角色管理模块

# Unity中的使用说明

## 角色控制系统说明

角色物体下面包含所有该角色的攻击模式、移动模式、被击模式，放置层级如下图所示，其中AttackMode、MoveMode、HitMode仅做归类作用的空物体，没有任何脚本。

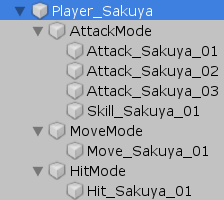


图1 角色物体Hierarchy

每个模式物体中挂一个模式脚本，并可以在Inspector窗口中设置相关参数，如下图所示，具体参数含义可以参见2.5-2.7节。

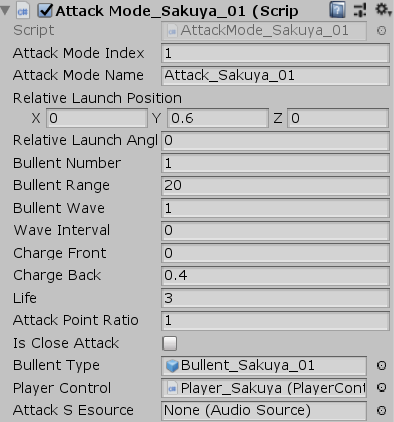


图2 攻击模式脚本

每个角色物体上挂一个控制脚本，玩家角色为PlayerControl，敌方杂鱼为EnemyControl，敌方Boss为BossControl（BOSS的为暂定），后文如果未单独说明的话，均用PlayerControl举例，三者基本内容是相同。控制脚本中记录角色的HP、P点等各种基本参数，如下图左，具体参数含义可以参见2.2-2.4节。控制脚本中还包含AttackMode、MoveMode、HitMode三个变量，用来记录当前起作用的攻击、移动、被击模式。其中，由于攻击技能有多种，因此使用的是List，同时，对应的有List变量attackButtons记录每个攻击模式的按键字符串名称，代表了Unity的InputManager里的按键名称。

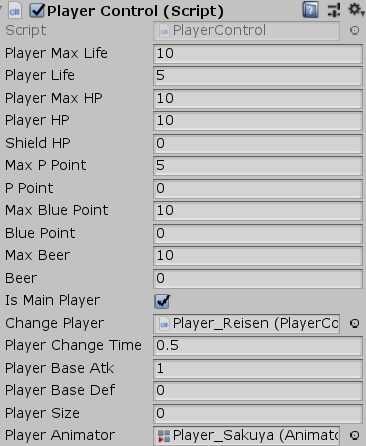
 

图3 PlayerControl脚本

## Buff系统说明

buff通过子弹命中或者角色技能释放添加到角色身上，子弹的buff效果通过OnTriggerEnter2D方法判定命中时添加，使用BuffGroup.CreateBuff方法生成buff类，并运行PlayerControl.AddBuff方法添加到buffList里，并执行OnBuffAdd方法。PlayerControl的Update方法中会执行buffList中buff的OnBuffUpdate方法，满足条件清除buff时，将buff添加到PlayerControl的buffRemoveList中，PlayerControl会在所有buffList的buff更新后，从中移除buffRemoveList中的buff，并执行OnBuffRemove方法。玩家释放护盾技能时，以相同方式添加到角色身上。

# 脚本变量和方法

## CharacterManager（角色管理，待定，目前用场景管理器DemoSceneManager代替功能）

### 变量

public GameObject **mainPlayer**：主角色GameObject

public GameObject **subPlayer**：副角色GameObject

public List<GameObject> **enemies**：场景内的所有敌人GameObject

public GameObject **boss**：BOSS的GameObject

public event EventHandler **gameOverEvent**：角色死亡后执行的所有操作（暂未使用）

### 方法

void **GameOver**()：角色死亡时执行gameOverEvent（暂未使用）

## PlayerControl（玩家角色控制器，执行各种行为）

### 变量

public float **playerMaxLife**：玩家残机最大值

public float **playerLife**：玩家残机

public float **playerMaxHP**：玩家最大HP

public float **playerHP**：玩家当前HP

public float **maxPPoint**：角色最大P点

public float **pPoint**：角色当前P点

public float **maxBluePoint**：角色最大蓝点

public float **bluePoint**：角色当前蓝点

public int **maxBeer**：回血道具最大携带量

public int **beer**：回血道具当前数量

public bool **isMainPlayer：**是否是主角色

public PlayerControl **changePlayer**：可更换的角色的PlayerControl组件

public float **playerChangeTime**：两次更换的间隔时间

public float **playerBaseAtk**：角色基础攻击力

public float **playerBaseDef**：角色基础防御力

public Animator **playerAnimator**：角色动画机

public List<string> **attackButtons**：角色攻击按键列表

public List<AAttackMode> **playerAttackModes**：角色攻击模块列表

public AMoveMode **playerMoveMode**：角色移动模块

public AHitMode **playerHitMode**：角色受伤模块

public float **playerSize**：角色判定大小（暂未使用，使用的胶囊形碰撞体）

[HideInInspector]

