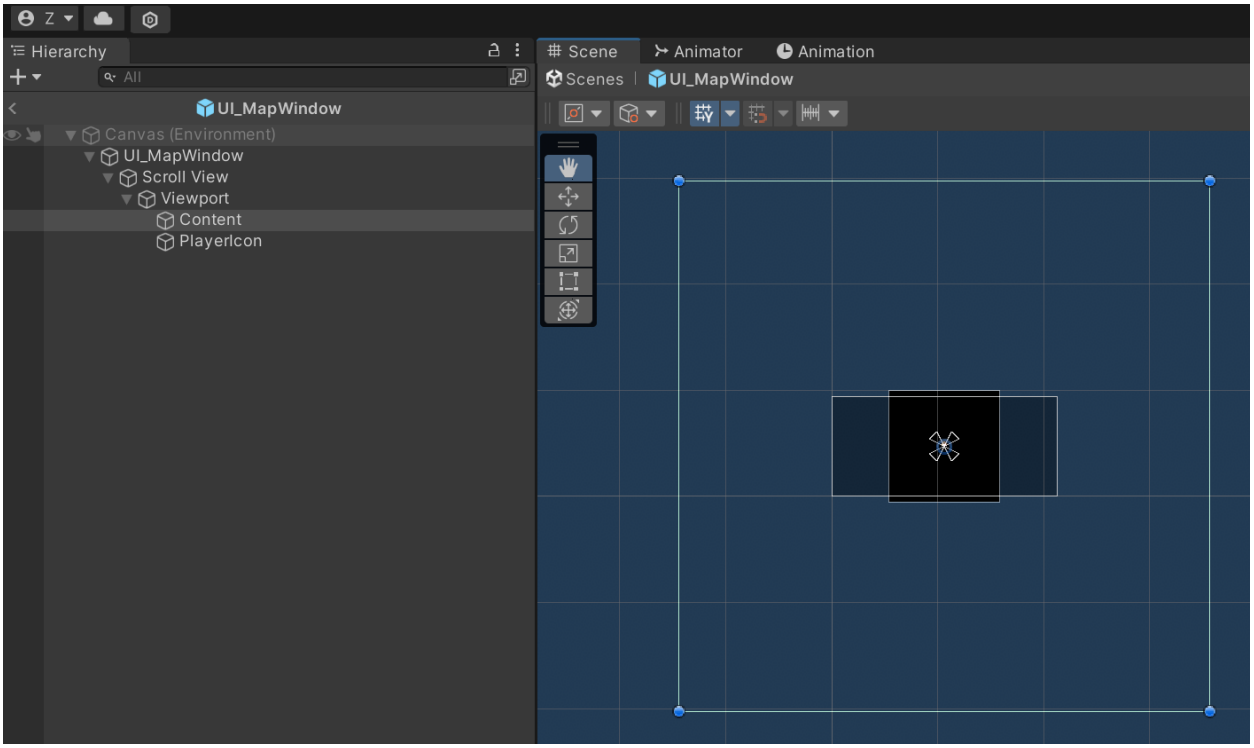
1. UI_MapWindow.cs

地图UI窗口

说明：

1. 放大缩小地图：在游戏中按下M键后显示小地图（红框部分），滚动鼠标滚轮后小地图会在一定范围内变大和缩小
2. 战争迷雾：玩家如果探索过当前地图块则会在小地图中显示，如果没有探索过则不显示。还需要注意当小地图打开时会屏蔽其他游戏防止误操作
3. 小地图内容：1. 玩家icon：表示玩家当前位置，会随着玩家移动而移动，即地图中心点为玩家坐标；2. 特定地图对象icon：例如树木、石头、蘑菇、树枝等；3. 地形：森林、沼泽

4. 小地图显示做法：小地图组件中Content范围很大，整个content对应一整个地图，但是viewport很小，对应的是游戏中的小地图窗口，当玩家移动时，不断修改content的pivot以实现小地图跟随玩家移动而移动



