1. 在脚本文件中类前添加[AddComponentMenu(“相对路径”)]即可将脚本文件作为一个组件，可供unity在相应的逻辑菜单下添加。
2. 对象的移动与旋转可以通过改变其属性，也可以通过调用相应方法。参数为向量vector
3. Input获取按键时可以在projectsetting中打开标准输入
4. Time.deltaTime一帧所执行的时间：移动或旋转的时候最好也乘上，可以降低设备差异造成的差异。
5. 一个变量，私有级别无法再unity访问，public就可以。或者私有时在前面加上序列化
6. GameObject的静态方法CreatePrimitive、Destroy
7. Shader多色器，告诉显卡怎么去显示纹理。Legacy里的diffuse最经典，漫反射。
8. 导入模型：FBX模型

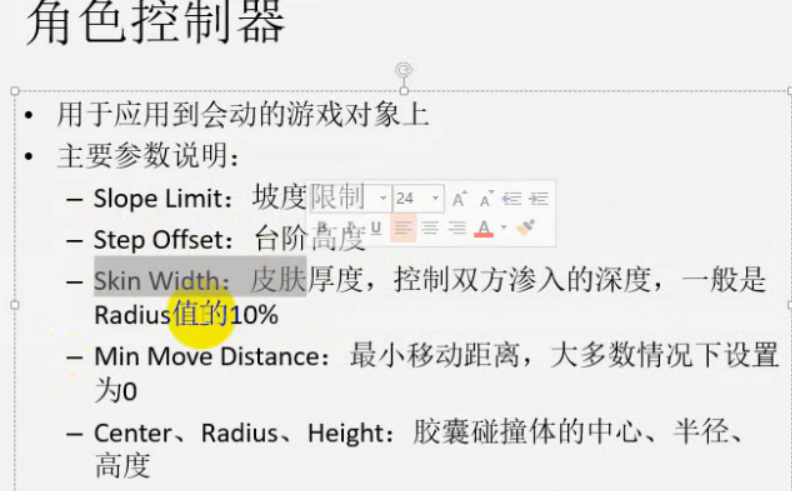
把文件夹拖入之后会自动生成一些必须文件以及模型，点击模型会发现模型有很多动画、总共有999帧，需要哪些动作就通过预览并由自己选出开始于结束的帧数。并在模型中填入开始与结束的帧数，添加剪辑。Loop表示不断循环

打开AnimatorController，并拖动剪辑到其上即可实现对剪辑的操作，实现条件可以在Paramaters中设置。要使条件生效，只需要获得animator对象，并使用setbool等方法实现。

1. 对应8的第二种方法：把模型中的Rig里一个属性由Generic（动画机）改为Legacy

获得animation对象，并调用play方法。或者CrossFade，此方法会加过渡

1. 模型可以导出，否则下次要用还得重新设置，导出格式为unitypackage
2. 范围光需要烘焙：静态物体才需要烘焙
3. 第一人称：只需要把相机拖到游戏对象即可，且相机与游戏对象位置一致。而第三人称也类似，将相机拖到游戏对象，但是相机和游戏对象的位置不同，要将相机正对着游戏对象，即把相机往后挪挪。但是当相机作为子对象时，只能看到一个面，因此最好的办法是相机独立出来，然后写脚本控制移动
4. LateUpdate是一个内置函数，可用来控制摄像机。通过Camera的静态属性main获得摄像机。Vector3的静态方法Lerp可以通过差值取向量。这里的差值越大，延迟越小
5. Terrain地形、Package包（如天空盒子）、粒子系统Particle System
6. 物理引擎Physics：刚体RigidBody（受力效果）、角色控制器Character Controller（发出力的作用）、物体碰撞检测



1. 刚体：AddForce可以给刚体添加力（实际是添加一个方向的位移），ForceMode力的类型。使用velocity属性能给物体赋予速度
2. 碰撞检测：碰撞体Collider，至少有一方是有刚体的。碰撞分三种状态：Enter、Stay、Exit。两种消息类型：触发Trigger——没有力的作用，会穿透物体、非触发Collision
3. 射线Ray：先通过主摄像机获得一条射线-ScreenPointToRay。再通过Physics物理引擎发射射线Raycast，out出一个RaycastHit对象，通过这个对象可以获得碰撞的一个点，还可以通过另一个参数获得指定层的游戏对象（LayerMask）。线渲染器LineRenderer。
4. Instantiate可复制游戏对象。打箱子时会产生很大子弹，而且被打掉的箱子对象还在，浪费资源，此时可以使用事件OnBecameInvisible将游戏对象在相机视野外时清除。
5. 将光标变成指定形状，先将Textures中对应光标图片的Type改为Cursor。代码中获得Texture对象后，通过Cursor类的SetCursor方法即可设置
6. 本地坐标系：transform有三个方向。Vector点出来的是世界坐标系。
7. 音频：aiff、wav——短音效格式，mp3、ogg——长音效音乐。aiff和ogg格式的比较逼真。组件：AudioListener、AudioSource。播放音乐：a）使用AudioSource组件的对象调用Play方法播放 b）使用AudioClip音乐剪辑类，使用AudioSource组件的静态方法PlayClipAtPoint播放，在摄像机处播放是最大声的。
8. 预保存：Playerprefs类提供了键值对的读写，即使关闭游戏也会存储（存储到注册表）。
9. 资源加载：必须把文件保存到Resources文件夹中。使用Resources类，Load加载（路径为Resources文件夹下开始的路径）。不过一般不使用这个方法加载资源，而是用之前学习的那样使用资源。此方法加载FBX模型的话，需要使用复制才能显示出来。
10. 多场景切换：File-Build Setting中添加所有需要呈现的场景，第一个为初始场景。后面的顺序不重要，是通过代码切换的。使用Application类的方法LoadLevel方法