Unity3D物体移动

内容提要：

1、Vector3向量
2、transform对象与移动
3、世界坐标系与局部坐标系
4、刚体与力系统

大数据与物联网学院　邵亮

移动的前提：认识向量

向量的概念

1、向量的几何意义

向量是有大小和方向的线段。

案例：

男生上课线路：$\overrightarrow{BA}$　向北，走281米；

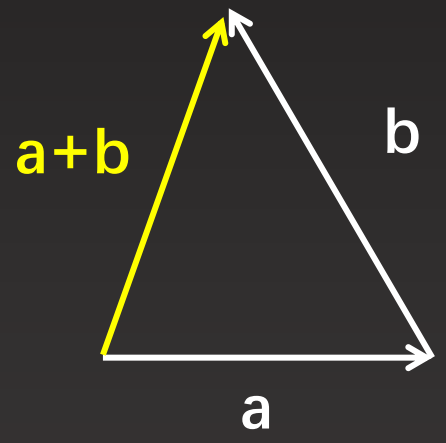
女生上课线路：$\overrightarrow{CA}$　向东北，走314米；

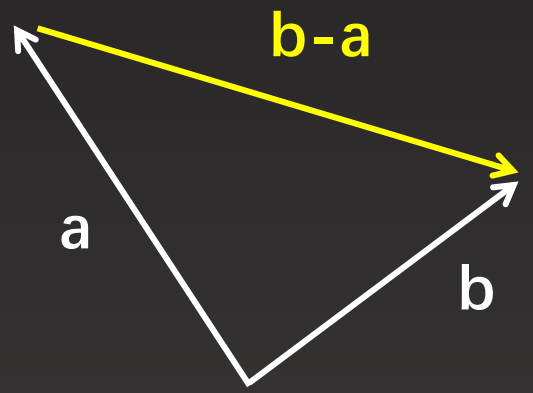
点与向量的区别：

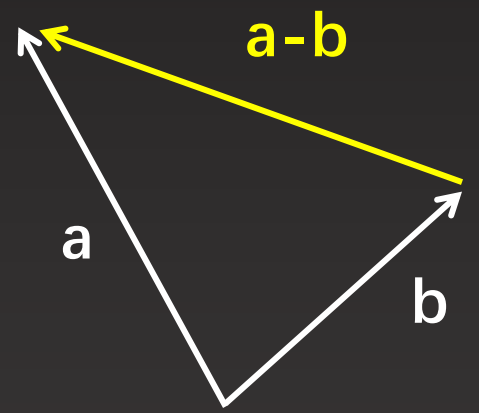
- “点”有位置，但没有大小和方向；
- “向量”有大小和方向，但没有位置；
- “点”描述位置，“向量”描述位移。

