《游戏开发》
期末课题实验报告

姓　　名：　刘婧

班　　级：　2018211601

学　　号：　2018211744

指导教师：　李学明

北京邮电大学数字媒体与设计艺术学院

2021 年 **6** 月 **27** 日

目录

一、场景搭建

1. 场景描述

这里涉及到的场景为两个，一个是 NPC 场景，用于游戏先导动画的制作和游戏剧情的引出，另一个是游戏的第二个关卡洞穴场景，包含了角色、敌人、陷阱和机关及一些装饰元素。

2. 场景资源

利用 TileMap 对场景进行了搭建，这里搭建了最开始先导动画的场景和洞穴关卡的场景。因为场景为 2D 像素风格，我们利用了对多个精灵图进行切片拼接，以此合成一个场景。

使用到的资源包为 Gothicvania-Town(NPC 场景)、Pixel Fantasy Caves（洞穴关卡），皆为 Unity Asset Store 免费资源。以下为资源最初精灵图的展示：

图 1-1 洞穴关卡精灵图资源包

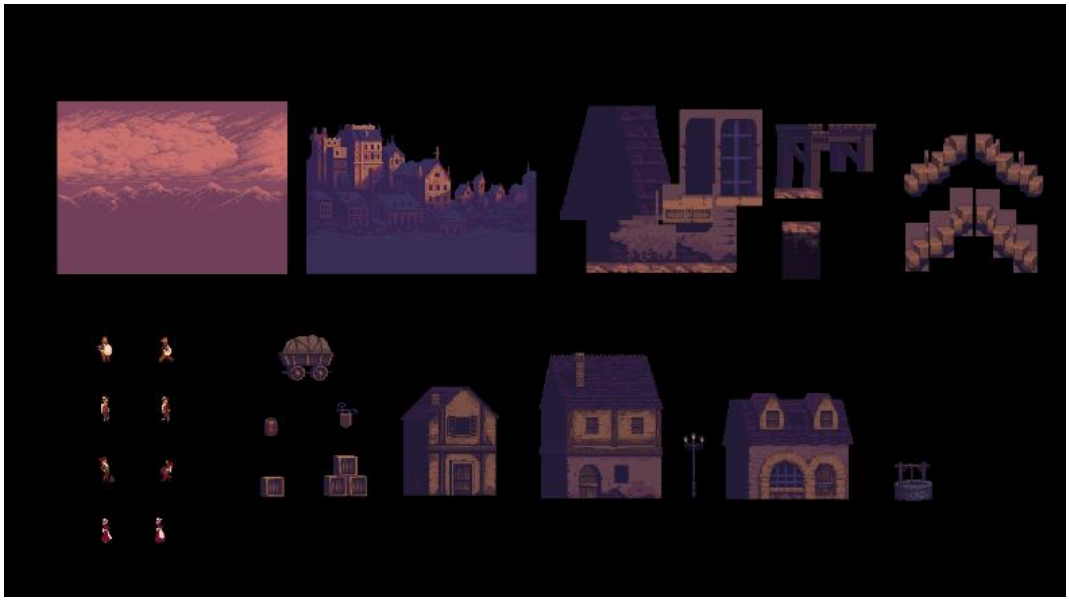

图 1-2 NPC 场景精灵图资源包

3. 场景制作

