**《类银河恶魔城demo开发日志》**

**2025.11.8**

开发进度：

完成游戏工程建设，下载角色（骑士）素材，制作序列帧动画，设置基础状态机，制作基础脚本，学习尝试新版输入系统

新建脚本：

***Entity（后面改为EntityState）类***

作用是所有实体（比如敌人和角色）基类，本身是抽象类，子类无法直接实例化出对象。

参数：

player用于访问玩家的属性和方法

stateMachine用于状态切换

stateName用于识别状态名称

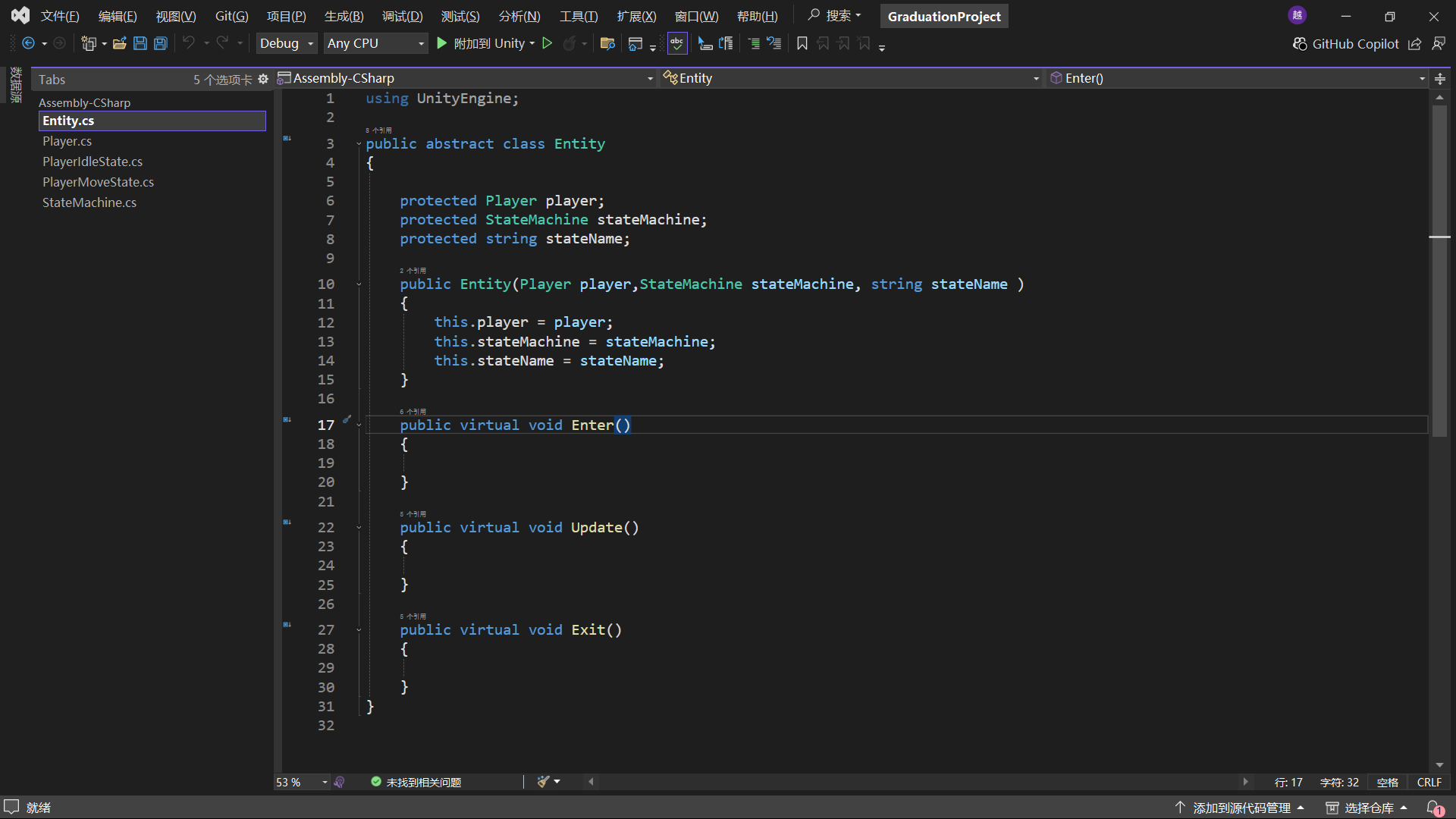
方法：

构造函数，初始化状态所需的引用和状态名

Enter用于进入该状态

Update用于更新该状态

Exit用于退出该状态



***Player类（继承自MonoBehavior）***

作用是用于代表玩家对象，用于控制玩家输入和状态机

参数：

Input作用是处理玩家输入（使用了新版本的InputSystem）

stateMachine是玩家的状态机

idleState是待机状态

moveState是移动状态

moveInput是玩家控制移动时的输入

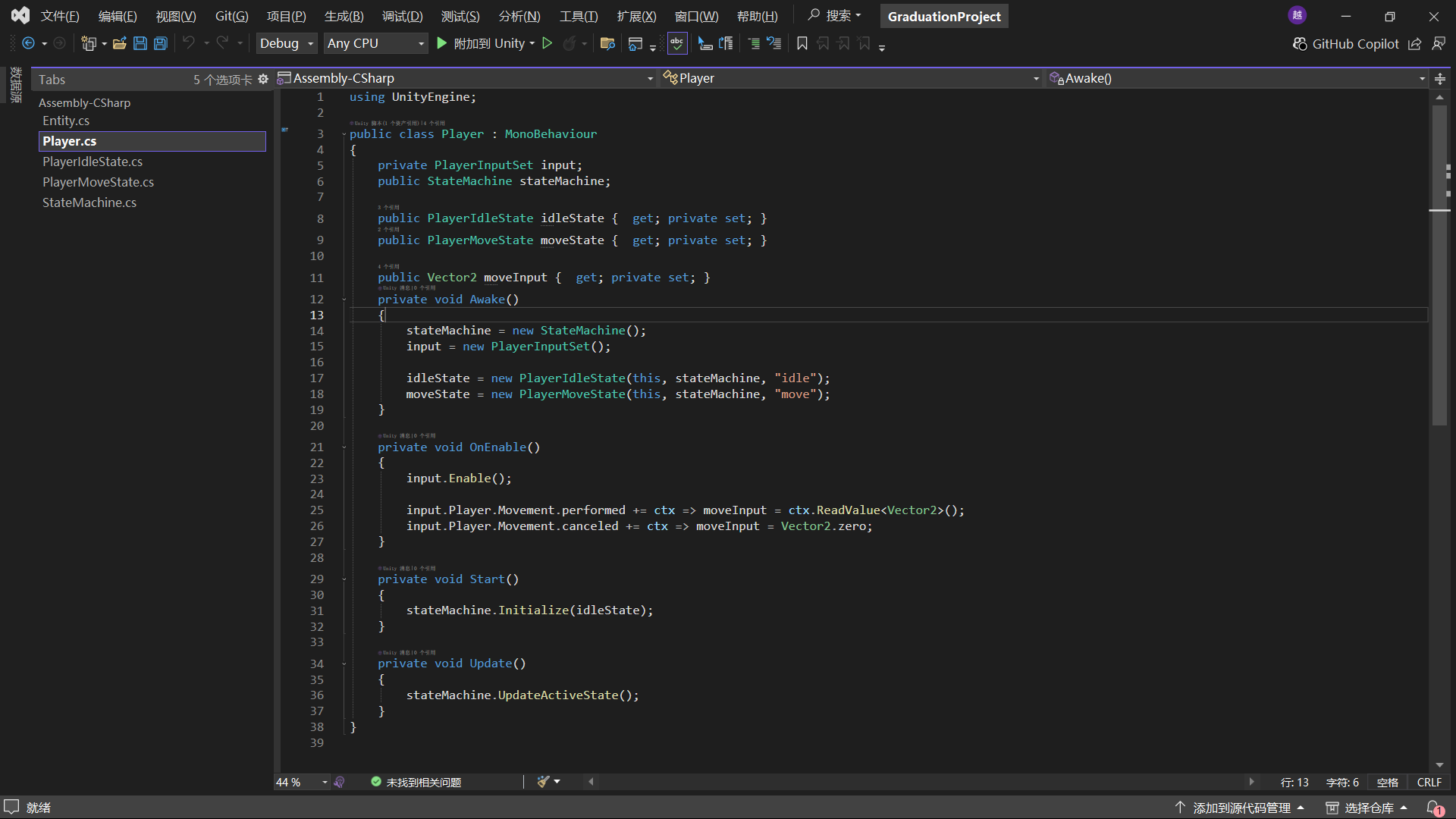
方法：

Awake在启动时调用，用于初始化各个状态、状态机和输入系统

OnEnable在脚本启用时，启动输入系统并注册输入事件

Start在第一次帧更新之前初始化状态机状态，从空闲状态开始

Update每一帧更新，以此更新状态机状态



***PlayerIdleState类（继承Entity）***

作用是继承自基类，定义了玩家空闲状态

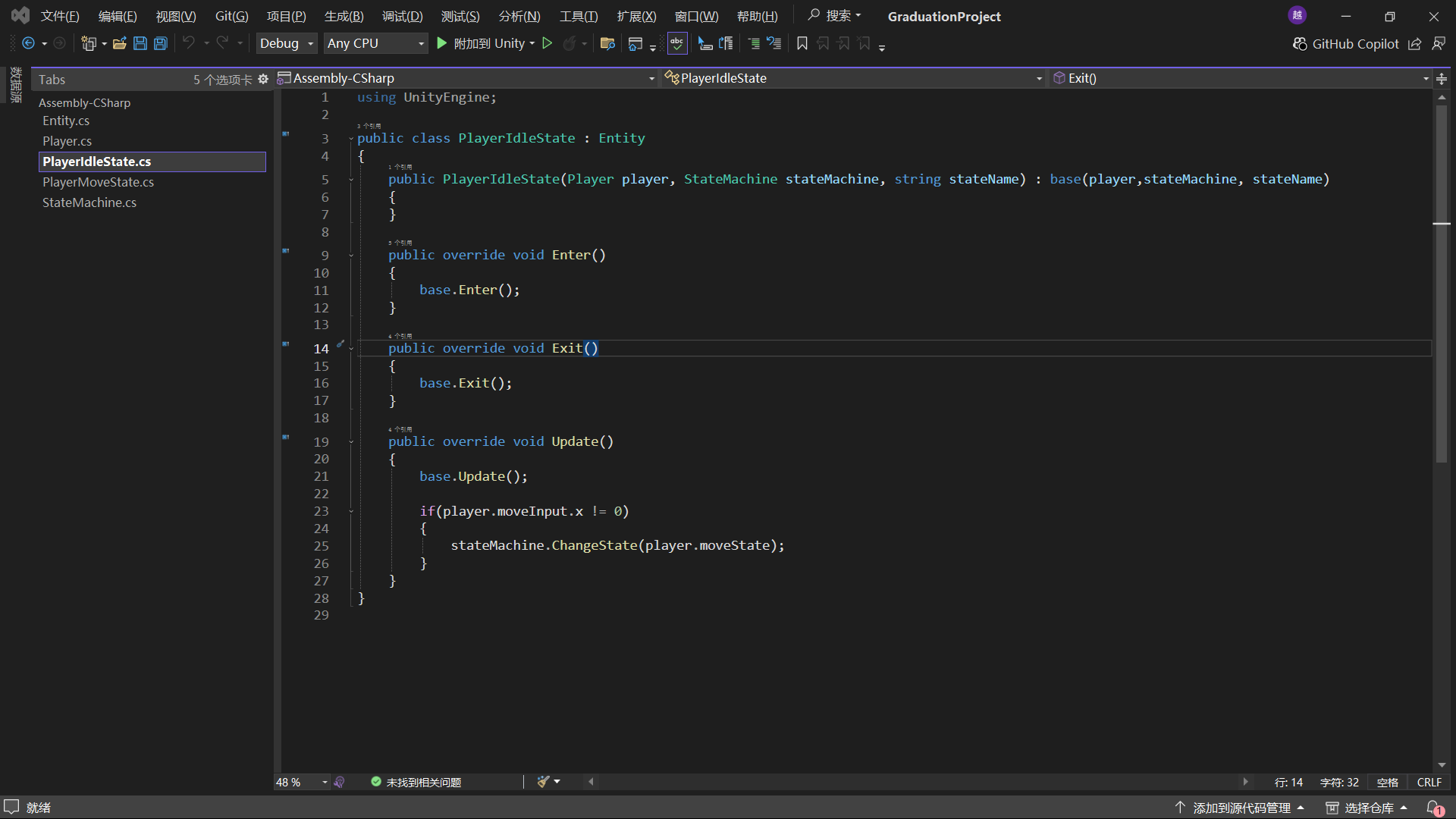
参数：

从构造方法中继承自基类

方法：

Enter，Update，Exit，均继承自Entity类

其中Update方法设定了当玩家的输入不为null时，状态机切换到行走状态



***PlayerMoveState类（继承Entity）***

作用是定义玩家的移动状态

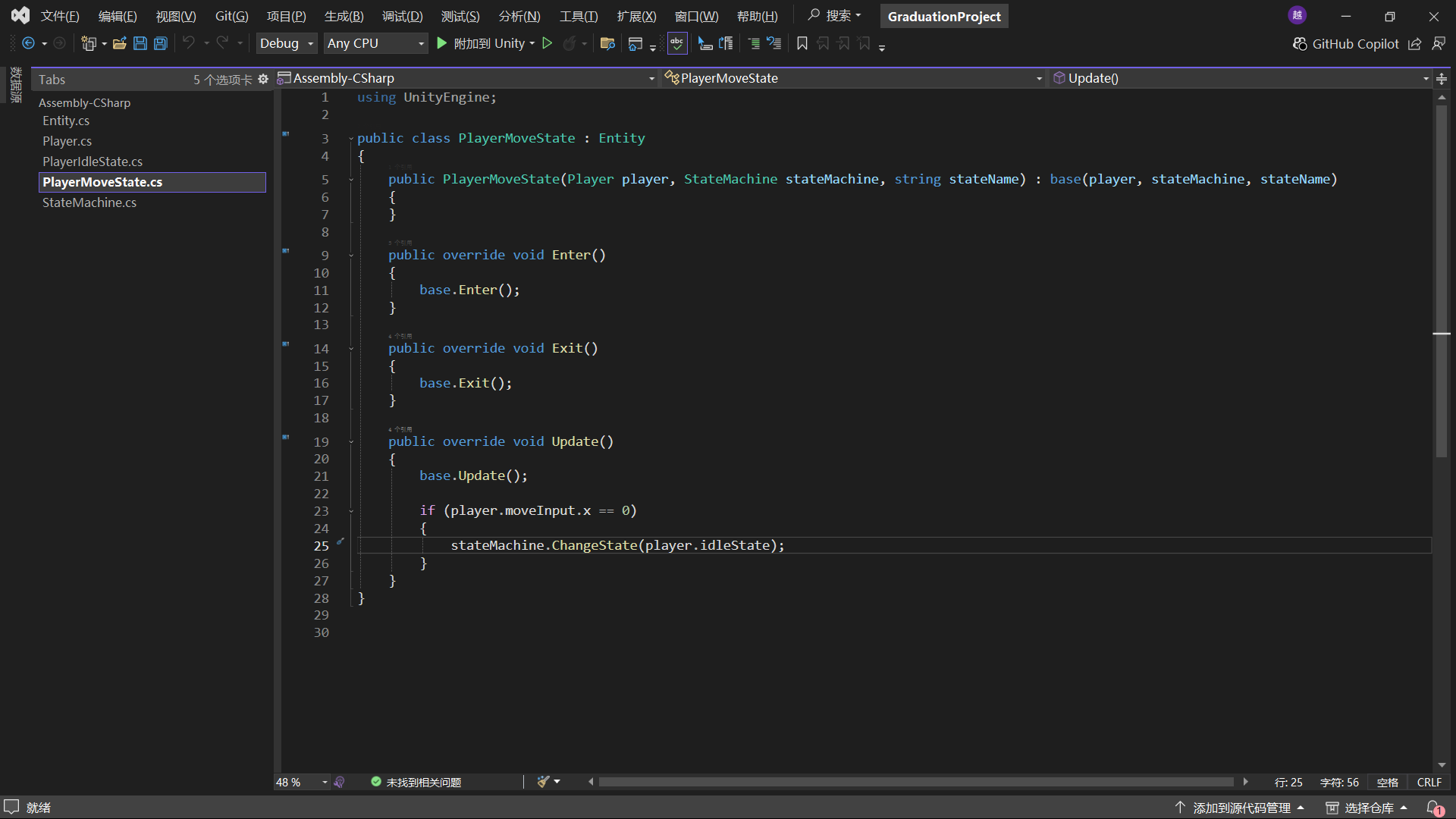
参数：

通过构造函数继承自基类

方法：

Enter，Update，Exit均继承自基类

其中Update方法定义了当玩家的移动输入为零时状态切换为空闲。



