储物箱

核心脚本：

1. StorageBox_Controller.cs
2. StorageBoxData.cs

重要资源：

1. UI_StorageBoxInventoryWindow：箱子打开后的UI窗口

储物箱控制器 StorageBox_Controller

说明：

1. 储物箱是一个建筑，继承自`MapObjectBase, IBuilding`，相较于标准的建筑物，储物箱特殊之处在于当玩家点击储物箱物体时会弹出一个仓库UI界面，并且玩家能跟仓库中的物体进行交互

2. 储物箱建造前提是需要有一个科学机器，并且在建造普通储物箱后会解锁高级储物箱
3. 玩家距离仓库过远会关闭仓库

初始化方法

1. 调用基类MapObjectBase初始化方法，记录当前mapChunk和对应的mapObjectId，如果当前初始化对象是建筑物则解锁对应科技树id
2. 查看当前是否是从存档中恢复的，如果是从存档中恢复的则读取存档数据，否则创建一份新的仓库数据并将其添加到存档管理器中

建筑物选中逻辑

1. 打开储物箱UI窗口，传入当前储物箱的数据和UI窗口尺寸

储物箱UI窗口

说明：储物箱UI窗口继承自`UI_InventoryWindowBase` 其中有储物箱当前格子UI的和仓库数据

重要属性：

1. closeButton：储物箱UI界面右上角关闭按钮
2. itemParent：物品格子挂载父物体
3. storageBox：储物箱控制器

初始化方法

1. 记录传入的储物箱控制器对象和储物箱数据
2. 根据传入的windowSize<***设置窗口大小***>
3. 遍历所有的储物箱数据生成窗口格子
    a. 实例化格子`UI_ItemSlot itemSlot = ResManager.Load<UI_ItemSlot>("UI/UI_ItemSlot", itemParent);`
    b. 初始化每个格子中的物品数据和UI信息并添加到物品槽数据中心

Update

1. 检查当前玩家跟储物箱距离是否超过配置距离，当超过配置距离时<***关闭储物箱***>

设置窗口大小

1. 根据传入的windowSize设置窗口大小，其中每个格子100*100，宽度 = 两边15 + 中间格子区域，高度 = 顶部50 + 中间格子区域 + 底部15

关闭储物箱

1. 清空基类中持有的数据：遍历基类中slots列表数据将每个元素都放入对象池后清空slots和inventoryData

关闭按钮点击逻辑

1. 播放关闭音效
2. <***关闭储物箱***>

储物箱数据 StorageBoxData

```
public class StorageBoxData : IMapObjectTypeData
{
    private InventoryData inventoryData;    // 储物箱数据
    public InventoryData InventoryData { get => inventoryData; }

    public StorageBoxData(int itemCount) {
        inventoryData = new InventoryData(itemCount);
    }
}

// 通用物品栏格子
[Serializable]
public class InventoryData
{
    // 物品栏中装的物品
    public ItemData[] itemDatas { get; protected set; }

    public InventoryData(int itemCount) {
```

```
        itemDatas = new ItemData[itemCount];
    }

    // 移除某一个物品
    public void RemoveItem(int index) {
        itemDatas[index] = null;
    }

    // 放置某一个物品
    public void SetItem(int index, ItemData itemData) {
        itemDatas[index] = itemData;
    }

}
```