Unity3D

专业的游戏引擎 综合型游戏开发工具 跨平台游戏引擎

应用领域：

游戏开发 VRAR开发 电影 APP等

最新大版本：5.X

特点：

跨平台

更新迭代速度快

第三方技术持开放态度

开发成本更低

开发相对简单 减少开发周期

资源共享 UnityAssetStore

Scene 场景 一个游戏项目可以包含多个场景 Project->NScene

Export导出UnityPackage

Import导入UnityPackage

Transform 变换组件

控制物体的移动Position 旋转rotation和缩放scale

在场景中任何一个物体都必须具有的组件

MeshFilter 网格过滤器组件

决定物体是什么形状 什么样子

简单了解就是模型

Collider 碰撞器组件

一般情况碰撞器和Mesh本身大小保持一致

除非有特殊情况

更多需要的判定是碰撞器是否有接触 不会去判定物体Mesh本身

MeshRenderer 网格渲染器组件

任何一个物体如果需要被渲染 都必须携带Render组件

一个物体可以包含多个组件 组件可以被移除和再次添加 也可以调整顺序

空物体 GameObject

空物体是一种仅仅只包含transform组件的物体

一般用于记录位置 执行脚本 用作父节点等

即 不参与渲染 但是可以执行很多的逻辑的物体

Unity脚本：

```
//默认所有Unity的脚本都继承于Mono
//保证脚本可以像组件一样绑定在游戏物体身上
public class Text : MonoBehaviour {

    //Start方法 在项目开始运行时调用一次
    void Start () {

    }

    //在Start之后 Update每一帧调用一次
    void Update () {
    }
}
```

GameObject 游戏物体类型

Transform 控制物体在场景中的唯一 旋转 缩放等功能

Vector3 向量 用于3维物体的计算

控制物体前后左右平移

```
public class Text : MonoBehaviour {
    //在Start之后 Update每一帧调用一次
    void Update () {
        //当W按键被按下时激活
        if (Input.GetKey(KeyCode.W))
        {
            transform.Translate(new Vector3(0, 0, 0.1f));
        }
        if (Input.GetKey(KeyCode.S))
        {
            transform.Translate(new Vector3(0, 0, -0.1f));
        }
        if (Input.GetKey(KeyCode.A))
        {
            transform.Translate(new Vector3(-0.1f, 0, 0));
        }
        if(Input.GetKey(KeyCode.D))
        {
            transform.Translate(new Vector3(0.1f, 0, 0));
        }
    }
}
```

使用Public 外部赋值

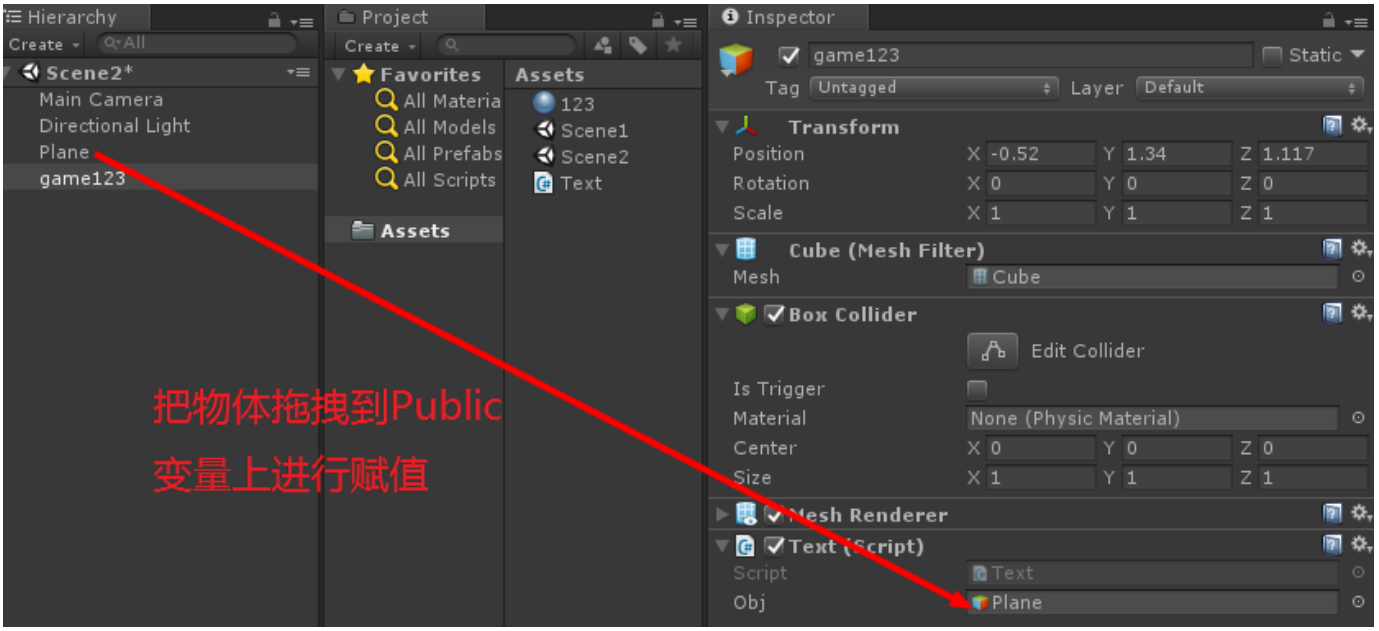
```
public class Text : MonoBehaviour {

    //使用Public定义成员 在编辑器外部进行复制
    public GameObject obj;

    void Start () {
```

```
        //此时obj就是在外部赋值的游戏物体对象
        Debug.Log(obj.name);
    }

    //在Start之后 Update每一帧调用一次
    void Update () {
        //当W按键被按下时激活
        if (Input.GetKey(KeyCode.W))
        {
            //控制obj游戏物体的transform组件的平移
            obj.transform.Translate(new Vector3(0, 0, 0.1f));
        }
    }
}
```



