## 单元测试报告

测试模块：游戏状态初始化

测试工具：unity

测试人员：吴子乔

### 程序结构（主结构）

if(timeval >= 3600)

{

timeval = 0;

people\_times = people\_times + 1;

item\_times = item\_times + 1;

//选择狼人吃人类

if (gameChoice == 0)

{

//最多生成4次人物以及10次新道具

if (people\_times <= 4)

{

//6区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(-4, 3, 0), Quaternion.identity);

//7区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(4, 3, 0), Quaternion.identity);

//10区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(-4, -5, 0), Quaternion.identity);

//11区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(4, -5, 0), Quaternion.identity);

}

else

{

if(GameObject.Find("MapCreation/People(Clone)") == null)

{

//场上所有人物都被吃光，进入下一关

MainCharacter.youWin = true;

MainCharacter.isOver = true;

}

}

//最多生成10次道具

if(item\_times <= 10)

{

CreateItems(5);

}

}

//选择了人类

else

{

//最多生成10次道具

if (item\_times <= 10)

{

CreateItems(5);

}

}

}

else

{

timeval = timeval + 1;

}

### 2.测试结果分析

测试1

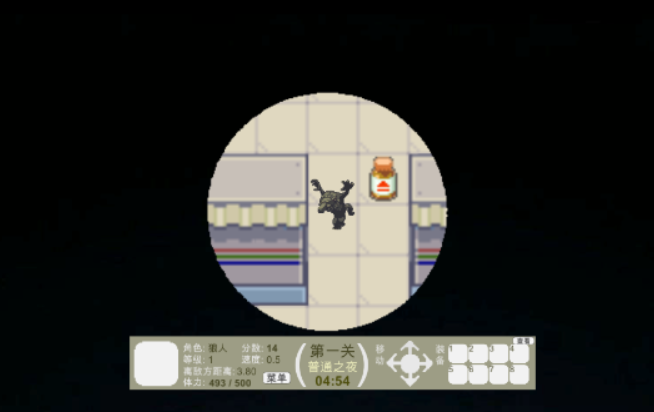
进入游戏场景人物生成、地图生成、道具生成、操作栏操作是否正常



分析：在进入游戏是查看上几类测试信息，控制角色移动，测试地图内碰撞块是否正常生成。

测试2

控制角色碰撞墙体



分析：角色在碰撞墙体时是否可通过墙体，不可通过墙体，是否会嵌入墙体无法移动，是否在碰撞墙体后能正常移动。

测试3

检验游戏道具是否生成正常，是否可以拾取



分析：游戏道具再地图中是否正常生成并且是否可视，游戏道具是否可以拾取

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例名称 | | 游戏初始化 | | | |
| 测试类型 | | 单元测试 | | | |
| 测试目的 | | 测试在游戏开始时，游戏状态是否初始化正常 | | | |
| 测试方法 | | 在游戏界面控制角色在地图上移动、拾取道具、碰撞墙体等 | | | |
| 测试用例设计原则 | | 条件组合覆盖 | | | |
| 正常测试情况 | | | | | |
| 编号 | 测试数据 | 数据描述 | 预期结果 | 实际测试结果 | |
|  | 游戏操作栏控制角色移动 | 角色在地图中走动 | 角色在地图中走动，并且相应参数发生正常变化 | 角色在地图中走动，并且相应参数正常改变 | |
|  | 操作游戏角色碰撞墙体并在全地图行走 | 检验地图碰撞块生成是否正常 | 能够在地图内自由行走，并且游戏墙体能阻隔游戏角色行走，并且地图内没有走不通的路 | 满足预期结果 | |
|  | 查看地图道具是否生成异常，是否可以拾取道具 | 检验道具生成是否正常 | 能够顺利拾取道具 | 满足预期结果 | |
| 测试结论 | |  |  |  |  |
| 状态 | | 正常 |  |  |  |

测试模块：游戏道具使用及效果

测试工具：unity

测试人员：吴子乔

### 1.程序结构（核心代码）

if (isOver == false)

{

Vector2 temp = Vector2.MoveTowards(transform.position, dest, speed);

GetComponent<Rigidbody2D>().MovePosition(temp);

//在角色能移动的前提下进行移动

if (canMove == true)

{

if ((Vector2)transform.position == dest)

{

GameObject btnObjup = GameObject.Find("upBtn");

Button btnup = btnObjup.GetComponent<Button>();

btnup.onClick.AddListener(Ononeup);

GameObject btnObjdown = GameObject.Find("downBtn");

Button btndown = btnObjdown.GetComponent<Button>();

btndown.onClick.AddListener(Ononedown);

GameObject btnObjright = GameObject.Find("rightBtn");

Button btnright = btnObjright.GetComponent<Button>();

btnright.onClick.AddListener(Ononeright);

