**1、建立自己的类，并给其他类对象使用：**

1）写一个独立的类MyClass;

public class MyClass

{

public static int age;

private string name;

public MyClass(string str)

{

Debug.log(str);

}

public void getName()

{

Debug.log(name);

}

public void string setName(string str)

{

name = str;

}

}

2）在其他脚本中调用我们的类：

MyClass myclass = new MyClass("Its me");

void Start()

{

myclass.setName("I am New!"); //类对象调用类成员函数

myclass.getName();

MyClass.age = 10; //静态成员变量

Debug.log(MyClass.age);

}

**2、获取别的游戏对象，摧毁、实例化一个游戏对象：**

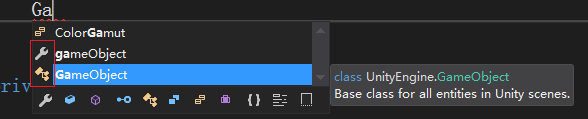
1)摧毁一个游戏对象

Destroy(gameObject);//摧毁自己本身

Destroy(GameObject.Find("Box"));//摧毁名字为Box的物体,Find为静态方法

Destroy(GameObject.Find("Box"),2);//2秒后摧毁

一个是类对象（扳手图标），一个是类（树形图标）



2）实例化（创造）一个游戏对象

public GameObject shot; //将要创建的对象赋予变量

public Transform shotSqawn; //将创建在哪里赋予坐标

Instantiate(shot,shotSqawn.position,shotSqawn.rotation); //调用函数

3）通过对象名字找到此游戏对象

GameObject.Find("xxxx");

4）通过tag找到一个或者多个游戏对象；

游戏对象新建自己的tag，别人可以通过你的tag找到你，以下是在别的游戏对象的脚本中：

GameObject cube = GameObject.FindGameObjectWithTag("mytag");//找到Tag标记为mytag的对象，付给cube  
 void OnGUI()

{

if(GUILayout.Button("CUBE")) // 生成一个CUBER的按钮，如果被按下, RepeatButton可以按住不动

{

Destroy(cube); //销毁 cube对象

}

}

一次tag返回多个tag标记的对象

GameObject cube[];

cube = GameObject.FindGameObjectsWithTag("mytag");

for (int i=0;i<cube.Length;i++)

{

Debug.Log(cube[i].name);

}

**3、代码给一个游戏对象添加和删除组件：**

1）给游戏对象增加一个组件

private void OnGUI()

{

if(GUILayout.Button("ADDCOMP"))

{

gameObject.AddComponent<Lesson2>();//给绑定代码的游戏对象增加一个组件 Lesson2

}

}

2）删除一个游戏对象的组件

if(GUILayout.Button("REMOVE"))

{

Object script = gameObject.GetComponent("Lesson2");

Destroy(script);

}

**4、脚本生命周期结束的时候打印出信息**

void OnDestroy()

{

Debug.Log("OnDestroy");

}

**5、SendMessage函数给其他游戏对象传递消息**

1）在发送消息的脚本中写入gameObject.SendMessage("Function1","1000");

2）其他的需要接收此消息游戏对象的脚本中：

void Function1(string str)

{

Debug.Log(str);

}

**6、脚本中函数的运行次序**

void Awake();

void Start();

void Update();

void LateUpdate();

void FixedUpdate();//设置更新时间在 Edit->Project Settings->Time 默认是0.02秒更新一次

void OnGUI();

void OnDestory();

**7、游戏对象位置和方位的改变（移动和旋转）**

1）改变位置

①

float speed = 5.0f;

transform.Translate(Vector3.forward \* speed \* Time.deltaTime);

②加力的方式使一个物体移动

float power = 100.0f;

GetComponent<Rigidbody>().AddForce(new Vector3(0, 1, 0) \* power \* Time.deltaTime);

③GetComponent<Rigidbody>().velocity = Vector3.forward \* speed \* Time.deltaTime;

④让游戏对象移动到一个坐标

gameObject.transform.position = new Vector3(10,10,10);

2）旋转

①gameObject.transform.Rotate(new Vector3(10, 0, 0));

③让对象旋转到哪个角度

transform.rotation = Quaternion.Euler(0,50,0);

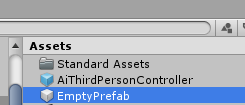
**8、用脚本创建一个立方体，并给这个立方体赋予0的贴图**

GameObject obj = GameObject.CreatePrimitive(PrimitiveType.Cube);   
 obj.name = "123456";

obj.renderer.material.mainTexture = (Texture)Resources.Load("0");

**9、prefabs**

新建一个空的预设，显示为灰色的正方体，将模型拖入预设，编程蓝色的正方体，操作完成。



**10、检测物体的碰撞**

物体的碰撞总是一个物体对另外一个物体，函数是

private void OnCollisionEnter(Collision collision) //另一个与自己碰撞的物体被传进来了

{

if (collision.gameObject.tag == "PickUp")

//按tag和按名字检查碰撞

if (collision.gameObject.name == "Plane")

