



AI系统

重点部分：

1. AI状态逻辑：移动、攻击、追击、受伤、动画切换、模型更改等
2. AI巡逻与领地逻辑
3. AI死亡掉落物品逻辑
4. AI数据和模型的保存与恢复

核心脚本：

1. AIBase.cs
 2. AIState.cs
 3. AIStateBase.cs
 - a. AI_AttackState.cs
 - b. AI_DeadState.cs
 - c. AI_Hurt.cs
 - d. AI_Idle.cs
 - e. AI_Patrol.cs
 - f. AI_PursueState.cs
 4. MapObjectData.cs
 5. AIConfig.cs
 6. Boar_Controller.cs
 7. Spider_Controller.cs
 8. Spider_PursueState.cs
-

AI状态和AI状态基类 AIState/AIStateBase

说明：

1. 目前有7种状态，分别是None、Idle、Patrol、Hurt、Pursue、Attack、Dead
2. AI状态基类继承自StateBase，该类为抽象类，其中有一些涉及状态Enter、Exit、Update...的虚方法
3. AIStateBase中包含一个AIBase的成员，目的是方便后续各个AI状态调用AIBase方法

AI待机状态 AI_Idle

进入状态

1. 播放待机动画，有1/3的几率播放AI物体待机声音
2. 原地待机一小段时间后使用协程让AI物体进入巡逻状态

Update

1. 检查玩家是否进入了警戒范围并且当前游戏还在继续
 - a. 判断AI和玩家距离是否在仇恨距离范围之内
 - i. AI切换到<追击状态>

使用协程让AI物体进入巡逻状态

1. 等待一定时间
2. 切换到<巡逻状态>

退出状态

1. 如果当前巡逻协程未停止则需要停止当前协程

AI攻击状态 AI_AttackState

进入状态

1. 将AI对象转向玩家
2. 随机选择一个攻击动作并播放动画
3. 播放攻击音效
4. <添加动画事件> 分别将<开始攻击>、<停止攻击>、<攻击结束>添加到AIBase动画事件中
5. 通过Unity OnTriggerStay添加<武器伤害检测方法>

武器伤害检测方法

说明：可攻击的AI物体都会包含一个攻击部位，下面统一称呼为“武器”

1. 设置一个bool变量 `isAttacked` 用于检测当前是否已经攻击过，避免一次攻击产生多次伤害
2. 通过传入的 `Collider` 信息判断当前gameObject的Tag是否为“Player”
 - a. `isAttacked` 设置为true，AI播放攻击音效
 - b. 通过玩家控制器调用玩家<受伤方法>，此时应更新玩家血量、触发右上角玩家UI更新并让玩家进入受伤状态

开始攻击

1. 开启AI物体武器 `AI.Weapon.gameObject.SetActive(true);`

停止攻击

1. 关闭AI物体武器 `AI.Weapon.gameObject.SetActive(false);`
2. `isAttacked=false;`

攻击结束

说明：AI物体在攻击后会一直追随玩家不断进行攻击，直到玩家脱离AI物体仇恨范围才停止攻击

1. 将AI状态切换至追踪（Pursue）状态

退出状态

1. <移除动画事件>，分别将<开始攻击>、<停止攻击>、<攻击结束>从AIBase动画事件中移除
2. 在Unity OnTriggerStay中移除<武器伤害检测方法>

AI巡逻状态 AI_Patrol

进入状态

1. 播放AI移动动画
2. <添加动画事件> 将<脚步声>添加到AIBase动画事件中
3. 开启NavMeshAgent，<获取AI可以到达的随机坐标>作为巡逻目标点，使用NavMeshAgent.SetDestination设置AI物体目的地

Update

1. 保存AI坐标
2. 检查玩家是否进入了警戒范围并且当前游戏还在继续
 - a. 判断AI和玩家距离是否在仇恨距离范围之内
 - i. AI切换到<追击状态>
3. 检查是否到达目标，如果AI到达目标则切换到<待机状态>

