第一次版本更新

1.血量与伤害

1.1主角将有3点血，与怪物第一次接触时掉1点血，持续接触时每秒掉1点血

实现方式：

1.用OnCollisionEnter判断当发生碰撞检测时就调用Damage方法掉血。

2.中OnCollisionStay中每次增加Time.deltaTime,当积累的时间超过1秒时持续掉血。

遇到的问题：

最开始我在主角身上写的碰撞检测怪物，但是无法获得怪物重生后经过的时间，所以我把碰撞检测放到了怪物的逻辑中

1.2怪物血量为1点，被灯光照射或子弹命中后受1点伤害

实现方式：

当玩家开枪时，向前发射子弹，并进行射线检测，如果检测到碰撞的物体的tag是"Monster"就会调用怪物的Damage方法掉血

2.死亡与重生

2.1 血量降至0的主角应播放死亡动画，其模型会淡出场景，游戏场景随后将重新加载

实现方式：

1.角色收到伤害后调用Damage方法，如果HP为0则设置Animator中的变量为true，然后状态机切换到死亡动画，Update中判断IsDead，如果为true就会调用gameEnding.CaughtPlayer方法，

2.随后弹出淡出的UI，然后使用SceneManager.LoadScene (0)重新加载场景。

```
private void Die()
{
//禁用移动
    GetComponent<PlayerMovement>().enabled=false;
    m_IsDead=true;
    m_Animator.SetBool(PlayerAniPara.IS_DEAD,m_IsDead);
}
```

2.2场景中有重生点，死亡的怪物在符合特定条件下会在特定地点重生，重生后的前3秒不会伤害玩家

实现方式：

1.Awake中记录怪物的初始位置和旋转，当怪物死亡前就把怪物放到一个**列表**中，然后setactive为false表示死亡

2.重生时记录时间，在碰撞检测时判断如果记录时间没有到达3秒则不会执行伤害方法。

```
public void Reactive()
{
    for(int i=0;i<EnemyList.Count;i++)
    {
        EnemyList[i].SetActive(true);
        EnemyList[i].GetComponent<MonsterDamage>().init();
    }
}
```

遇到的问题：

开始想传递参数后销毁怪物对象，后续通过Instantiate生成怪物，但是由于销毁了怪物对象无法使用位置和旋转，最后改为不销毁对象直接设置active属性

3.场景机制

3.1每个房间有一盏灯，灯亮10秒钟后自动熄灭，灯灭15秒后自动亮起；在灯为亮起状态时，进入灯光5米范围内的怪物将会死亡；灯灭3秒后，相应房间内处于死亡状态的怪物将重生

实现方式：

1.使用一个变量记录灯积累的时间，如果大于10s就setactive为false，切换标志为true，下次如果大于15s时就设置为true，实现灯的亮和灭。

2.灯灭后另外积累一个变量，当到3秒后遍历之前的怪物**列表**，setactive为true，然后调用初始化方法，初始位置，旋转和生命值。

```
void Update()
{
    m_nowTime+=Time.deltaTime;
    if(m_NextStatus==false)//现在是关灯
    {
        m_nowTimeReactive+=Time.deltaTime;
        if(m_nowTimeReactive>=m_ReactiveTime)
        {
            m_nowTimeReactive-=ConstSetting.MAX_INF_TIME;
            //关灯后三秒复活怪物
            m_EnemyMana.Reactive();

        }
        //15秒后开灯
        if(m_nowTime>=ConstSetting.SWITCH_ON_TIME)
        {
```

```
            m_nowTime-=ConstSetting.SWITCH_ON_TIME;
            m_NextStatus=true;
            OpenLight();
        }
    }
    else if(m_NextStatus==true)//现在在开灯
    {
        m_nowTimeReactive=0;
        //10秒后关灯
        if(m_nowTime>=ConstSetting.SWITCH_OFF_TIME)
        {
            m_nowTime-=ConstSetting.SWITCH_OFF_TIME;
            m_NextStatus=false;
            CloseLight();
        }
    }
}
```

遇到的问题：

灯光照射时，想添加碰撞器然后触发检测，但是没有锥形的碰撞器，其他碰撞器检测不精准。

解决方案：

1.通过圆形碰撞器加raycast检测怪物是否在灯光照射范围内。

2.找一个锥形网格，然后通过网格碰撞体可以实现锥形检测

第二次版本更新

4.机关

4.1开关：主角操作开关可改变与之相连的灯的状态（关->开，开->关）并重置其计时情况

实现方式：

主角进入开关范围时，检测如果按下E键，则调用SwitchLight切换灯光状态

```
public void SwitchLight(object o)
{
    m_PlayerAnimator=o as Animator;
    if(!m_LightControl.m_NextStatus==true)
    {
        m_PlayerAnimator.SetBool(PlayerAniPara.SWITCH_ON,true);
        StartCoroutine(ReLightOn());
    }
    else if(!m_LightControl.m_NextStatus==false)
    {
        m_PlayerAnimator.SetBool(PlayerAniPara.SWITCH_OFF,true);
        StartCoroutine(ReLightOff());
    }
    m_SwitchSource.PlayOneShot(m_SwitchClip);

 m_Animator.SetBool(SwitchAniPara.SWTICH_STATUS,!m_LightControl.m_NextStatus);
```

```
    m_LightControl.SwitchStatu();
}
```

