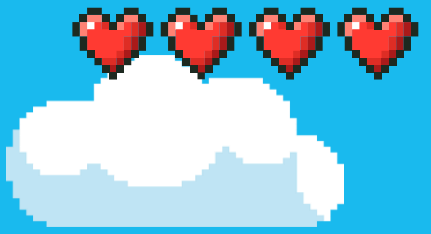
游戏设计与设计思维 课程汇报

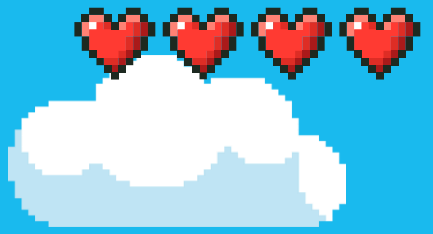
汇报人：黄勖
（混合组）张璐冰、黄子安、曹志逸
黄勖、应驰骅、吕芸鹤



目录

CONTENTS


×15
×06
游戏简介
GAME OVERVIEW



背景概述
BACKGROUND

目的

本游戏从童年的通话入手，旨在帮助游戏玩家回忆童年的快乐，也能在繁重的学业或工作中享受真正的自己

立意

从童话本身的意义出发，我们希望玩家在这个奇幻的世界中找到真正的自我，能更勇敢的面对现实世界的挑战。

工程

游戏的技术部分我们使用unity的2D模块进行实现，搭建了多个场景构建一个完整的游戏。

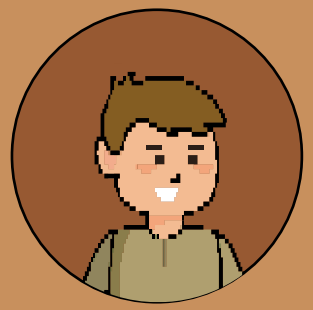
核心

我们的核心技术是一个镜像shader，可以反射水面上的内容，另外怪物的刷新和出现我们利用了退火算法，使游戏更加合理。

梦游仙境

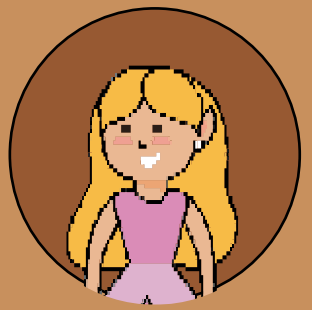
用户画像

USER PORTRAIT



12~22岁的年轻人

喜好挑战



对童话兴趣浓厚


×15
×06
设定概述
SETTING OVERVIEW

道具设定汇总：

我们的几
境产出外，剩
率，以此来增
12个金币作为

道具名
神秘兔子的
喝一杯就会变
吃了变成巨人
巨大蘑
兔子先生丢下胡萝卜
蜈蚣的炮

角色设定汇总：

角色
爱丽丝
神化爱
红桃卫
黑桃卫
胡桃夹
剑鱼
长毛怪

商店设定：

由于在这个世界，我们将会遇到无数的挑战与危险，而爱丽丝将会在冒险中不同程度的受到伤害，所以我们在所有的BOSS场景前提供神秘商人服务。玩家可以使用本剧掉落的金币进行道具的兑换，以此来回复血量或收取相应收益。具体设定如下：

道具名	作用	价格
体力面包	回复25%生命值	3金币
香喷喷的烤鸡	回复50%生命值	5金币
塑金魔杖	下一场景回蓝速度提升20%	6金币
无情拳套	下一场景攻击伤害提高20%	6金币
乌龟壳	可以抵挡2次伤害，消耗品	10金币
随机刷新BOSS掉落物品	BOSS的掉落物，都有极其强大的作用。	25金币