***StateMachine类***

作用是管理实体的状态切换和更新

参数：

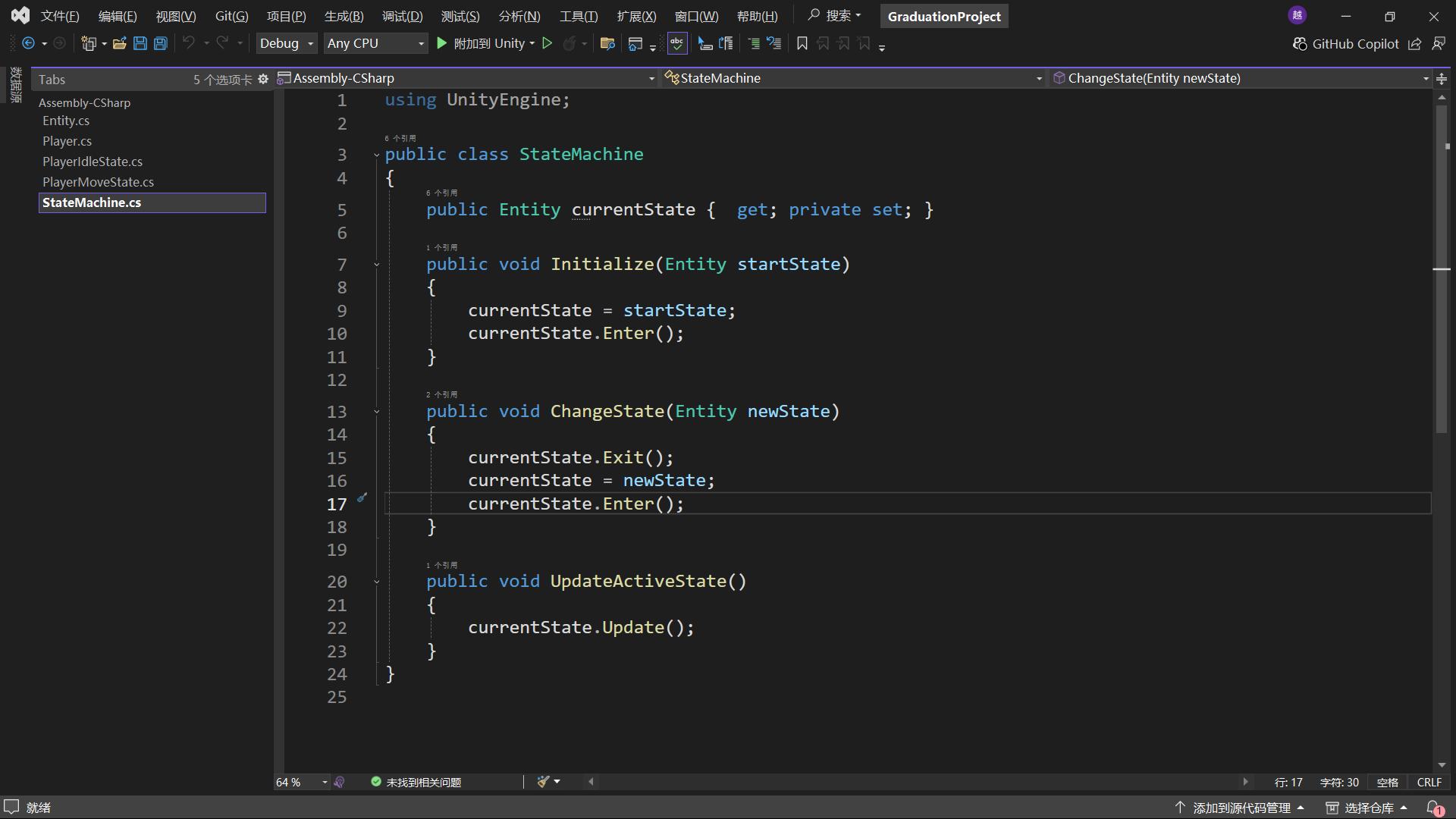
currentState用于设定当前激活的状态

方法：

Initialize用于初始化状态机，设置初始状态并调用该状态的Enter方法。

ChangeState用于控制状态机切换状态，流程为获取新状态——退出当前状态——设置新状态——进入新状态

UpdateActiveState用于更新当前状态，每帧调用当前状态的Update方法（其实这一步可以省略，代价时每次调用Update时要声明 当前状态.Update，现在只需要直接调用这个方法）



目前设定的流程是：

Player对象在Awake()中创建状态机，在Start()中激活并初始化，最开始的默认状态是空闲状态。

每帧更新时，Player调用stateMachine.UpdateActiveState，从而调用当前状态的Update方法

在PlayerIdleState状态下，如果检测到玩家的移动输入，则切换为移动状态。

**现在有什么目标？**

关联状态脚本和序列帧动画，这花不了太多时间

制作其他状态机，比如跳跃、攀墙、翻滚等，还有它们的动画

**有什么期待？**

期待自己写世界观，还有这个角色的动作不是我想要的，我希望之后可以找到更方便的

**总结？**

项目已经上传Github，本篇日志也会上传并且长期更新。

**有什么想对自己说的？**

**请明天的你也如此慷慨激昂。**

0.4616288 0.2409944(这是精灵图片的轴心位置)

**2025.11.9**

开发进度：实现了空闲、行走和跳跃的基础功能，并完成了三个状态之间的切换。其中行走功能添加了面向检测和转向功能，跳跃设置了射线检测用于检测地面，还有通过Blend Tree检测上升/下降动画。

新建的脚本：

PlayerAirState类

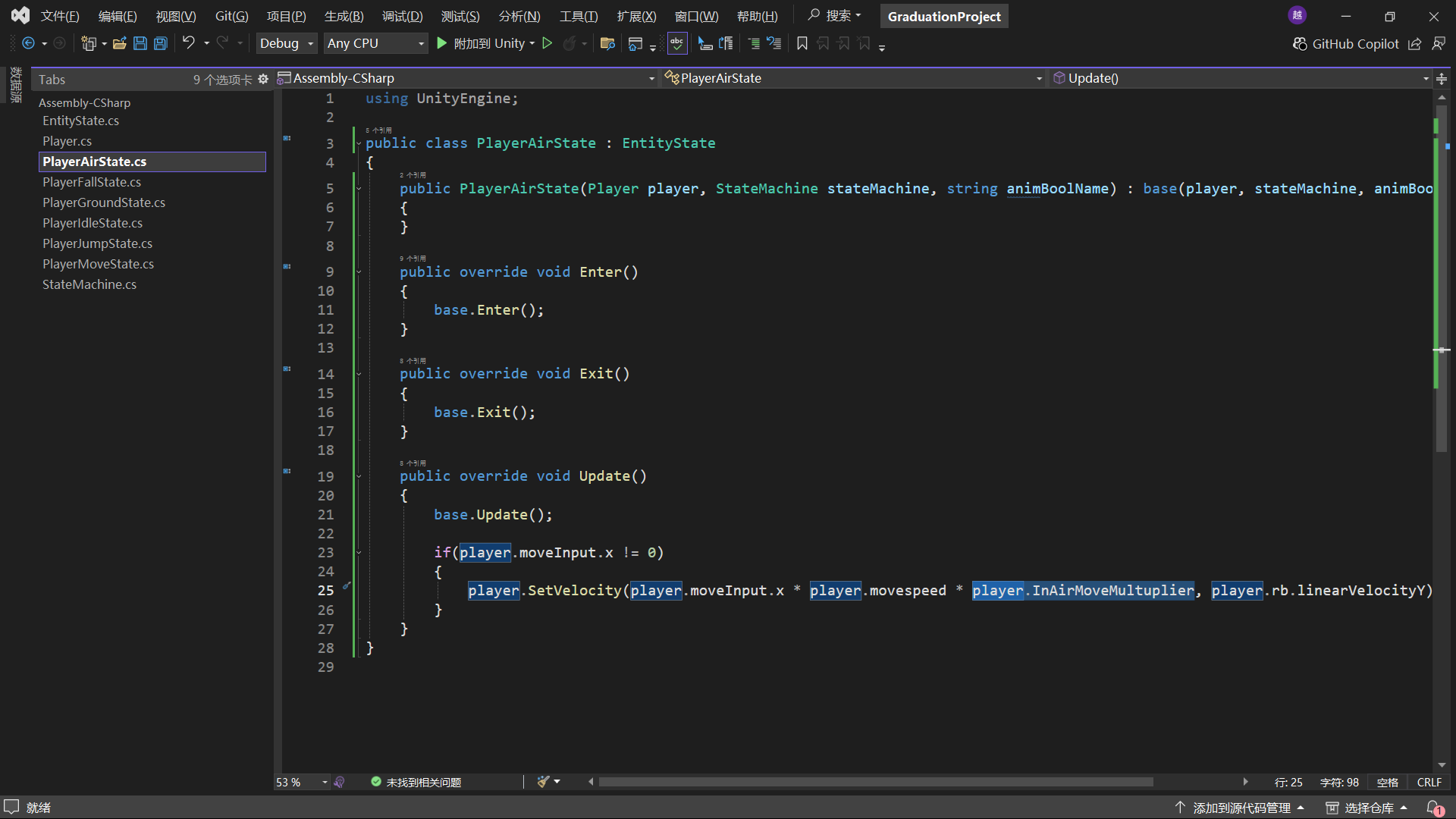
作用是统合目前有的Jump状态（即滞空），后面可能要在里面加一些其他东西，比如空中技能，空中攻击。

参数：

重写自父类Entity

方法：

重写自父类，在Update中设定了如果水平输入不为零，则设定玩家在空中的速度。其中player.InAirMoveMultuplier为0.7，是不想让角色在空中时像在地面一样保持水平速度（会显得有些快，我想做得慢一些，魂系角色那样）