脚步声

1. 随机播放脚步声片段

退出状态

1. 关闭NavMeshAgent
2. <移除动画事件>，将<脚步声>从AIBase动画事件中移除

AI追击状态 AI_PursueState

注意：巡逻状态与追击状态非常相似

进入状态

1. 开启NavMeshAgent
2. 播放AI移动动画
3. <添加动画事件> 将<脚步声>添加到AIBase动画事件中

Update

1. 检查当前AI和玩家距离是否小于AI物体半径+AI攻击距离
 - a. 如果小于则进入<攻击状态>
 - b. 否则则保存AI位置，设置玩家坐标为AI物体新的目标点
`AI.NavMeshAgent.SetDestination(Player_Controller.Instance.transform.position);`，并检查当前AI是否出现了迁移地图的情况<检查并迁移地图块>

检查并迁移地图块

1. 通过AI物体坐标得到当前地图块控制器
2. 如果AI所在地图块控制器和之前的地图块控制器不一致说明AI物体从A地图块跑到B地图块，需要迁移AI数据
 - a. <删除一个AI物体(物体迁移)>，从旧的地图块AIDataDict/AIObjectDict中删除数据
 - b. <添加一个AI物体(物体迁移)>，在新的地图块AIDataDict/AIObjectDict中添加数据、设置AI物体Parent以及引用当前新的地图块

脚步声

1. 随机播放脚步声片段

退出状态

1. 关闭NavMeshAgent
2. <移除动画事件>，将<脚步声>从AIBase动画事件中移除

AI受伤状态 AI_Hurt

进入状态

1. 先播放受伤动画和受伤音效
2. <添加动画事件> 将<受伤结束>添加到AIBase动画事件中

受伤结束

1. 将状态切换为<追击状态>

退出状态

1. 将<受伤结束>从AIBase动画事件中移除

AI死亡状态

进入状态

1. 关闭AI物体Collider防止玩家继续攻击
2. 播放死亡动画
3. 将<死亡事件>添加到AIBase动画事件中

死亡事件

1. 调用AI物体<死亡逻辑>，例如告知AI所在地图块管理器移除该物体，根据物品掉落配置掉落不同物品等

退出状态

1. 从AIBase动画事件中移除<死亡事件>

2. 开启AI物体Collider

AI对象基类

必要属性

1. 引用组件：

- a. animator：动画控制器，AI对象各个状态
- b. navMeshAgent：导航网格代理
- c. audioClips：音频片段，例如脚步声
- d. collider：当AI物体死亡时需要关闭碰撞体
- e. weapon：在AI物体攻击部位创建的gameobject
- f. stateMachin：包含控制AI状态的各种方法

2. 属性值：

- a. maxHP：AI物体血量
- b. attackDistance：攻击距离
- c. attackValue：攻击力
- d. **hostileDistance**：敌对距离
- e. radius：AI物体半径

3. 其他数据：

- a. mapVertexType：AI物体所在的地图cell顶点类型
- b. mapChunk：AI物体所在的地图块控制器
- c. lootObjectConfig：掉落物品配置id
- d. aiData：AI地图数据，包括id、configId等

初始化方法

1. 根据传入的地图块控制器和AI地图数据设置当前属性和数据，例如设置hp、位置等
2. 设置默认状态为<待机状态>

切换状态

1. 根据传入的AIState类型调用stateMachine执行不同状态对应的类里Enter、Update、Exit等方法来切换状态

播放动画

1. 根据传入的AnimationName播放对应动画 `animator.CrossFadeInFixedTime(animationName, fixedTime);`

播放音效

1. 使用AudioManager在指定位置使用指定音量播放对应的音效片段

获取AI物体能到达的随机坐标

1. (MapChunkController) <获取AI物体能到达的随机坐标>, 根据地图块顶点列表随机选择一个顶点作为目的地, 通过 `NavMesh.SamplePosition` 判断能否到达, 如果能到达则该顶点数据中的position作为新的AI物体目的地

保存AI物体坐标

1. 设置aiData中缓存的position为当前AI物体position, 由于在游戏结束时每个地图块都会保存数据, 而aiData数据在地图块数据中, 所以AI物体位置也可以保存