GameObject btnObjleft = GameObject.Find("leftBtn");

Button btnleft = btnObjleft.GetComponent<Button>();

btnleft.onClick.AddListener(Ononeleft);

}

else

{

//移动完毕，改变分数和体力等属性状态

changeOnMove();

if (healthPoint <= 0)

{

//选择狼人

if (MapCreation.gameChoice == 0)

{

//狼人体力耗尽，狼人死掉，游戏结束

isOver = true;

youWin = false;

canMove = false;

}

//选择人类

else

{

//人类体力耗尽，不能移动，等着被狼人吃掉

canMove = false;

}

}

}

}

//道具箱按钮的监听事件

GameObject bag1 = GameObject.Find("bag1");

Button button\_bag1 = bag1.GetComponent<Button>();

button\_bag1.onClick.AddListener(bag1Event);

GameObject bag2 = GameObject.Find("bag2");

Button button\_bag2 = bag2.GetComponent<Button>();

button\_bag2.onClick.AddListener(bag2Event);

GameObject bag3 = GameObject.Find("bag3");

Button button\_bag3 = bag3.GetComponent<Button>();

button\_bag3.onClick.AddListener(bag3Event);

GameObject bag4 = GameObject.Find("bag4");

Button button\_bag4 = bag4.GetComponent<Button>();

button\_bag4.onClick.AddListener(bag4Event);

GameObject bag5 = GameObject.Find("bag5");

Button button\_bag5 = bag5.GetComponent<Button>();

button\_bag5.onClick.AddListener(bag5Event);

GameObject bag6 = GameObject.Find("bag6");

Button button\_bag6 = bag6.GetComponent<Button>();

button\_bag6.onClick.AddListener(bag6Event);

GameObject bag7 = GameObject.Find("bag7");

Button button\_bag7 = bag7.GetComponent<Button>();

button\_bag7.onClick.AddListener(bag7Event);

GameObject bag8 = GameObject.Find("bag8");

Button button\_bag8 = bag8.GetComponent<Button>();

button\_bag8.onClick.AddListener(bag8Event);

//showCharacterProperties();

//print(itemNum);

//道具栏状态（改变等级，离敌方距离（(玩家进入区域时)找玩家所属区域的敌方），体力，速度，分数等属性）

//等级

level = (int)Mathf.Floor(score / 2500) + 1;

GameObject.Find("rank\_value").GetComponent<Text>().text = level.ToString();

//距离

GameObject.Find("distance\_value").GetComponent<Text>().text = distance.ToString();

//体力

//print(score);

string healthPoint\_str1 = healthPoint.ToString();

string healthPoint\_str2 = max\_healthPoint.ToString();

string healthPoint\_str = healthPoint\_str1 + " / " + healthPoint\_str2;

GameObject.Find("blood\_value").GetComponent<Text>().text = healthPoint\_str;

//速度

speed = (float)(0.5 + (level - 1) \* 0.05);

GameObject.Find("speed\_value").GetComponent<Text>().text = speed.ToString();

//离（最近的）敌方距离

//选择狼人，获取与最近人类的距离

if(MapCreation.gameChoice == 0)

distance = getNearestDistance("hunter");

//选择人类，获取与最近狼人的距离

else

distance = getNearestDistance("wolf");

if (distance <= 3.5f)

{

this.gameObject.GetComponent<AudioSource>().mute = false;

//this.gameObject.GetComponent<AudioSource>().Play();

}

else

{

this.gameObject.GetComponent<AudioSource>().mute = true;

//this.gameObject.GetComponent<AudioSource>().Pause();

}

GameObject.Find("distance\_value").GetComponent<Text>().text = distance.ToString();

//分数

GameObject.Find("grade\_value").GetComponent<Text>().text = score.ToString();

//剩余时间

currentTime = currentTime - 1;

if (koo == 1) //照明弹计时

{

timer = timer - 1;

}

if (timer <= 0)

{

sd.SetTexture("\_Mask", mask1);

timer = 300f;

koo = 0;

}

int currentTime\_minute = (int)currentTime / 3600;

int currentTime\_second = (int)(currentTime - currentTime\_minute \* 3600) / 60;

string currentTime\_minute\_str = currentTime\_minute.ToString();

string currentTime\_second\_str;

string currentTime\_str;

//秒数<10，前面+0

if (currentTime\_second < 10)

currentTime\_second\_str = "0" + currentTime\_second.ToString();

else

currentTime\_second\_str = currentTime\_second.ToString();

//分钟数<10，前面+0

if (currentTime\_minute < 10)

currentTime\_str = "0" + currentTime\_minute\_str + ":" + currentTime\_second\_str;

else

currentTime\_str = currentTime\_minute\_str + ":" + currentTime\_second\_str;

