建造系统

重点部分：

1. 建筑物建造预览方法和如何移除资源
2. 建造系统与UI面板交互

核心脚本：

1. BuildManager.cs
2. IBuilding.cs
3. BuildingBase.cs
4. BuildingConfig.cs

目前建筑物仅有三个：储物箱、篝火、科学机器

建筑物对象接口类 IBuilding.cs

必要属性：

1. gameObject：由子类赋值，当前建筑物物体
2. collider：预览时需要先关闭碰撞体
3. materialList：保存所有MeshRenderer中的Material
4. 默认颜色 Green/Red

初始化预览方法

1. 预览时关闭碰撞体防止拖拽物体时碰撞到其他物体
2. 初始化材质球和MeshRenderer
    a. 获得当前和子物体中的所有MeshRenderer中的material
    b. 将所有material颜色改为红色并改成支持更改透明通道的rendering mode
    c. 显示预览

设置预览颜色

1. 遍历所有的MeshRenderer中的material，如果当前传入bool代表红色则设置为红色否则为绿色

建筑物对象基类 BuildingBase.cs

注意：该类继承自MapObjectBase和IBuilding，开始时会设置IBuilding的属性

必要属性：

1. 建筑物能解锁的科技id

初始化方法

1. 根据传入的mapChunk、mapObjectId、isFromBuild调用基类初始化方法
2. 如果`isFromBuild==true`则调用科技树管理器添加已解锁科技id

建筑物管理器 BuildingManager.cs

在游戏中如何进入预览模式：在左侧UI界面中点击建造建筑物后会进入预览模式，该模式下建筑物会变为透明，如果当前区域可以建造建筑物则显示为透明绿色，否则为透明红色

必要属性：

1. 虚拟网格吸附范围
2. Mask层：LootObject, BigObject, Buiding

初始化

1. 初始化建造UI面板
2. 添加建造事件，当在UI界面点击建造时进入建造状态

开始建造

1. 关闭鼠标与地图对象交互功能`InputManager.Instance.SetCheckState(false);`
2. 关闭游戏UI交互功能`graphicRaycaster.enabled=false;`
3. 生成建筑物预览物体并开启预览模式
    a. 获取建筑物配置中包含的预制体
    b. 实例化该预制体后初始化预览方法
    c. 开启预览模式

4. 预览物体跟随鼠标，跟随时预览物体需要”吸附”到地图格子上
    a. 检查是否取消建造（右键）
        i. 如果取消建造则需要关闭预览物体、开启鼠标和地图对象交互功能、开启游戏UI交互功能，break死循环
    b. 获取鼠标在当前世界中的坐标，将鼠标坐标转换为格子坐标（需要取整）
    c. 根据不同collider类型进行检测（CheckBox/CheckCapsule/CheckSphere），一般传入位置、半径、LayerMask，如果当前collider与其他collider重叠则返回True
    d. 如果当前collider与其他collider重叠则将预览模式设置为红色
    e. 如果当前collider与其他collider不重叠
        i. 将预览模式设置为绿色
        ii. 如果玩家按下鼠标左键则建造建筑物并移除对应资源
            1. 关闭预览物体、开启鼠标和地图对象交互功能、开启游戏UI交互功能
            2. 放置建筑物 `MapManager.Instance.GenerateMapObject(buildConfig.targetId, previewBuilding.gameObject.transform.position, true);`
            3. 根据建筑物配置减少背包中的物品
            `InventoryManager.Instance.UpdateItemsForBuild(buildConfig);`
        iii. break出循环

建筑物配置：

```
[CreateAssetMenu(fileName = "建造合成配置", menuName = "Config/建造与合成配置")]
public class BuildConfig : ConfigBase
{
    [LabelText("合成类型")]
    public BuildType buildType;
    [LabelText("前置科技")]
    public List<ulong> preconditionScienceIdList = new List<ulong>();
    [LabelText("合成条件")]
    public List<BuildConfigCondition> buildConfigConditions = new List<BuildConfigCondition>();
    [LabelText("合成产物")]
    public int targetId;

    // 检查建筑物是否满足前置科技条件
    public bool CheckPreconditionScienceId() {
        for (int i = 0; i < preconditionScienceIdList.Count; i++) {
            if (ArchiveManager.Instance.scienceData.CheckUnlock(preconditionScienceIdList[i]) == false) {
                return false;
            }
        }
        return true;
    }

    // 检查资源是否满足当前建造/合成配置
```

```
    public bool CheckBuildConfigCondition() {
        for (int j = 0; j < buildConfigConditions.Count; j++) {
            int currentCount = InventoryManager.Instance.GetItemCount(buildConfigConditions[j].itemId);
            if (currentCount < buildConfigConditions[j].count) {
                return false;
            }
        }
        return true;
    }
}


public class BuildConfigCondition
{
    [LabelText("物品Id"), HorizontalGroup]
    public int itemId;
    [LabelText("物品数量"), HorizontalGroup]
    public int count;
}
```