PlayerJumpState类

作用是设定了玩家在从起跳到上升之间的逻辑

参数：

继承自父类PlayerAirState

方法：

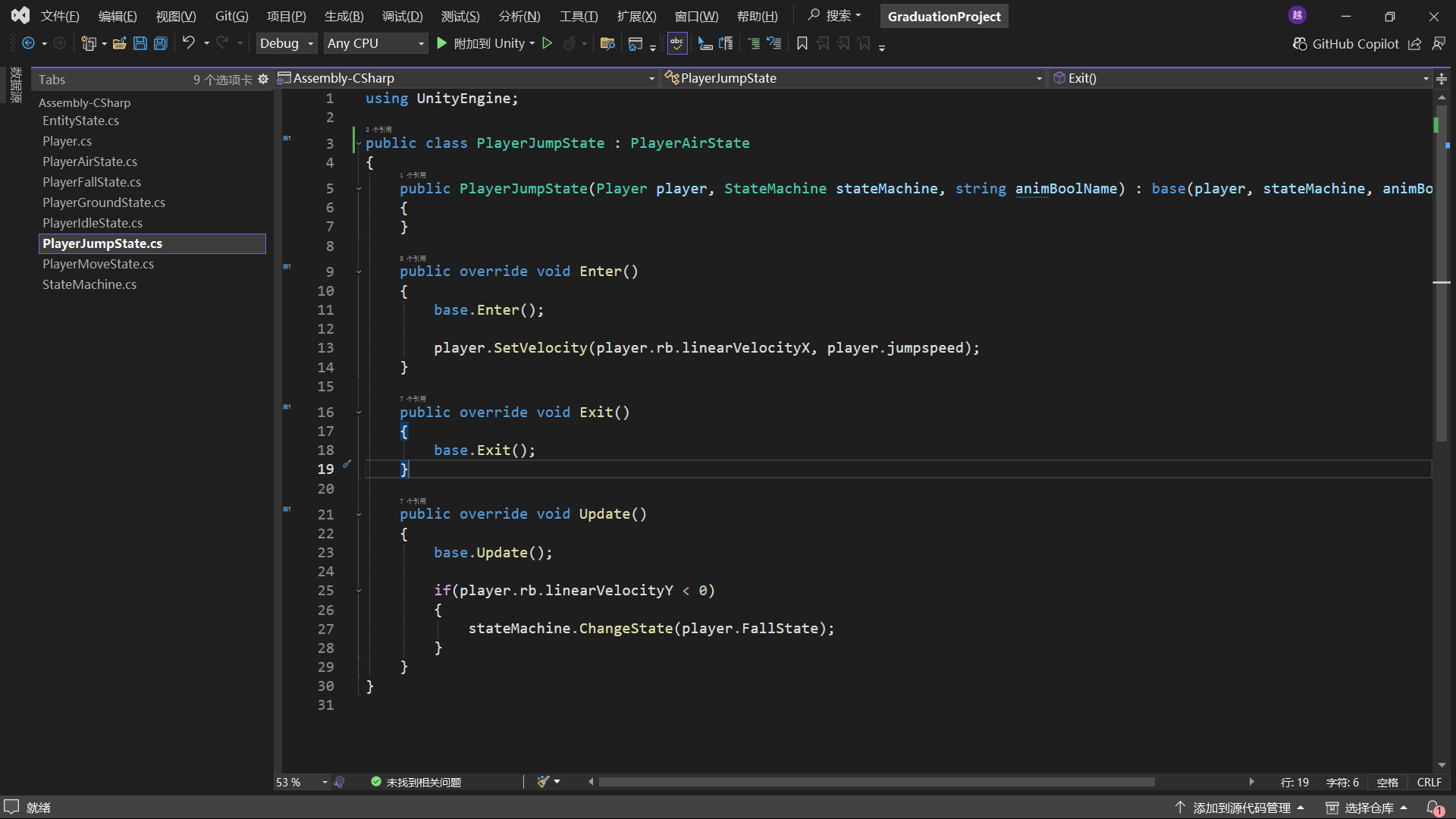
重写自父类，其中Enter设定进入状态时会给予对象一个y轴的上升速度，Update会判断对象下落时进入FallState状态。

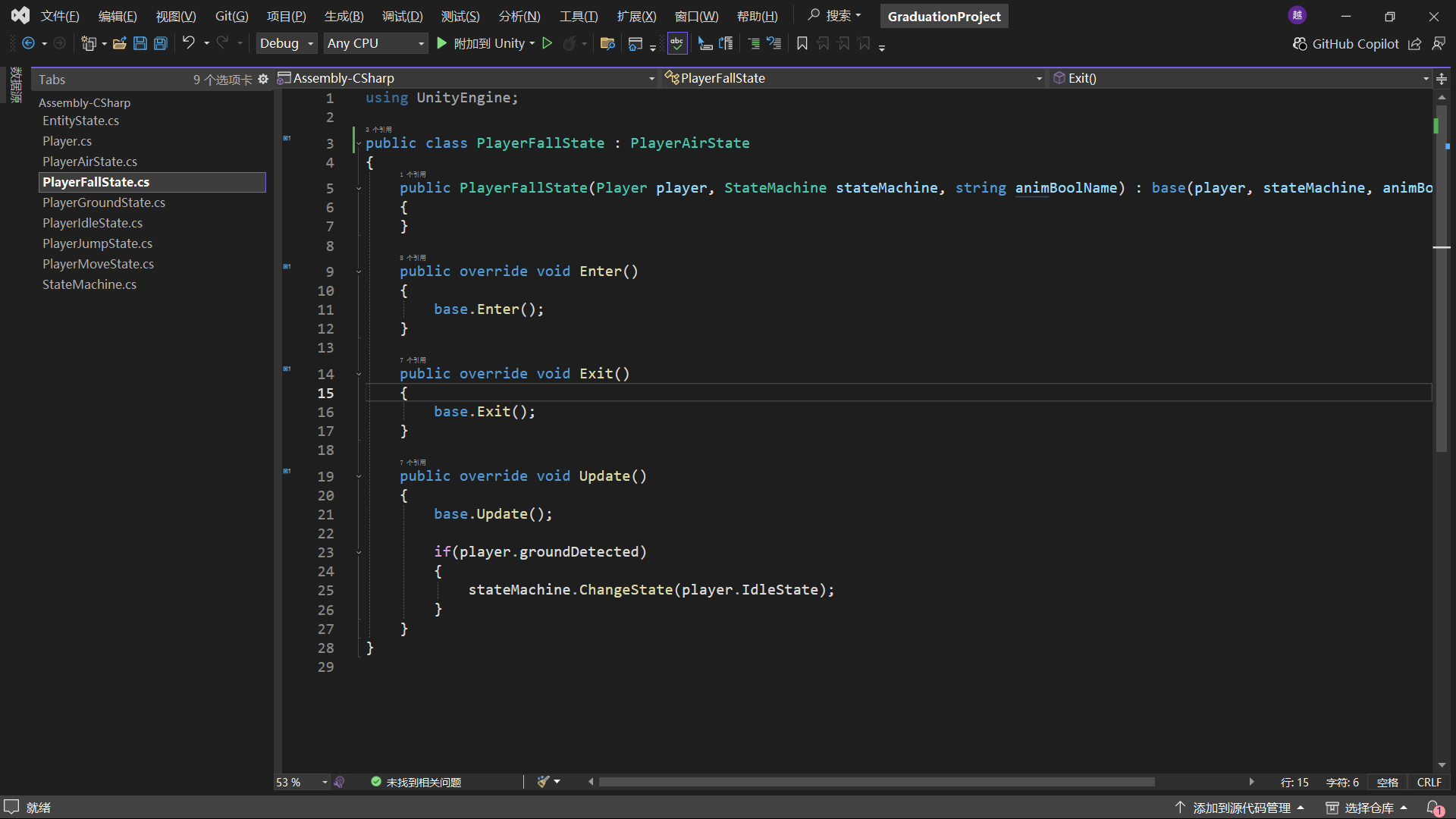
PlayerFallState类

作用是设定了玩家下落的逻辑

参数和方法均继承/重写自父类

Update中设定对象的地面检测为真时进入空闲状态





PlayerGroundState类

作用是统合了玩家的地面状态，目前包括空闲和移动状态。

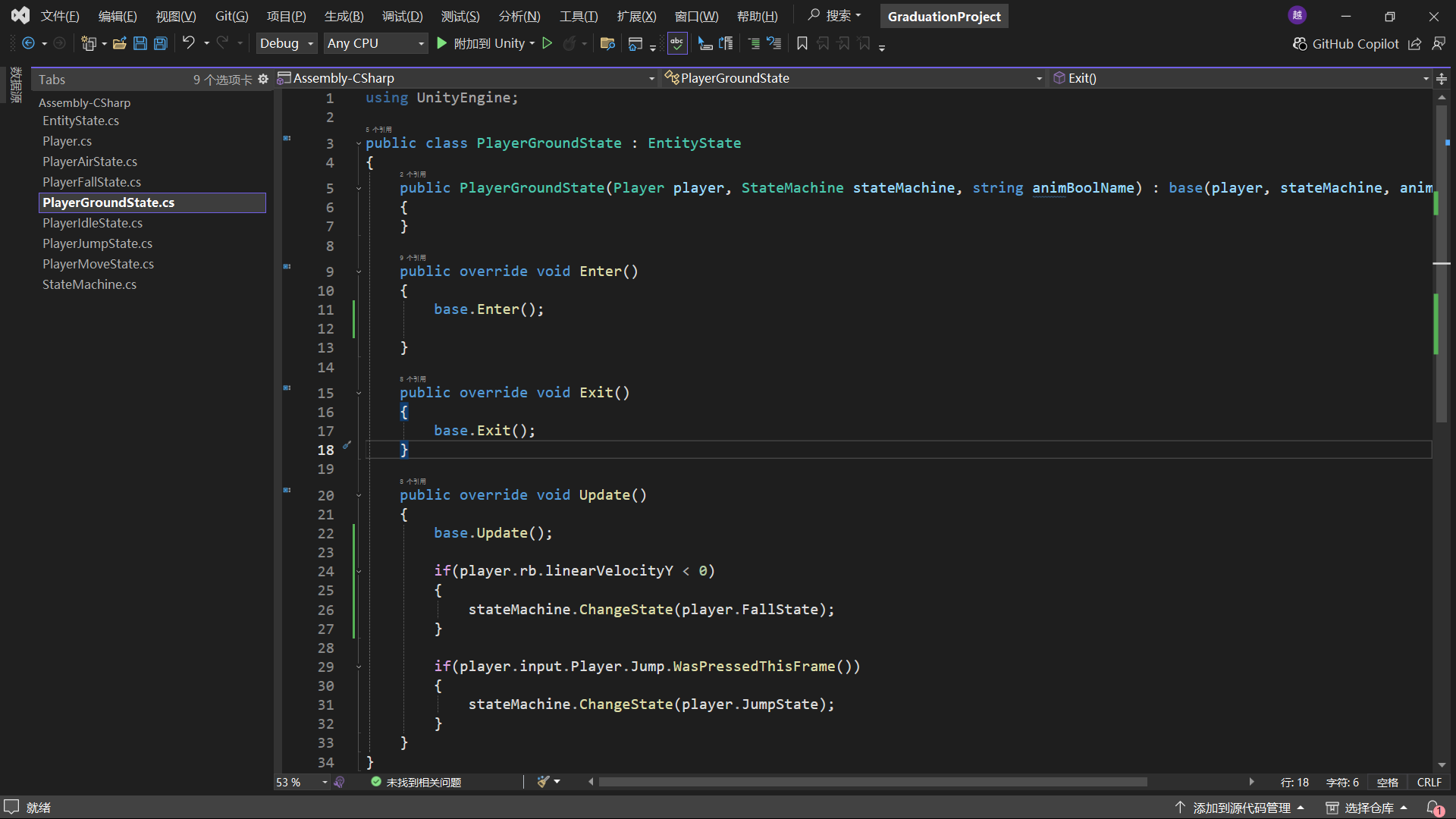
参数：

继承自父类EntityState类

方法：

重写自父类

方法中的设定了角色y轴速度小于1时切换下落状态，这不是跳跃中的下落状态，而是角色从高处自然掉落时的下落状态。还有当检测到对象进行了跳跃输入时会切换到跳跃上升状态。

