**Unity射线应用（第三人称Demo制作）**

开发环境

Window7

[Unity3D](http://www.unitymanual.com/)3.4

MB525defy Android 2.2.1

       又一个星期又过去了，写写博客慢慢成了一个习惯吧，可以记点事和交流一下技术还是挺不错的，话说昨天是小假生日，先送上衷心的祝福，希望小假工作顺利，感情美满，身体健康-0- 到目前为止狼人已经到了73级，还算顺利吧，期间得到一些台湾同胞的帮忙，还是很感动的，也许你觉得微不足道的帮助，对于别人来说可能会是很大的支持，也不知道用什么动词好(⊙o⊙)…玩玩魔兽也算故地重游吧，心境变了，人没变而已。从学习Unity到现在差不多一个月了，想想也许有些收获，让我更加接近梦想吧，我还差得很远，最近才知道世界上有个*Global GameJam*，很想去组队参加看看试试自己的实力如何-0- 人的生命能有多长，最后如何看待生命，羽化最近得知二姨爹因为癌症只有半年的时间，羽化想无论如何都要回去看看，一定要回去，一定，一定，一定。

       牢骚发完了，这次推荐一个羽化仿照混乱与秩序制作的一个控制器，包括移动与视角，代码部分送上，资源是公司提供的，所以只送上一个Demo的apk，做得还不成熟，很多地方需要改进，欢迎交流，希望大家喜欢~ ~

本次学习：

1.第三人称视角研究

2.摇杆区域制作

3.视角限制设计

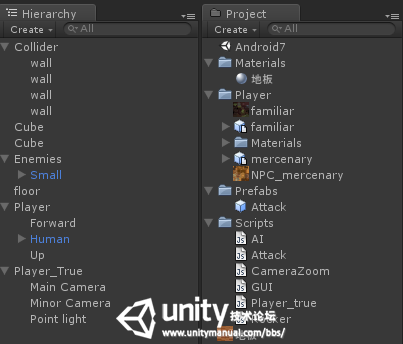
4.简单的战斗部分

**1.第三人称视角研究**

第三人称视角是什么大家玩玩游戏就懂了-0- 羽化不是专业制作人，只是一个玩家，下面是羽化玩家身份的总结出来的一些经验，很多词汇非专业，见谅。。。

1.人物移动 玩过魔兽世界的玩家可能都知道W A S D可以控制人物往8个方向移动，但人的面向不会改变，意味着有左移右移和后退等动作，羽化认为这样设计比较有真实感，而且效果很好，但有些游戏比如神鬼寓言就有所改变，W A S D不仅可以控制移动方向，而且还能改变人的面向，这种设计更加灵活，适合ACT，不适合ARPG，所以羽化这是做的一个ARPG，使用的是魔兽世界的移动标准，摇杆第一次点进区域就可自由移动，原来羽化就是这么想的，看到混乱与秩序的摇杆后觉得更加合理，所以就借鉴了下。

2.视角移动 ARPG也有很多种视角，有斜45°类似火炬之光这种，也有魔兽世界一样360°的，羽化原来做过斜45°的Demo，固定视角可以解决很多问题，但也伴随着更棘手的问题，既然公司要求，所以就做了这个360°的视角转换，包括手势放大缩小。视角改变很容易，但当视角更改以后问题出现了，玩过魔兽世界包括众多自由视角游戏的玩家都知道视角虽然可以随便移动，但存在着很多约束，比如你向上滑动时，视角不可能穿过地面而到地下，一定会沿着人的方向放大，这是前人给我们的经验，比如有一个物体挡在人物和玩家中间时，视角应该拉近，这样就不会产生视觉死角，诸如此类的情况很多，就运用到了射线Ray，进入我们今天的主题。



**1.jpg** (31.51 KB, 下载次数: 0)

[下载附件](http://www.unitymanual.com/bbs/forum.php?mod=attachment&aid=MTExM3w0MmFkYzc3N3wxMzc0ODE2MTc0fDM4N3w0OTU%3D&nothumb=yes)

2013-7-5 10:00 上传

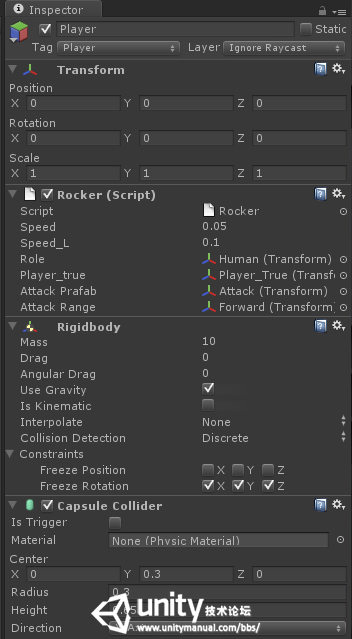
         大家可以看到，羽化这个Demo用的东西也不多，这是前期的一个Demo，还不算完整，如果有机会给大家分享下羽化现在的Demo。