public List<ABuff> **buffList**：所有buff的列表

public List<ABuff> **buffRemoveList**：当前要移除的buff列表

public float **playerAttackPoint**：计算后的角色攻击力

public float **playerDefensePoint**：计算后的角色防御力

public bool **isDead**：玩家角色是否死亡

public bool **isChanging**：玩家正在切换角色

public bool **isInvulnerable**：玩家角色是否无敌（暂未使用，准备在受攻击后无敌时间或其它无敌道具技能中使用）

### 方法

public void **AddBuff**(ABuff buff)：添加buff

public void **RemoveBuff**(ABuff buff)：移除buff

public void **SetAttackMode**(AAttackMode attackMode, int index)：设置角色攻击模块

public void **SetMoveMode**(AMoveMode moveMode)：设置角色移动模块

public void **SetHitMode**(AHitMode hitMode)：设置角色受伤模块

public int **GetAttackModeIndex**(int index) 获取当前第index号攻击模式编号

public string **GetAttackModeName**(int index) 获取当前第index号攻击模式名称

## EnemyControl（敌人角色控制器）

### 变量

public float **enemyMaxHP**：敌人最大HP

public float **enemyHP**：敌人当前HP

public Animator **enemyAnimator**：敌人角色动画机

public List<AAttackMode> **enemyAttackModes**：敌人攻击模块

public AMoveMode **enemyMoveMode**：敌人移动模块

public AHitMode **enemyHitMode**：敌人受伤模块

public float **enemySize**：敌人判定大小（暂未使用，使用的圆形碰撞体）

public float **enemyAttackPoint**：敌人攻击力

public float **enemyDefensePoint**：敌人防御力

public float **hatredRange**：仇恨距离

public List<GameObject> **itemDropped**：掉落物品

public List<float> **droppedRate**：掉落率

[HideInInspector]

public List<ABuff> **buffList**：所有buff的列表

public List<ABuff> **buffRemoveList**：当前要移除的buff列表

public bool **isDead**：敌人角色是否死亡

public bool **isInvulnerable**：敌人角色是否无敌（暂未使用，准备无敌技能中使用）

### 方法

public void **AddBuff**(ABuff buff)：添加BUFF

public void **RemoveBuff**(ABuff buff)：移除BUFF

public void **SetAttackMode**(AAttackMode attackMode, int index)：设置角色攻击模块

public void **SetMoveMode**(AMoveMode moveMode)：设置角色移动模块

public void **SetHitMode**(AHitMode hitMode)：设置角色受伤模块

## BossControl（暂未使用）

## AAttackMode（攻击模块抽象类）

### 变量

public bool **isCannotAttack**：角色是否处于不能攻击状态

public int **attackModeIndex**：攻击模式的编号

public string **attackModeName**：攻击模式的名字

### 方法

public abstract void **AttackButtonDown**()：攻击键按下时的操作

public abstract void **AttackButtonUp**()：攻击键未按下时的操作

public abstract void **PowerUp**(int power)：增加攻击力

## AttackMode\_Sakuya\_01 : AattackMode（咲夜的第一种攻击模式，目前铃仙也使用此模式）

### 子弹参数相关变量

public Vector3 **relativeLaunchPosition**：发射点相对角色中心的位置

public float **relativeLaunchAngle**：发射方向相对角色朝向的角度（角度值，向前为0度，逆时针为正）

public int **bullentNumber**：一轮发射子弹数量

public float **bullentRange**：子弹覆盖范围，最两边子弹发射方向之间的夹角

public int **bullentWave**：一轮发射波数

public float **waveInterval**：每波间隔

public float **chargeFront**：攻击前摇，按下攻击键到发射出第一轮之间的时间

public float **chargeBack**：攻击后摇，最后一轮发出到下一次能产生前摇之间的时间（最好大于waveInterval）

public float **life**：子弹生存时间

public float **attackPointRatio**：子弹攻击力比角色攻击力的倍数

public bool **isCloseAttack**：是否近战

### 其它变量

public int **attackModeIndex**：攻击模式的编号

public string **attackModeName**：攻击模式的名字

public GameObject **bullentType**：发射的子弹预设体

public PlayerControl **playerControl**：角色控制器

public AudioSource **attackSEsource**：攻击音效（暂未使用）

[HideInInspector]

public bool **isCannotAttack**：角色是否处于不能攻击状态

### 方法

public override void **AttackButtonDown**()：攻击键按下时的操作(由PlayerControl每帧调用)

public override void **AttackButtonUp**()：攻击键未按下时的操作(由PlayerControl每帧调用)

public override void **PowerUp**(int power)：增加攻击力（暂未使用，目前直接修改PlayerControl.pPoint）