GameObject.Find("剩余时间").GetComponent<Text>().text = currentTime\_str;

if (currentTime <= 0)

{

//时间结束，你没有被狼人碰到，游戏结束，并且你在这个关卡中获胜

isOver = true;

youWin = true;

}

else

{

if (currentTime % 3600 == 0)

{

state = Random.Range(0, 3);

if (state == 0)

{

GameObject.Find("夜晚状态").GetComponent<Text>().text = "宿 夜";

}

else if (state == 1)

{

GameObject.Find("夜晚状态").GetComponent<Text>().text = "普通之夜";

}

else

{

GameObject.Find("夜晚状态").GetComponent<Text>().text = "月圆之夜";

}

}

}

//装备剩余使用时间（>0的-1）

for (int i = 0; i < 10; i++)

{

if (item\_remainedTime[i] > 0)

{

item\_remainedTime[i] = item\_remainedTime[i] - 1;

if (i == 5 && item\_remainedTime[i] <= 0)

{

GetComponent<Renderer>().material.color = new Color(1.0f, 1.0f, 1.0f, 1.0f);

}

else if(i == 6 && item\_remainedTime[i] <= 0)

{

GameObject obj = GameObject.FindGameObjectWithTag("dummy");

if (obj != null)

{

Destroy(obj);

}

}

}

}

}

else

{

this.gameObject.GetComponent<AudioSource>().Stop();

}

//使用结果作为返回值，成功使用返回true，否返回false

bool useAns = true;

int itemName = itemBox[index - 1];

GameObject obj = null;

switch (itemName)

{

//人类和狼人共同拥有的五个道具：时光药水、体力药水、铲子、照明弹、手电筒

case 1:

//拳头：假如附近区域有狼（敌人），附近区域的狼消失

//选择人类，消灭在这块区域内的狼人

if(MapCreation.gameChoice == 1)

{

obj = FindClosestObject("wolf", 4.0f);

if(obj != null)

{

Destroy(obj);

}

else

{

useAns = false;

}

}

//选择狼人：

//能用则改变角色属性

if(useAns == true) {

if (state == 0)

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 180;

//人类

else

score = score + 220;

}

else if (state == 1)

score = score + 200;

else

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 220;

//人类

else

score = score + 180;

}

}

break;

case 2:

//防狼喷雾（×）：改变狼朝你方向行走的概率（和隐身衣功能差不多，可以取消）

if (state == 0)

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 80;

//人类

else

score = score + 120;

}

else if (state == 1)

score = score + 100;

else

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 120;

//人类

else

score = score + 80;

}

//item\_remainedTime[2] = item\_canUseTime[2];

break;

case 3:

//铲子（√）：玩家附近的障碍消失

obj = FindClosestObject("destroyWall", 1.0f);

if (obj != null)

{

Destroy(obj);

}

else

{

useAns = false;

}

//能用则改变角色属性

if(useAns == true)

{

if (state == 0)

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 80;

//人类

else

score = score + 120;

}

else if (state == 1)

score = score + 100;

else

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 120;

//人类

else

score = score + 80;

}

}

break;

case 4:

//照明弹（O）：3秒内整个地图亮度调为1

//狼人

if (MapCreation.gameChoice == 0)

{

sd.SetTexture("\_Mask", mask2);

koo = 1;//确定是否使用照明弹

}

//人类

else

{

sd.SetTexture("\_Mask", mask2);

koo = 1;//确定是否使用照明

}

break;

case 5:

//隐身衣：隐身自己，隐身期间敌人碰到你照样也没关系

item\_remainedTime[5] = item\_canUseTime[5];

GetComponent<Renderer>().material.color = new Color(1.0f, 1.0f, 1.0f, 0.5f);

if (state == 0)

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 80;

//人类

else

score = score + 120;

}

else if (state == 1)

score = score + 100;

else

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 120;

//人类

else

score = score + 80;

}

break;

case 6:

//傀儡：让最近的狼人去追新生成的人类

if(MapCreation.gameChoice == 1)

{

Instantiate(prefabList[2], this.gameObject.transform.position, Quaternion.identity);

//让距离最近的狼人去追傀儡

obj = FindClosestObject("wolf", 100.0f);

obj.GetComponent<Wolf>().traceDummy = true;

}

if (state == 0)

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 180;

//人类

else

score = score + 220;

}

else if (state == 1)

score = score + 200;

else

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 220;

//人类

else

score = score + 180;

}

break;

case 7:

//（√）时光药水：关卡时间(加减十秒的整数倍，绝对值不超过60)s（已做）

currentTime = currentTime + Random.Range(-6, 7) \* 600;

if (state == 0)

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 40;

//人类

else

score = score + 60;