**2.摇杆区域制作**

         触摸屏发展至今，羽化见过最好大的移动摇杆莫过于“混乱与秩序”的移动摇杆了，本身摇杆制作并不困难，主要是学习下这种为玩家带来方便的思维模式，先把代码送上：

Rocker.js：

<font color="#333333"><font face="Arial">//移动速度var Speed : float = 0.05;//视角转动速度var Speed\_L : float = 0.1;//角色var Role : Transform;//人称视角var Player\_true : Transform;//攻击预设var AttackPrafab : Transform;//攻击方向var AttackRange : Transform; //第一点按下的初始点private var PO\_X : int = 0;private var PO\_Y : int = 0;//第二点按下的初始点private var PT\_X : int = 0;private var PT\_Y : int = 0;//判断人是否移动static var PR\_M : boolean = false;//触摸点的记录private var PM\_X : int = 0;private var PM\_Y : int = 0;//第二点按下的初始点private var PS\_X : int = 0;private var PS\_Y : int = 0;//触摸点与中心点的距离private var M\_X : int = 0;private var M\_Y : int = 0;//中心的坐标 private var PC\_X : int = 0;private var PC\_Y : int = 0;//旋转变量private var touchDeltaPosition : Vector2;//两点静态和动态距离private var Distance : int = 0;private var Distance\_D : int = 0; //记录摄像机距离static var Camera\_Record : float = -2.0;//两点的角度static var Angles : int = 0;//战斗焦点static var Focus : boolean = false;//打击点private var hit : RaycastHit;//敌人方位private var Enemie : Vector3;//攻击模式范围private var Range : float = 6.0;//记录时间private var CreationTime : double = -10.0;//处于攻击动作private var Attacking : boolean = false;//Teststatic var Test = 0.0; function Update(){//记录游戏时间 //Test = Time.time;//战斗状态 if(Focus) { if(Vector3.Distance(transform.position,Enemie) > Range) { Focus = false; Camera.main.depth = -1; return; } Camera.main.depth = -3; transform.LookAt(Enemie); Player\_true.transform.LookAt(Enemie); }//攻击预设 if(Time.time > (CreationTime + 0.5) && Role.animation.IsPlaying("Attack") && Attacking) { Instantiate(AttackPrafab, AttackRange.transform.position,Quaternion.identity); Attacking = false; }//单点触摸时 if (Input.touchCount == 1) { Player\_Move(); Attack(); //退出 if(Input.GetTouch(0).position.y > Screen.height - 50 && Input.GetTouch(0).position.x > Screen.width - 50) { Application.Quit(); } }//多点触摸时 else if (Input.touchCount == 2) { Player\_Move(); Player\_Look(); Attack(); }} //角色移动和静态视角function Player\_Move(){ switch(Input.GetTouch(0).phase) { case TouchPhase.Began: PO\_X = Input.GetTouch(0).position.x; PO\_Y = Input.GetTouch(0).position.y; PC\_X = PO\_X; PC\_Y = PO\_Y; var ray = Camera.main.ScreenPointToRay (Input.GetTouch(0).position); if (Physics.Raycast (ray, hit)) { if(hit.collider.gameObject.tag == "Enemies") { Focus = true; Enemie = hit.transform.position; } else if((PO\_X > 200 || PO\_Y > 200) && (PO\_X < Screen.width - 100 || PO\_Y > 100)) { Focus = false; Camera.main.depth = -1; } } break; case TouchPhase.Moved: PM\_X = Input.GetTouch(0).position.x; PM\_Y = Input.GetTouch(0).position.y; if(PO\_X < 200 && PO\_Y < 200 && PO\_X > 0 && PO\_Y > 0) { PR\_M = true; } else if(Input.touchCount == 1) { touchDeltaPosition = Input.GetTouch(0).deltaPosition; //视角限制 if(Player\_true.transform.localEulerAngles.x <= 60 || Player\_true.transform.localEulerAngles.x >= 310) { Player\_true.transform.Rotate(Vector3(touchDeltaPosition.y,touchDeltaPosition.x,0) \* Time.deltaTime \* 100 \* Speed\_L, Space.World); } else if(Player\_true.transform.localEulerAngles.x < 310 && Player\_true.transform.localEulerAngles.x > 200) { Player\_true.transform.localEulerAngles.x = 311; } else if(Player\_true.transform.localEulerAngles.x > 60 && Player\_true.transform.localEulerAngles.x < 200) { Player\_true.transform.localEulerAngles.x = 59; } Player\_true.transform.localEulerAngles.z = 0; } break; case TouchPhase.Ended: PO\_X = 0; PO\_Y = 0; PM\_X = 0; PM\_Y = 0; PC\_X = 0; PC\_Y = 0; PR\_M = false; Role.animation.Stop("Run"); Role.animation.Stop("Back"); Role.animation.Stop("Right"); Role.animation.Stop("Left"); Role.animation.PlayQueued("Wait", QueueMode.CompleteOthers); break; } if(PR\_M) { M\_X = PM\_X - PC\_X; M\_Y = PM\_Y - PC\_Y; //求距离 Distance = Mathf.Sqrt((M\_X \* M\_X) + (M\_Y \* M\_Y)); //求角度 Angles = Mathf.Atan2(M\_X, M\_Y)\* Mathf.Rad2Deg; if(Distance >= 50) { PC\_X = PC\_X\*15/16 + PM\_X/16; PC\_Y = PC\_Y\*15/16 + PM\_Y/16; } //最大速度限制 if(M\_X > 40) { M\_X = 40; } else if(M\_X < -40) { M\_X = -40; } if(M\_Y > 50) { M\_Y = 50; } else if(M\_Y < -30) { M\_Y = -30; } //移动判断和优化 if(Angles>=-45 && Angles <=45) { Role.animation.CrossFade("Run"); if(Angles>=-20 && Angles <=20) { M\_X = 0; } } else if(Angles>=135 || Angles <=-135) { Role.animation.CrossFade("Back"); if(Angles <=-160 || Angles>=160) { M\_X = 0; } } else if(Angles > 45 && Angles < 135) { Role.animation.CrossFade("Right"); if(Angles>=70 && Angles <=110) { M\_Y = 0; } } else if(Angles > -135 && Angles < -45) { Role.animation.CrossFade("Left"); if(Angles>=-110 && Angles <=-70) { M\_Y = 0; } } //转身 transform.localRotation = Quaternion.Euler(0, Player\_true.transform.localEulerAngles.y, 0); //var target = Quaternion.Euler (0, Player\_true.transform.localEulerAngles.y, 0); //transform.rotation = Quaternion.Slerp(transform.rotation, target, Time.deltaTime \* 2); //移动 transform.Translate(M\_X \* Time.deltaTime \* Speed,0, M\_Y \* Time.deltaTime \*Speed); }} //游戏移动视角function Player\_Look(){ switch(Input.GetTouch(1).phase) { case TouchPhase.Began: PT\_X = Input.GetTouch(1).position.x; PT\_Y = Input.GetTouch(1).position.y; M\_X = PO\_X - PT\_X; M\_Y = PO\_Y - PT\_Y; Distance = Mathf.Sqrt((M\_X \* M\_X) + (M\_Y \* M\_Y)); break; case TouchPhase.Moved: PS\_X = Input.GetTouch(1).position.x; PS\_Y = Input.GetTouch(1).position.y; if(PR\_M && !Focus) { touchDeltaPosition = Input.GetTouch(1).deltaPosition; Player\_true.transform.Rotate(Vector3(0,touchDeltaPosition.x,0) \* Time.deltaTime \* 160 \* Speed\_L, Space.World); Player\_true.transform.Rotate(Vector3(touchDeltaPosition.y,0,0) \* Time.deltaTime \* 90 \* Speed\_L, Space.World); Player\_true.transform.localEulerAngles.z = 0; } else if((PO\_X > 200 || PO\_Y > 200) && (PT\_X > 200 || PT\_Y > 200)) { M\_X = PS\_X - PM\_X; M\_Y = PS\_Y - PM\_Y; Distance\_D = Mathf.Sqrt((M\_X \* M\_X) + (M\_Y \* M\_Y)); if(Distance - Distance\_D > 20 && Camera.main.transform.localPosition.z <= 0) { Camera.main.transform.localPosition.z += 0.1; Camera\_Record = Camera.main.transform.localPosition.z; Distance = Distance\_D; } else if(Distance - Distance\_D < -20 && Camera.main.transform.localPosition.z >= -4) { Camera.main.transform.localPosition.z -= 0.1; Camera\_Record = Camera.main.transform.localPosition.z; Distance = Distance\_D; } } break; case TouchPhase.Ended: PT\_X = 0; PT\_Y = 0; PS\_X = 0; PS\_Y = 0; Distance = 0; Distance\_D = 0; break; }} //攻击判断function Attack(){ //普通攻击 if(Input.GetTouch(0).position.y < 100 && Input.GetTouch(0).position.x > Screen.width - 100 && !(Role.animation.IsPlaying("Attack"))) { Role.animation.Play("Attack"); Role.animation.CrossFadeQueued("Wait", 0.3,QueueMode.CompleteOthers); CreationTime = Time.time; Attacking = true; } //移动攻击 else if(PR\_M && Input.GetTouch(1).position.y < 100 && Input.GetTouch(1).position.x > Screen.width - 100 && !(Role.animation.IsPlaying("Attack"))) { PR\_M = false; Role.animation.CrossFadeQueued("Attack", 0.3, QueueMode.PlayNow); Role.animation.CrossFadeQueued("Wait", 0.3, QueueMode.CompleteOthers); CreationTime = Time.time; Attacking = true; }} function OnGUI () { GUI.Box (Rect(0,Screen.height - 200,200,200), "Rocker"); GUI.Box (Rect(Screen.width - 50,0,50,50), "Quit"); GUI.Box (Rect(Screen.width - 100,Screen.height - 100,100,100), "Attack");}</font></font>