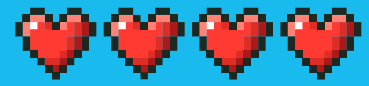
图 x10

敌人 x08

神奇的游戏道具 x 14

强大的玩家技能 x 02

有趣的游戏成就 x17



功能&特点

FUNCTION

机制

镜像冒险机制，符合年轻用户喜爱探索的特点。

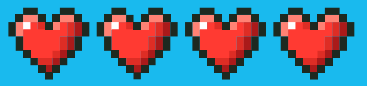
原创

故事的剧情，设定，素材与音乐均为我们组内成员自主创作

多样

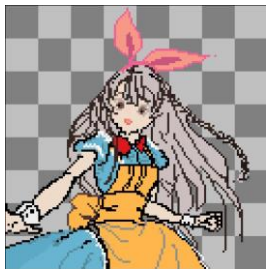
游戏中的角色与成就多样化，并融入了当下的梗元素，期待能更加吸引年轻玩家的注意。

 ×15  ×06



游戏的美工是我们的游戏设计组自行绘画，采用板绘等方式进行实现，具体呈现如下，希望能给玩家带来更好的游戏体验。













美术设计

ART

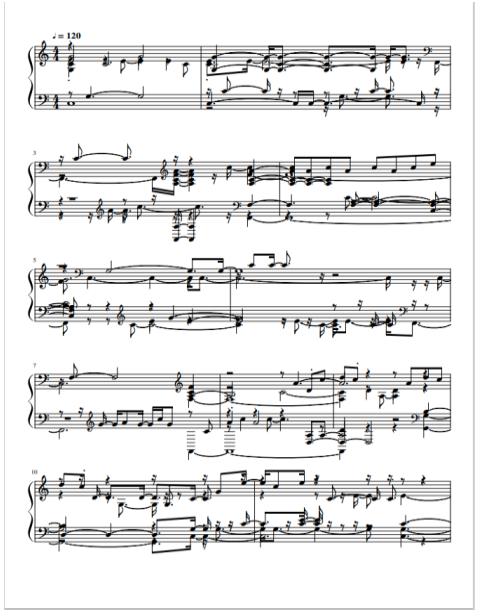
×15 ×06

音乐设计

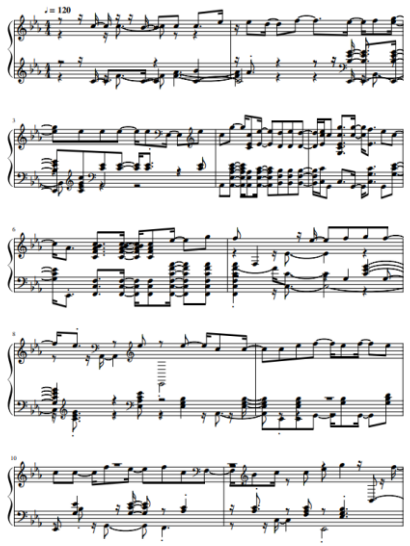
MUSIC

我们的游戏背景音乐是由组内成员使用AIGC技术完成，以及进一步remix加入混响和自然音，或舒缓或激昂，配合不同的场景进行播放，旨在为玩家提供更加沉浸式的体验。

开局音乐《幻境序曲》



战斗音乐《龙骑之战》




×15
×06
Boss概述
BOSS
多样化的设计
总计4只BOSS每只都拥有自己独特的技能，甚至有独属于自己的特性，在击败他们以后更是有机会获得与他们能力里相关的道具
与道具的配合
在挑战BOSS时使用道具或许会有出人意料的收获，也许会有出乎意料的友军，也许能够直接溜走，增强游戏的自由性。