{

Debug.Log("Hit Plane!");

collision.gameObject.SetActive(false);

}

else

{

Debug.Log("Hit Somethine!");

}

}

**11、输入检查和简单计时器**

void Update () {

if (Input.GetButtonUp("Jump"))

{

Debug.Log("We have hit the spacedd bar!");

}

//myTime每次减去一个固定时间

myTime -= Time.deltaTime;

if (myTime < 0)

{

Debug.Log("Game Over");

}

}

**12、播放声音**

需要注意2点，监听器audio lessoner（是摄像机的一个组件）和音源audio souse

给一个游戏组件添加一个音源组件，给组件赋予一个声音文件就可以播放了。

1）给游戏对象添加Audio Source的组件，将声音文件拖入AudioClip，调用GetComponent<AudioSource>().Play();

2）给游戏对象添加Audio Source的组件，声明一个类对象public AudioClip myClip;在unity界面给my clip赋值，调用函数GetComponent<AudioSource>().PlayOneShot(myClip);

3）声明一个类对象public AudioClip myClip;在unity界面给my clip赋值，调用函数AudioSource.PlayClipAtPoint(myClip, transform.position);（不需要添加Audio Source组件）

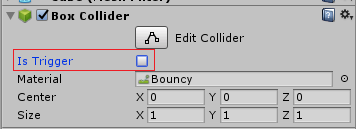
**13、Joints关节，轴类型（Axis），**

添加组件Physics->Hinge Joint将要连接的游戏对象拖入到Connected Body中。

轴为键盘的输入

**14、Trigger 触发器**

物体的碰撞，因为组件Collider碰撞体，如果不想让碰撞检测影响物体移动，但又想检测到碰撞，怎么办



如果勾选Is Trigger，删除物体的Mesh Renderer组件，这样这个物体就看不见了。加入函数：

private void OnTriggerEnter(Collider other)

{

if (other.gameObject.name == "Cube")

{

Debug.Log("Box Went through!");

}

}

**15、GUI**

GUI是展示游戏状态的常用方式

1)GUI Text图形界面文本

2)GUI创建一个按钮

if(GUILayout.Button("CUBE",GUILayout.Height(50),GUILayout.Width(100)))

｛

//如果按下了按钮在这里填入代码

｝

增加一个UI计分板

public Text countText;

GameObject增加UI的Text将此ui拖入到添加了Text代码的组件中

countText.text = "分数：" + count;

修改count的值就可以修改UI的计分板了。

**16、通过控制键盘让一个物体移动**

让一个物体移动：

public float speed;

void FixedUpdate()

{

float H = Input.GetAxis("Horizontal");

float V = Input.GetAxis("Vertical");

Vector3 movement = new Vector3(H,0.0f,V);

GetComponent<Rigidbody>().AddForce(movement \* speed \* Time.deltaTime);

}

检测物品碰撞：

被碰撞的物体设置Tag为PickUp2，运动的物体加入以下代码

void OnTriggerEnter(Collider other)

{

if(other.gameObject.tag == "PickUp2")

{

other.gameObject.SetActive(false);//被碰到的物体消失

count++;

countText.text = "分数：" + count;

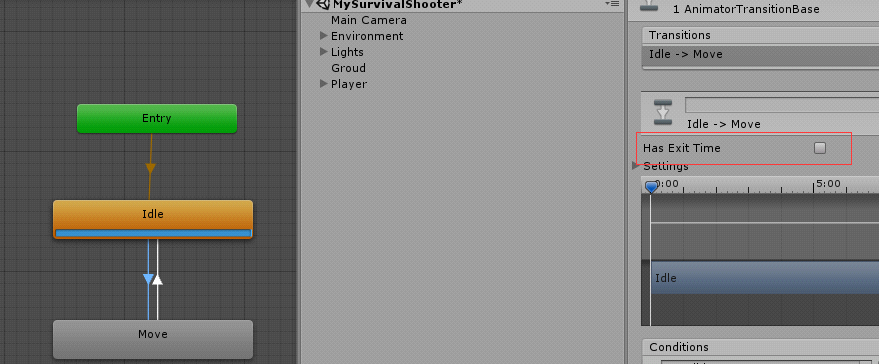
}

}

Gravity重力的方向是沿着Transform.Position的Y轴的无限小方位运动的，把自己想象成为Unity场景中的物体可以更好的理解xyz三轴所表达的意义。  
Transform.Position  
当我们由于地心引力往下掉，那么Y轴减少，相反则Y轴增加  
纵向的改变就好比我们往前走，那么Z轴增加，后退则是Z轴减少  
横向的改变就好比左右的移动，往左即X轴减少，往右X轴增加  
Transform.Rotation  
Y轴表示物体左右旋转  
X 表示为物体向前后旋转  
Z轴表示水平旋转

Survival Shooter Tutorial官方教程2

从Idle状态触发了Move的条件，但是还是在Idle的动画播放，不播放Move



Animation的Has Exit Time复选框需要取消，否则在触发了Move条件后不会退出Idle的动画播放。