}

else if (state == 1)

score = score + 50;

else

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 60;

//人类

else

score = score + 40;

}

break;

case 8:

//（√）体力药水：增加角色体力（已做）

healthPoint = healthPoint + Random.Range(0, max\_healthPoint - healthPoint + 1);

if (state == 0)

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 40;

//人类

else

score = score + 60;

}

else if (state == 1)

score = score + 50;

else

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 60;

//人类

else

score = score + 40;

}

break;

case 9:

//手电筒（O）：视野范围+1

//狼人

if (MapCreation.gameChoice == 0)

{

sd.SetTexture("\_Mask", mask3);

}

//人类

else

{

sd.SetTexture("\_Mask", mask3);

}

break;

default:

useAns = false;

break;

}

### 2.测试结果分析

测试1

道具1体力药水的使用



分析：使用体力药水可以回复一定的体力值。

测试2

道具2手电筒的使用



分析：使用手电筒可以使视野范围变得更广。

测试3

道具3时光药水的使用



分析：使用时光药水可以增加一定的游戏时间或者减少一定的游戏时间，由一定概率决定。

测试4

道具4照明弹的使用



分析：使用照明弹可以使当前区域在一定时间内照亮。

测试5

道具5拳头的使用



分析：使用拳头可以再一定范围内将怪物杀死。

测试6

道具6隐身衣的使用



分析：使用隐身衣后，怪物将在一段时间内不可视你。

测试7

道具7铲子的使用



分析：使用铲子可以破坏一定的墙体。

测试8

道具8傀儡的使用



分析：使用傀儡可以引开敌人。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例名称 | | 游戏道具使用及效果分析 | | | |
| 测试类型 | | 单元测试 | | | |
| 测试目的 | | 测试在游戏进行中时，游戏道具使用是否正常 | | | |
| 测试方法 | | 使用相应道具 | | | |
| 测试用例设计原则 | | 条件组合覆盖 | | | |
| 正常测试情况 | | | | | |
| 编号 | 测试数据 | 数据描述 | 预期结果 | 实际测试结果 | |
|  | 使用道具1 | 使用道具1，点击事件 | 增加一定体力 | 增加一定体力 | |
|  | 使用道具2 | 使用道具2，点击事件 | 视野变广 | 视野变广 | |
|  | 使用道具3 | 使用道具3，点击事件 | 游戏时长增加或者减少 | 游戏时长增加和减少有一定的概率性 | |
|  | 使用道具4 | 使用道具4，点击事件 | 在一定时间内照亮区域 | 在一定时间内照亮区域 | |
|  | 使用道具5 | 使用道具5，点击事件 | 在一定区域内将怪物击杀 | 在一定区域内将怪物击杀 | |
|  | 使用道具6 | 使用道具6，点击事件 | 在一定时间内实现隐身效果 | 在一定时间内实现隐身效果 | |
|  | 使用道具7 | 使用道具7，点击事件 | 消除特定方向的障碍物 | 消除特定方向的障碍物 | |
|  | 使用道具8 | 使用道具8，点击事件 | 使怪物变化追逐目标 | 使怪物变化追逐目标 | |
| 测试结论 | |  |  |  |  |
| 状态 | | 部分道具效果出现BUG同时道具的效果不够明显。 | | |  |

测试模块：游戏内NPC移动及其效果

测试工具：unity

测试人员：吴子乔

### 1.程序结构（主结构）

state = laststate = 1;

speed\_f = 45;

//根据x，y位置决定人物的活动范围（只存左上到右下）

x = this.transform.position.x;

y = this.transform.position.y;

is\_inRange = false;

//1,2,5,6号大块

if (x >= -16.01f && x <= -15.99f && y >= 9.99f && y <= 10.01f)

{

start\_x = -18;

start\_y = 12;

end\_x = -1;

end\_y = 1;

}

//3,4,7,8号大块

else if (x >= 15.99f && x <= 16.01f && y >= 9.99f && y <= 10.01f)

{

start\_x = 1;

start\_y = 12;

end\_x = 18;

end\_y = 1;

}

//9,10,13,14号大块

else if (x >= -16.01f && x <= -15.99f && y >= -12.01f && y <= -11.99f)

{

start\_x = -18;

start\_y = -3;

end\_x = -1;

end\_y = -14;

}

//11,12,15,16号大块

else if (x >= 15.99f && x <= 16.01f && y >= -12.01f && y <= -11.99f)

{

start\_x = 1;

start\_y = -3;

end\_x = 18;

end\_y = -14;

}

maze = new int[37, 29];

for (int i = 0; i < 37; i++)

{

for (int j = 0; j < 29; j++)

{

maze[i, j] = MapCreation.getMaze(i, j);