增加旋转

```
    void Update () {

        //当W按键被按下时激活
        if (Input.GetKey(KeyCode.W))
        {
            transform.Translate(new Vector3(0, 0, 0.1f));
        }
        if (Input.GetKey(KeyCode.S))
```

```
        {
            transform.Translate(new Vector3(0, 0, -0.1f));
        }
        if (Input.GetKey(KeyCode.Q))
        {
            transform.Rotate(new Vector3(0,-1,0));
        }
        if (Input.GetKey(KeyCode.E))
        {
            transform.Rotate(new Vector3(0, 1, 0));
        }
    }
```

Vector3 向量

即可以表示向量 又可以表示位置

```
public struct Vector3
{
    public const float kEpsilon = 1e-005f;

    public float x;
    public float y;
    public float z;

    public Vector3(float x, float y);
    public Vector3(float x, float y, float z);

    public static Vector3 operator -(Vector3 a);
    public static Vector3 operator -(Vector3 a, Vector3 b);
    public static bool operator !=(Vector3 lhs, Vector3 rhs);
    public static Vector3 operator *(float d, Vector3 a);
    public static Vector3 operator *(Vector3 a, float d);
    public static Vector3 operator /(Vector3 a, float d);
    public static Vector3 operator +(Vector3 a, Vector3 b);
    public static bool operator ==(Vector3 lhs, Vector3 rhs);

    public static Vector3 back { get; }
    public static Vector3 down { get; }
    public static Vector3 forward { get; }
    [Obsolete("Use Vector3.forward instead.")]
    public static Vector3 fwd { get; }
    public static Vector3 left { get; }
    public float magnitude { get; }
```

零向量

3个维度都为0的向量 0，0，0

```
    void Start () {
        Debug.Log(Vector3.forward);// 0 0  1
        Debug.Log(Vector3.back);// 0 0 -1
        Debug.Log(Vector3.left);// -1 0 0
        Debug.Log(Vector3.right);// 1 0 0
        Debug.Log(Vector3.up);//0 1 0
        Debug.Log(Vector3.down);// 0 -1 0
    }
```

向量的模（长度 大小）

```
        Vector3 v1 = new Vector3(2, 3, 4);
        Debug.Log(v1.magnitude);

        //先得到两个点的相对向量 根据向量的模 求得距离
        Vector3 dir = transform.position - sphereObj.transform.position;
        Debug.Log(dir.magnitude);

        //直接通过静态方法Distance求得两个点的距离 返回float
        float distance=Vector3.Distance(transform.position, sphereObj.transform.position);
        Debug.Log(distance);
```

看向物体 并向物体移动

```
   if (Vector3.Distance(transform.position, sphereObj.transform.position) > 0.1f)
   {
            transform.LookAt(sphereObj.transform);
            transform.Translate(Vector3.forward * 0.1f);
   }
```