原创动漫游戏造型设计

游戏设计与实现

鲁周民

20141824222

**一、游戏设计部分：**

**1.游戏概述**

像素风格的主人公勇敢的大战一百层地狱的过程。主人公有攻击属性，可以击退从天而降的怪物，他通过踩从下而上的地板来往下走，在游戏过程中会出现四种不同的物资，可以增加或者减弱主人公的生命值和能量。能量决定着跳跃高度。

**2.游戏主要玩法**

主人公通过避免与怪物触碰或者攻击怪物来保护自己的生命值，尽量吃到有积极作用的物资，来下降更多的层数，从而获得更高的分数。

Z键为攻击，左右方向键控制角色左右移动，空格键为跳跃键。

**3.游戏流程**

开始界面

玩法介绍界面



关于我界面



地图选择界面



游戏界面



暂停界面



评分界面



**3.游戏元素**

1.上升的地板



2.怪物



3.掉落物资



**4.游戏特色**

1.有五种不同的地图可以选择，每种地图的背景音乐各不相同，音乐风格与地图的风格相同。

2.角色攻击有动画效果，攻击到怪物有爆炸效果。

3.游戏可以暂停

**二、游戏实现部分**

1.游戏工程目录



**2.开发流程：**

1.制作人物站立动画

2.制作人物奔跑动画

3.制作人物拳击动画

4.制作人物动画控制器，把三种状态联系起来，设置三种状态过渡条件，过渡时间

5.编写障碍物碰撞监测脚本

6.编写物资碰撞监测脚本

7.把碰撞监测脚本整合到主角脚本中，编写主角脚本，让脚本能控制动画控制器

8.绘制血条，在主角脚本中编写血条变化的函数

9.建立游戏场景，给物资和地板的预设贴上图片资源

10.建立主菜单场景，建立主菜单画布，选择地图画布，玩法介绍画布，关于我画布。

11.设置各个按钮的事件。

12.添加音乐音效资源。

13.游戏测试。

**3.脚本与代码**

bg.js(背景移动脚本)

var speed = 1f;

function FixedUpdate () {

if(transform.position.y>=10){

transform.position = Vector3(0,-10,1);

}

transform.Translate(0,speed\*Time.deltaTime,0);

}

BlocksMove.js(障碍物移动脚本)

var MoveSpeed = 0.6;

function Update () {

transform.Translate(0,MoveSpeed\*Time.deltaTime,0);

}

ButtonEvents.js(游戏界面按钮onClick事件)

public var PausePanel:RectTransform;

public var MainPanel:RectTransform;

function Awake (){

}

function pause\_btn () { //暂停按钮事件

Time.timeScale = 0;

PausePanel.anchoredPosition = Vector2(0,0);

}

function play\_btn () { //继续游戏按钮事件

Time.timeScale = 1;

PausePanel.anchoredPosition = Vector2(800f,0);

}

function replay\_btn(){ //重新开始按钮事件

Application.LoadLevel("main");

Time.timeScale = 1;

PausePanel.anchoredPosition = Vector2(800f,0);

}

function menu\_btn(){ //返回主菜单按钮事件

Application.LoadLevel("hello");

}

CreateBlocks.js（产生地板的脚本）

var blocks : Transform[];

private var time = 0f;

var BlocksParent : Transform;

var CreateBlockDeltaTime = 100f;

function Update () {

time++;

if(time==CreateBlockDeltaTime){

var i = Random.Range(0, blocks.Length);

var position = Vector3(Random.Range(-2.0, 2.0),-5,0);

Instantiate(blocks[i],position,transform.rotation);

time = 0;

}

}

CreateGoods.js（产生掉落物资的脚本）

var goods : Transform[];

private var time = 0f;

var CreateGoodsDeltaTime = 500f;

function Update () {

time++;

if(time==CreateGoodsDeltaTime){

var i = Random.Range(0, goods.Length);

var position = Vector3(Random.Range(-2.0, 2.0),Random.Range(-9.0, 9.0),0);

Instantiate(goods[i],position,transform.rotation);

time = 0;