**2.jpg** (48.63 KB, 下载次数: 0)

[下载附件](http://www.unitymanual.com/bbs/forum.php?mod=attachment&aid=MTExNHxjZmRjNzRiNnwxMzc0ODE2MTc0fDM4N3w0OTU%3D&nothumb=yes)

2013-7-5 10:01 上传

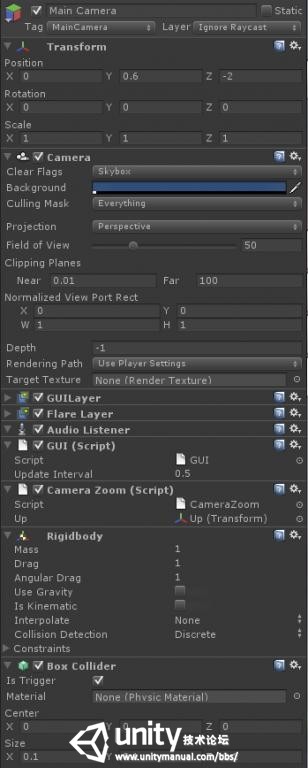
         啊，好长，其实也不多，就300行+，上面有羽化的一些注释，大家可以看看这个脚本，Rocker是绑在Player上的，大家可以从上一张图看到，羽化见了两个Player，一个叫Player\_True的物体其实是一个空物体里面放置了一些基本东西包括光和摄像机，旋转视角的时候就是旋转的Player\_True，其中的好处只有用过的人才能了解吧～　～。羽化使用了预设制作攻击，这是个权益之计，以后说不定会改，这里最多判断了两个点的触控情况，羽化是分别判断的，这是吸取了前一个游戏的经验，这样不会乱，因为移动，攻击，转视角全部如果写在一个判断里面会造成很多冲突，羽化深有体会。

**3.视角限制设计**

         前面部分羽化用不了半天就完成了，后面的视角限制花了将近２天时间，当然个人能力有限也是原因吧。这是我这次博客的重头，如何运用射线，我们主要说说[Physics](http://www.unitymanual.com/bbs/file:/D:/Unity/Editor/Data/Documentation/Documentation/ScriptReference/Physics.html" \t "_blank).Raycast这个脚本的应用，在Unity帮助文档里面提到了，这里羽化尝试了下，最好使用Raycast (origin : [Vector3](http://www.unitymanual.com/bbs/file:/D:/Unity/Editor/Data/Documentation/Documentation/ScriptReference/Vector3.html),direction : [Vector3](http://www.unitymanual.com/bbs/file:/D:/Unity/Editor/Data/Documentation/Documentation/ScriptReference/Vector3.html),out hitInfo : [RaycastHit](http://www.unitymanual.com/bbs/file:/D:/Unity/Editor/Data/Documentation/Documentation/ScriptReference/RaycastHit.html),distance : float = [Mathf.Infinity](http://www.unitymanual.com/bbs/file:/D:/Unity/Editor/Data/Documentation/Documentation/ScriptReference/Mathf.Infinity.html?from=Physics),layerMask : int = kDefaultRaycastLayers)，这个方法，别的不一定靠谱。。。有的地面会莫名其妙穿过去。