×15
×06
游戏设计
GAME DESIGN

×15 ×06

游戏中，整体的场景设计将融合童话、幻想和神秘元素，打造一个充满奇幻和冒险的世界。

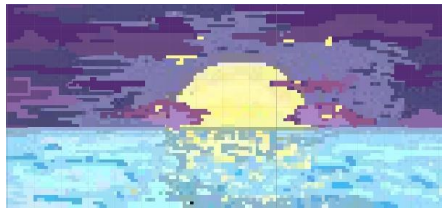
多样场景 x10

初始场景—树洞

第一场景—木屋

第二场景—蘑菇林

第三场景—月海



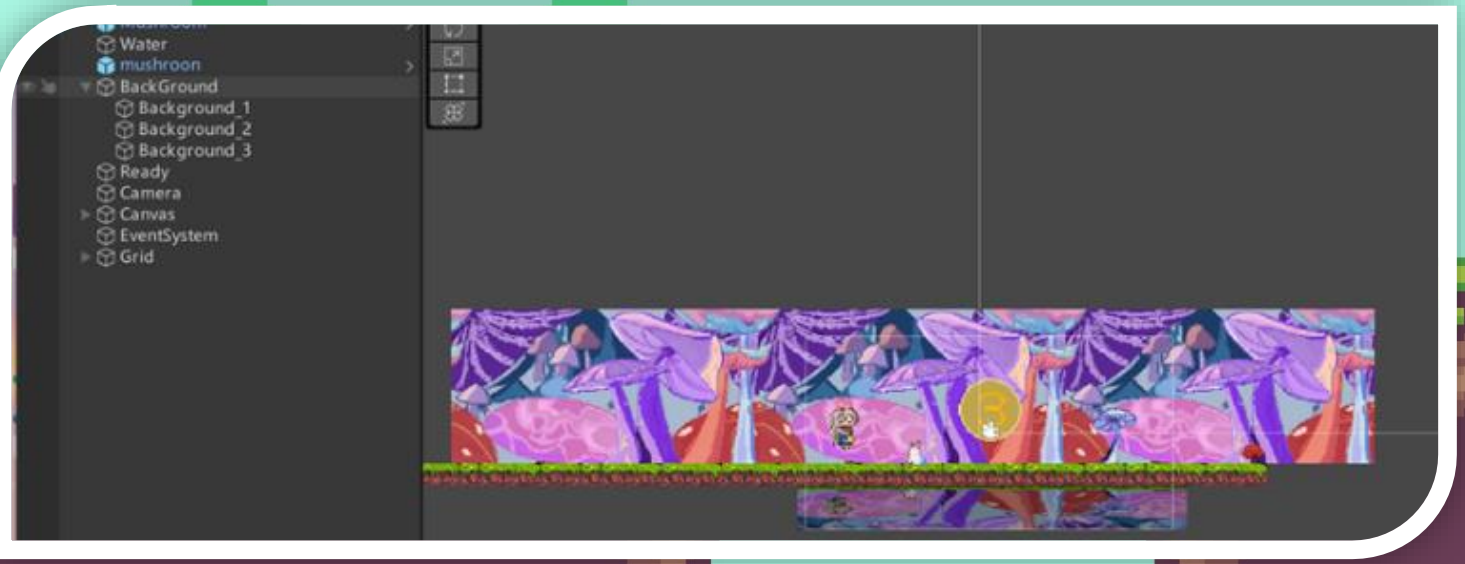
第五场景—暗林

第七场景—皇宫

01 背景滚动脚本

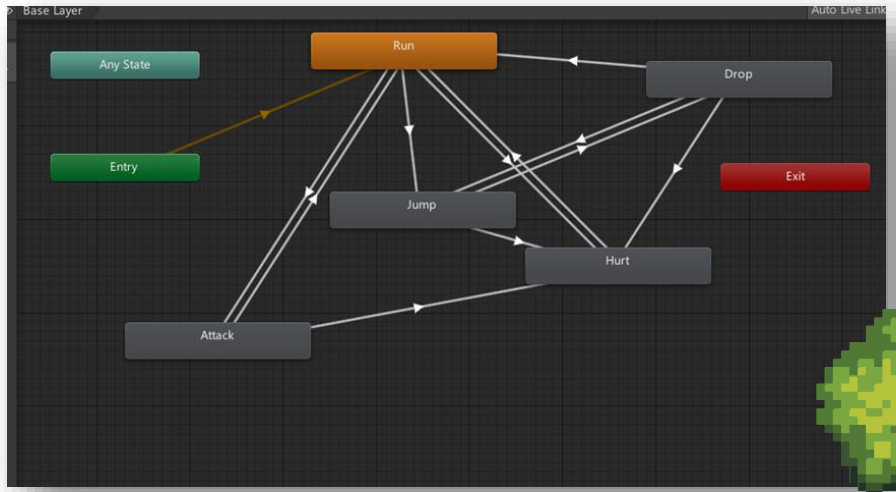
作为一款动作类游戏，我们自然需要Alice能往前移动，对应的为了实现背景的高效利用，我们在绘制背景的过程中使得背景可以在左右无缝拼接起来，之后运行时只需要使用代码让靠左的背景在合适的时机移动到最右边就可以实现背景的无限滚动

```
public class BackgroundMove : MonoBehaviour
{
    private GameObject[] background=new GameObject[3];
    public float Speed=0.05f;
    private float ystart,zstart;
    float lasttime;
    void Start()
    {
        lasttime = Time.time;
        for (int i=0;i<3;++i)
        {
            background[i] = GameObject.Find("Background_"+(i+1));
        }
        ystart=background[0].transform.position.y;
        zstart=background[0].transform.position.z;
    }

    void Update()
    {
        if (Time.time - lasttime >= 3)
        {
            for (int i = 0; i < 3; ++i)
            {
                if (background[i].transform.position.x < -18.5f)
                {
                    background[i].transform.position = new Vector3(27.0f, ystart, zstart);