重要属性：

1. content：content组件，所有小地图上icon和地面UI的父物体
2. contentSize：小地图尺寸
3. minScale：地图最小缩放尺寸
4. maxScale：地图最大缩放尺寸
5. mapScaleFactorNum：跟原始地图之前的比例系数
6. mapItemPrefab：单个地图块在UI中的预制体
7. mapIconPrefab：单个icon在UI中的预制体
8. mapChunkImageSize：地图块UI贴图尺寸

初始化

1. 设置ScrollRect当值改变时<***更新玩家icon位置***>

Update

1. 得到当前鼠标滚轮的值`Input.GetAxis("Mouse ScrollWheel");`
2. 根据当前滚轮的值设置当前content的localScale

生成地图块Sprite

1. 根据传入的texture生成sprite，需要设置sprite的texture、rect的位置和宽高以及pivot

初始化小地图UI

1. 根据传入的地图大小设置content尺寸，content尺寸是地图尺寸*比例系数（默认是10）
2. 设置content.sizeDelta和localScale
3. 计算当前一个地图块UI应该多大，contentSize/mapSize
4. 计算当前maxScale和minScale

更新地图中心点

1. 根据传入的玩家玩家位置计算玩家坐标在整个地图坐标中的位置 `x = viewerPosition.x / mapSizeOnWorld;` y同理
2. 设置 `content.pivot = new Vector2(x, y);` ，由于content添加了 `onValueChanged` 的事件，所以当content值改变时也会触发<***更新玩家icon位置***>

更新玩家icon位置

1. 将玩家icon anchoredPosition3D设置为当前UI窗口中content的anchoredPosition3D，其中content是用来显示地图上的物体的，即上图红框部分。

添加地图对象icon

1. 根据传入的mapObjectData拿到当前地图对象配置id然后找到对应配置
2. 如果当前地图对象icon配置为空代表不需要在小地图上显示对应icon
3. 实例化出mapIconPrefab对象，将其挂载到content上
4. 设置当前对象image.sprite为配置中的mapIconSprite，设置当前image大小为配置中的mapIconSize
5. 通过传入的mapObjectData获得小地图与地图之间的比例系数，同比计算icon在小地图中的位置，将计算后的位置设置为前地图对象icon的anchoredPosition
6. 将设置完的地图对象icon添加到mapObjectIconDict中

添加地图块UI

1. 先实例化出mapItemPrefab对象，将其挂载到content上，然后设置RectTransform
2. 由于当前地图块UI在小地图中贴图尺寸已经算好，因此可以直接使用地图块索引乘上每个地图块贴图大小作为当前地图块UI的anchoredPosition，同时设置sizeDelta为mapChunkImageSize

```
// 确定地图块在UI界面中的位置和大小(宽高)
mapChunkRect.anchoredPosition = new Vector2(chunkIndex.x * mapChunkImageSize, chunkIndex.y * mapChunkImageSize);
mapChunkRect.sizeDelta = new Vector2(mapChunkImageSize, mapChunkImageSize);
```

3. 根据传入的texture判断当前是否是森林贴图还是包含沼泽的贴图
    a. 森林贴图：控制图片缩放，让一个森林sprite填充满当前chunk
    b. 包含沼泽的贴图：根据传入的沼泽texture生成sprite<***生成地图块Sprite***>
4. 将设置完的地图块UI对象添加到mapChunkImageDict中
5. 遍历传入的mapObjectDict获取地图块上的所有地图物体，<***添加地图对象icon***>

移除地图对象icon

1. 根据传入的地图对象id查看当前mapObjectDict中是否有对应数据，如果有则将该icon对象放入对象池并从字典中删除对应数据

重置窗口

说明：目的是当游戏结束时回收资源

1. 回收icon：遍历mapObjectIconDict，将所有icon放入对象池，最后清空字典
2. 回收地图块UI：遍历mapChunkImageDict，Destory(Image.gameObject)，最后清空字典