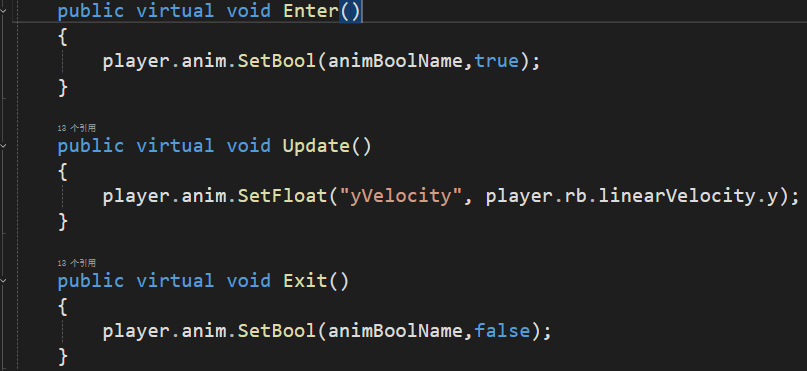


改动了那些类？

Entity类：

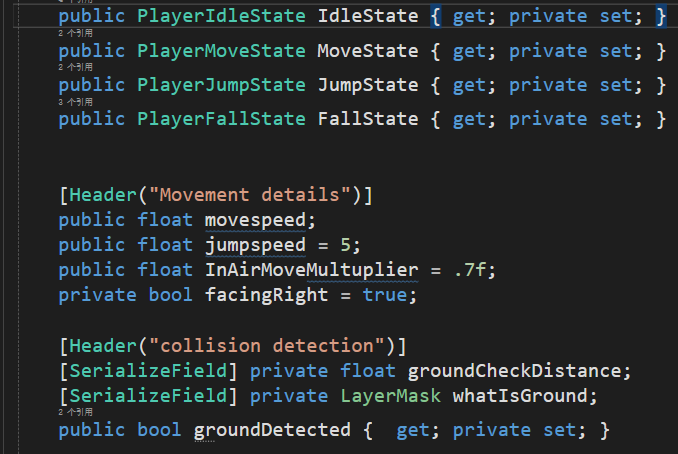
Enter和Exit用于设定状态机动画的进入/推出判断条件

Update中加入了实时检测对象速度，并将其设置为动画BlendTree中的关键参数yVelocity，用于判断上升/下落状态。

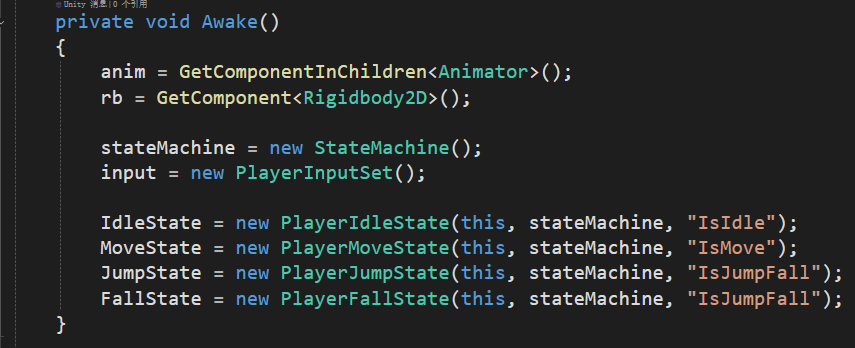


Player类

加了很多东西，参数方面是各类状态、移动和跳跃速度、面朝向，还有射线距离、地面层级定义和地面检测。



Awake方法初始化了各个状态



Flip方法用于定义对象反转，

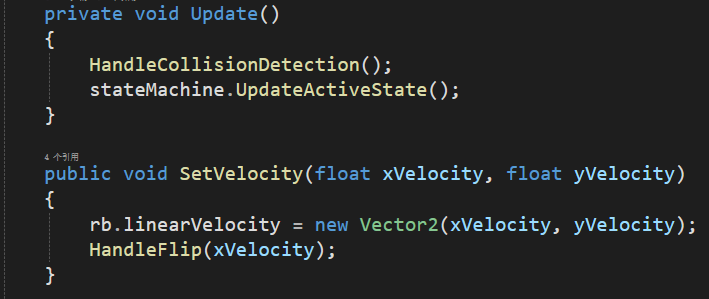
HandleFlip方法传入x轴方向速度，控制调用Flip方法，

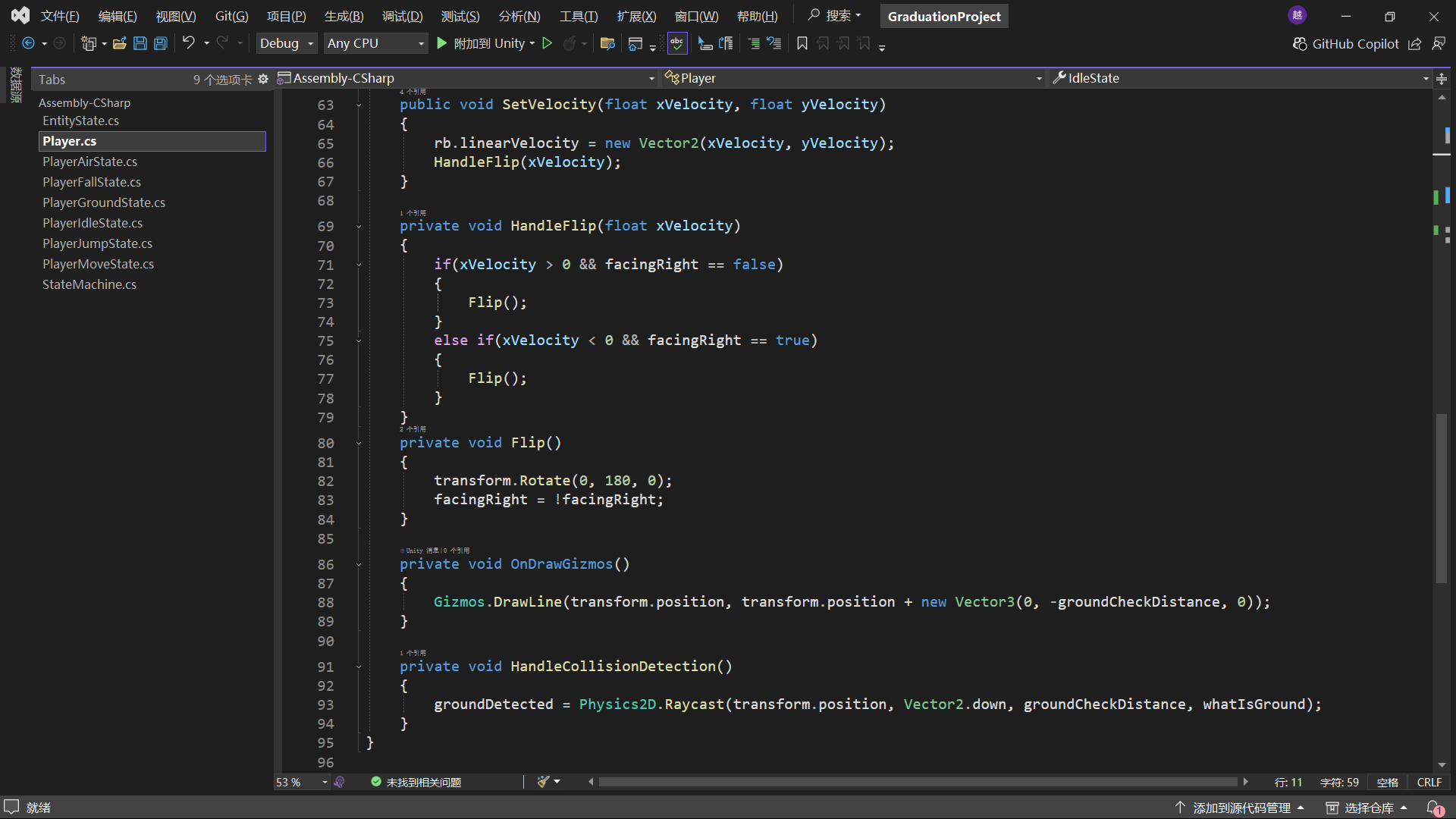
OnDrawGizmos方法用于画射线，

HandleCollisionDetection方法用于进行地面检测（设定groundDetected的真假）

还要主义参数的传入和方法的调用。

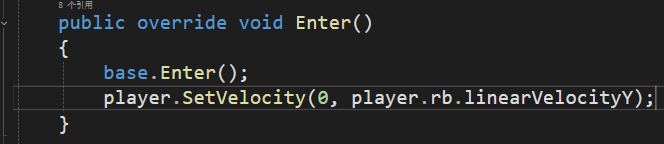
我要特别强调OnDrawGizmos方法的两个参数，第一个transform.position是射线的原点，第二个是射线的检测，由于地面检测射线向下，y值为负，所以长度相加时一定要是负数。





IdleState类

没什么变化，主要就是新设置了当进入空闲状态时对象的x速度为0，用于防止对象空闲时因惯性而平移。



**接下来有什么目标？**

没什么了，应该继续把角色状态做完吧，比如角色滑墙、攻击、翻滚之类的。不过接下来要连上五天班（好累），六点就要起床，每天八点才能回寝室，估计每天写不了多少吧，多学学策划案的东西吧。

**总结？**

呼，加油吧，就这样先写，我想想怎么设计玩家的行为和战斗逻辑，我不想搞太难，就好就是黑魂那种。但是角色的攻击方式只有两种，可恶……

**有什么想对自己说的？**

**不要害怕死亡，去穷尽每一种活着的可能吧！**

**2025.11.10**

开发进度：

尝试设置对象翻滚，遇到了诸多问题，亟待解决。

新建脚本：

PlayerRollState类

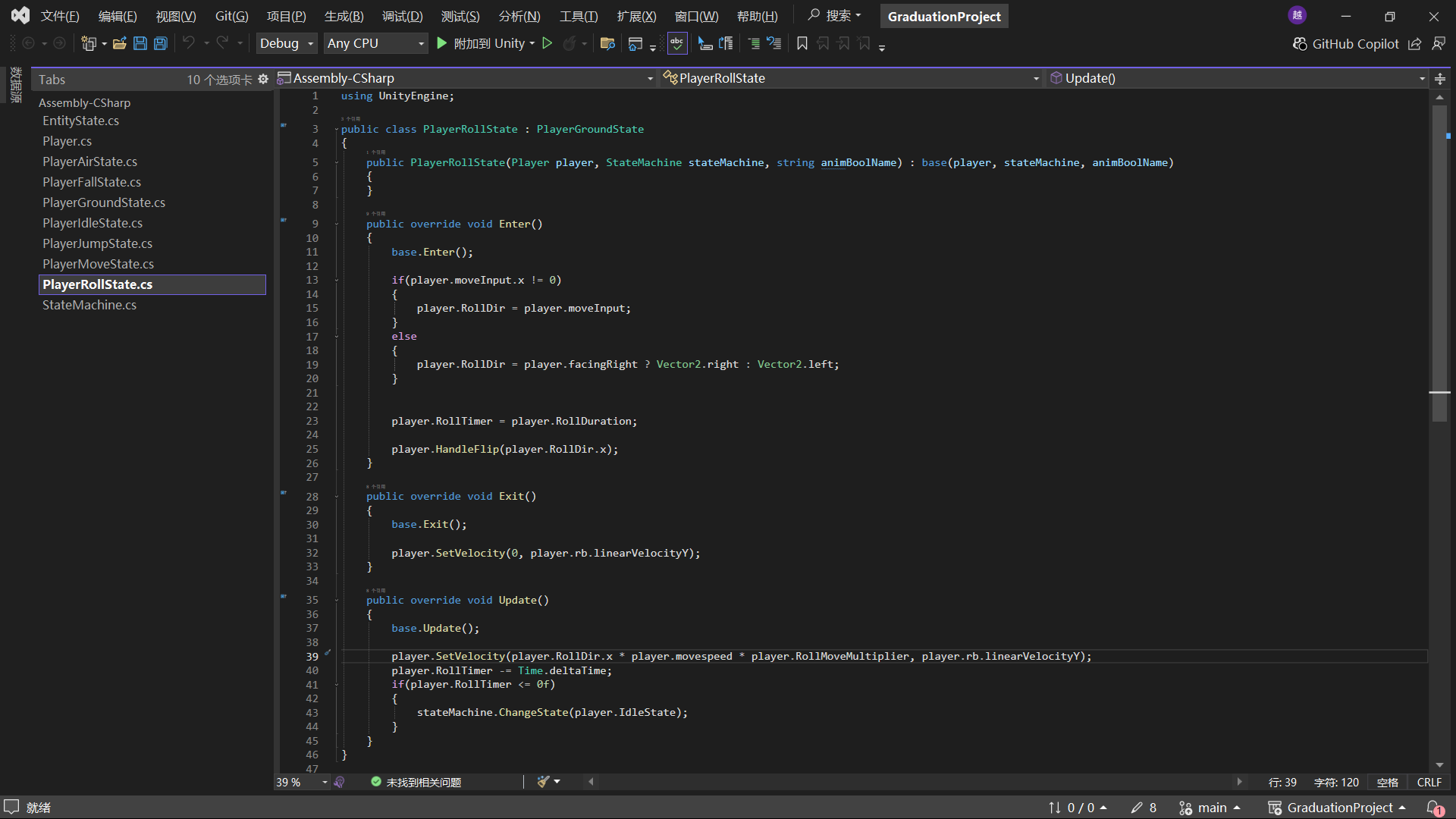
作用是控制玩家翻滚状态。

参数：

继承自父类EntityState类

方法：

继承自父类，其中Enter设置了玩家进入翻滚状态后的面朝向/反转，如果玩家有x轴输入则根据x轴方向决定翻滚方向，如果玩家没有x轴输入，则根据玩家的面朝向，还有初始化玩家的翻滚动画计时器。Update方法中设置了角色的速度、计时器和退出状态。



目前这个类还有很多问题：

1. 角色翻滚时会原地不动，目测是player.RollDir.x为0导致的
2. 角色在翻滚期间改变方向会立刻生效，目测需要加入当前状态判断，如果是翻滚状态则禁用翻滚状态更新。

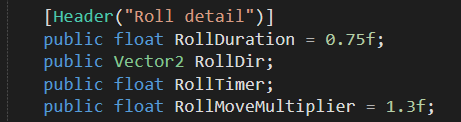
3.角色在翻滚中应该有无敌帧，这几帧期间取消受击判定，这个功能在之后攻击功能完善了再做吧。

改动的类：

Player类

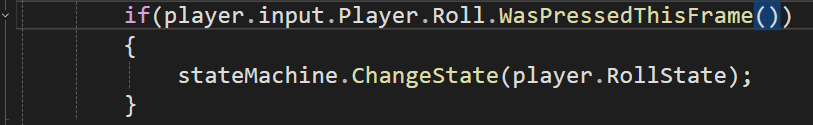
声明了新的状态并初始化，还有加了几个翻滚相关参数。

还有为了方便反转的使用，将HandleFlip方法公开了



PlayerGroundState类

新增了翻滚状态更新，但是这个不完善，导致了PlayerRollState类的问题2



后续应该会加入当前状态判定。

**总结？**

现在没多少时间能用在完善项目上了，每天3小时通勤要累死我了，敲代码的时间也只剩下9点到11点的两小时，根本写不了多少。明天得想办法摸鱼，但公司的电脑一没梯子二是老古董，怎么摸鱼是个问题。

也许我应该回顾一下滑铲的代码，那个跟翻滚很像，应该可以给我提供很好的思路。

**有什么想说的?**

我构思了一个关于奈亚拉托提普的故事，然后想把它写出来，但是我文笔太烂了；想把它画出来，但是我不会画画；想把它唱出来，可是我不会唱歌。

这些并不是遗憾，这是我未来的目标们。

今日文档和代码已更新，明天也是充满希望的一天。

**2025.11.11**

开发进度：

完善角色的翻滚状态，修复了昨天遇到的两个问题，我会在具体的脚本更新中说

新建脚本：

无，本来想单独建立一个闪避父类用于派生翻滚和冲刺两个能力的，但是目前还没这个需求，等后面冲刺能力做出来再说吧。

修改脚本：

PlayerRollState类

这个类的修改主要是加入了计时器，用于解决昨天单次翻滚不能完整结束就可以被第二次翻滚打断的问题，现在在翻滚的时间段内不能再使用翻滚，也不能使用跳跃，后面什么不能攻击之类的也会实现。