CameraZoom.js：

<font color="#333333"><font face="Arial">var Up : Transform;private var Forward : boolean = false;private var Back : boolean = false; function Update () { //通过向下和前后激光判断视角 var layerMask = 1 << 2; layerMask = ~layerMask; var hit : RaycastHit; //向前激光　判断中间阻挡的镜头靠近 if(Physics.Raycast (transform.position, Camera.main.transform.TransformDirection (Vector3.forward), hit, (-Camera.main.transform.localPosition.z))) { Forward = true; var distanceToForward = hit.distance; if(Camera.main.transform.localPosition.z < - 0.5) { Camera.main.transform.localPosition.z += 0.05; } } else { Forward = false; } //向后激光　判断后退时靠墙的镜头拉近 if(Physics.Raycast (transform.position, Camera.main.transform.TransformDirection (Vector3.forward \* (-1)), hit, 0.2,layerMask)) { Back = true; var distanceToBack = hit.distance; if(distanceToBack > 0.1) { Back = false; } } //向下激光　用于判断下拉镜头的拉近 if (Physics.Raycast (transform.position, -Vector3.up, hit , (-2) \* Rocker.Camera\_Record, layerMask)) { var distanceToGround = hit.distance; //print(distanceToGround); } if(distanceToGround < 0.1) { Camera.main.transform.localPosition.z += 0.1; } else if(Camera.main.transform.localPosition.z > Rocker.Camera\_Record && distanceToGround > 0.2 && !Back && !Forward) { Camera.main.transform.localPosition.z -= 0.02; Camera.main.transform.localPosition.y = 0.6; } if(!Rocker.Focus) { transform.LookAt(Up); }} function OnTriggerStay (other : Collider) { if (/\*other.attachedRigidbody &&\*/ Camera.main.transform.localPosition.z < - 0.4) { Camera.main.transform.localPosition.z += 0.02; } Back = true;} function OnTriggerExit (other : Collider) { Back = false;}</font></font>



**3.jpg** (49.58 KB, 下载次数: 0)

[下载附件](http://www.unitymanual.com/bbs/forum.php?mod=attachment&aid=MTExNXwyMzRmYjQyNXwxMzc0ODE2MTc0fDM4N3w0OTU%3D&nothumb=yes)

2013-7-5 10:01 上传

        首先，羽化开始认为只有一个射线就可以搞定了，结果一个射线什么都判断不了，后来羽化仔细想了想就写成了三个射线，加上一个判断，终于效果出现了，话说羽化在摄像机设置Collider，是怕一个不小心，玩家把整个地图看透了-0-　如果你想视角更舒畅点可以把other.attachedRigidbody的注释去掉。这里有个函数layerMask是判断射线穿过那个层的，以后肯定用得上，1<<2代码射线只能穿过前两层。

**4.简单的战斗部分**

       战斗部分很简单，做一个攻击预设，触发攻击动作时就产生预设，在极短时间给怪物造成伤害。绝大多数代码都在移动部分了，这里看看一个简单的AI。

AI.js：

<font color="#333333"><font face="Arial">var range = 4;var Player : Transform;private var health = 10; function Update() { Discover();} function OnTriggerEnter (other : Collider) { if (other.attachedRigidbody) { if(other.gameObject.tag == "Attack") { Rocker.Test = 10.2; transform.Find("Small").animation.CrossFade("Hit"); transform.Find("Small").animation.CrossFadeQueued("Wait", 0.3,QueueMode.CompleteOthers); health--; if(health <= 0) { transform.Find("Small").animation.CrossFade("Dead"); Rocker.Focus = false; Camera.main.depth = -1; transform.tag = "Dead"; } } }} function Discover(){ var layerMask = 1 << 3; layerMask = ~layerMask; if(Vector3.Distance(transform.position,Player.position) > range) { return false; } var hit : RaycastHit; if(Physics.Linecast(transform.position, Player.position,hit,layerMask)) { if(hit.collider.gameObject.tag == "Player" && !(transform.Find("Small").animation.IsPlaying("Dead"))) { transform.LookAt(Player); return true; } else { return false; } } return true;}</font></font>

怪物在一定范围内会发现你，面朝向敌人，简单的AI，可以添加怪物行走等，这就看大家对自己什么要求了，当玩家点击怪物后会进入战斗视角，这时的移动时围绕怪物的，是不是很像Fable~ ~羽化的Demo就初步完成了，最后来张截图。



**4.jpg** (353.99 KB, 下载次数: 0)

[下载附件](http://www.unitymanual.com/bbs/forum.php?mod=attachment&aid=MTExNnw2MmJiMTBjNXwxMzc0ODE2MTc0fDM4N3w0OTU%3D&nothumb=yes)

2013-7-5 10:02 上传

自己认为视角做得很棒，有什么不足还望指出~ ~

作者：羽化  http://blog.csdn.net/libeifs/article/details/6696424