Unity酱模型使用文档

Unity酱 是一个由Unity公司——日本分公司开发出的开放而免费的模型资源，在Unity3D软件的Asset Store里可以免费下载使用。以下的使用文档都是基于Unity3D软件中的编辑和操作，如文中出现错误或者版权问题，请至末尾联系笔者。

引用声明

本文中的文字部分来自笔者对官方Unity酱代码和文档的原创理解，本文实例使用的代码截取自官方DEMO。

Unity酱的导出模型在Inspector中所需添加的基本属性

- **Animator**
- **Rigidbody**
- **CapsuleCollider**

Animator：

- **Controller:** UnitychanLocomotions (在Animators文件夹中可以找到)

Rigidbody：

- **Mass:** 10
- **Angular Drag** 0
- **FreezeRotation:** x=true y=true z=true (true意味着勾选)

CapsuleCollider：

- **Center:** x=0 y=0.75 z=0
- **Radius:** 0.5
- **Height:** 1.5

脚本文件变量

```
public float animSpeed = 1.5f;              // 动画再生速度設定
public float lookSmoother = 3.0f;           // 摄像头平滑移动参数
public bool useCurves = true;               // Mecanim的Curve设定
                                            // 不设置这个开关就无法使用Curve，Curve可以编辑曲线，
                                            // 在这里被用作编辑动画曲线，使得动作更自然
public float useCurvesHeight = 0.5f;        // Curve修正，容易打滑时，加大这个参数

// 以下是角色控制用的参数
// 前進速度
public float forwardSpeed = 7.0f;
// 後退速度
public float backwardSpeed = 2.0f;
// 旋回速度
public float rotateSpeed = 2.0f;
// 起跳威力
public float jumpPower = 3.0f;
// 角色控制（胶囊碰撞器）的参照
private CapsuleCollider col;
private Rigidbody rb;
// 角色控制（胶囊碰撞器）的移動量
private Vector3 velocity;
// 存放CapsuleCollider中Heiht、Center的初期数值
private float orgColHight;
private Vector3 orgVectColCenter;
private Animator anim;
private AnimatorStateInfo currentBaseState;         // 使用base layer、Animator現在状態的参照
private GameObject cameraObject;                    // mainCamera的参照
```

```
//Animator各个状态的参照
static int idleState = Animator.StringToHash("Base Layer.Idle");
static int locoState = Animator.StringToHash("Base Layer.Locomotion");
static int jumpState = Animator.StringToHash("Base Layer.Jump");
static int restState = Animator.StringToHash("Base Layer.Rest");
```

调用方法

在void Start()函数中进行如下初始化：

```
//获取Animator组件
anim = GetComponent<Animator>();
//获取CapsuleCollider和Rigidbody组件
col = GetComponent<CapsuleCollider>();
rb = GetComponent<Rigidbody>();
// CapsuleCollider组件初期的高度和中心
orgColHight = col.height;
orgVectColCenter = col.center;
```