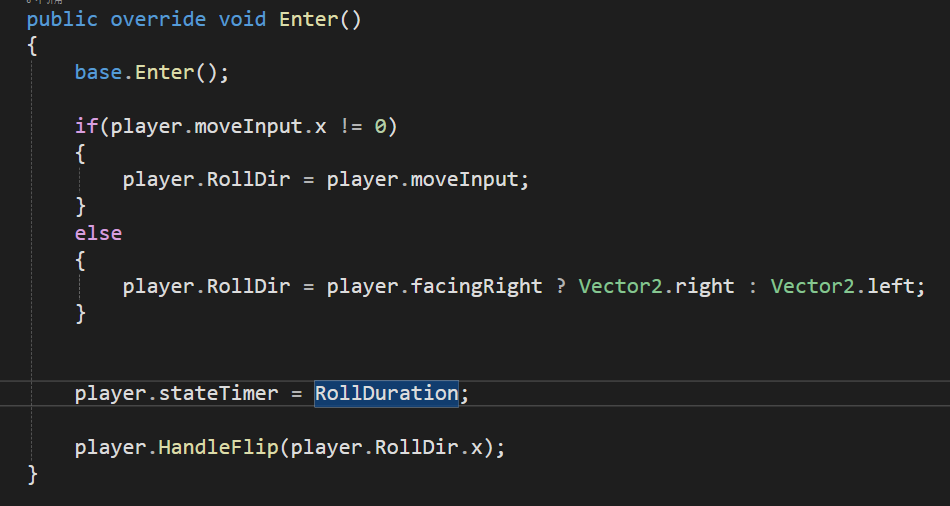
属性：

新增了RollDuration参数，用于记录翻滚所需的动画时长。

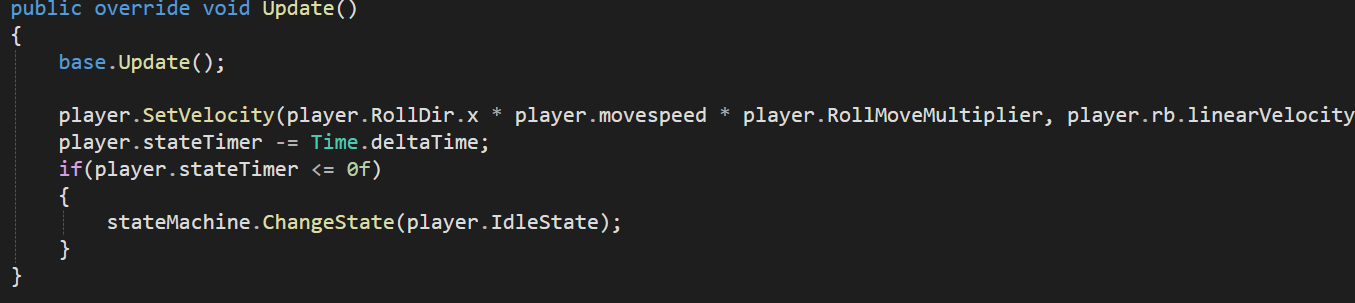
其实算不得新增，主要是把player类里的移过来了，本来计划的是在player类里实现一个计时器方法，其他状态需要时调用，但是考虑到翻滚和冲刺



计时器初始化



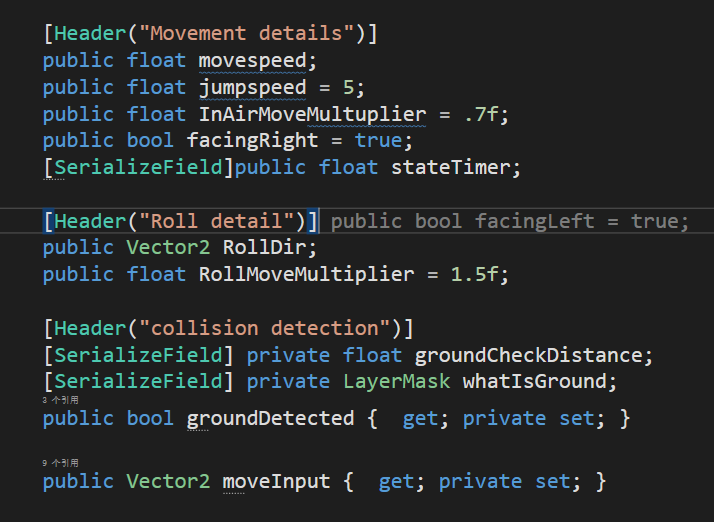
计时器更新



此外还有退出状态时将计时器归零，防止计时器数据残留。以及速度更新，解决了原地翻滚不位移的问题。

Player类

主要就是修改了一下参数，现在stateTimer作为通用计时器可以在各个类中访问，但是可能存在不同状态数据污染的问题，这个日后解决。

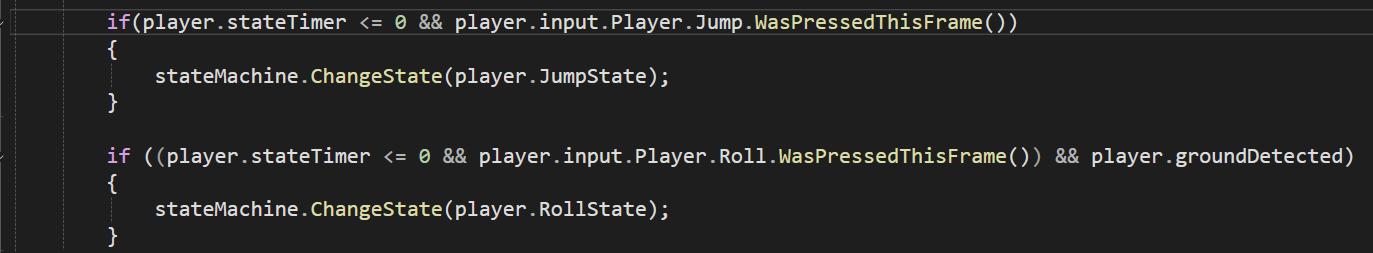


PlayerGroundState类

改动主要是更正了跳跃和翻滚的判定条件，防止冲突。

跳跃方面是必须计时器为0且在地面才能跳跃。

翻滚方面是必须计时器为0、按下翻滚键（目前绑定空格——我觉得空格翻滚比空格跳跃更好用，鉴定为玩黑夜君临玩的）和在地面才能翻滚。



**总结？**

上班好累，我想回家，通勤三小时太痛苦了，租房子又没钱，真是烂完了。

今天基本上完成了昨天的遗留问题，我太tm棒了，真的。

**想说的话？**

我想成为一个英雄主义者，兼存在主义者，兼理想主义者。

**2025.11.12**

开发进度：

制作角色的冲刺状态，看起来与翻滚没什么区别，只是触发方式不同。我规定翻滚只能在角色接触地面，而冲刺可以在空中使用，两个位移的时间不同，我设想的是给冲刺设置cd，但是目前还没这个需求。

新建脚本：

PlayerDashState类

这个类主要用于控制玩家的冲刺，其内容与翻滚差不多。我还没想好要怎么做冲刺与翻滚的区分，目前翻滚应该是战斗中常用的位移手段，而冲刺也许应该加一些其他效果。再看吧。

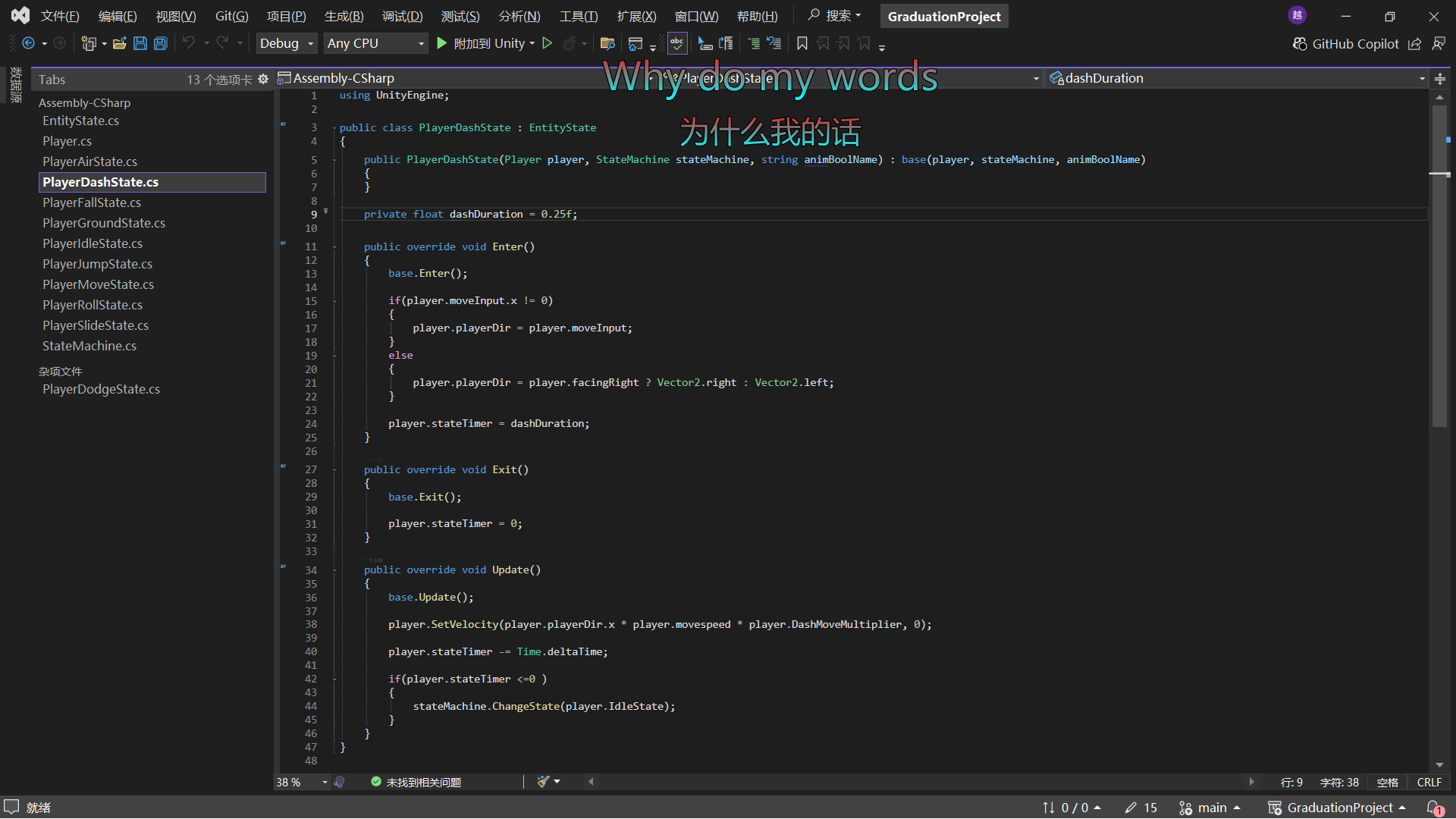
参数：

继承自父类EntityState类

dashDuration，用于确定冲刺持续时间。

方法：

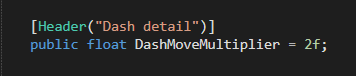
均继承自父类，几乎所有部分都与PlayerRollState一样。



修改脚本：

Player类

主要新增了冲刺速度修正参数



PlayerRollState类

主要修改了计时器，将RollDuration改为rollDuration，我希望用驼峰命名法保证规范性。



总结？

今天的工作不多，明天有一天假期，可以多做一些事情。我希望明天把攻击做了。还有一个位移是滑铲，我希望把那作为能力，目前设想是这样的：如果滑铲成功躲开了敌人的攻击，角色的下一次攻击会击晕敌人——不过这个部分可能需要先做敌人的逻辑了。

你，可还有话要说？

他一无所有了……他连一无所有都将没有了。

**2025.11.13**

**开发进度：**

今天请假做了一下午呢，虽然很累但是很充实哈。

那么今天做了角色的滑墙系统，不过做得也不是很完善，主要是做了滑墙、解决滑墙和冲刺的冲突问题、滑墙跳的功能，还有一些角色攻击的功能，但是我打算明天做，因为要把combo和攻击判定做出来。

新建脚本：

PlayerWallSlideState类

这个类就是处理了玩家滑墙的一些功能，还有一些细节，会在下面的具体参数和方法中说明。

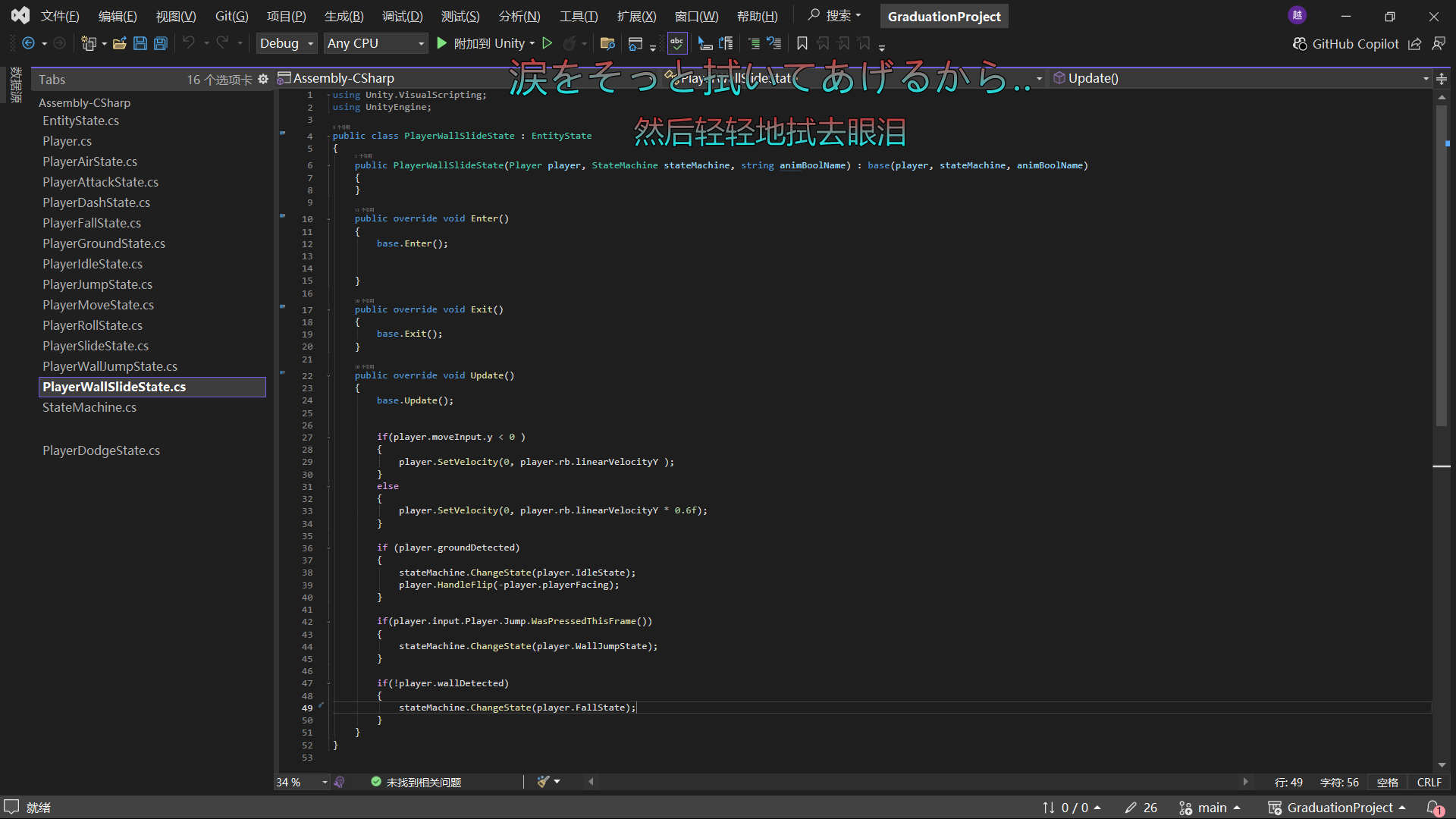
参数：

继承自父类EntityState类

方法：

继承自父类

其中update里实现了这些功能：通过监听玩家是否按X键(方向向下)决定是缓慢滑落还是快速滑落、当玩家接触地面时，切换为站立姿态，同时角色反转（反转的原因是角色的滑墙图集与站立面向是相反的，站立面向墙，滑墙时背向墙）、当玩家再滑墙状态跳跃时，他会向背离墙的方向获得速度，并切换为滑墙跳状态。最后还有一个墙壁碰撞检测，玩家的墙壁检测射线没有检测到墙的时候，会进入下降状态，这一点主要是解决dash期间按反向会导致角色反转但是依旧攀墙的状态。