4.2陷阱：主角移动到陷阱2米范围内将会触发陷阱

4.2.1伤害陷阱：主角触发陷阱时掉1点血，持续接触时每秒掉1点血

实现方式：

主角进入陷阱的触发器范围时，就会触发Damage造成伤害，然后积累时间持续掉血

```
private void OnTriggerEnter(Collider other)
{
    if(other.gameObject.tag==ObjectTag.PLAYER)
    {
        m_animator.SetBool(SpikeAniPara.IS_TRIGGER,true);
        m_DamageSource.PlayOneShot(m_DamageClip);
        m_damage[0]=ConstSetting.DAMAGE_UNIT;
        other.gameObject.GetComponent<PlayerLogic>().Damage(m_damage);
    }
}
private void OnTriggerStay(Collider other)
{
    if(other.gameObject.tag==ObjectTag.PLAYER)
    {
        DamageTime+=Time.deltaTime;
        if(DamageTime>=ConstSetting.Damage_SPEED)
        {
            DamageTime-=ConstSetting.Damage_SPEED;
            other.gameObject.GetComponent<PlayerLogic>().Damage(m_damage);
        }
    }
}
```

4.2.2减速陷阱：在陷阱范围内主角减少50%的移动速度；减速效果将在主角离开减速陷阱范围后结束，恢复正常移动速度

实现方式：

在进入和离开触发器时调整移动的速度倍率

```
private void OnTriggerEnter(Collider other)
{
    if(other.gameObject.tag==ObjectTag.PLAYER)
    {
        //减速
        other.gameObject.GetComponent<PlayerMovement>
().ChangeMoveRatio(ConstSetting.SLOW_SPEED_RATIO);
    }
}
private void OnTriggerExit(Collider other)
{
    if(other.gameObject.tag==ObjectTag.PLAYER)
```

```
    {
        //离开后加速
        other.gameObject.GetComponent<PlayerMovement>
().ChangeMoveRatio(ConstSetting.NORMAL_SPEED_RATIO);
    }
}
```

5.场景物件和角色道具

5.1宝箱：打开宝箱后主角将获得武器。未开启的宝箱是障碍物；已开启的宝箱不是障碍物。

实现方式：

当角色进入宝箱范围时并按下按键宝箱就会播放动画并修改碰撞属性为触发器。

```
public void SetTrigger()
{
    animator.SetBool(WoodBoxAniPara.IS_OPEN,true);
    GetComponent<BoxCollider>().isTrigger=true;
}
```

6.射击

6.1在拥有武器后，玩家点击鼠标右键，主角将会抬手向鼠标所在方向发射子弹

6.2子弹会沿直线飞行，在碰到障碍物、怪物、墙壁时消失

6.3射击应有1秒的冷却时间

实现方式：

1.用一个m_Open记录时间，当玩家点击右键时，判断m_Open是否准备就绪来控制射击的冷却时间，并调用Shoot()

2.通过ScreenPointToRay从屏幕到mouseposition进行射线检测，获取鼠标点击的位置

```
Vector3 Point=Vector3.zero;
Ray mousePos = Camera.main.ScreenPointToRay(Input.mousePosition);
RaycastHit hit02;//检测碰到的点
if(Physics.Raycast(mousePos, out
hit02,ConstSetting.MAX_RAYCAST_DISTANCE,LayerMask.GetMask(ConstSetting.LAYER_MAS
K_DEFAULT)))//函数是对射线碰撞的检测
{
    Point = hit02.point;//得到碰撞点的坐标
}
```

3.生成子弹，当子弹碰撞到物体时销毁

```
GameObject
go=Instantiate(m_Bullet,m_OpenFirePosition.position,Quaternion.identity);
go.GetComponent<Rigidbody>().velocity=(Point-
m_OpenFirePosition.transform.position)*ConstSetting.SHOOT_SPEED;
```

4.播放音效和粒子,通知冷却时间UI更新动画

```
public void UpdateUI(float m_OpenTime)
{
    m_Image.fillAmount=m_OpenTime/ConstSetting.SHOOT_SPEED;
}
```

7.游戏提示

7.1 游戏开始时，显示提示文字，5秒后关闭

实现方式：

添加TextUI设置文字，记录时间，达到5秒后销毁

```
private void Update() {
    m_NowTime+=Time.deltaTime;
    m_Text.color=Color32.Lerp(bottomColor, topColor,
m_NowTime/ConstSetting.UI_FADE_TIME);
    if(m_NowTime>=ConstSetting.UI_FADE_TIME)
    {
        Destroy(gameObject);
    }
}
```