单位化向量(标准化向量 法线)

许多向量 只关心方向不考虑大小

“我们面朝什么方向”

使用单位化向量 即大小为1的向量

```
    void Start () {
        Vector3 v1 = new Vector3(100,0,0);
        Vector3 v2= v1.normalized;//不改变自身 获得单位化向量
        v1.Normalize();//改变自身 把自身转化为单位化向量
        Debug.Log(v1);
        Debug.Log(v2);
    }
```

零向量不能被标准化 因为零向量没有方向

当玩家和目标点Y值不同时的移动方式

```
void Update()
    {
        //构建目标点
        //以目标的X和Z轴 自身的Y轴构建目标点
        Vector3 point = new Vector3(target.transform.position.x,
                                    transform.position.y,
                                    target.transform.position.z);
        transform.LookAt(point);
        transform.Translate(Vector3.forward * 0.1f);
    }
```

GameObject 游戏物体

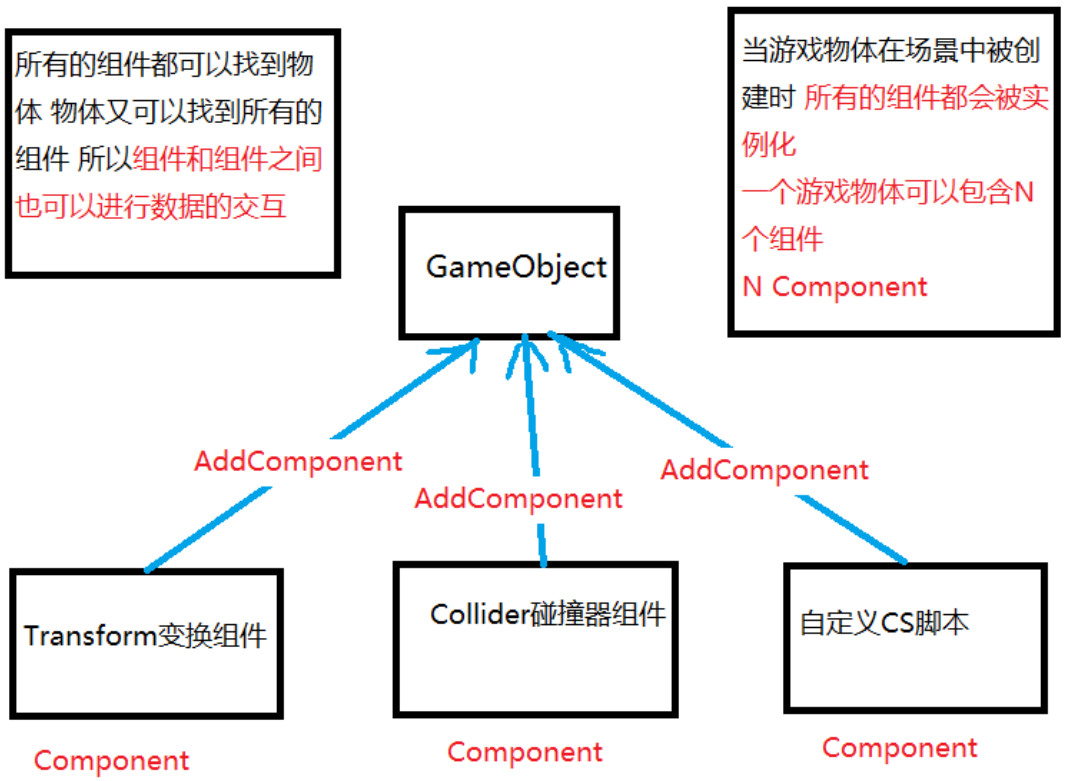
Unity场景中任何一个物体 都可以被看成是空物体 区别在于物体身上绑定的组件

游戏物体的实例化

```
    void Start () {
        //实例化一个空物体 默认名字 new game object
        GameObject obj = new GameObject();
        obj.name = "MyObj";

        //实例化空物体并且修改名字
        GameObject obj1 = new GameObject("MyObj2");
        //实例化空物体并且添加需要的组件
        GameObject obj3 = new GameObject("obj3",
typeof(BoxCollider),typeof(C1));
        //实例化几何体
        GameObject obj4 =
GameObject.CreatePrimitive(PrimitiveType.Cube);
    }
```