PlayerWallJumpState类

作用是解决滑墙时跳跃的问题。

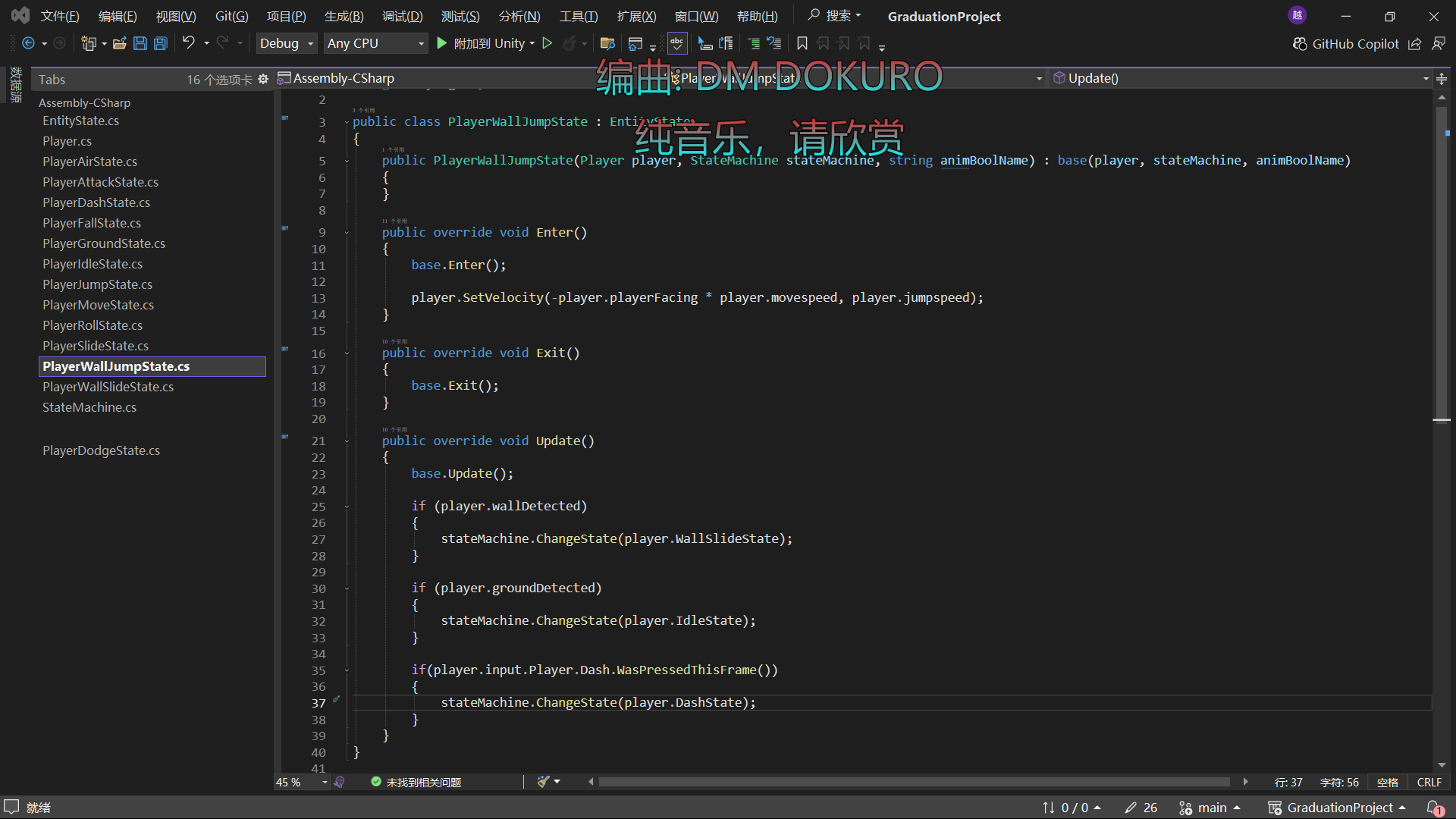
参数：

继承自父类

方法：

继承自父类，其中Enter在玩家攀墙跳时会赋予一个反向的初速度。Update类用于判断状态退出条件，玩家贴墙则进入攀墙状态，玩家接地则进入空闲状态，玩家冲刺则进入冲刺状态。

区分翻滚、冲刺和滑行的议程已经迫在眉睫了。

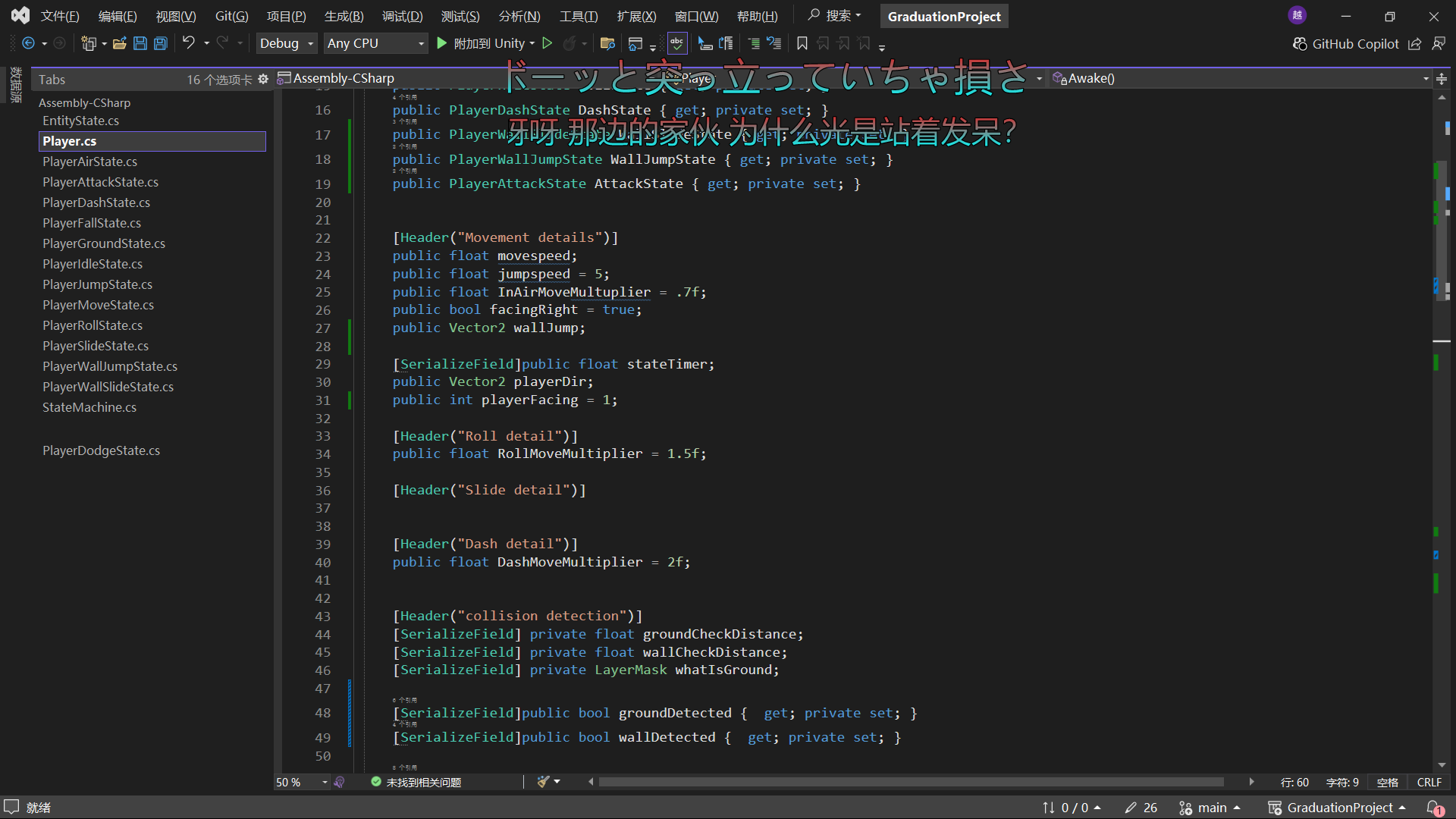


Attack类还没想好怎么做，明天再放吧。

修改脚本：

Player类

主要还是初始化状态机，新加的这个Vector2类型参数我觉得不是很有必要，因为跳跃的x轴和y轴速度可以直接给，而后面很少有需要修改这个参数的可能。



总结：

今天说不上充实，因为躺了一会。上班时候想着做自己的游戏，真能做了却像上床躺着……果然得不到的东西才是最好的吧。明天会把攻击做出来，明天可以多做半小时了，啊，生活多么美好，也许。