## AttackMode\_Enemy\_01 : AattackMode（敌人第一种攻击模式）

### 变量

变量与AttackMode\_Sakuya\_01相同，仅在给敌人添加此脚本组件时，填写参数不同。

### 方法

方法与AttackMode\_Sakuya\_01相同，由于敌人为自动AI，仅由EnemyControl每帧调用AttackButtonDown方法。

## ABullent（子弹抽象类）

### 变量

public float **life**：子弹生存时间（AttackMode会对其赋值）

public float **attackPoint**：弹幕攻击力（AttackMode会对其赋值）

## Bullent\_Sakuya\_01 : ABullent（咲夜的第一种子弹，前方扇形直线发射子弹）

### 变量

public float **life**：子弹生存时间（AttackMode会对其赋值）

public float **attackPoint**：弹幕攻击力（AttackMode会对其赋值）

public float **velocity**：子弹速度

public float **acceleration**：子弹加速度

public float **angularVelocity**：子弹角速度

public Vector3 **rotationCenter**：旋转中心，相对子弹自身的局部坐标

public float **bullentSize**：子弹圆形碰撞体半径

public GameObject **hitEffect**：命中特效产生的GameObject（暂未使用）

public List<GameObject> **enemies**：场景中敌人列表（暂未使用，目前使用Layer对碰撞体进行分组来识别是否需要进行命中判定）

public int **effect**：攻击效果的编号（暂未使用）

## Bullent\_Sakuya\_02 : ABullent（咲夜的第二种子弹，前方小刀划过）

### 变量

public float **life**：子弹生存时间（AttackMode会对其赋值）

public float **attackPoint**：弹幕攻击力（AttackMode会对其赋值）

public float **velocity**：子弹速度(近战子弹为零)

public float **acceleration**：子弹加速度(近战子弹为零)

public float **angularVelocity**：子弹角速度（近战武器砍人效果，子弹回旋效果）

public Vector3 **rotationCenter**：旋转中心，相对子弹自身的局部坐标

public float **bullentSize**：子弹圆形碰撞体半径

public GameObject **hitEffect**：命中特效产生的GameObject（暂未使用）

public List<GameObject> **enemies**：场景中敌人列表（暂未使用，目前使用Layer对碰撞体进行分组来识别是否需要进行命中判定）

public int **effect**：攻击效果的编号（暂未使用）

## Bullent\_Sakuya\_03 : ABullent（咲夜的第三种子弹，前方投掷抛物线子弹，落地后爆炸产生攻击判定）

### 变量

public float **life**：子弹生存时间（AttackMode会对其赋值，由于此子弹落地爆炸，生存时间由其它参数决定，因此life值只要足够大就行）

public float **attackPoint**：弹幕攻击力（AttackMode会对其赋值）

public float **distance**：子弹落地距离

public float **hight**：子弹抛物线高度

public float **velocity**：弹幕速度

public float **acceleration**：弹幕加速度

public float **angularVelocity**：弹幕角速度

public Vector3 **rotationCenter**：旋转中心

public float **bullentSize**：子弹圆形碰撞体半径

public GameObject **hitEffect**：命中特效产生的GameObject（暂未使用）

public List<GameObject> **enemies**：场景中敌人列表（暂未使用，目前使用Layer对碰撞体进行分组来识别是否需要进行命中判定）

public int **effect**：攻击效果的编号（暂未使用）

public Animator **bullentAnimator**：爆炸动画机

## Bullent\_Bat\_01 : ABullent（敌人的第一种子弹，前方扇形直线发射子弹）

### 变量

与第2.9节Bullent\_Sakuya\_01 : ABullent相同

### 说明

通过OnTriggerEnter2D方法获取命中的物体，原本角色和护盾都有碰撞体，但由于未知bug造成会角色和护盾各触发一次命中，因此目前取消了护盾的碰撞体。

## AMoveMode（移动模块抽象类）

### 变量

public float **moveSpeed**：角色移动速度

[HideInInspector]