                }
                else
                {
                    background[i].transform.position -= new Vector3(Speed, 0, 0);
                }
            }
        }
    }
}
```



角色动画 02

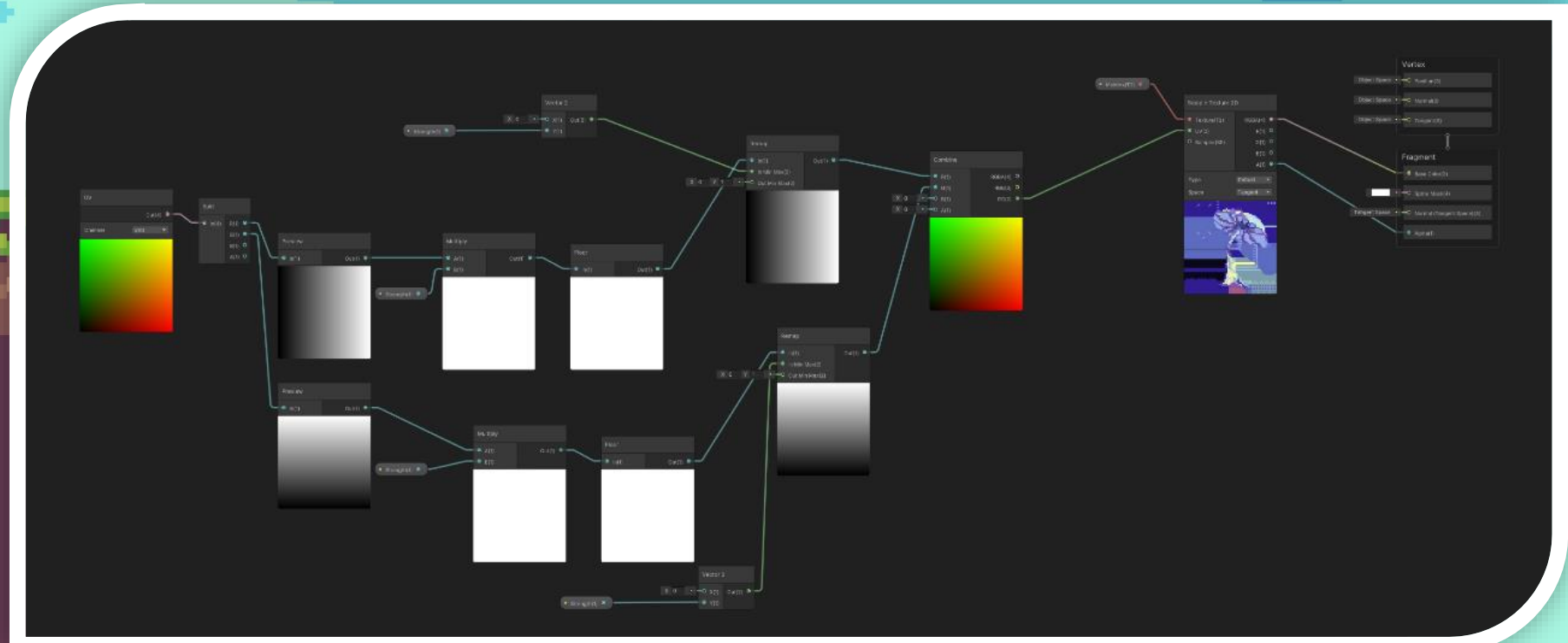
使用Unity的动画系统来实现Alice不同状态之间的流畅切换，跳跃过程中根据爱丽丝当前速度设置状态为上升或者降落，同时设定一个计数变量来实现Alice的二段跳




```
float verticalVelocity = rigid.velocity.y;
if (Input.GetKeyDown(KeyCode.Space)&&cnt<2)
{
    isJump = true;
    isGrounded = false;
    if (verticalVelocity < 0.0f)
    {
        animator.SetBool("DropToJump", true);
    }
    else animator.SetBool("RunToJump", true);
    isDrop = false;
    rigid.AddForce(Vector2.up * jumpForce, ForceMode2D.Impulse);
    cnt++;
}
if(verticalVelocity<0.0f)
{
    animator.SetBool("RunToJump", false);
    animator.SetBool("JumpToDrop",true);
    animator.SetBool("DropToRun",false);
    isDrop = true;
}

if(isGrounded)
{
    animator.SetBool("DropToRun",true);
    ump = false;
     = false;

MouseButtonDown(0)&&bulletNum>0)

etTrigger("Attack");
;
ttack");

et, transform.position + new Vector3(0.0f, -0.4f, 0), Quaternion.identity);
```