有什么想说？

全员，向着灯火闪烁的方向……

浪潮，会吞噬一切

**2025.11.14**

开发进度：

今天制作了初步的攻击动画，包括进入和退出，这个攻击动画的最后一帧设置了事件触发器，用于判断退出，顺便给方便明天的攻击combo判定，以及攻击事件的触发。

触发器的知识又忘了，今天看应该只是个类，当触发时实现类内设定的功能。

新建脚本：

PlayerAnimationTrigger类

作用是用作动画触发器，在添加事件之后，会根据触发器的触发执行相应代码逻辑

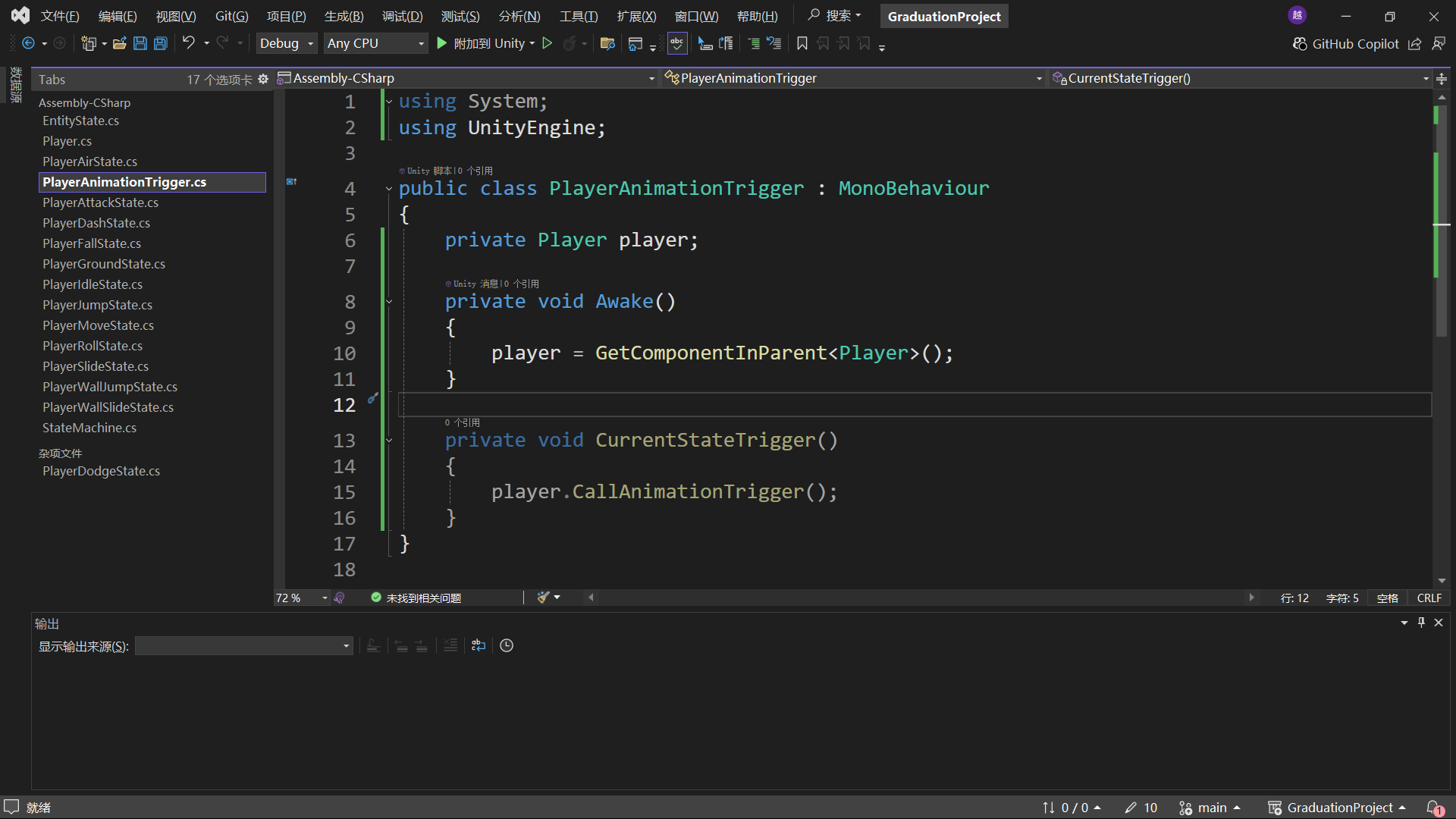
参数：

player，用于获取player对象的组件，传递给该类。

方法：

Awake，继承自MonoBehaviour的方法，用于初始化，这里是在初始化中获取Player组件。

CurrentStateTrigger，用于调用Player类中的CallAnimationTrigger方法，具体作用在Player改动中说明。



AttackState类

作用时用于控制攻击，目前只能控制第一段攻击，明天会做成“基础攻击”，与后面的攻击区分开来。

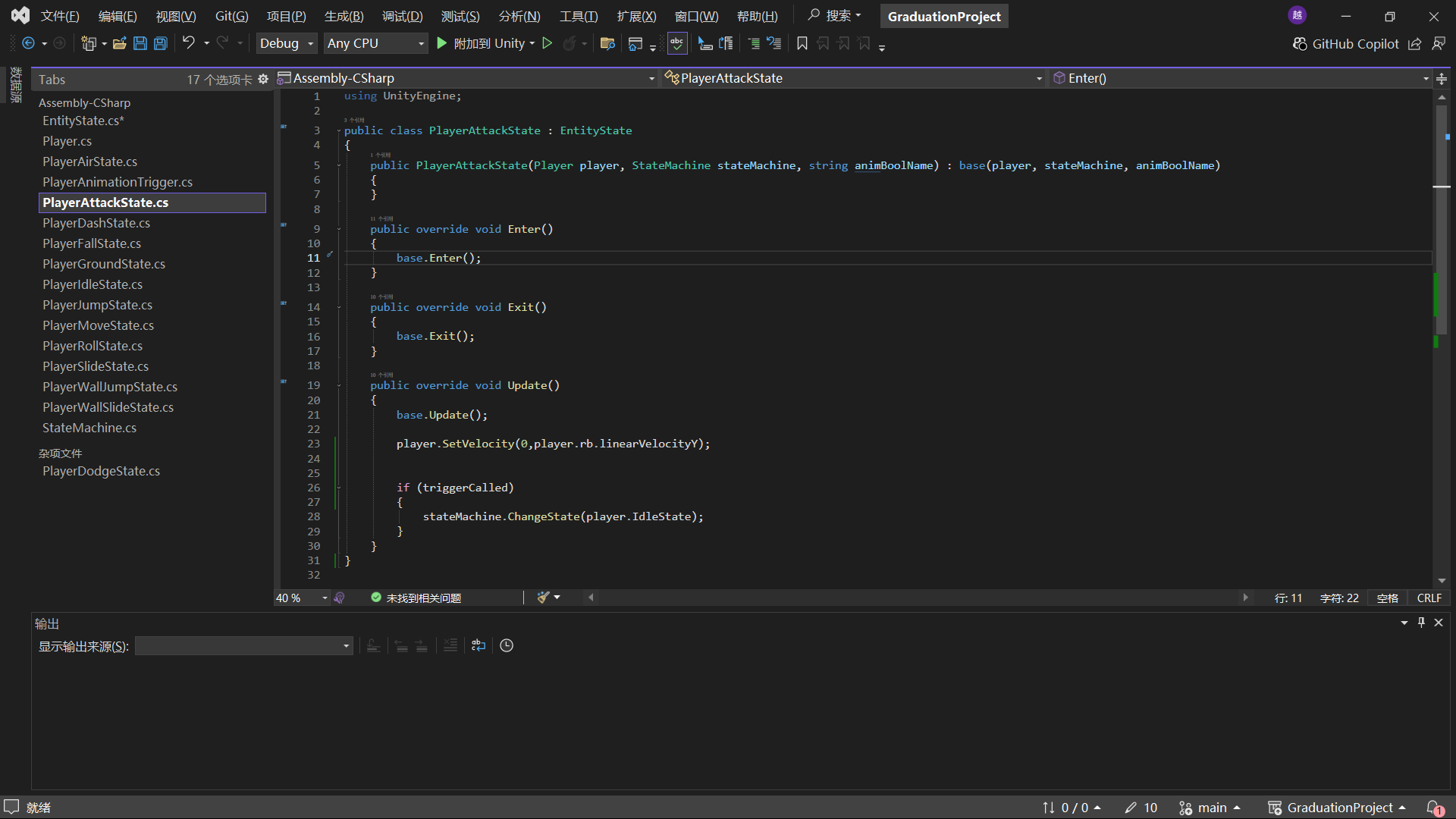
参数：

继承自父类EntityState类

方法：

继承自父类

在Update方法里进行了设置攻击时速度和攻击结束、触发器触发时，会进入IdleState状态。



改动脚本:

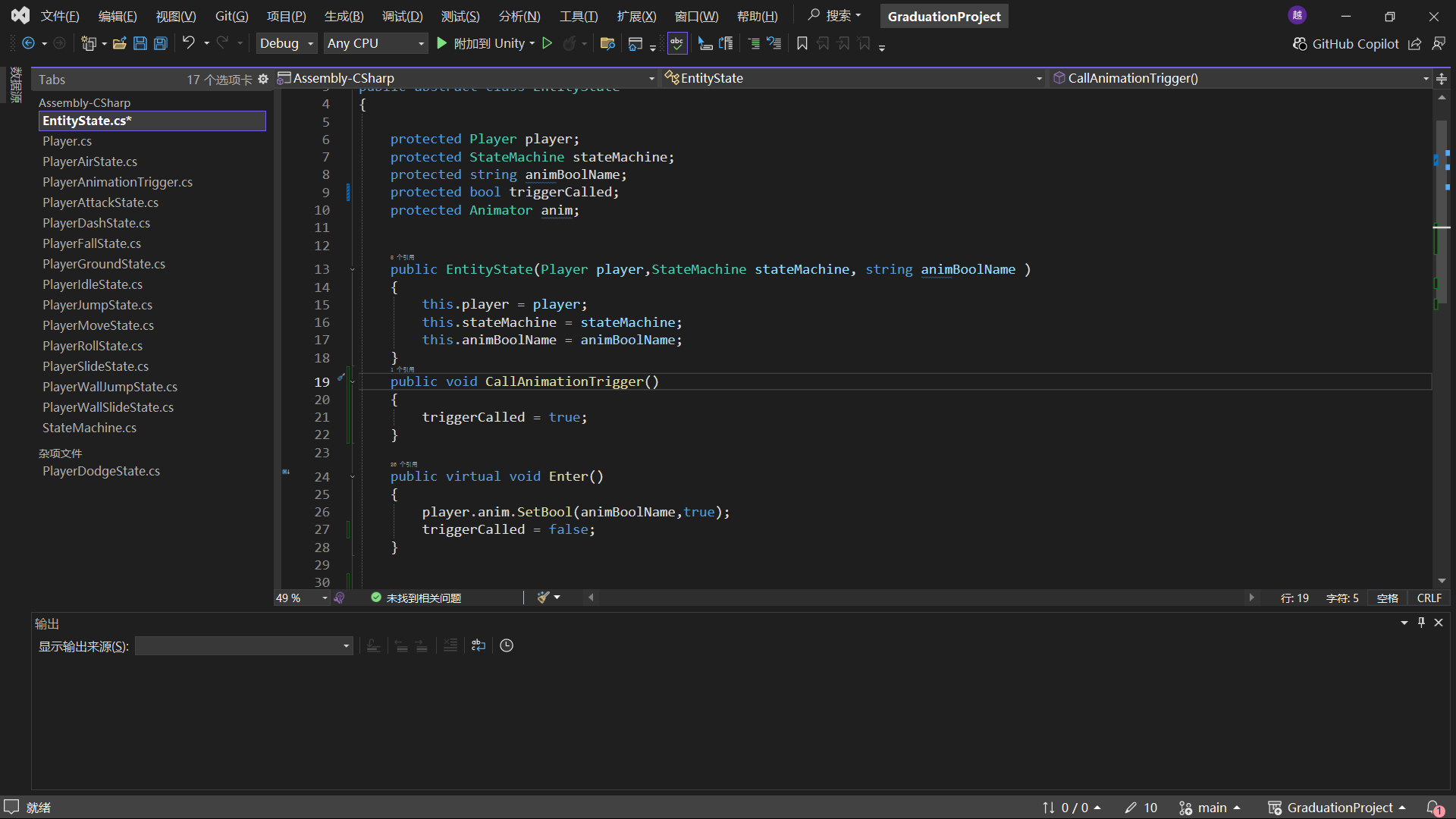
Entity类

参数：

新增布尔类型变量triggerCalled，用于记录触发器触情况。

方法：

CallAnimationTrigger方法，用于初始化triggerCalled，最开始是true



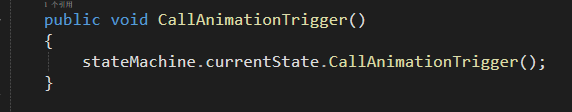
Player类

参数：

声明和初始化，我懒得截图了，就是声明那个状态，然后new一个

方法：

CallAnimationTrigger将动画事件传递给状态机当前激活的状态



总结：

明天是周！末！

我要把角色基础逻辑做完，最好还能做一点敌人逻辑……哦可能做不完，因为明天下午要和主程、主美讨论下美术风格，可惜明天副程序不在，我还想统一下代码分格来着。呼，明天真是充满了希望啊！

