Gameobject：游戏物件

Transform：用来描述物体的位置，大小，旋转等等信息，依附于每一个物体。也是当前游戏对象的一个组件

**游戏对象创建与销毁**

1. 直接创建

//创建正方体，可选类型有Cube、Capsule、Cylinder、Plane、Quad、Sphere

GameObject newGameObject = GameObject.CreatePrimitive(PrimitiveType.Cube);

newGameObject.name = "myCube";//赋名

newGameObject.tag = "myCubeTag";//标签

newgameObject.transform.position = new Vector3(0,0,0);//位置

1. 克隆

//克隆对象myGameObject，设置位置、角度

GameObject newGameObject = Instantiate(myGameObject, new Vector3(0,0,0), Quaternion.identity) as GameObject;

//克隆预制体,需手动在Assets文件夹下建立Resources文件夹

GameObject newGameObject = (GameObject)Instantiate(Resources.Load("Prefabs/myGameObject"));

1. 销毁

Destory(myGameObject);

**获取游戏对象**

1. 通过对象名称获取对象

GameObject myGameObject = GameObject.Find("myCube") ;

- 可以查找任何层级的物体

- 可以指定层级或不指定，不指定时若有同名物体，则取列表中最后一个

- 不能查找隐藏物体

GameObject myGameObject = transform.Find("myCube")

- 只能查找到子集物体

- 查找其他层级需要完整路径

- 可以查找隐藏物体

1. 通过标签获取游戏对象

GameObject = myGameObject;

GameObject[] myGameObjects;//声明

myGameObject = GameObjects.FindWithTag（"myCube"）//单个物体

myGameObjects = GameObjects.FindGameObjectsWithTag("myCube")//多个物体,附给数组变量

**改变游戏对象的颜色**

myObject.GetComponent<MeshRenderer>().material.color = new Color(1f, 0.4f, 0f, 0.7f);

myObject.GetComponent<MeshRenderer>().material.color = Color.red;

**游戏对象的平移、旋转、缩放**

1. 平移

myCube.transform.Translate(Vector3.right \* TransSpeed);

myCube.transform.position = new Vector3(-3.0f, -1.0f, 0f);

1. 旋转

//自转

myCube.transform.Rotate(Vector3.forward \* Time.deltaTime \* RotateSpeed);

//公转

myCube.transform.RotateAround(mySphere.transform.position, Vector3.right, RotateSpeed);

//直接到指定角度

myCube.transform.rotation = Quaternion.Euler(0f,1f,0f);

//平滑到指定角度

myCube.transform.rotation = Quaternion.Slerp(myCube.transform.rotation, Quaternion.Euler(0f, 270.0f, 0f), 0.1f);

myCube.transform.rotation = Quaternion.SlerpUnclamped(myCube.transform.rotation, Quaternion.Euler(10.0f, 27.0f, 50.0f), 0.1f);

//调整对象面向另一个物体

myCube.transform.LookAt(mySphere.transform.position);

1. 缩放

myCube.transform.localScale = new Vector3(2.0f, 2.0f, 2.0f);

**MonoBehaviour类**

Unity 3D 脚本中的常用函数如下：

- Update()

正常更新，创建 JavaScript 脚本时默认添加这个方法，每一帧都会由系统调用一次该方法。

- LateUpdate()

推迟更新，此方法在 Update() 方法执行完后调用，每一帧都调用一次。

- FixedUpdate()

置于这个函数中的代码每隔一定时间执行一次。

- Awake()

脚本唤醒，用于脚本的初始化，在脚本生命周期中执行一次。

- Start()

在 Update() 之前、Awake() 之后执行。Start() 函数和 Awake() 函数的不同点在于 Start() 函数仅在脚本启用时执行。

- OnDestroy()

当前脚本销毁时调用。

- OnGUI()

绘制游戏界面的函数，因为每一帧要执行多次，所以一些时间相关的函数要尽量避免直接在该函数内部使用。

- OnCollisionEnter()

当一个游戏对象与另外的游戏对象碰撞时执行这个函数。

- OnMouseDown()

当鼠标在一个载有 GUI 元素（GUI Element）或碰撞器（Collider）的游戏对象上按下时执行该函数。

- OnMouseOver()

当鼠标在一个载有 GUI 元素或碰撞器的游戏对象上经过时执行该函数。

- OnMouseEnter()

鼠标进入物体范围时执行该函数。和 OnMouseOver() 不同，OnMouseEnter() 函数只执行一次。

- OnMouseExit()

鼠标离开物体范围时执行该函数。

- OnMouseUp()

当鼠标释放时执行该函数。

- OnMouseDrag()

按住鼠标拖动对象时执行该函数。