}

}

}

// Update is called once per frame

void FixedUpdate () {

//状态的变化

laststate = state;

//（state=0(宿夜),state=1(普通之夜),state=2(月圆之夜)）

state = MainCharacter.state;

if (state == 0)

{

//狼人能力减弱

speed = 0.4f;

speed\_f = 50;

}

else if (state == 1)

{

speed = 0.5f;

speed\_f = 45;

}

else

{

//狼人能力增强

speed = 0.6f;

speed\_f = 40;

}

//设置一个时间变量来控制猎人行走（随机行走）

if (timeval >= speed\_f)

{

timeval = 0;

//设置一个变量判断行走能否走得通

bool can\_move = false;

int properbility = 0;

Vector2 t;

x = this.transform.position.x;

y = this.transform.position.y;

//print(x + " " + y);

//向四个方向走的概率

properbility\_up = -1;

properbility\_down = -1;

properbility\_left = -1;

properbility\_right = -1;

//int表示向0方向取整

int tx, ty;

tx = (int)x + 18;

ty = -(int)(Mathf.Ceil(speed) + y) + 14;

if (tx < 0 || ty < 0 || tx >= 37 || ty >= 29)

{

directions[0] = -1;

}

else

{

directions[0] = maze[tx, ty];

}

tx = (int)(Mathf.Ceil(speed) + x) + 18;

ty = -(int)y + 14;

if (tx < 0 || ty < 0 || tx >= 37 || ty >= 29)

{

directions[1] = -1;

}

else

{

directions[1] = maze[tx, ty];

}

tx = (int)x + 18;

ty = -(int)(-Mathf.Ceil(speed) + y) + 14;

if (tx < 0 || ty < 0 || tx >= 37 || ty >= 29)

{

directions[2] = -1;

}

else

{

directions[2] = maze[tx, ty];

}

tx = (int)(-Mathf.Ceil(speed) + x) + 18;

ty = -(int)y + 14;

if (tx < 0 || ty < 0 || tx >= 37 || ty >= 29)

{

directions[3] = -1;

}

else

{

directions[3] = maze[tx, ty];

}

//统计不能走的数量（走迷宫既然走到迷宫块来了就肯定存在回路的情况，count不会超过3）

int countDirections = 0;

if (directions[0] == -1)

{

properbility\_up = 0;

countDirections = countDirections + 1;

}

if (directions[1] == -1)

{

properbility\_right = 0;

countDirections = countDirections + 1;

}

if (directions[2] == -1)

{

properbility\_down = 0;

countDirections = countDirections + 1;

}

if (directions[3] == -1)

{

properbility\_left = 0;

countDirections = countDirections + 1;

}

//未赋值

if(properbility\_up == -1)

{

properbility\_up = (int)(100 / (4 - countDirections));

}

if (properbility\_right == -1)

{

properbility\_right = (int)(100 / (4 - countDirections));

}

if (properbility\_down == -1)

{

properbility\_down = (int)(100 / (4 - countDirections));

}

if (properbility\_left == -1)

{

properbility\_left = (int)(100 / (4 - countDirections));

}

can\_move = false;

while (!can\_move)

{

properbility = Random.Range(1, 101);

//往上走

if (properbility > 0 && properbility <= properbility\_up)

{

t = (Vector2)this.transform.position + Vector2.up \* speed;

if (t.y >= end\_y && t.y <= start\_y)

{

can\_move = true;

this.transform.position = t;

GetComponent<Animator>().SetFloat("drix", 0);

GetComponent<Animator>().SetFloat("driy", 0.5f);

}

}

//往右走

else if (properbility > properbility\_up && properbility <= properbility\_up+properbility\_right)

{

t = (Vector2)this.transform.position + Vector2.down \* speed;

if (t.y >= end\_y && t.y <= start\_y)

{

can\_move = true;

this.transform.position = t;

GetComponent<Animator>().SetFloat("drix", 0.5f);

GetComponent<Animator>().SetFloat("driy", 0);

}

}

//往下走

else if (properbility > properbility\_up + properbility\_right && properbility <= properbility\_up + properbility\_right + properbility\_down)

{

t = (Vector2)this.transform.position + Vector2.left \* speed;

if (t.x >= start\_x && t.x <= end\_x)

{

can\_move = true;

this.transform.position = t;

GetComponent<Animator>().SetFloat("drix", 0);

GetComponent<Animator>().SetFloat("driy", -0.5f);

}

}

//往左走

else if(properbility > properbility\_up + properbility\_right + properbility\_down && properbility <= 100)

{

t = (Vector2)this.transform.position + Vector2.right \* speed;

if (t.x >= start\_x && t.x <= end\_x)

{

can\_move = true;

this.transform.position = t;

GetComponent<Animator>().SetFloat("drix", -0.5f);