public bool **isCannontMove**：角色是否处于不能移动状态

public bool **isCannotSpecialMove**：是否无法特殊移动

public float **directionAngle**：角色朝向与y轴的夹角（弧度，逆时针为正向）

public Vector2 **characterDirection**：角色朝向

### 方法

public abstract void **Move**()：角色移动，每帧调用

public abstract void **IsDelayed**()：角色硬直时的操作（暂未使用）

public abstract void **SetDirection**(Vector2 direction)：设置角色朝向

## MoveMode\_Player\_Dash : AmoveMode（特殊移动为冲刺的移动模式）

### 变量

public float **moveSpeed**：角色移动速度

public PlayerControl **playerControl**：角色控制器

public Animator **playerAnimator**：角色动画机（后续可能被龙骨取代）

public Rigidbody2D **rb**：角色的刚体

public float **dashSpeed**：冲刺速度

public float **dashTime**：冲刺持续时间

public float **dashChargeTime**：冲刺后摇

[HideInInspector]

public bool **isCannontMove**：角色是否处于不能移动状态

public float **directionAngle**：角色朝向与y轴的夹角（弧度，逆时针为正向）

public Vector2 **characterDirection**：角色朝向

### 方法

public override void **Move**()：角色移动，每帧调用

public override void **IsDelayed**()：角色硬直时的操作（暂未使用）

public override void **SetDirection**(Vector2 direction)：设置角色朝向

## MoveMode\_Player\_Run : AmoveMode（特殊移动为加速跑的移动模式）

### 变量

public float **moveSpeed**：角色移动速度

public PlayerControl **playerControl**：角色控制器

public Animator **playerAnimator**：角色动画机（后续可能被龙骨取代）

public Rigidbody2D **rb**：角色的刚体

public float **dashSpeed**：冲刺速度

public float **dashTime**：冲刺持续时间

public float **dashChargeTime**：冲刺耗尽后的补充时间

[HideInInspector]

public bool **isCannontMove**：角色是否处于不能移动状态

public float **directionAngle**：角色朝向与y轴的夹角（弧度，逆时针为正向）

public Vector2 **characterDirection**：角色朝向

### 方法

public override void **Move**()：角色移动，每帧调用

public override void **IsDelayed**()：角色硬直时的操作（暂未使用）

public override void **SetDirection**(Vector2 direction)：设置角色朝向

## MoveMode\_Enemy\_01 : AmoveMode（敌方移动模式，仇恨距离外随机移动）

### 变量

public float **moveSpeed**：角色移动速度

public EnemyControl **enemyControl**：角色控制器

public Animator **enemyAnimator**：角色动画机

public Rigidbody2D **rb**：角色刚体

public float **moveTime**：移动时间

public float **moveInterval**：移动间隔时间

### 方法

public override void **Move**()：角色移动，每帧调用

public override void **IsDelayed**()：角色硬直时的操作（暂未使用）

public override void **SetDirection**(Vector2 direction)：设置角色朝向

## AhitMode（受伤模块抽象类）

### 方法

public abstract void **Hit**()：受伤后做的操作，每帧调用

public abstract void **BeHit**(int atkPoint, int effect)：触发角色受伤，atkPoint为攻击力，effect为攻击效果编号

## HitMode\_Player\_Normal : AHitMode（玩家通常状态的受伤模式）

### 变量

public PlayerControl **playerControl**：角色控制器

### 方法

public override void **Hit**()：受伤后做的操作，每帧调用（暂未使用）

public override void **BeHit**(int atkPoint, int effect)：触发角色受伤，atkPoint为攻击力，effect为攻击效果编号，攻击效果暂未使用

## HitMode\_Enemy\_Normal : AHitMode（敌人通常状态的受伤模式）

### 变量

public EnemyControl **enemyControl**：角色控制器

### 方法

public override void **Hit**()：受伤后做的操作，每帧调用（暂未使用）

public override void **BeHit**(int atkPoint, int effect)：触发角色受伤，atkPoint为攻击力，effect为攻击效果编号，攻击效果暂未使用