2、向量的运算

(1)向量的加减



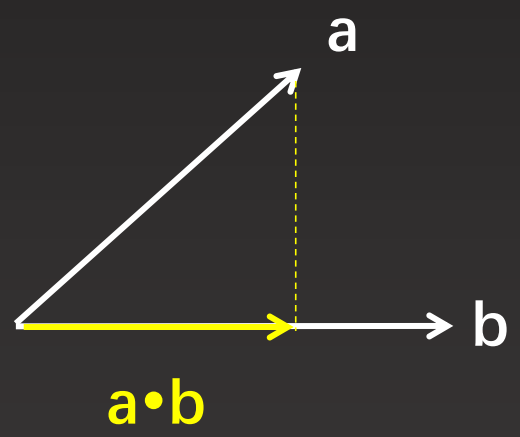




2、向量的运算

(2)向量的乘法运算

向量的点乘，算投影

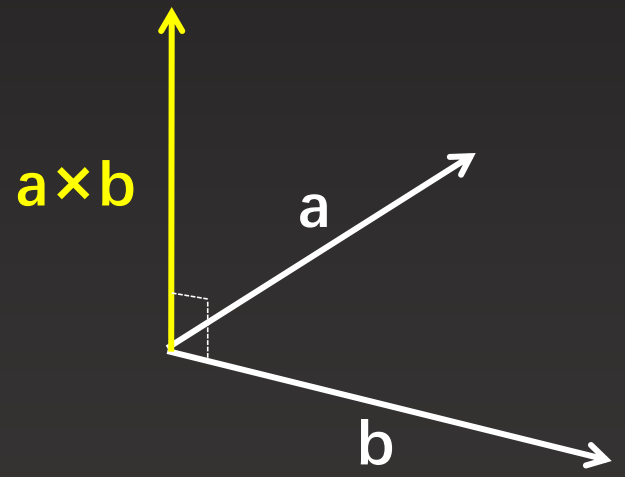


Unity3D代码表达：

```
//判断敌人与玩家的距离
float dot = Vector3.Dot((敌人位置 - 玩家位置).normalized, 玩家前方);
```

参考文献：https://blog.csdn.net/qq_46324811/article/details/128765739

向量的叉乘，算法线



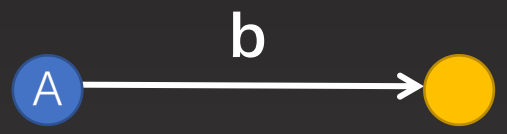
Unity3D代码表达：

```
//判断两个物体的左右关系
Vector3 cross = Vector3.Cross(目标物体距离, 当前物体前方);
 if (cross.y > 0)
{   Debug.Log("目标物体在当前物体的左方");    }
else if (cross.y < 0)
 {   Debug.Log("目标物体在当前物体右方");       }
```

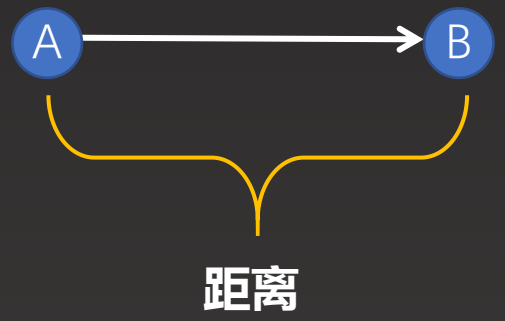
2、向量的运算

(3)向量与点的运算

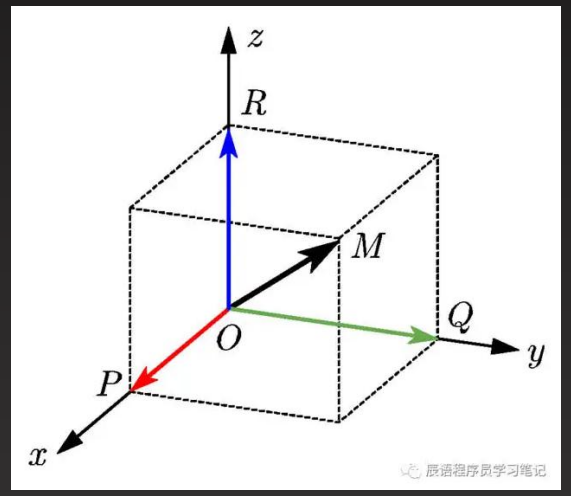
点（位置）+ 向量 = 新点 （新位置）



点（位置）- 点（位置）= 向量（距离）



3、三维向量



向量的大小就是向量各分量平方和的平方根
设三维向量OM(x,y,z),则OM向量的大小为：|OM|= √(OP^2 +OQ^2 +OR^2)

简化： |OM|= √(X^2 +Y^2 +Z^2)

Unity3D代码表达：

```
//创建向量
Vector3 pos;

// 开平方计算
Mathf.Sqrt(Mathf.Pow(pos.x, 2) + Mathf.Pow(pos.y, 2) + Mathf.Pow(pos.z, 2));
```