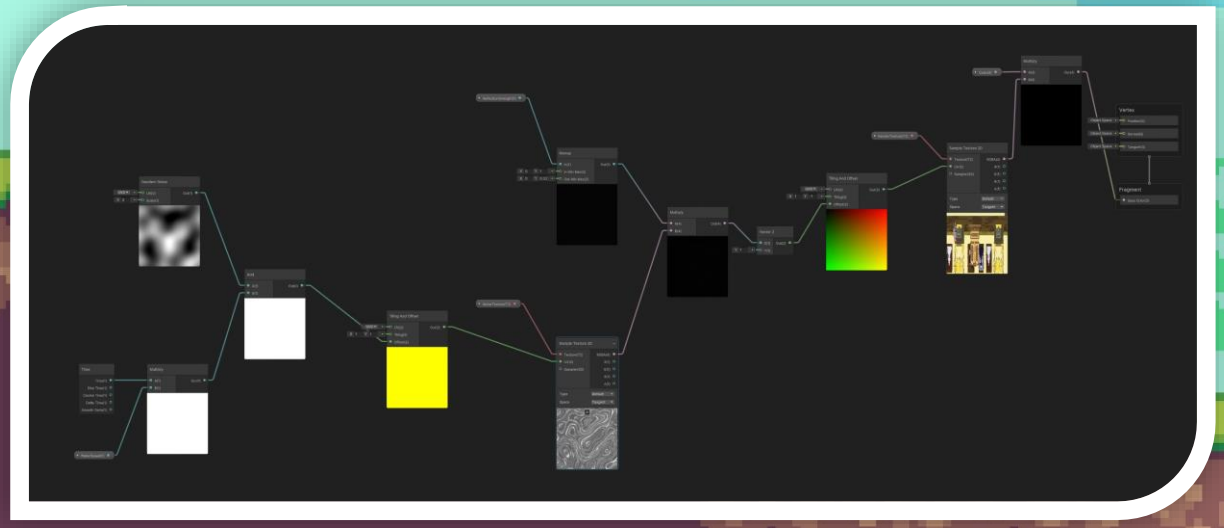
03 渲染和粒子特效

在击碎效果的shader graph中，我们通过texture2D获取被打碎的物体的texture并将其R和G进行重新组合，连接到图片的UV节点。而像素化的关键是将R和G的顺滑渐变变成阶梯式渐变，于是我们分别对R和G扩大一定范围后使用向下取整函数来改造，向下取整使小数部分被去除，从而得到阶梯式渐变。最后在场景中运用粒子系统爆发。