场景制作中，使用到的工具为 Tilemap、Paletee、Tile，分别为地图、调色盘和瓦片。

(1) 洞穴场景

洞穴场景占了 1920*1080 屏幕大小的六屏左右，如下图所示，每屏中 Background 分别由三个背景图片构成，第四、五屏为三个背景图片构成。

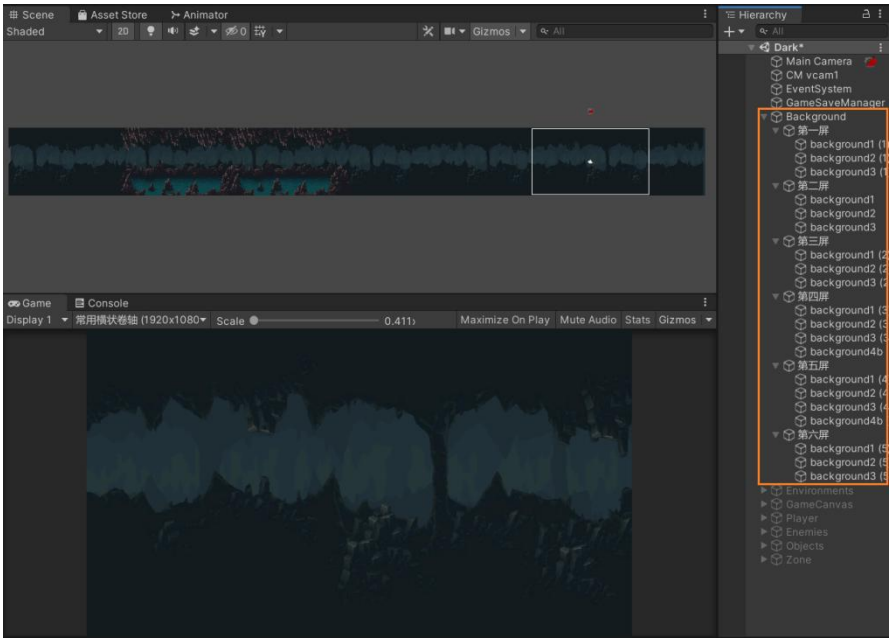

图 1-3 场景背景

设置了 7 个调色盘来对场景进行绘制。

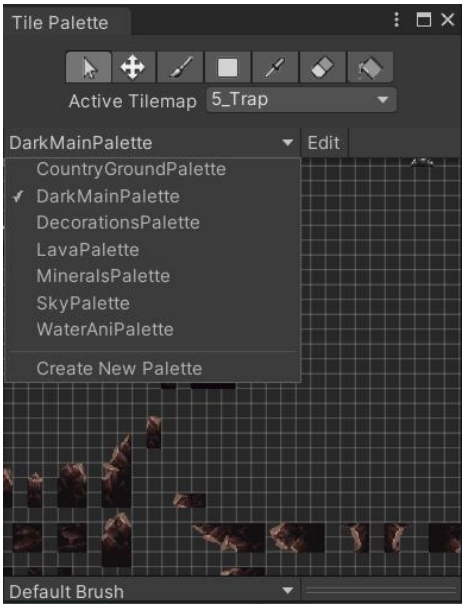

图 1-4 调色盘

以下为绘制的场景，依据不同功能，分为了七层。其中 2_DarkGround 为整个场景

的地面，3_Platform 和 6_Fly 为移动平台，5_Trap 为陷阱，其余层皆为装饰层。

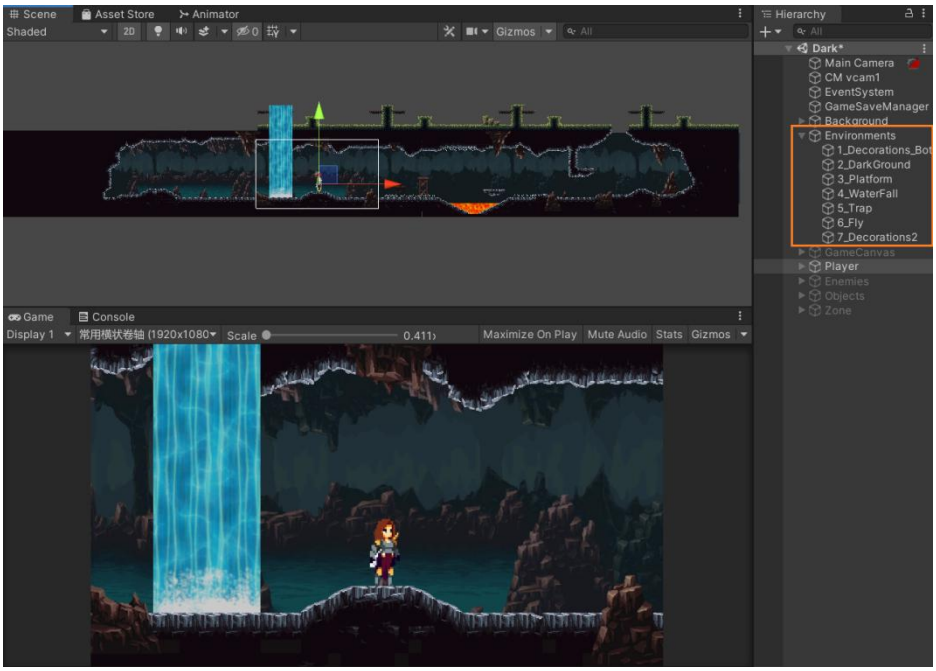

图 1-5 场景搭建

在装饰层，利用了 AnimationTile 来制作了动画，例如始终保持流动的瀑布。共分为两段，一段为流水，一段为浪花，每个瓦片动画包含了 5 张图片，通过控制动画播放速度，来使其具有动画的效果。