}

}

CreateMaster.js（产生怪物的脚本）

var masters : Transform[];

private var time = 0f;

var CreateMasterDeltaTime = 100f;

function Update () {

time++;

if(time==CreateMasterDeltaTime){

var i = Random.Range(0, masters.Length);

var position = Vector3(Random.Range(-2.0, 2.0),0,0);

Instantiate(masters[i],transform.position+position,transform.rotation);

time = 0;

}

}

MasterMove.js（怪物移动脚本）

public var MasterMoveSpeed = 1f;

private var isLanded = false;

private var Move = ["Left","Right"];

private var i;

function Start () {

i = Random.Range(0, 2);

}

function Update () {

if(isLanded) {

if(Move[i]=="Right")

transform.Translate(MasterMoveSpeed\*Time.deltaTime,0,0);

else

transform.Translate(-MasterMoveSpeed\*Time.deltaTime,0,0);

}

Destroy(gameObject,20);

}

function OnCollisionEnter2D(collision : Collision2D){

if(collision.gameObject.tag!="Master") isLanded = true;

if(collision.gameObject.tag=="Master") {

var theScale = transform.localScale;

theScale.x \*= -1;

transform.localScale = theScale;

}

}

Hero1.js（主角脚本）

#pragma strict

import UnityEngine.UI;

public var speed = 3; //游戏人物运动速度

private var facingRight = true; //生成时是否面向右面

private var run = false; //是否正在奔跑

private var stand = false; //是否站立

private var Ac1 = false; //是否拳击

var ScoresDisplay : Transform; //分数显示

var HealthDisplay : Transform; //生命值显示

var JumpDisplay : Transform; //能量显示

var bgmusic:AudioClip[]; //背景音乐

private var score = 0; //分数

var boom : Transform; //爆炸效果

var anim : Animator; //主角运动控制

var moveForce = 365f; //移动力

var maxSpeed = 5f; //最大速度

var JumpHeight = 300f; //跳跃力

private var health = 100f; //生命值

private var isJumping = false; //是否在跳跃

var rigi : Rigidbody2D; //刚体

var healthBar : SpriteRenderer; //生命条

private var healthScale : Vector3; //生命条的缩放向量

private var lastHitTime = 0f; //最后一次击打怪物的时间

var DestroyLifeDamagePeriod = 0.5f; //持续触碰减血物体时，减血的时间间隔

var DestroyLifeDamage = 20f; //减血数量

var PlusLifeMount = 10f; //加血数量

private var AttackTime = 0f; //攻击时间

var bg :Transform[]; //背景图片

var hitmusic:AudioClip[]; //击打音效

var diemusic:AudioClip; //死亡音效

function Start () {

//获得血条缩放比例

healthScale = healthBar.transform.parent.gameObject.transform.localScale;

//获得背景选择的存档

var i = PlayerPrefs.GetInt("bg");

if(i==-1) i=1;

GameObject.Find("Main Camera").gameObject.GetComponent(AudioSource).clip = bgmusic[i];

GameObject.Find("Main Camera").gameObject.GetComponent(AudioSource).Play();

//根据存档实例化背景

Instantiate(bg[i], Vector3(0,0,1),transform.rotation);

Instantiate(bg[i], Vector3(0,-10,1),transform.rotation);

}

function Update () {

//z按键攻击

if(Input.GetKey(KeyCode.Z)){

anim.SetBool("Ac1",true);

Ac1 = true;

AttackTime = Time.time;

}else

anim.SetBool("Ac1",false);

//左右控制移动

if(Input.GetKey(KeyCode.LeftArrow)||Input.GetKey(KeyCode.RightArrow))

anim.SetBool("Run",true);

else

anim.SetBool("Run",false);

//空格控制跳跃，而且只能跳一次，因为isJumping

if(Input.GetButtonDown("Jump")&&!isJumping) {

rigi.AddForce(new Vector2(0f, JumpHeight));

isJumping = true;

}

if(Ac1&&(Time.time>=AttackTime+1.7f)) Ac1=false;