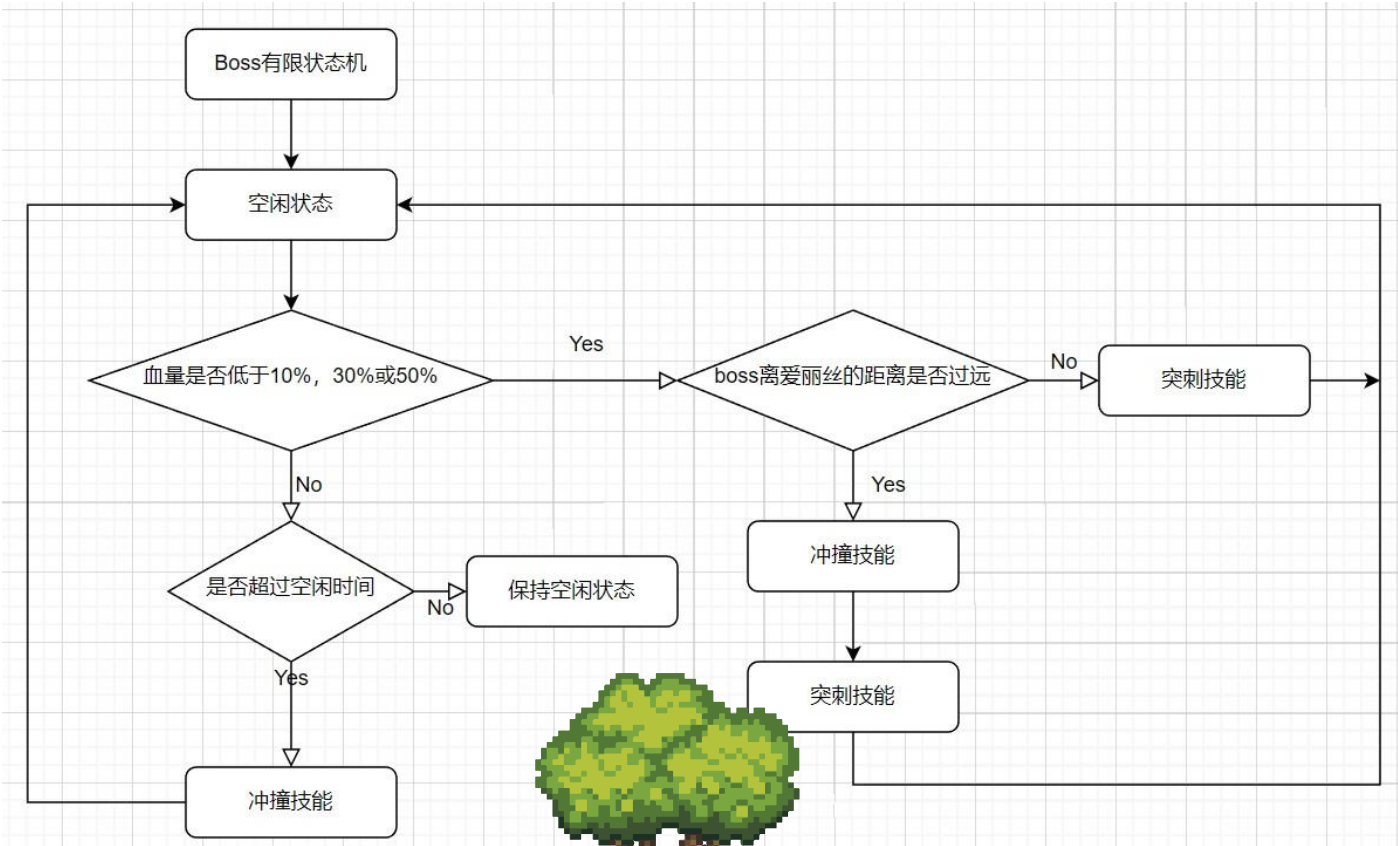
03 渲染和粒子特效

在水面效果中，我们通过通过texture2D获取摄像机的摄影内容，通过调整缩放和位置实现倒立，添加噪声和time结点实现水面的不规则波纹。



Boss机制 04

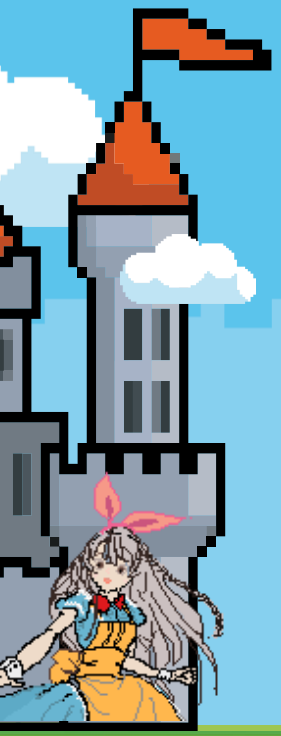
我们设计了boss的有限状态机实现了怪物的类智能。



难度设计
DIFFICULT

障碍物出现节点

采用模拟退火算法来构造障碍物出现的时间。模拟退火是一种随机化算法，我们会首先构造出障碍物大概出现的时间范围，然后实际测试针对各个玩家的游玩水平，根据退火的规律引入了更多随机因素，得到一个玩家能够较好体验游戏流程的最优解。在玩家测试的过程我们可以去记录这个过程，将目标函数作为能量函数。采用这种方法既满足了随机性，也使得游玩过程具有合理性，并且在跑酷的过程中爱丽丝的速度会越来越快，难度会自然而然地变大。



过程

先用一句话概括：如果新状态的解更优则修改答

我们定义当前
差为 ΔE （Δ

注意：我们有
随机状态，尝

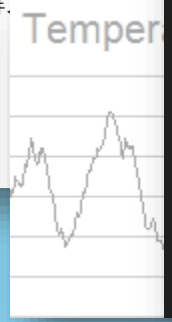
如何退火（降温）？

模拟退火时我们有三个参数:
数，d 是一个非常接近 1 但是

首先让温度 T = T_0，然后按
火过程结束，当前最优解即为

注意为了使得解
解的最优值。

引用一张 Wiki -
来越稳定）。



Assets > Scripts > TilemapMove.cs > TilemapMove

```
299         //初始状态的能量
300         double energy_new=0;
301         //最优状态
302         int[] mushRoomShow1_best=new int[8];
303         int[] mushRoomShow2_best=new int[7];
304         int[] smallMushRoomShow_best=new int[12];
305         int[] fishShow_best=new int[17];
306         int[] seaweedShow_best=new int[21];
307 >       for(int i=0;i<8;i++){…
310 >       for(int i=0;i<7;i++){…
313 >       for(int i=0;i<12;i++){…
316         for(int i=0;i<17;i++){
317             fishShow_best[i]=fishShow[i];