GetComponent<Animator>().SetFloat("driy", 0);

}

}

}

}

else

{

timeval = timeval + 1;

}

//获取玩家在地图当中生成的位置

float x, y;

//出口位置

float exit\_x, exit\_y;

//FixedUpdate计时工具

int timeval = 0;

//速度

float speed;

//人类运动的频率

int speed\_f = 60;

//这时候的状态和上一刻的状态

int state, laststate;

// Use this for initialization

void Start()

{

state = laststate = 1;

speed\_f = 60;

//根据x，y位置决定人物的活动范围（只存左上到右下）

x = this.transform.position.x;

y = this.transform.position.y;

//决定速度

speed = 0.5f;

//人物生成在固定区域内，但是会朝出口的方向行走

//根据endx,endy来决定exit\_x,exit\_y

int endx = MapCreation.endx;

int endy = MapCreation.endy;

if (endx == 0 && endy == 0)

{

exit\_x = -18;

exit\_y = 11;

}

else if (endx == 1 && endy == 0)

{

exit\_x = -18;

exit\_y = 3;

}

else if (endx == 2 && endy == 0)

{

exit\_x = -18;

exit\_y = -5;

}

else if (endx == 3 && endy == 0)

{

exit\_x = -18;

exit\_y = -12;

}

else if (endx == 0 && endy == 1)

{

exit\_x = -5;

exit\_y = 11;

}

else if (endx == 3 && endy == 1)

{

exit\_x = -5;

exit\_y = -12;

}

else if (endx == 0 && endy == 2)

{

exit\_x = 5;

exit\_y = 11;

}

else if (endx == 3 && endy == 2)

{

exit\_x = 5;

exit\_y = -12;

}

else if (endx == 0 && endy == 3)

{

exit\_x = 18;

exit\_y = 11;

}

else if (endx == 1 && endy == 3)

{

exit\_x = 18;

exit\_y = 3;

}

else if (endx == 2 && endy == 3)

{

exit\_x = 18;

exit\_y = -5;

}

else

{

exit\_x = 18;

exit\_y = -12;

}

}

// Update is called once per frame

void FixedUpdate()

{

//状态的变化

laststate = state;

state = MainCharacter.state;

//（state=0(宿夜),state=1(普通之夜),state=2(月圆之夜)）

if (state == 0)

{

//人类能力增强

speed = 0.6f;

speed\_f = 55;

}

else if (state == 1)

{

speed = 0.5f;

speed\_f = 60;

}

else

{

//人类能力减弱

speed = 0.4f;

speed\_f = 65;

}

//通过transform.position获取主角的位置

if (timeval >= speed\_f)

{

timeval = 0;

//人类向出口走

int properbility;

int properbility\_up, properbility\_down, properbility\_left, properbility\_right;

float dy, dx;

Vector2 t;

dx = exit\_x - this.transform.position.x;

dy = exit\_y - this.transform.position.y;

//根据水平和垂直偏移，确定向四个方向移动的概率

if (dy >= 0)

{

properbility\_up = (int)(dy / (dy + Mathf.Abs(dx)) \* 90);

properbility\_down = 10;

}

else

{

properbility\_up = 10;

properbility\_down = (int)(-dy / (-dy + Mathf.Abs(dx)) \* 90);

}

if (dx >= 0)

{

properbility\_right = 80 - Mathf.Max(properbility\_up, properbility\_down);

properbility\_left = 10;

}

else

{

properbility\_right = 10;

properbility\_left = 80 - Mathf.Max(properbility\_up, properbility\_down);

}

//人类朝四个方向随机走

properbility = Random.Range(0, 101);

//往上走

if (properbility >= 0 && properbility <= properbility\_up)

{

t = (Vector2)this.transform.position + Vector2.up \* speed;

this.transform.position = t;

GetComponent<Animator>().SetFloat("drix", 0);

GetComponent<Animator>().SetFloat("driy", 0.5f);

}

//往下走

else if (properbility > properbility\_up && properbility <= (properbility\_up + properbility\_down))

{

t = (Vector2)this.transform.position + Vector2.down \* speed;

this.transform.position = t;

GetComponent<Animator>().SetFloat("drix", 0);

GetComponent<Animator>().SetFloat("driy", -0.5f);

}

//往左走

else if (properbility > (properbility\_up + properbility\_down) && properbility <= (properbility\_up + properbility\_down + properbility\_left))

{

t = (Vector2)this.transform.position + Vector2.left \* speed;

this.transform.position = t;

GetComponent<Animator>().SetFloat("drix", -0.5f);

GetComponent<Animator>().SetFloat("driy", 0);

}

//往右走

else

{

t = (Vector2)this.transform.position + Vector2.right \* speed;

this.transform.position = t;