ScoresDisplay.GetComponent(Text).text = score+"";

HealthDisplay.GetComponent(Text).text = health+"";

JumpDisplay.GetComponent(Text).text = JumpHeight+"";

}

function FixedUpdate () {

//随时间增加加分

score = score+Time.deltaTime\*2;

var h = Input.GetAxis("Horizontal");

//控制人物移动速度

if(h \* rigi.velocity.x < maxSpeed)

//rigi.AddForce(Vector2.right \* h \* moveForce);

transform.Translate(h\*Time.deltaTime\*speed,0,0);

if(Mathf.Abs(rigi.velocity.x) > maxSpeed)

rigi.velocity = new Vector2(Mathf.Sign(rigi.velocity.x) \* maxSpeed, rigi.velocity.y);

//控制人物朝向，向左或者向右

if(h > 0 && !facingRight)

Flip();

else if(h < 0 && facingRight)

Flip();

}

//改变朝向的函数

function Flip ()

{

facingRight = !facingRight;

var theScale = transform.localScale;

theScale.x \*= -1;

transform.localScale = theScale;

}

//改变生命函数，第一个参数如果为true则加生命，如果为false则减生命，第二个参数为加或者减的数量。

function LifeChange(t:int,b){

if(!b)

health -= t;

else {

if(health+t<=100) health += t;

}

if(health<=0) Die();

else{

healthBar.material.color = Color.red;//血条颜色

//血条长度缩放依据生命值

healthBar.transform.parent.gameObject.transform.localScale = new Vector3(healthScale.x \* health \* 0.01f, healthScale.y, healthScale.z);

}

}

function OnCollisionStay2D(collision : Collision2D){

//如果踩到阻力地板，根据人物朝向添加阻力

if(collision.gameObject.tag=="RowBack2Side"){

if(!facingRight)

rigi.AddForce(Vector2.right\*5);

else

rigi.AddForce(Vector2.left\*5);

}

//踩到消失地板，地板1s后消失

if(collision.gameObject.tag=="PushDown"){

Destroy(collision.gameObject,1);

}

//踩到减生命值地板，减少生命值

if(collision.gameObject.tag=="DestroyLife"){

if (Time.time > lastHitTime + DestroyLifeDamagePeriod) {

LifeChange(DestroyLifeDamage,false);

lastHitTime = Time.time;

}

}

}

function OnCollisionEnter2D(collision : Collision2D){

isJumping = false;//去掉跳跃状态

//得分

if(collision.gameObject.tag=="RowBack2Side") score+=6;

if(collision.gameObject.tag=="PushDown") score+=10;

if(collision.gameObject.tag=="DestroyLife") score+=2;

//踩到弹力地板，添加向上弹力

if(collision.gameObject.tag=="PushUp"){

rigi.AddForce(new Vector2(0f, 200f));

score+=1;

}

//踩到加生命的地板，加血

if(collision.gameObject.tag=="LifeBox"){

LifeChange(PlusLifeMount,true);

score+=5;

}

//撞到怪物

if(collision.gameObject.tag=="Master"){

//如果主角处于攻击状态，则消灭怪物并加分 if(Ac1&&collision.gameObject.transform.position.y<=transform.position.y){

AudioSource.PlayClipAtPoint(hitmusic[Random.Range (0, hitmusic.Length)], transform.position);

Instantiate(boom, transform.position,transform.rotation);

Destroy(collision.gameObject);

score+=10;

}else{//否则扣除生命值

LifeChange(DestroyLifeDamage,false);

}

}

}

function OnTriggerEnter2D (other : Collider2D) {

//如果碰撞物体为屏幕底部，则死亡

if(other.gameObject.name=="DownWall"){

Die();

}

//碰到上墙，向下移动两个单位

if(other.gameObject.name=="Wall"){

transform.Translate(0,-2,0);

}

//如果碰到苹果，加血

if(other.gameObject.tag=="Apple"){

LifeChange(5,true);

Destroy(other.gameObject);

}

//如果碰到葡萄，加能量

if(other.gameObject.tag=="Grape"){

if(JumpHeight<=500) JumpHeight +=20f;