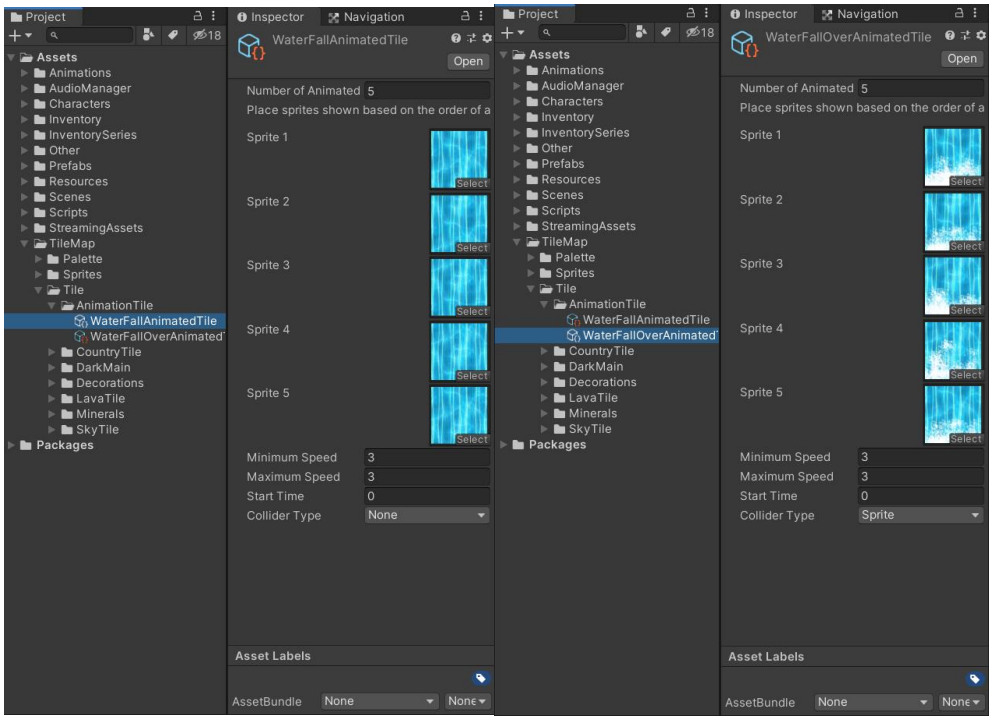

图 1-6 瀑布瓦片动画

加入角色和敌人后，部分场景大致如下：

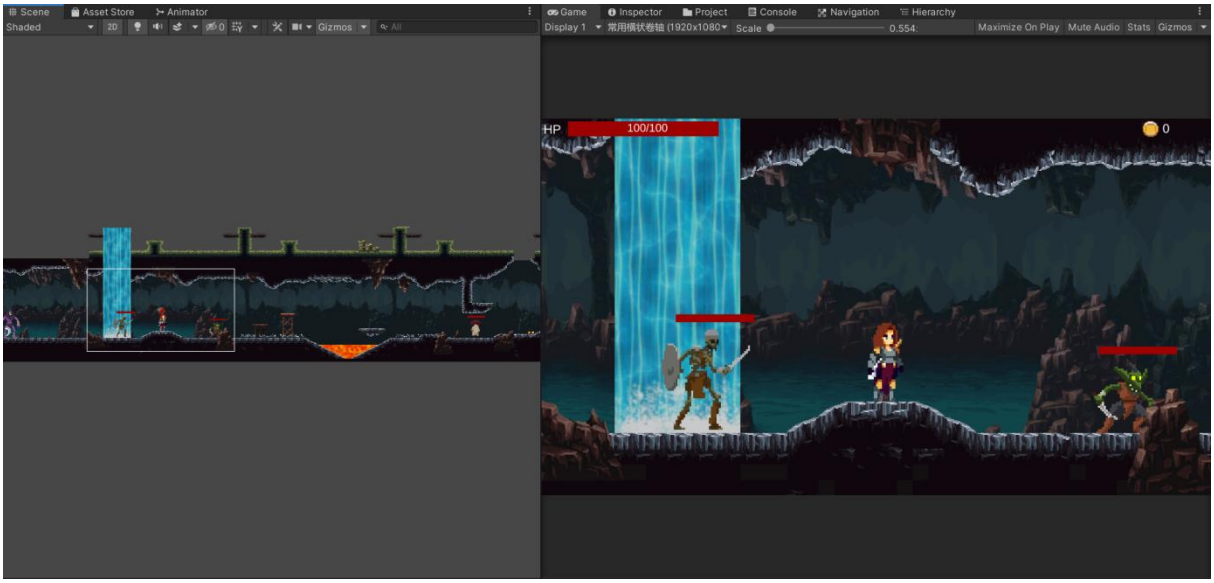

图 1-7 洞穴场景展示

(2) NPC 场景

NPC 场景共三屏。背景分为两层，场景层为走路平台，装饰层为房屋、路灯、马车等等。NPC 包含了四个角色。

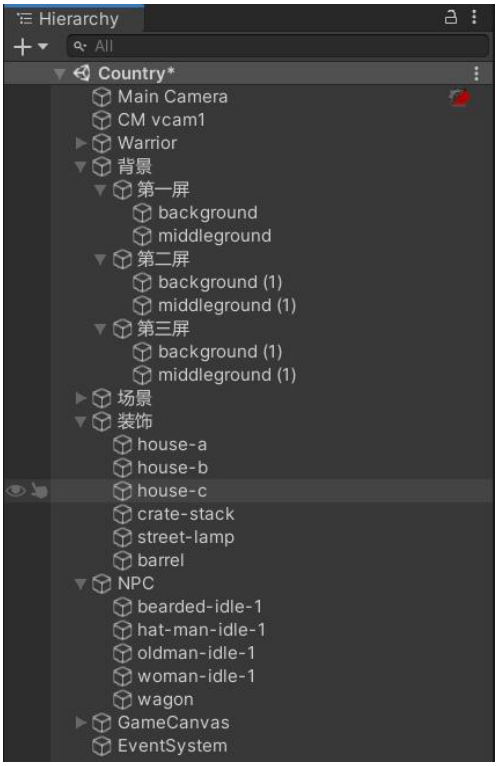

图 1-8 场景构成

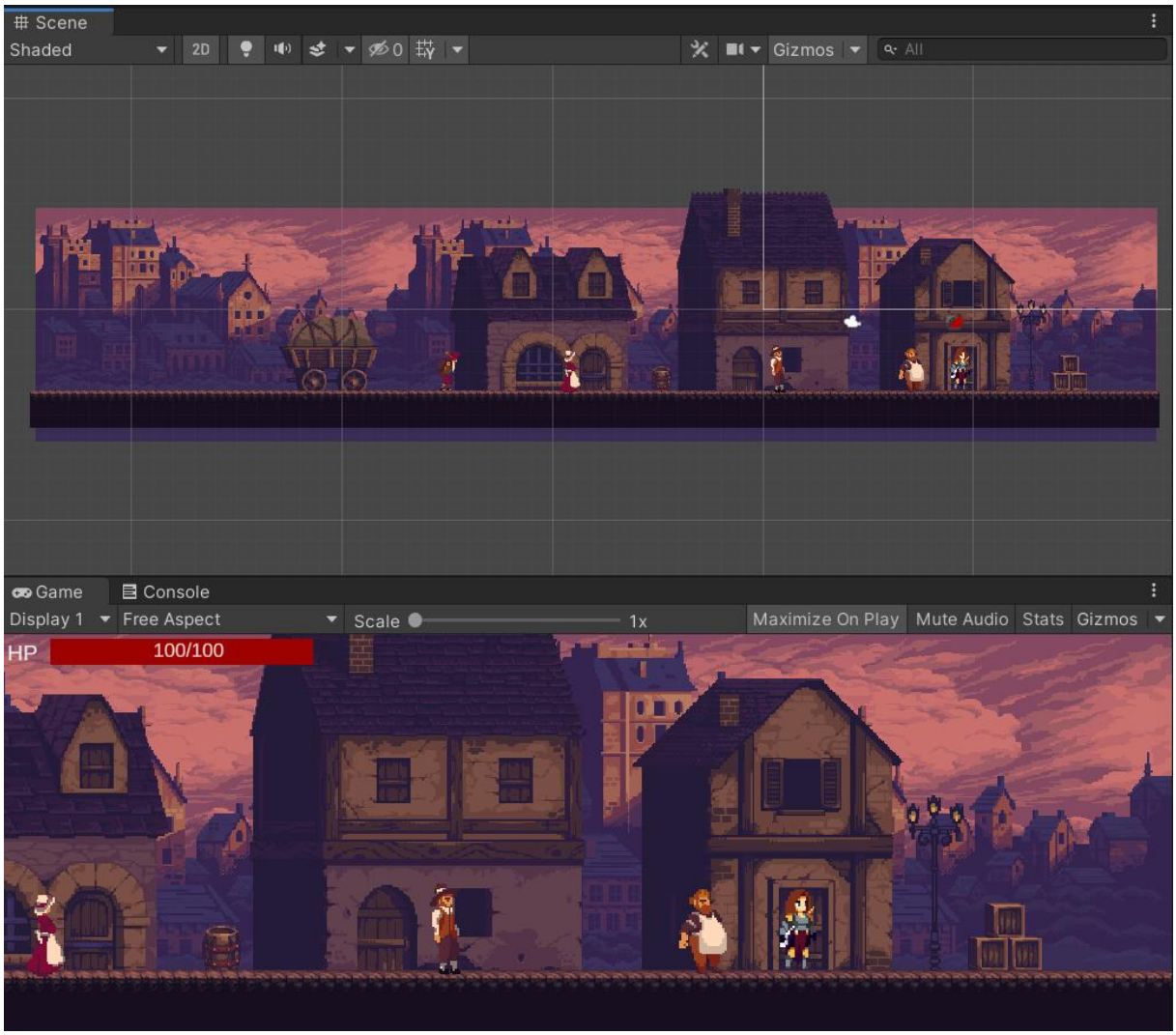

图 1-9 NPC 场景展示

二、相机跟踪

相机跟踪上，我们使用了 CineMachine 插件，以这种简便的方式来制作相机跟踪效果。

首先在 2D 场景中通过 Package Manager 导入 CineMachine 插件，新建一个 2D Camera。