Unity所有可以绑定在游戏物体上的类型 都被称为 组件（Component）

Component是所有组件的父类 继承于Component类型才可以绑定在游戏物体上



获取组件：

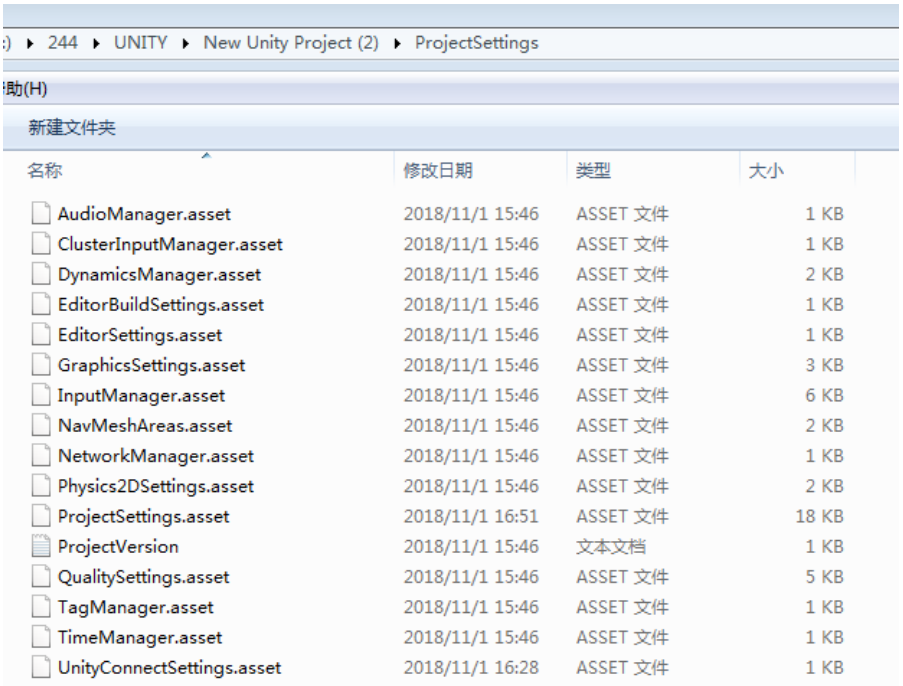
```
void Start () {
        //获取系统组件
        Rigidbody rigid=gameObject.GetComponent<Rigidbody>();
        BoxCollider co = gameObject.GetComponent<BoxCollider>();

        //获取自定义组件
        C1 c = gameObject.GetComponent<C1>();
        c.Fun();

        //获取Transform组件 两种方法一样
        Debug.Log(transform.position);
        Transform trans = gameObject.GetComponent<Transform>();
        Debug.Log(trans.position);
    }

```

Tag标签

游戏中的物体都可以被添加标签

主要目的是用来区分不同类型的物体 把物体有效的归类

标签的数据 存放在ProjectSettings->TagManager.asset文件中




```
    void Start () {
```

```
        //通过游戏物体名字 查找游戏物体
        obj=GameObject.Find("s1");
        Debug.Log(obj.name);
        //使用标签来查找场景中的物体
        GameObject obj2=GameObject.FindGameObjectWithTag("Player");
        //使用标签来查找场景中所有的物体
        GameObject[] objs=
GameObject.FindGameObjectsWithTag("Player");

        transform.GetComponentInChildren<Text>();
    }
```

Prefab 预设(预制体)

在游戏开发过程中会经常涉及到一些复用的重复的游戏物体

Prefab是Unity中最常用到的一种资源类型

具有 加载速度快 体积小 方便克隆等特点

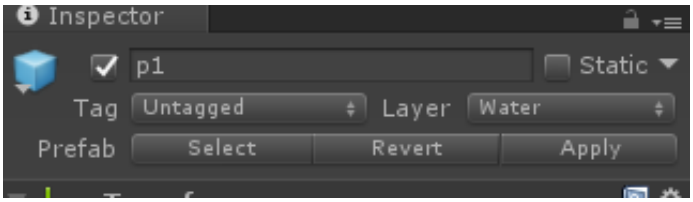
在游戏中频繁动态的创建物体(重复物体)