在void Update()函数中进行调用:

```
float h = Input.GetAxis("Horizontal");              // 输入设备水平軸h(键盘A)
float v = Input.GetAxis("Vertical");                // 输入设备水平軸v（键盘D）
anim.SetFloat("Speed", v);                          // Animator设定"Speed"参数
anim.SetFloat("Direction", h);                      // Animator设定“Direction"参数
anim.speed = animSpeed;                             // Animator播放速度，使用animSpeed設定
currentBaseState = anim.GetCurrentAnimatorStateInfo(0); //状态机中使用Base Layer (0)设置为当前状态
rb.useGravity = true;                               //由于有跳跃，所以可以使用重力



// 以下是角色的移动处理
velocity = new Vector3(0, 0, v);        // 根据上下键（键盘W，S）的输入来取得移动偏移量
// 映射到角色的本地空间
velocity = transform.TransformDirection(velocity);
//以下的v的阈值0.1、Mecanim的transition一起调整
if (v > 0.1) {
    velocity *= forwardSpeed;       // 移動速度正
} else if (v < -0.1) {
    velocity *= backwardSpeed;  // 移動速度负
}

if (Input.GetButtonDown("Jump")) {  // 键盘Space空格键

    //Animation的状态在Locomotion的名字要对应
    if (currentBaseState.nameHash == locoState){
        //只有在移动中才可以进行跳跃操作
        if(!anim.IsInTransition(0))
        {
                rb.AddForce(Vector3.up * jumpPower, ForceMode.VelocityChange);
                anim.SetBool("Jump", true);     // Animator中播放跳跃动画
        }
    }
}


// 通过上下键（键盘W,S键）来移动角色
transform.localPosition += velocity * Time.fixedDeltaTime;

// 通过左右键（键盘A,D键）来旋转角色
transform.Rotate(0, h * rotateSpeed, 0);
```

```
// 以下是Animator中各个状态的处理
// Locomotion中
// 当前基础层是locoState的时候
if (currentBaseState.nameHash == locoState){
    //使用curve，要重置collider
    if(useCurves){
        resetCollider();
    }
}
// 跳跃中
// 现在的基础层是jumpState的时候
else if(currentBaseState.nameHash == jumpState)
{
    cameraObject.SendMessage("setCameraPositionJumpView");  // 跳跃中要设置摄像头跳跃视角
    //状态不在transition当中
    if(!anim.IsInTransition(0))
    {

        //以下是使用curve时，对重力的处理。
        if(useCurves){
            // 以下是JUMP00动画相关的Curve JumpHeight和GravityControl
            // JumpHeight:JUMP00中跳跃高度（0～1）
            // GravityControl:1⇒跳跃中（重力无效）、0⇒重力有效
            float jumpHeight = anim.GetFloat("JumpHeight");
            float gravityControl = anim.GetFloat("GravityControl");
            if(gravityControl > 0)
                rb.useGravity = false;  //切断跳跃中重力的影响

            // raycast 投在角色中间
            Ray ray = new Ray(transform.position + Vector3.up, -Vector3.up);
            RaycastHit hitInfo = new RaycastHit();
            // 高度在 useCurvesHeight 之上的时候，将collider的中心调整为JUMP00动画的curve
            if (Physics.Raycast(ray, out hitInfo))
            {
                if (hitInfo.distance > useCurvesHeight)
                {
                    col.height = orgColHight - jumpHeight;          // 调整后的collider高度
                    float adjCenterY = orgVectColCenter.y + jumpHeight;
                    col.center = new Vector3(0, adjCenterY, 0); // 调整后的collider中心
                }
                else{
                    //比阈值低的时候充值collider（以防万一）
                    resetCollider();
                }
            }
        }
        // 设置jump的布尔值（为了不重复播放跳跃动画）
        anim.SetBool("Jump", false);
    }
}
// IDLE处理
// 现在基础层是idleState时
else if (currentBaseState.nameHash == idleState)
{
    //使用curve调整collider时，以防万一重置collider
    if(useCurves){
        resetCollider();
    }
    // 输入jump，重置
    if (Input.GetButtonDown("Jump")) {
        anim.SetBool("Rest", true);
    }
}
// REST 处理
// 现在的状态是restState
```

```
else if (currentBaseState.nameHash == restState)
{
    //cameraObject.SendMessage("setCameraPositionFrontView");       // 摄像头切回正面
    //如果不在移动中，则设置休息状态
    if(!anim.IsInTransition(0))
    {
        anim.SetBool("Rest", false);
    }
}
```

最终的控制脚本

相信有些读者看了以上代码还不懂怎么应用，此时应该去AssetStore中下载Unity酱的资源，打开Example观看源码。下载方法：在搜索框中搜索**Unity chan**

联系方式

个人邮箱：**suyupeng1991@gmail.com**，个人生活博客：♂→_→♂。