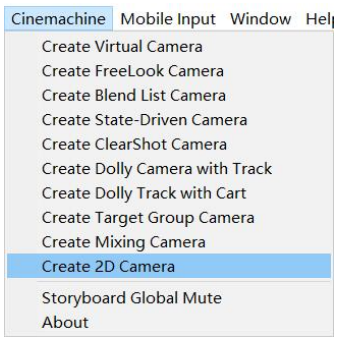

图 2-1 2D Camera

对跟踪对象、相机画面大小进行调节。并根据场景需求对相机进行设置，调整红色区域、蓝色区域、透明区域。当角色处于透明区域时，相机不移动；处于蓝色区域时，相机跟随角色缓慢移动；处于红色时，相机跟随角色瞬间进行移动。

因为我们游戏的场景需求仅仅要求角色所处场景在 X 轴方向上有变化，Y 轴方向固定，所以只设定了透明区域和蓝色区域，保证了用户在游戏时，不会由于瞬移而感到不适。

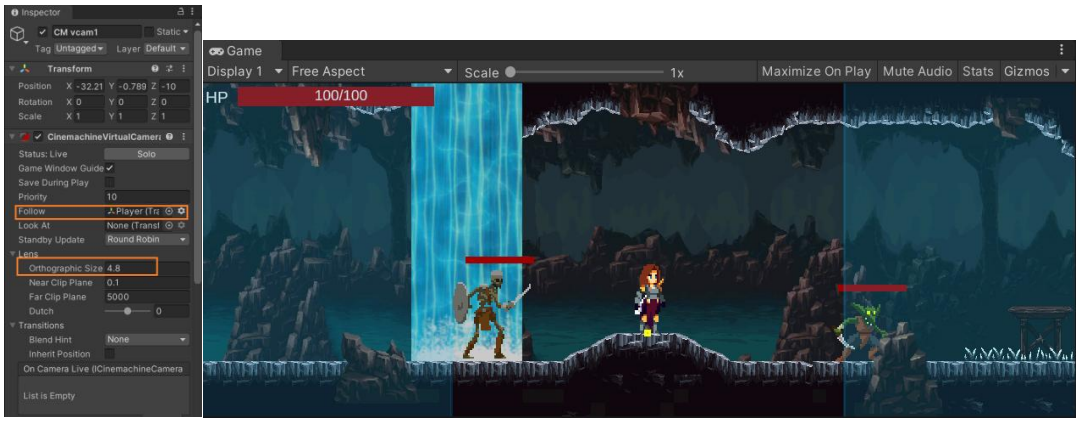

图 2-2 相机跟踪设置

针对区域，我们在所需要的区域设置了一个 Polygon Collider，当角色运动时，画面只能位于这个 Polygon Collider 中，以此限制了游戏画面区域，避免了在角色移动时显示不需要的场景区域。