Destroy(other.gameObject);

}

//如果碰到胡萝卜，减血

if(other.gameObject.tag=="Carrot"){

LifeChange(5,false);

Destroy(other.gameObject);

}

//如果碰到蘑菇，减能量

if(other.gameObject.tag=="Mushroom"){

if(JumpHeight>=20) JumpHeight -=20f;

Destroy(other.gameObject);

}

}

//死亡函数

function Die(){

AudioSource.PlayClipAtPoint(diemusic,transform.position);

PlayerPrefs.SetInt("Score", score);//游戏存档

Application.LoadLevel("end");

}

**三、开发过程中关键问题及解决方法：**

**1.传统gui按钮位置设置不方便，无法自定义图片；**

解决方法：通过上网查找，了解到gui形式有三种，一是最古老的onGUI,二是公司普遍用的NGUI插件，三是unity4.6之后才有的ugui.所以打算用ugui来建立按钮。

**2.ugui默认画布与场景中物体大小差别悬殊，不兼容，运行后看不到按钮。**

解决方法：在画布设置中把render mode 改为 camera模式，并且选择render camera为场景中的主相机。

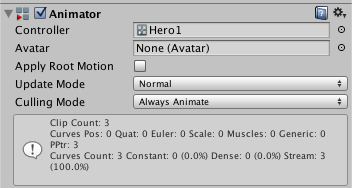
**3.游戏运行时按钮的位置与大小跟随游戏分辨率改变，与设计时的效果差别悬殊。**

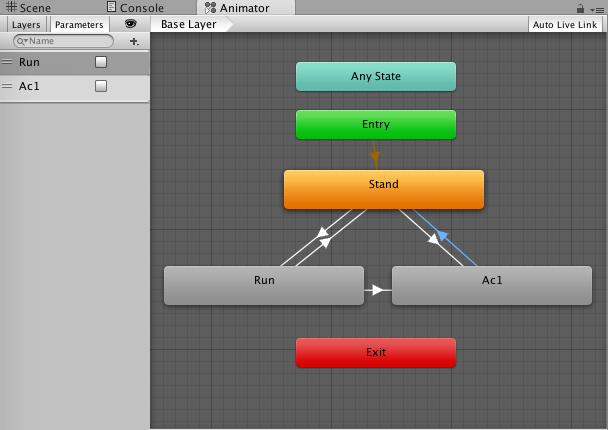
解决方法：在画布设置中把canvas scaler组件中ui scale mode参数改为   
“scale with screen size”。这样设置之后，运行后按钮缩放与设计时相同，能达到自适应分辨率的效果。

**4.unity无法加载gif动画**

解决方法：把gif每一帧都导出来，生成一个图片序列，然后倒入unity ，全选图片序列，并且拖入scene场景中，unity会自动生成动画资源。

**5.想让主角有不同的运动状态，但是不知道怎么实现。**

解决方法：通过上网查找，发现可以给主角添加animator组件，并在该组件中选择一个动画控制器，双击动画控制器，可以看到有任何状态，和默认状态，可以右击空白添加一个跑的状态和攻击的状态。并且设置两个bool类型的变量，来控制状态的转换。只要在脚本中改变该bool变量就能达到状态的转换。

**6.不同场景之间传递参数，来达到选择不同地图就加载不通地图和声音资源。**

解决方法：通过游戏存档可以做到。用playerprefs类，在选择游戏地图的场景中，添加一个游戏存档变量bg，设置为int类型，在第二个场景加载之前读取该变量，通过存储的int数值来读取事先设定好的背景数组，和背景音乐数组就能达到该效果。

用同样的方法，可以做到把游戏场景中的分数传递到结算场景中。

地图选择场景设置存档：



游戏场景读取存档：



**7.背景循环移问题**

解决方法：通过两张背景图片循环播放来达到循环的效果。给两张背景添加向上移动的脚本，当一个背景向上移动出视野范围时，重新设置位置，让他返回屏幕底部，接替屏幕中背景移动。