GetComponent<Animator>().SetFloat("drix", 0.5f);

GetComponent<Animator>().SetFloat("driy", 0);

}

}

else

{

timeval = timeval + 1;

}

}

//触发事件

void OnTriggerEnter2D(Collider2D collider)

{

//碰到狼人（主要针对傀儡道具来说）

if (collider.tag.Equals("wolf") == true)

{

Destroy(this.gameObject);

}

//不是人撞人的情况以及碰到道具的情况

else if (collider.tag.Equals("people") == false && collider.tag.Contains("item")==false)

{

float dirx = this.GetComponent<Animator>().GetFloat("drix");

float diry = this.GetComponent<Animator>().GetFloat("driy");

Vector2 t;

//往下过头

if(diry <= -0.1)

{

this.GetComponent<Animator>().SetFloat("drix", 0f);

this.GetComponent<Animator>().SetFloat("driy", 0.5f);

t = (Vector2)this.transform.position + Vector2.up \* speed \* 4;

this.transform.position = t;

}

//往上过头

else if(diry >= 0.1)

{

this.GetComponent<Animator>().SetFloat("drix", 0f);

this.GetComponent<Animator>().SetFloat("driy", -0.5f);

t = (Vector2)this.transform.position + Vector2.down \* speed \* 4;

this.transform.position = t;

}

//往左过头

if (dirx <= -0.1)

{

this.GetComponent<Animator>().SetFloat("drix", 0.5f);

this.GetComponent<Animator>().SetFloat("driy", 0f);

t = (Vector2)this.transform.position + Vector2.right \* speed \* 2;

this.transform.position = t;

}

//往右过头

else if (dirx >= 0.1)

{

this.GetComponent<Animator>().SetFloat("drix", -0.5f);

this.GetComponent<Animator>().SetFloat("driy", 0f);

t = (Vector2)this.transform.position + Vector2.left \* speed \* 2;

this.transform.position = t;

}

}

//状态的变化

laststate = state;

//（state=0(宿夜),state=1(普通之夜),state=2(月圆之夜)）

state = MainCharacter.state;

if (state == 0)

{

//狼人能力减弱

speed = 0.4f;

speed\_f = 50;

}

else if (state == 1)

{

speed = 0.5f;

speed\_f = 45;

}

else

{

//狼人能力增强

speed = 0.6f;

speed\_f = 40;

}

/\*

start\_x = start\_x - (state - laststate) \* 1;

start\_y = start\_y + (state - laststate) \* 1;

end\_x = end\_x + (state - laststate) \* 1;

end\_y = end\_y - (state - laststate) \* 1;

\*/

//设置一个时间变量来控制狼人行走

if (timeval >= speed\_f)

{

timeval = 0;

GameObject obj = null;

if(traceDummy == false)

{

obj = GameObject.Find("MainCharacter");

}

else

{

obj = GameObject.FindGameObjectWithTag("dummy");

//暂时性的

if(obj == null)

{

traceDummy = false;

obj = GameObject.Find("MainCharacter");

}

}

//设置一个变量判断行走能否走得通

bool can\_move = false;

int properbility = 0;

Vector2 t;

float mainCharacter\_postion\_x = obj.transform.position.x;

float mainCharacter\_postion\_y = obj.transform.position.y;

//判断玩家的位置是否进入狼人所管辖的范围内

if(mainCharacter\_postion\_x>=start\_x && mainCharacter\_postion\_x <= end\_x && mainCharacter\_postion\_y>=end\_y && mainCharacter\_postion\_y <= start\_y)

{

is\_inRange = true;

}

else

{

is\_inRange = false;

}

//根据玩家是否进入到狼人所管辖到狼人的区域内来判断狼人所走的位置

//根据关卡的不同，狼人行动的概率也不同

if(is\_inRange == true)

{

int properbility\_up, properbility\_down, properbility\_left, properbility\_right;

float dy, dx;

dx = mainCharacter\_postion\_x - this.transform.position.x;

dy = mainCharacter\_postion\_y - this.transform.position.y;

//根据水平和垂直偏移，确定向四个方向移动的概率

if (dy >= 0)

{

properbility\_up = (int)(dy/(dy+Mathf.Abs(dx))\*90);

properbility\_down = 5;

}

else

{

properbility\_up = 5;

properbility\_down = (int)(-dy / (-dy + Mathf.Abs(dx)) \* 90);

}

if(dx >= 0)

{

properbility\_right = 90 - Mathf.Max(properbility\_up, properbility\_down);

properbility\_left = 5;

}

else

{

properbility\_right = 5;

properbility\_left = 90 - Mathf.Max(properbility\_up, properbility\_down);

}

//循环，直到狼人能走得通（这里是正方形区域，能够保证狼人能走得通）

while (!can\_move)