## AItem（物品抽象类）

### 变量

public int **effectPoint**：物品产生的效果量

public AudioClip **itemSE**：物品拾取时的音效

### 方法

public abstract void **ItemEffect**(GameObject player)：物品拾取到时产生的效果

## ItemPowUp : Aitem（P点）

### 变量

public int **effectPoint**：物品产生的效果量

public AudioClip **itemSE**：物品拾取时的音效

### 方法

public override void **ItemEffect**(GameObject player)：物品拾取到时产生的效果

## ItemDrop（物品掉落四散掉落效果）

### 变量

public AItem **item**：物品

public float **distance**：物品落地距离

public float **hight**：物品抛物线高度

public float **velocity**：物品掉落速度

## Abuff（Buff抽象类）

### 变量

public BuffType **buffType**：Buff类型

### 方法

public abstract void **OnBuffAdd**()：角色添加上buff时调用的方法

public abstract void **OnBuffUpdate**()：buff持续过程中作用的方法

public abstract void **OnBuffRemove**()：buff移除时调用的方法

## BuffGroup（Buff组，定义了BuffType，负责创建Buff）

### 方法

public static ABuff **CreateBuff**(PlayerControl playerControl, BuffType buffType, List<float> buffParaList)：通过buff类型和参数创建并返回作用于玩家的buff

public static ABuff **CreateBuff**(EnemyControl enemyControl, BuffType buffType, List<float> buffParaList)：通过buff类型和参数创建并返回作用于敌人的buff

buff参数在子弹预设体中设定，如果buffParaList的元素数量和buff应有的参数数量不相同时，会使用默认值创建buff。

## BuffBlank（空buff，用于没有buff效果的子弹）

### 方法

public override void **OnBuffAdd**()：添加空buff时，直接将次buff添加到角色buffRemoveList中

public override void **OnBuffUpdate**()：空buff此方法为空

public override void **OnBuffRemove**()：从角色控制模块中移除

## BuffFrozen（冰冻buff，减速效果）

### 变量

private float **buffTime**：buff持续时间

private float **slowdownScale**：减速比例

### 方法

public override void **OnBuffAdd**()：先判断角色是否已经有这种buff，没有的话，将buff添加到角色buffList中，角色变蓝，按slowdownScale降低moveSpeed，如果已经有这种buff，原buff持续时间刷新，此buff不添加到角色buffList中，此buff不再被访问，不知道会不会在gc时被回收

public override void **OnBuffUpdate**()：保持角色颜色为蓝色，如果到达持续时间，将次buff添加到角色buffRemoveList中

public override void **OnBuffRemove**()： 角色moveSpeed恢复，颜色恢复，从角色控制模块中移除此buff

## BuffPoison（中毒buff，持续掉血效果）

### 变量

private float **buffTime**：buff持续时间

private float **minusHP**：单次减血量

private float **effectGapTime**：减血间隔时间

### 方法

public override void **OnBuffAdd**()：先判断角色是否已经有这种buff，没有的话，将buff添加到角色buffList中，角色变绿，如果已经有这种buff，原buff持续时间刷新，此buff不添加到角色buffList中，此buff不再被访问，不知道会不会在gc时被回收

public override void **OnBuffUpdate**()：保持角色颜色为绿色，每到间隔时间，触发一次减血，如果到达持续时间，将次buff添加到角色buffRemoveList中

public override void **OnBuffRemove**()：角色颜色恢复，从角色控制模块中移除此buff

## BuffStun（晕眩buff，无法移动和攻击效果）

### 变量

private float **buffTime**：buff持续时间

### 方法

public override void **OnBuffAdd**()：先判断角色是否已经有这种buff，没有的话，将buff添加到角色buffList中，将角色所有attackMode和moveMode的停止执行判断变量改为true，角色动画改为idle状态，添加晕眩动画预设体，如果已经有这种buff，原buff持续时间刷新，此buff不添加到角色buffList中，此buff不再被访问，不知道会不会在gc时被回收

public override void **OnBuffUpdate**()：如果到达持续时间，将次buff添加到角色buffRemoveList中

public override void **OnBuffRemove**()：将角色所有attackMode和moveMode的停止执行判断变量改为false，从角色控制模块中移除此buff，销毁晕眩动画预设体

## BuffShield（护盾buff，产生有一定血量的护盾，护盾能防止敌方子弹的debuff效果）

### 变量

private float **shieldHP**：护盾血量

private float **shieldTime**：护盾持续时间

### 方法

public override void **OnBuffAdd**()：先判断角色是否已经有这种buff，没有的话，将buff添加到角色buffList中，增加角色的shieldHP，添加护盾动画预设体，如果已经有这种buff，原buff持续时间刷新，角色shieldHP刷新，此buff不添加到角色buffList中，此buff不再被访问，不知道会不会在gc时被回收

public override void **OnBuffUpdate**()：如果到达持续时间或角色shieldHP小于等于0，将次buff添加到角色buffRemoveList中

public override void **OnBuffRemove**()：从角色控制模块中移除此buff，shieldHP归零，销毁护盾动画预设体