就是克隆体或者叫模板

游戏中经常会用到的：敌人 子弹 特效 道具等

默认生成的Prefab是一模一样的

往往他们的位置角度等一些属性可以在生成后发生改变



Select: 定位到Prefab源文件

Revert: 取消修改

Apply: 应用修改

特点：

1 可以被置入场景中 也可以在多个场景中使用

2 当一个场景增加一个prefab时 即使实例化一个prefab

3 所有的prefab的实例 都是源prefab的一个克隆体

4 当源prefab发生变化 所有的prefab克隆体都改变

在游戏刚开始运行时生成实例化5个物体(prefab)

```
public class Text : MonoBehaviour {

    public GameObject prefab;
    void Start () {
        for (int i = 0; i < 5; i++)
        {
            GameObject obj=Instantiate(prefab);
            obj.name = "Cube" + i.ToString();
        }
    }
}
```

鼠标点击触发生成游戏物体 并 3秒中后销毁游戏物体

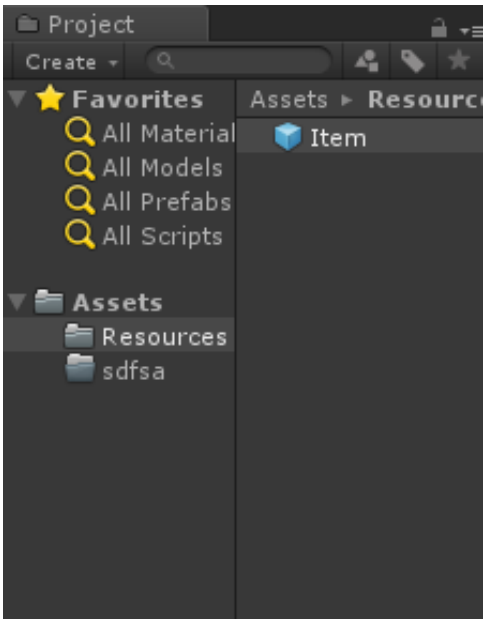
```
     //预制体
    public GameObject prefab;

    // Update is called once per frame
    void Update () {
        //当鼠标左键按下
        if (Input.GetMouseButtonDown(0))
        {
            //参数1 要生成的游戏预制体
            //参数2 生成物体的所在位置
            //参数3 生成物体的旋转值
            GameObject obj=
GameObject.Instantiate(prefab,transform.position,transform.rotation);
            Destroy(obj,3.0f);//3秒以后销毁物体
```

```
            //Destroy（obj） 立即销毁物体
        }
    }
```

直接从文件夹加载Prefab

Resources文件夹的作用： 实现动态加载

Unity可以识别的文件夹之一




```
    void Update () {
        //当鼠标左键按下
        if (Input.GetMouseButtonDown(0))
        {
            //Resources.Load方法 返回Object类型
            //Instantiate方法 当参数是Object类型 返回类型也是Object类型 所以需要 as 强转
            GameObject obj = GameObject.Instantiate(Resources.Load("Item")) as GameObject;
            // GameObject obj = GameObject.Instantiate(Resources.Load<GameObject>("Item"));
        }
    }
```

Unity的时间

Time类

```
void Update () {
        //Unity开始运行到目前的时间
        Debug.Log(Time.time);

        //一帧所消耗的时间
        Debug.Log(Time.deltaTime);
    }
```

每3秒生成一个预设

```
    private float time;
    void Update () {
        //每一帧 都累加一帧的时间
        time  += Time.deltaTime;
        //当时间大于3秒时 进行实例化
        if (time > 3.0f)
        {
            GameObject obj =
GameObject.Instantiate(Resources.Load("Item")) as GameObject;
            //把计时器归零
            time = 0f;
        }
    }
```

制作3个游戏物体的预设

当鼠标按下 在3个不同的位置

随机生成一个预设的实例

第一步 在场景中创建三个预设 存放在Resources文件夹

第三步：代码实现

```
public class Spwan : MonoBehaviour {
    //生成物体的3个位置点
    private GameObject[] points;
    private float time;
    void Start()
    {
       points=GameObject.FindGameObjectsWithTag("Point");
    }

    void Update () {
        time += Time.deltaTime;
        if (time>1.0f)
        {
            time = 0f;
            //随机得到预设的名字
            int index = Random.Range(1, 4);
            string path = "Item" + index.ToString();

            //随机得到点数组的索引
            int arrayIndex = Random.Range(0, points.Length);

            Object obj= GameObject.Instantiate(Resources.Load(path),

points[arrayIndex].transform.position,
```

```
24
points[arrayIndex].transform.rotation);
25              Destroy(obj, 3.0f);
26          }
27      }
28  }
```