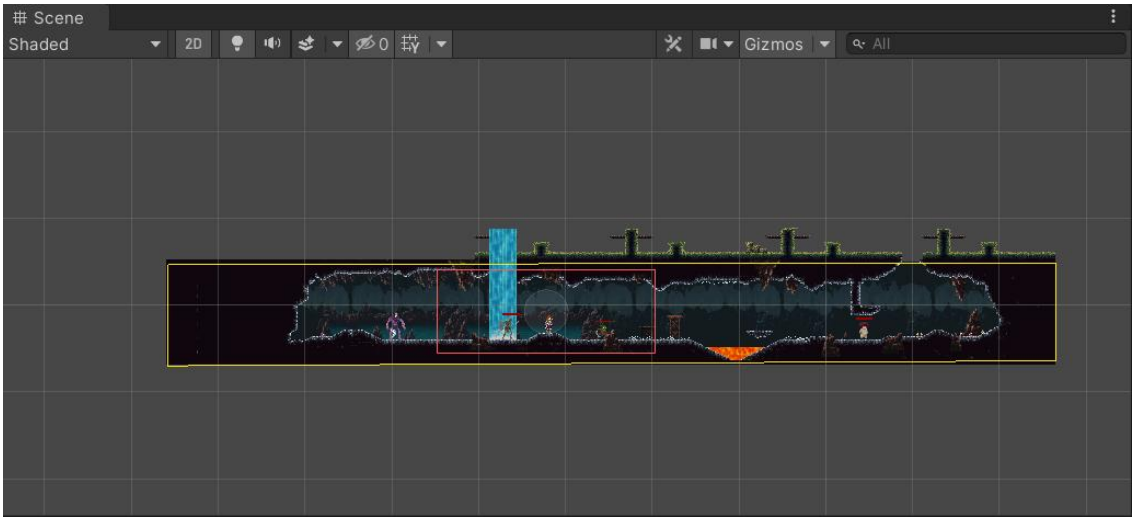

图 2-3 限制游戏画面区域

三、血条及缓冲效果制作

战斗系统中，制作了角色及怪物的血条，并为血条添加了缓冲效果。

1. 血条分层

基础血条共分为三层，Hp、HpEffect 及 HpBackground。如下图所示：

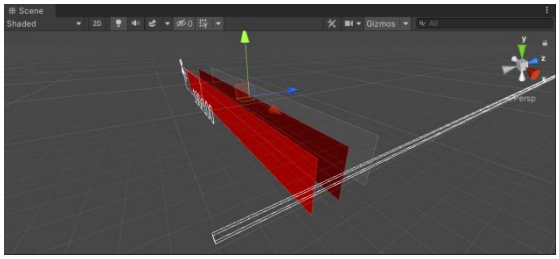

图 3-1 血条分层

这里前方的红色层为血条的颜色，中间暗红层位缓冲效果，最后灰色层为血条背景。

2. 血条属性设置

在设置中，将血条图片更改为 Filled，将角色和怪物的血量进行展示和变化。

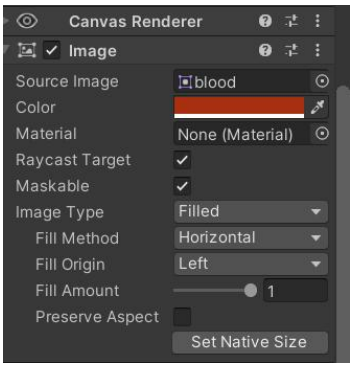

图 3-2 血条设置

3. 血条脚本制作

(1) 角色血条

角色血条新增了一层 HpNumber。通过增加 HurtSpeed 来制作掉血缓冲效果。

```
void Update()
{
    hpNumber.text = player.currentHp.ToString() + "/100";
    hpImage.fillAmount = player.currentHp / player.maxHp;
    if (hpEffectImage.fillAmount > hpImage.fillAmount)
    {
        hpEffectImage.fillAmount -= hurtSpeed;
    }
    else
    {
        hpEffectImage.fillAmount = hpImage.fillAmount;
    }
}
```

(2) 怪物血条

与角色血条相似，但置于怪物子集下，置于怪物头顶，并需要在怪物转身时同

时改变血条方向，放置血条方向反向。

图 3-3 血条缓冲效果展示

4. 战斗系统中血量控制

在攻击普通怪物和 Boss，在怪物的普通攻击、重击和 Player 的普通攻击和重击下，对怪物和 Player 的伤害值、受伤害值进行了设定，以下为部分示例：

```
//普攻下 Boss 血量减少
boss.GetComponent<EnemyHealthBar>().hp -= 6f;
boss.GetComponent<BossController>().Health =
boss.GetComponent<EnemyHealthBar>().hp;
if (boss.GetComponent<EnemyHealthBar>().hp <= 0)
{
    boss.GetComponent<BossController>().Health = 0;
}

//重击下怪兽血量减少
enemyController.GetComponentInChildren<EnemyHealthBar>().hp -=
16f;
if (enemyController.GetComponentInChildren<EnemyHealthBar>().hp
<= 0)
{
    enemyController.Health = 0;
}
```

通过获取怪物和 Player 的 Hp 来控制战斗系统中血量的变化。

四、背包系统

1. 背包系统 UI

游戏增加背包系统，玩家能够收集药水和草药，并置于背包中。背包 UI 如下：

图 4-1 背包 UI

2. 背包系统脚本

在背包这一系统中，我们使用到了 ScriptableObject。用来保存游戏中角色得到物体的一些数据。

(1) 创建背包

```
//背包系统为列表
[CreateAssetMenu(fileName ="New Inventory",menuName ="Inventory/New Inventory")]
public class Inventory : ScriptableObject
{
    public List<Item> itemlist = new List<Item>();
}
```

(2) 创建背包格（物体）

```
[CreateAssetMenu(fileName = "New Item",menuName ="Inventory/New Item")]

//使用 ScriptableObject
public class Item : ScriptableObject
{
    public string itemName;
    public Sprite itemImage;
    public int itemHeld; //物品的数量
    [TextArea] //多行文字
    public string itemInfo;
}
```

(3) 背包管理器

```
//销毁并重新创建
public static void RefreshItem()
{
    for (int i = 0; i < instance.slotGrid.transform.childCount; i++)
    {
```

```
                if (instance.slotGrid.transform.childCount == 0)
                    break;
                Destroy(instance.slotGrid.transform.GetChild(i).gameObject);
                instance.slots.Clear(); //清空列表
            }

            for (int i = 0; i < instance.myBag.itemlist.Count; i++)
            {
                //循环创建回 Item 列表中的所有 Item
                //CreateNewItem(instance.myBag.itemlist[i]);
                instance.slots.Add(Instantiate(instance.emptySlot));       //生成空格
子
                instance.slots[i].transform.SetParent(instance.slotGrid.transform);
//摆放好位置
                instance.slots[i].GetComponent<Slot>().slotID = i;    //获得每个格子
的 ID

instance.slots[i].GetComponent<Slot>().SetupSlot(instance.myBag.itemlist[i]);

            }
        }
```