AI物体受伤方法

1. 根据传入的伤害值更新当前血量
2. 如果当前血量 ≤ 0 则切换到<死亡状态>
3. 如果当前血量 > 0 则切换到<受伤状态>

AI物体死亡方法

1. 告知地图块控制器<删除一个AI物体>

2. 根据掉落物品配置id获得对应配置数据，在当前地图块的AI物体位置附近<生成地图掉落物品>

AI对象销毁方法

1. 将自身放入对象池中

添加动画事件

1. 根据传入的事件名和事件Action去检查当前动画事件字典中是否包含对应事件名，如果包含事件名则将当前传入事件Action += 之前事件
2. 如果时间名不在事件字典中则添加一个事件

移除动画事件

1. 根据传入的事件名和事件Action去检查当前动画事件字典中是否包含对应事件名，如果包含事件则将之前事件 -= 当前传入事件Action

特殊AI-野猪 Boar_Controller.cs

说明：野猪没有“领地”意识，只会在当前地图块上随机巡逻，当收到攻击时才会主动攻击玩家，因此AIBase中的逻辑完全满足野猪控制器条件不需要再单独添加其他逻辑

特殊AI-蜘蛛 Spider_Controller.cs

说明：蜘蛛有“领地”意识，当玩家进入蜘蛛领地后蜘蛛会主动追击玩家，直到玩家死亡/蜘蛛死亡/玩家跟蜘蛛之间的距离超过仇恨范围，且蜘蛛在追击时会加快移动速度，因此需要重写蜘蛛追击状态逻辑Spider_PursueState.cs

Spider_Controller.cs 必要属性：

1. walkSpeed：蜘蛛移动速度
2. runSpeed：蜘蛛奔跑速度

3. retreatDistance：蜘蛛撤退距离

蜘蛛追击逻辑重写部分 Spider_PursueState.cs

进入状态

1. 开启NavMeshAgent，修改AI移动速度 `AI.NavMeshAgent.speed = spider.RunSpeed;`
2. 播放蜘蛛奔跑动画，添加脚步声事件

Update

1. 判断当前玩家和AI的距离
 - a. 如果玩家和AI距离 \leq AI半径+AI攻击距离则蜘蛛进入<攻击状态>
 - b. 如果玩家和AI距离 $>$ AI半径+AI攻击距离
 - i. 保存蜘蛛当前坐标
 - ii. 设置玩家位置为AI目的地
 - iii. <检查并迁移地图块>
 - iv. 如果当前距离超过蜘蛛撤退距离，即仇恨范围则蜘蛛切换到<待机状态>

退出状态

1. 调用基类<退出状态>方法，关闭NavMeshAgent，移除脚步声事件等
2. 恢复移动速度

AI对象配置

```
[CreateAssetMenu(fileName = "生物配置", menuName = "Config/生物配置")]
public class AIConfig : ConfigBase
{
    [LabelText("空的(不生成物品)")]
    public bool isEmpty = false;
    [LabelText("地图顶点类型")]
    public MapVertexType mapVertexType;
    [LabelText("预制体")]
}
```

```

public GameObject prefab;
[LabelText("生成概率(百分比类型)")]
public int probability;
[LabelText("腐烂天数")]
public int destoryDay = -1;           // -1代表无效
}

```

Key	Value	
0	森林空 (AI Config)	✎ 🔍 ✕
1	野猪1 (AI Config)	✎ 🔍 ✕
2	野猪2 (AI Config)	✎ 🔍 ✕
3	野猪3 (AI Config)	✎ 🔍 ✕
4	蜘蛛5 (AI Config)	✎ 🔍 ✕
5	沼泽空 (AI Config)	✎ 🔍 ✕

Script	AIConfig
空的(不生成物品)	<input type="checkbox"/>
地图顶点类型	Forest
预制体	Bora_1
生成概率(百分比类型)	10
腐烂天数	-1

AI对象数据

注意：AI对象数据没有单独设置一个类

```

// 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.ConvertToVector3();
        set => sv_position = value.ConvertToSVector3();
    }
}

```