有什么想说的吗？

那些听不见音乐的人，都说跳舞的人是疯子。

但是每一个不曾起舞的日子，都是对我生命的辜负。

**2025.11.15**

开发进度：今天把连招系统做了一下，顺便用协程确保了攻击之间更加流畅（比我想象的难，不得不又回去看了下思路）。

新建脚本：

今天没有新建脚本，都是在原来的脚本上进行的修改。

修改脚本：

EntityState类

参数：

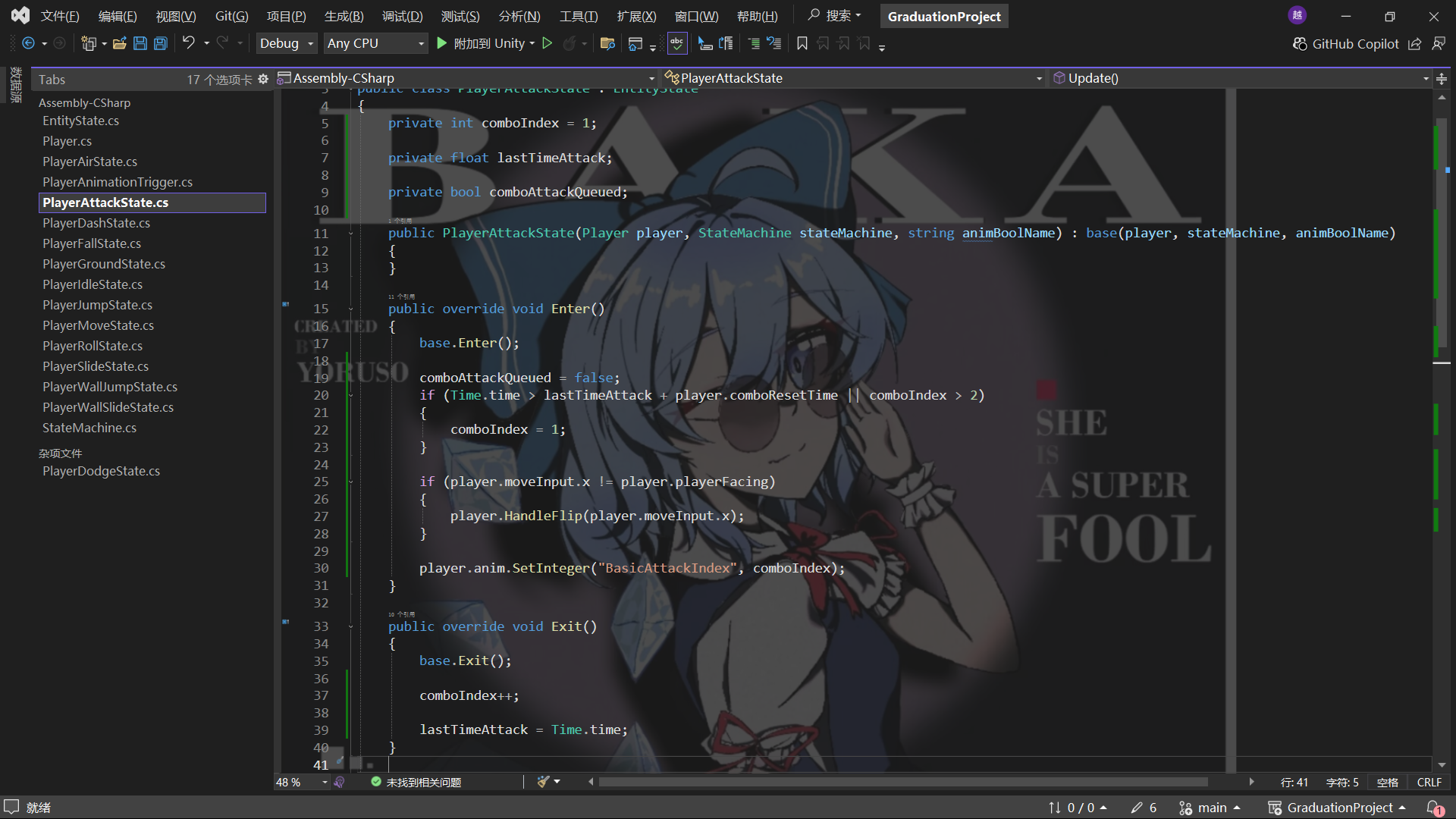
comboIndex，用于记录当前攻击段数

lastTimeAttack，用于记录最后一次攻击的时间

comboAttackQueue，用于判断已输入下一段攻击

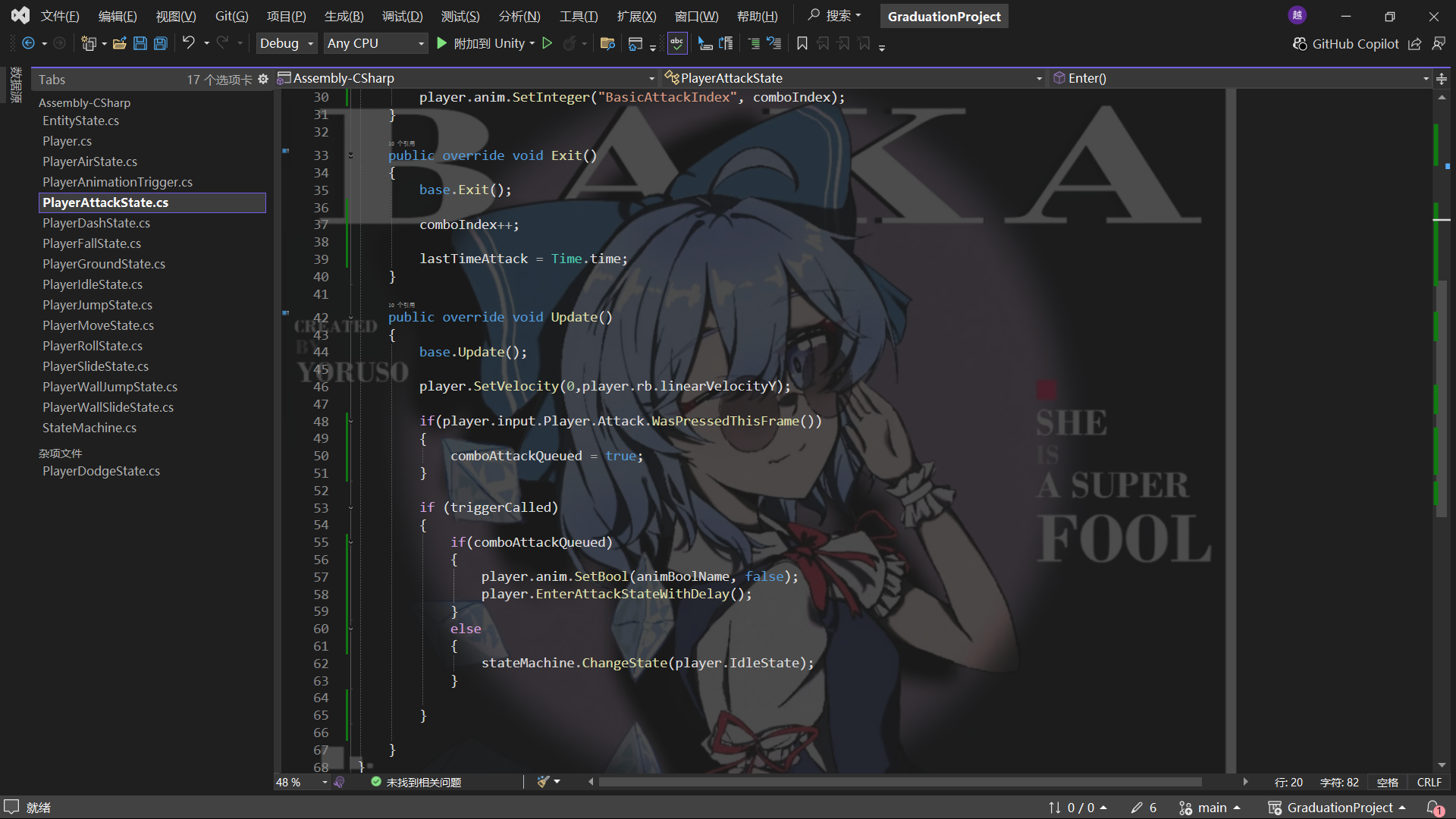
方法：

Enter中新增了设置攻击指示，会把当前的comboIndex传给动画系统的BasicAttackIndex，再由BasicAttackIndex去判断使用哪段攻击动画。



Exit方法中，将最后一次攻击的时间记录为当前游戏时间，用于判断下次攻击的时间间隔。

Update方法中，监听玩家是否存在连击，如果有连击，就将连击指示设置为true。后面的判定中如果有连击，会强制停止要进入的状态（这里表现为IdleState），并在延迟后进入下次攻击。

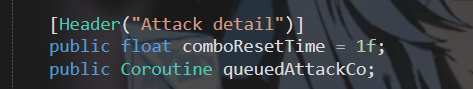


Player类

参数：

comboResetTime，攻击重置时间，如果自最后一次起到下一次攻击大于这个数，攻击次数重置。

queuedAttackCo，用于储存当前的攻击协程引用，具体作用为跟踪当前正在运行的攻击协程、防止多个攻击协程同时运行、提供取消未执行攻击的能力。如果没有这个参数，多次点击将触发多个攻击协程。

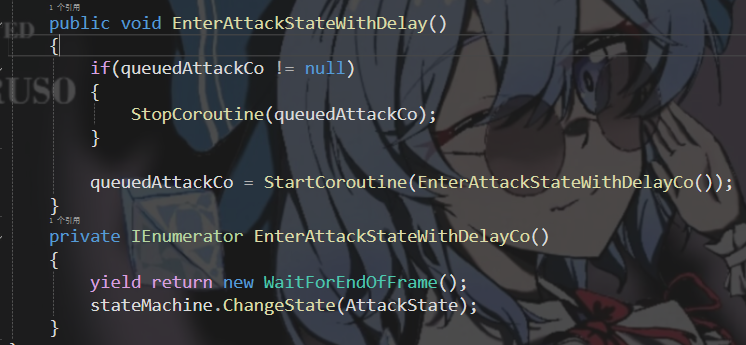


方法：

EnterAttackStateWithDelay()，如果以及存在攻击协程，则停止它，以此确保每次只有一个攻击协程生效。然后再调用攻击协程。

EnterAttackStateWithDelayCo()，等待当前帧结束（即上一个攻击帧），然后切换到攻击状态。

这个协程的意义在于避免了攻击1-空闲-攻击2的空闲导致动作不连贯问题，现在如果玩家连击，会直接再攻击1-攻击2之间切换，如果攻击段数更多也差不多。

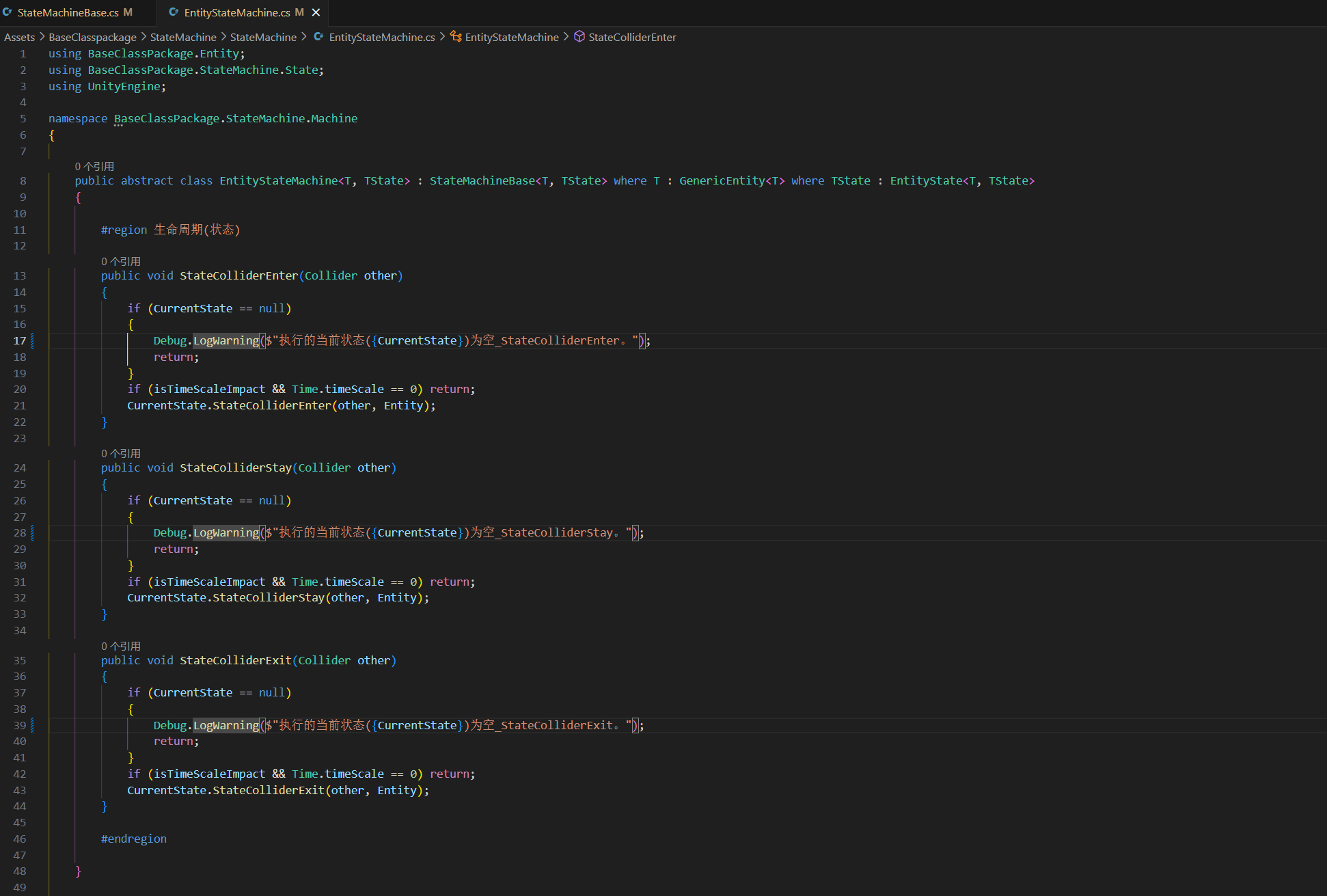


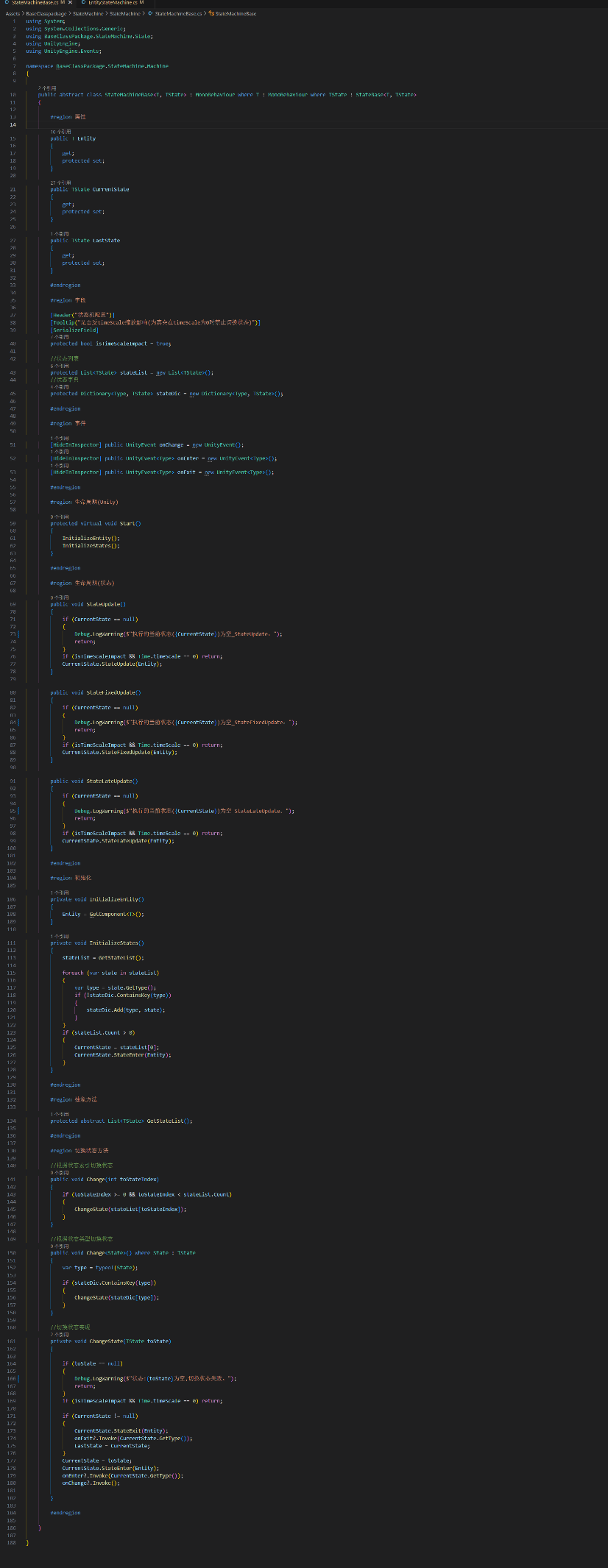
总结：

今天和一位老资历谈了一下，指出了代码的一些可以改进的地方

1. 用泛型替代具体类型，方便从EntityState中派生，不需要再特地实例化角色和敌人
2. 目前Player类负责了状态的声明和初始化，这部分不应该由Player类负责，而是应该由单独的状态机类来管理，以此降低代码耦合
3. 状态不用考虑缓存引用，让状态机缓存就行，状态方法加上参数就好

这是他给我的一些参考：







明天先把攻击写完吧（应该还会加个跳跃攻击，然后再参考一下这些建议修改一下代码）

另外今天GGJ队伍一致决定要先打一场磨合一下，我得找个大伙都有时间的时候……还有要学点策划要会的东西。

有什么想说的？

不要成为空想者。