{

properbility = Random.Range(0, 101);

//往上走

if (properbility >= 0 && properbility <= properbility\_up)

{

t = (Vector2)this.transform.position + Vector2.up \* speed;

if (t.y >= end\_y && t.y <= start\_y)

{

can\_move = true;

this.transform.position = t;

GetComponent<Animator>().SetFloat("drix", 0);

GetComponent<Animator>().SetFloat("driy", 0.5f);

}

}

//往下走

else if (properbility > properbility\_up && properbility <= (properbility\_up+ properbility\_down))

{

t = (Vector2)this.transform.position + Vector2.down \* speed;

if (t.y >= end\_y && t.y <= start\_y)

{

can\_move = true;

this.transform.position = t;

GetComponent<Animator>().SetFloat("drix", 0);

GetComponent<Animator>().SetFloat("driy", -0.5f);

}

}

//往左走

else if (properbility > (properbility\_up + properbility\_down) && properbility <= (properbility\_up + properbility\_down + properbility\_left))

{

t = (Vector2)this.transform.position + Vector2.left \* speed;

if (t.x >= start\_x && t.x <= end\_x)

{

can\_move = true;

this.transform.position = t;

GetComponent<Animator>().SetFloat("drix", -0.5f);

GetComponent<Animator>().SetFloat("driy", 0);

}

}

//往右走

else

{

t = (Vector2)this.transform.position + Vector2.right \* speed;

if (t.x >= start\_x && t.x <= end\_x)

{

can\_move = true;

this.transform.position = t;

GetComponent<Animator>().SetFloat("drix", 0.5f);

GetComponent<Animator>().SetFloat("driy", 0);

}

}

}

}

else

{

//循环，直到狼人能走得通（这里是正方形区域，能够保证狼人能走得通）

while (!can\_move)

{

properbility = Random.Range(0,101);

//往上走

if(properbility >= 0 && properbility <= 25)

{

t = (Vector2)this.transform.position + Vector2.up \* speed;

if(t.y>=end\_y && t.y <= start\_y)

{

can\_move = true;

this.transform.position = t;

GetComponent<Animator>().SetFloat("drix", 0);

GetComponent<Animator>().SetFloat("driy", 0.5f);

}

}

//往下走

else if(properbility > 25 && properbility <= 50)

{

t = (Vector2)this.transform.position + Vector2.down \* speed;

if (t.y >= end\_y && t.y <= start\_y)

{

can\_move = true;

this.transform.position = t;

GetComponent<Animator>().SetFloat("drix", 0);

GetComponent<Animator>().SetFloat("driy", -0.5f);

}

}

//往左走

else if (properbility > 50 && properbility <= 75)

{

t = (Vector2)this.transform.position + Vector2.left \* speed;

if (t.x >= start\_x && t.x <= end\_x)

{

can\_move = true;

this.transform.position = t;

GetComponent<Animator>().SetFloat("drix", -0.5f);

GetComponent<Animator>().SetFloat("driy", 0);

}

}

//往右走

else {

t = (Vector2)this.transform.position + Vector2.right \* speed;

if (t.x >= start\_x && t.x <= end\_x)

{

can\_move = true;

this.transform.position = t;

GetComponent<Animator>().SetFloat("drix", 0.5f);

GetComponent<Animator>().SetFloat("driy", 0);

}

}

}

}

}

else

{

timeval = timeval + 1;

}

### 2.测试结果分析

测试一

检查猎人移动



分析：NPC出现瞬移，动画播放过快等问题。

测试二

检查人类移动



分析：NPC出现瞬移，生成时便卡在墙体的问题。

测试三

检查狼人移动



分析：NPC出现瞬移，生成时便卡在墙体的问题。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例名称 | | 游戏内NPC | | |
| 测试类型 | | 单元测试 | | |
| 测试目的 | | 测试在游戏进行中时，观察NPC移动及效果 | | |
| 测试方法 | | 观察NPC移动 | | |
| 测试用例设计原则 | | 条件组合覆盖 | | |
| 正常测试情况 | | | | |
| 编号 | 测试数据 | 数据描述 | 预期结果 | 实际测试结果 |
|  | 猎人 | NPC | 移动正常，动画播放正常，运动逻辑正常 | 出现瞬移、动画播放过快、生成时卡在墙体 |
|  | 村民 | NPC | 移动正常，动画播放正常，运动逻辑正常 | 出现瞬移、动画播放过快、生成时卡在墙体 |
|  | 狼人 | NPC | 移动正常，动画播放正常，运动逻辑正常 | 出现瞬移、动画播放过快、生成时卡在墙体 |
| 测试结论 | |  |  |  |
| 状态 | | NPC出现瞬移、动画播放过快、生成时卡在墙体的问题 | | |