Unity3D中的向量：Vector3类/对象

属性/方法	说明
x	向量的x轴分量
y	向量的y轴分量
z	向量的z轴分量
magnitude	三维向量长度 （无方向，表距离）
normalized	标准化向量 （有方向，长度为1）

Vector3类

```
Vector3  d=new Vector3();

d.x=1f;
d.y=1f;
d.z=0f;
```

Vector3(x,y,z);
Vector3(1f,1f,0f);

Vector3对象

Vector3.forward
Vector3.right
Vector3.back
Vector3.up

非刚体的移动（坐标）

非刚体的移动

变型对象

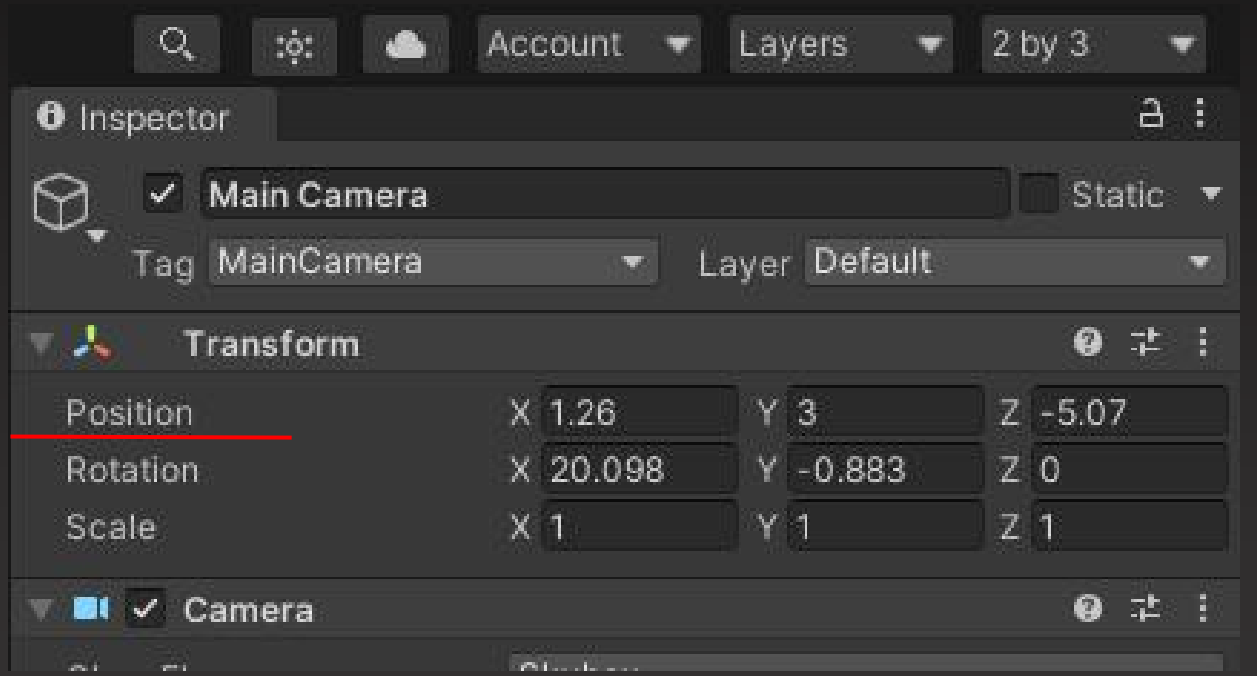
transform

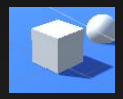
位置对象

向量（Vector3）

position

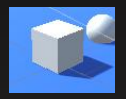
- **x**
- **y**
- **z**
- **magnitude** 物体距原点的距离





物体平移：moveCube

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class moveCube : MonoBehaviour
{
    float offsetZ;
    void Start()
    {       }

    void Update()
    {

       offsetZ=0.01f;

       transform.position+=new Vector3(0f,0f,offsetZ);