318         }
319         for(int i=0;i<21;i++){
320             seaweedShow_best[i]=seaweedShow[i];
321         }
322         //最优状态的能量
323         double energy_best=0;
324         //迭代次数
325         int k=0;
326         //迭代终止条件
327         while(T>T_end){
328             //产生新状态
329 >           for(int i=0;i<8;i++){…
332 >           for(int i=0;i<7;i++){…
335 >           for(int i=0;i<12;i++){…
338 >           for(int i=0;i<17;i++){…
341 >           for(int i=0;i<21;i++){…
344             //计算新状态的能量
345             energy_new=Energy(1,mushRoomShow1_new[0])+Energy(1,mushRoomShow1_new[1])+Energy
346             +Energy(2,mushRoomShow2_new[0])+Energy(2,mushRoomShow2_new[1])+Energy(2,mushRoo
347             +Energy(3,smallMushRoomShow_new[0])+Energy(3,smallMushRoomShow_new[1])+Energy(3
348             +Energy(4,fishShow_new[0])+Energy(4,fishShow_new[1])+Energy(4,fishShow_new[2])+
349             +Energy(5,seaweedShow_new[0])+Energy(5,seaweedShow_new[1])+Energy(5,seaweedShow
350             //计算能量差
351             double delta=energy_new-energy;
352             //判断是否接受新状态
353             if(delta<0){
354 >               for(int i=0;i<8;i++){…
```