(4) 背包格子

当点击背包格子时，显示物品对应描述。当没有物品时，禁用此格子，在背包中，格子显示为空。

否则便得到物体信息，显示到背包对应格子中。

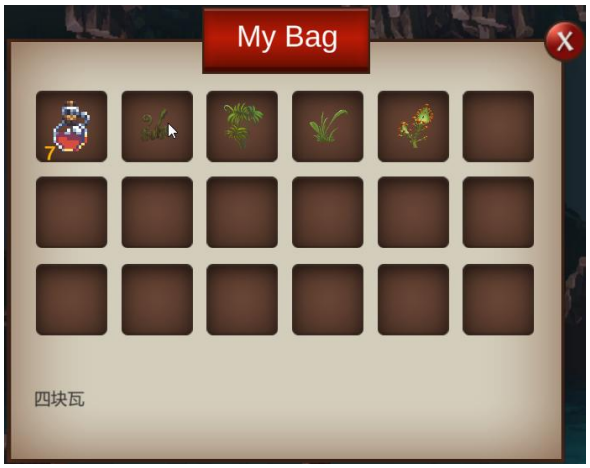

图 4-2 物体描述显示

```
    public void OnPointerClick(PointerEventData eventData)
        {
            if(eventData.button == PointerEventData.InputButton.Right)
            {
```

```
        }
    }


    public void SetupSlot(Item item)
    {
        if(item == null) //没有物品时
        {
            itemInSlot.SetActive(false);
            return;
        }

        slotImage.sprite = item.itemImage;
        slotNum.text = item.itemHeld.ToString();
        slotInfo = item.itemInfo;     //获得物品信息
    }
```

(5) 背包的移动

引入 Unity 事件管理库，获得背包的坐标，使用 OnDrag 事件在原来背包坐标的中心锚点位置增加鼠标移动的值：

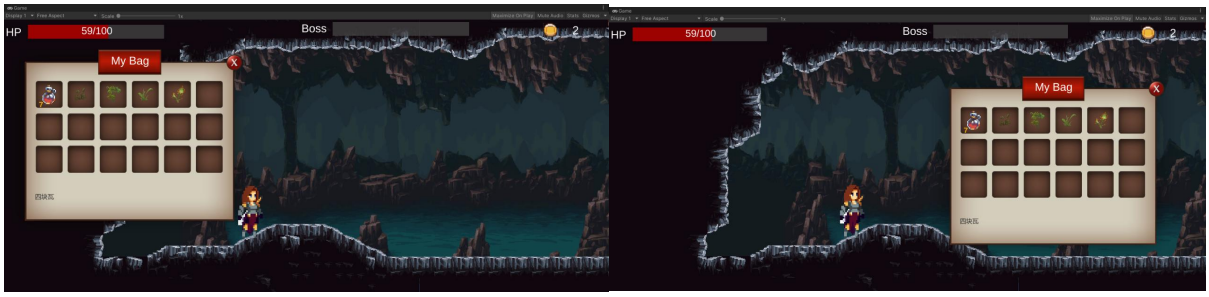

图 4-3 鼠标拖动背包

```
    //获得背包坐标
    RectTransform currentRect;    //当前背包的坐标

    public void OnDrag(PointerEventData eventData)
    {
        //中心锚点的位置加上鼠标移动的值
        currentRect.anchoredPosition += eventData.delta;
    }
```

(6) 背包中物体拖拽与交换位置

当在背包中拖拽某个物体到另外一个位置时，此物体位置与另一个位置的物体或是空格子进行位置的交换时，通过 IBeginDragHandler,IDragHandler,IEndDragHandler 事件来实现。

- 拖拽开始：

图 4-4 物品拖拽开始

```
public void OnBeginDrag(PointerEventData eventData)
{
    originalParent = transform.parent;  //原始父级等于当前父级，置换两个格子的位置
    currentItemID = originalParent.GetComponent<Slot>().slotID; //当前物品的 ID 就是 背包格子对应的 ID
    transform.SetParent(transform.parent.parent);    //更改父级避免拖拽被其他格子挡住
    transform.position = eventData.position;    //获得鼠标的位置
    GetComponent<CanvasGroup>().blocksRaycasts = false
}
```

- 拖拽中：

  此时物品的位置为鼠标所在的位置。

```
public void OnDrag(PointerEventData eventData)
{
    transform.position = eventData.position;

//Debug.Log(eventData.pointerCurrentRaycast.gameObject.name);//鼠标当前射线
}
```

- 拖拽后：

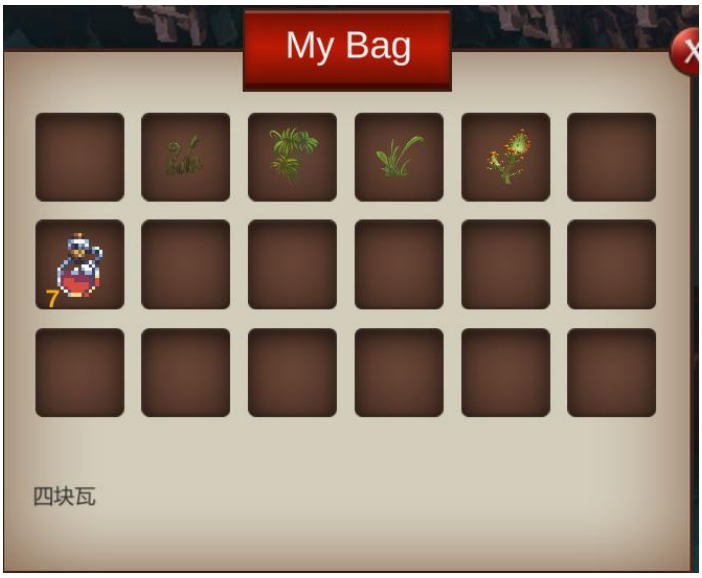

图 4-5 物品拖拽结束

此时物品位置已置于拖拽后位置。

如果拖拽后位置为另一个物品，需要将此时拖拽的物品的 slot 与目标位置物体的 slot 置换，并将双方 slotID 进行对调。在最终完成置换后，需要开启射线阻挡。

```
//鼠标指向的图片是物品
if (eventData.pointerCurrentRaycast.gameObject.name == "Item Image")
{   //得到 slot

transform.SetParent(eventData.pointerCurrentRaycast.gameObject.transform.parent.parent);
transform.position = eventData.pointerCurrentRaycast.gameObject.transform.parent.parent.position;
//itemlist 的物品存储位置改变
var temp = myBag.itemlist[currentItemID];
//实现物品 ID 对调
//当前存储位置的 ID 改变为鼠标点击的格子的 ID
myBag.itemlist[currentItemID] = myBag.itemlist[eventData.pointerCurrentRaycast.gameObject.GetComponentInParent<Slot>().slotID];
//鼠标点击格子的 ID 变为原存储格子的 ID

myBag.itemlist[eventData.pointerCurrentRaycast.gameObject.GetComponentInParent<Slot>().slotID] = temp;


eventData.pointerCurrentRaycast.gameObject.transform.parent.position =
```

```
originalParent.position;

eventData.pointerCurrentRaycast.gameObject.transform.parent.SetParent(origi
nalParent);
                GetComponent<CanvasGroup>().blocksRaycasts = true;  //射线阻
挡开启，不然无法再次选中移动的物品
                return;
 }
```