难度设计
DIFFICULT DESIGN

跑动速度与关卡设置

作为一款跑酷游戏，随着时间推移，游戏的难度应该越来越大，对应的表现就是跑动的速度会不断加快同时生成的障碍物愈发密集为了实现这一个特性，我们在脚本开启了一个协程用来计算当前的游戏时间，之后根据当前的时间节点生成对应的障碍物来实现关卡设计

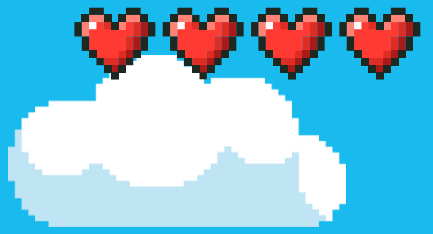
对于场景滚动的加速，我们只需要将对应的地面和障碍物移动速度设置为一个变量Speed，并将该变量和当前的时间建立绑定关系，就可以实现随着时间的推移物体运动的速度不断加快

```
void BarrierController(int t)
{
    Speed = v0 + 6.0f * t / 60;
    if (t >= 60)//finished
    {
        //todo
    }
    else
    {
        if (isMushRoomAppear(t) && barrier == null && t % 2 == 1) barrier = Instantiate(mushRoomPreferb1, transform.position + new Vector3(20.0f, 2.00f, 0), Quaternion.ident
        else if (isMushRoomAppear(t) && barrier == null && t % 2 == 0) barrier = Instantiate(mushRoomPreferb2, transform.position + new Vector3(20.0f, 2.00f, 0), Quaternion.
        if(isSmallMushRoomAppear(t) && barrier2 == null) barrier2 = Instantiate(smallMashroomPreferb, transform.position + new Vector3(20.0f, 1.18f, 0), Quaternion.identity
        if (isFishAppear(t) && fish == null) fish = Instantiate(fishPreferb, transform.position + new Vector3(20.0f, 3.5f, 0), Quaternion.identity);
        //下面控制水草生成
        if (isSeaweedAppear(t) && seaweed == null && t % 2 == 1) seaweed = Instantiate(seaweedPreferb1, transform.position + new Vector3(20.0f, -3f, 0), Quaternion.identity
        else if (isSeaweedAppear(t) && seaweed == null && t % 2 == 0) seaweed = Instantiate(seaweedPreferb2, transform.position + new Vector3(20.0f, -3f, 0), Quaternion.ide
    }
}
```

```
void Update()
{
    if (Time.time - lasttime >= 3)
    {
        for (int i = 0; i < 2; ++i)
        {
            if (tilemap[i].transform.position.x < -20.69f)
            {
                tilemap[i].transform.position = new Vector3(20.58f, ystart, zstart);
            }
            else
            {
                tilemap[i].transform.position -= new Vector3(Time.deltaTime*Speed, 0, 0);
            }
        }
    }
    BarrierController((int)currentTime);
    BarrierDestroy();
}
void BarrierDestroy()
{
    if (barrier != null && barrier.transform.position.x < -11.0f) Destroy(barrier);
}
void BarrierController(int t)
{
    Speed = v0 + 6.0f * t / 60;
    if (t >= 60)//finished
```

WORK

人员分工



姓名	负责
张璐冰	各素材绘画、渲染管线与状态机书写
曹志逸	世界观构架、游戏玩法设计与文档书写
黄子安	Unity场景及代码构建
应驰骅	Unity场景及代码构建
黄勖	音乐、关卡脚本、代码等杂项
吕芸鹤	部分场景素材绘画

×15 ×06

DEMO

游戏演示


×15
×06
Game Design
感谢您的观看
THANKS FOR WATCHING