如果拖拽后指向位置为空，则检测到 Slot 下方，将拖动物体的位置直接设置于鼠标检测到的位置。

在 Itemlist 中，当前鼠标指向的位置的 ID 设置为原物体所在的 ID。当拖动到不是自己的其他格子上时，当前物品原 ID 需要设置为空。

```
if (eventData.pointerCurrentRaycast.gameObject.name == "slot(Clone)")
            {
                //如果鼠标指向的位置是空的，直接检测到 Slot 下面

transform.SetParent(eventData.pointerCurrentRaycast.gameObject.transform)
;
                transform.position =
eventData.pointerCurrentRaycast.gameObject.transform.position;
                //itemlist 物品存储位置改变

myBag.itemlist[eventData.pointerCurrentRaycast.gameObject.GetComponentIn
Parent<Slot>().slotID] = myBag.itemlist[currentItemID];
                //当拖动到不是自己的其他空格子上时
                if
(eventData.pointerCurrentRaycast.gameObject.GetComponent<Slot>().slotID !
= currentItemID)
                    myBag.itemlist[currentItemID] = null;

                GetComponent<CanvasGroup>().blocksRaycasts = true;
                return;
    }
```

当物体拖拽到其他非 slot 位置时，归回原位：

```
transform.SetParent(originalParent);
transform.position = originalParent.position;
GetComponent<CanvasGroup>().blocksRaycasts = true;
```

(7) 世界物体装入背包

playerInventory 为角色所拥有的背包，thisItem 为物品属性所属的数据库，判断背包中是否有此物体。如果没有，则加入背包，如果有，便进行叠加。持有数目皆加 1。

```
public void AddNewItem()
```

```
        {
            //如果背包中没有物体,背包内加入这个物体
            if(!playerInventory.itemlist.Contains(thisItem))
            {
                for (int i = 0; i < playerInventory.itemlist.Count; i++)
                {
                    if(playerInventory.itemlist[i] == null)
                    {
                        playerInventory.itemlist[i] = thisItem;
                        thisItem.itemHeld += 1;
                        break;
                    }
                }
            }
            else
            {//背包中有该物体，则将该物体数目增加 1
                thisItem.itemHeld += 1;
            }

            InventoryManager.RefreshItem();
        }
```

(8) 使用背包中物体

在背包中，生命药水持有量大于 1，且血量低于 100 时，按下快捷键 1，生命值增加，相应背包中生命药水数量减少。

```
void UseLifePotion()
{
    //回血药水
    if (playerInventory.itemlist.Contains(thisItem))  //角色背包里存在生命药水
    {
        if (thisItem.itemHeld > 0)  //持有量大于 0
        {
            if (currentHp < 100) //此刻血量小于 100
            {
                if (Input.GetKeyDown(KeyCode.Alpha1))   //按下 1 回血
                {
                    thisItem.itemHeld -= 1; //持有量减 1
                    currentHp += 10.0f; //当前血量+10
                }
            }
        }
        else   //持有量如果小于等于 0
        {
```

```
                    for (int i = 0; i < playerInventory.itemlist.Count; i++)
//遍历背包列表
                    {
                        if (playerInventory.itemlist[i] == thisItem)    //将
生命药水从列表中移除
                        {
                            playerInventory.itemlist[i] = null;
                        }
                    }
                }
            }

    }
```

(9) 单个物体显示

当收集到单个物体时，设置其不显示左下角数字，除非持有量大于 1，才显示数量。

```
if(item.itemHeld == 1)
{
    slotNum.text = null;
}
else
{
    slotNum.text = item.itemHeld.ToString();
}
```

图 4-6 单个物品显示

五、物品收集

1. 物品收集描述

(1) 在游戏物品方面涉及到了金币、生命药水和药草。

(2) 金币显示在右上角，当人物碰撞金币时，右上角金币数目实时改变。
(3) 当角色撞击到生命药水时，将生命药水存储到背包中。
(4) 药草为任务道具，收集后再背包中显示。

2. 物品收集

(1) 金币

为金币设置了旋转动画，在收集金币前，金币能够在原位置一直旋转。

图 5-1 金币收集

```
if(other.gameObject.tag == "Coin")
{
    //other.transform.GetComponent<Animator>().SetTrigger("coinGet");
    Destroy(other.gameObject);
    coinNum += 1;
    coinNumText.text = coinNum.ToString();

}
```

(2) 药水

图 5-2 药水收集

```
if (other.gameObject.tag == "LifePotion")
{
        other.GetComponent<ItemOnWorld>().AddNewItem();
        Destroy(other.gameObject);
}
```

(3) 草药

因为收集的草药只能出现一次，且收集只能收集到一个。故重写草药收集判断的脚本，并挂载到草药上。

- 草药收集

在草药子集中加入 pickUpText，当 Player 靠近草药时，出现收集提示文本。

图 5-3 草药收集

```
private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.gameObject.name.Equals("Player"))
        {
            pickUpText.gameObject.SetActive(true);
            pickUpAllowed = true;
        }
    }

    private void OnTriggerExit2D(Collider2D other)
    {
        if (other.gameObject.Equals("Player"))
        {
            pickUpText.gameObject.SetActive(false);
            pickUpAllowed = false;
        }
```

当出现收集提示后，按下 E 键进行收集：

```
    void Update()
    {
        if (pickUpAllowed && Input.GetKeyDown(KeyCode.E))
            PickUp();
    }

    private void PickUp()
    {
        gameObject.GetComponent<ItemOnWorld>().AddNewItem();
        Destroy(gameObject);
    }
```

- 检测草药持有量

当收集到该种草药后，后续关卡场景中去除该种草药。

```
void ExamHerbs()
{
    if(playerInventory.itemlist.Contains(Item_herb1))
    {

        Destroy(herb1);
    }
    if (playerInventory.itemlist.Contains(Item_herb2))
    {

        Destroy(herb2);
    }
    if (playerInventory.itemlist.Contains(Item_herb3))
    {

        Destroy(herb3);
    }
    if (playerInventory.itemlist.Contains(Item_herb4))
    {

        Destroy(herb4);
    }
}
```

六、关卡机关

洞穴关卡涉及到刺刀、岩浆、移动平台三个机关。当 Player 走进刺刀区域时，持续损失生命值；掉入岩浆时，判定死亡，回到关卡起始点；移动平台保持移动，角色踏上平台后，能被承载着进行一段距离的移动。

- 刺刀

设置了单独的刺刀层，为其添加 Collider 和刚体，角色在进入或在刺刀上时，持续丢失生命值。

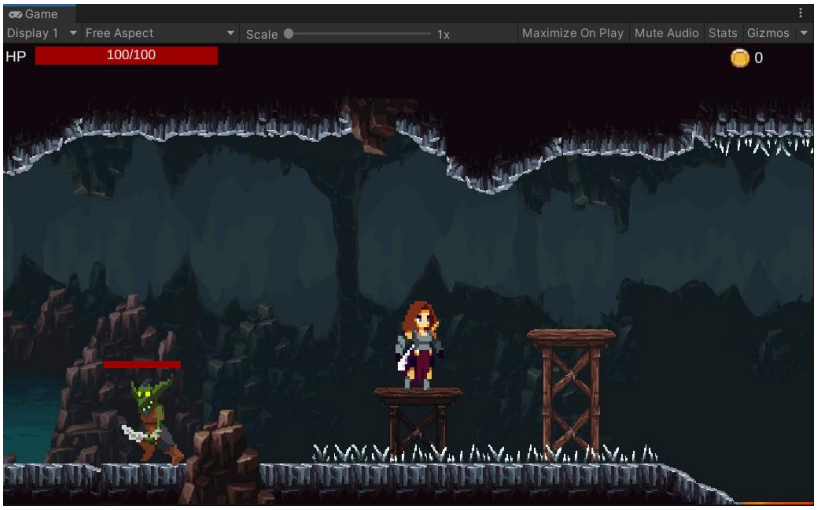

图 6-1 刺刀机关

```
private void OnTriggerStay2D(Collider2D other)
        {
            //刺刀
            if (other.tag == "Traps")
            {
                player.GetComponent<PlayerScript>().currentHp -= 1f;
        }
}
```

- 岩浆

为岩浆设置 Collider，判定在碰撞到 Collider 时，马上重启关卡。

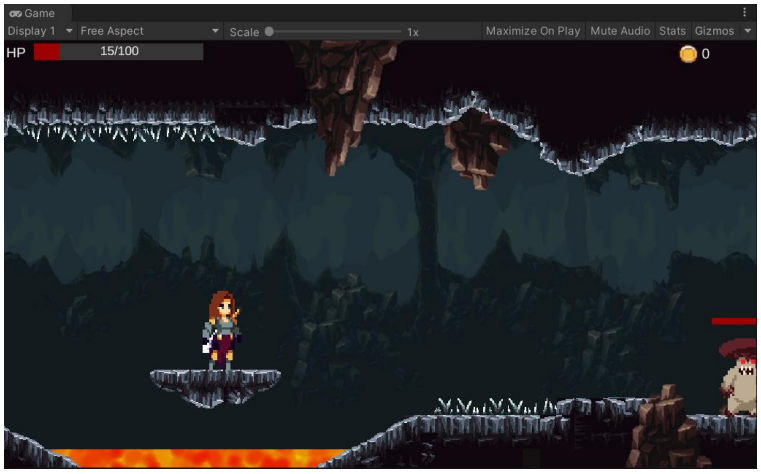

图 6-2 岩浆

```
    //进入死亡区域
    if(other.tag == "KillZone")
    {
```

SceneManager.LoadScene(SceneManager.GetSceneAt(0).name);

七、未来改进点

时间限制，项目效果还未达到预期。针对于个人部分，就目前功能而言，可有以下改进点与个人部分所存在的一些问题：

1. 相机跟踪效果改进

在进入洞穴关卡时，角色由洞穴顶端下落，目前相机的设置是固定了 Y 轴，在下落时并不跟随角色移动。尝试更改相机跟随角色在 Y 轴移动（也就是在 Y 轴设置了蓝色区域，相机随角色缓慢移动），这种设置下，虽然相机能够跟随角色在下落时移动，但下落后，相机视角太过于靠下。

后期会继续学习 Cinemachine 的使用，更改目前项目不合理的一些配置。

2. 血条缓冲效果改进

已实现角色、小怪、Boss 的血条缓冲效果，但忽略了加血效果。这个会在后面加上。

3. 血量控制改进

在写血量的时候，思考的太少，在测试过程中，对于角色对怪物进行打击以及后续关卡的进行有一些影响，比如说因为血量减少的不合理使得游戏太难无法通关又或是太简单而太容易通关。

4. 背包系统改进

初次接触 ScriptableObject，接触时间太短，还未来得及熟练。但也体会到了 ScriptableObject 储存游戏某些数据的便利之处。

在背包系统这一块上，出现了很多问题，比如说金钱的数据未一起同步，然后同一个按钮控制不同物体的丢弃和使用还未实现，目前是简单用按键来控制生命药水的使用。

后续我们还能把不同的物体分成不同的类别、描述等等。

根据背包系统，我们还能产生多个背包，也就是商店系统的实现，后续会继续跟进！

5. 物体收集改进

这里存在需要改进的点是物体的重力问题，以及增加一些物体收集的特效动画。比如：收集金币后，金币飞到金币收集数处，能够增加视觉体验效果。