    }
}
```



物体平移（限制边界）：moveCube

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class moveCube : MonoBehaviour
{
    float offsetX;
    float offsetZ;
    void Start()
    {       }

    void Update()
    {

        offsetX=0.01f;
        offsetZ=0.01f;
        if(transform.position.magnitude<6)
        { transform.position+=new Vector3(offsetX,0,offsetZ);}

    }
}
```

案例：荒漠飞车

功能要求：

1、上下键控制车辆前后移动

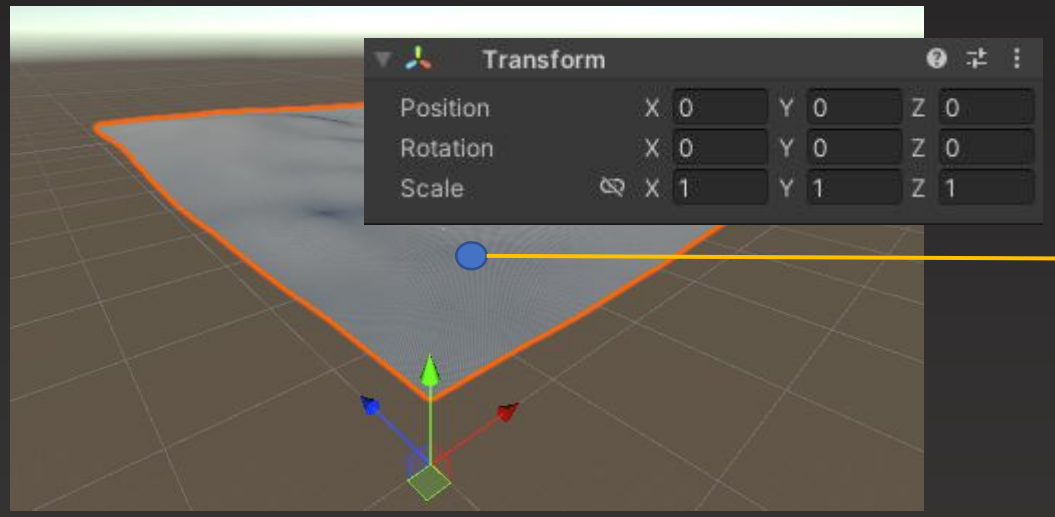
2、左右键控制车辆左右转向

扩展提醒：

transform.**Rotate()**

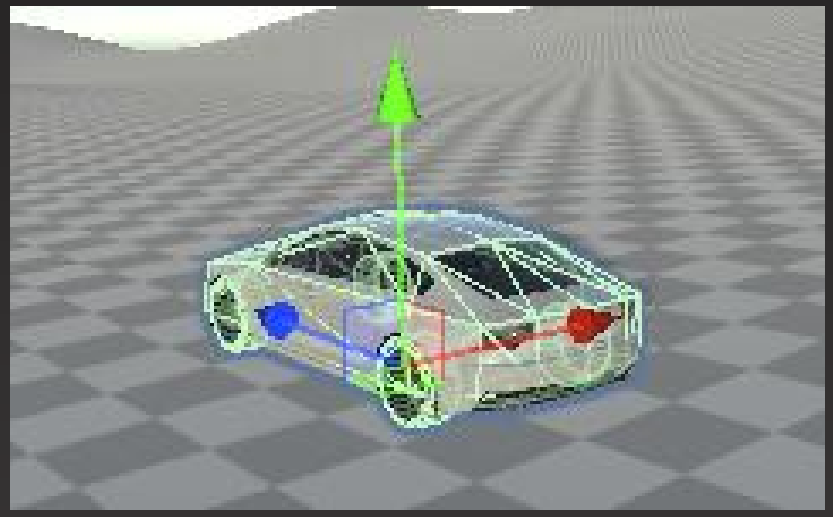
transform对象的不同坐标系应用

世界坐标系



子对象：position
角度对象(四元数)：rotation

局部坐标系

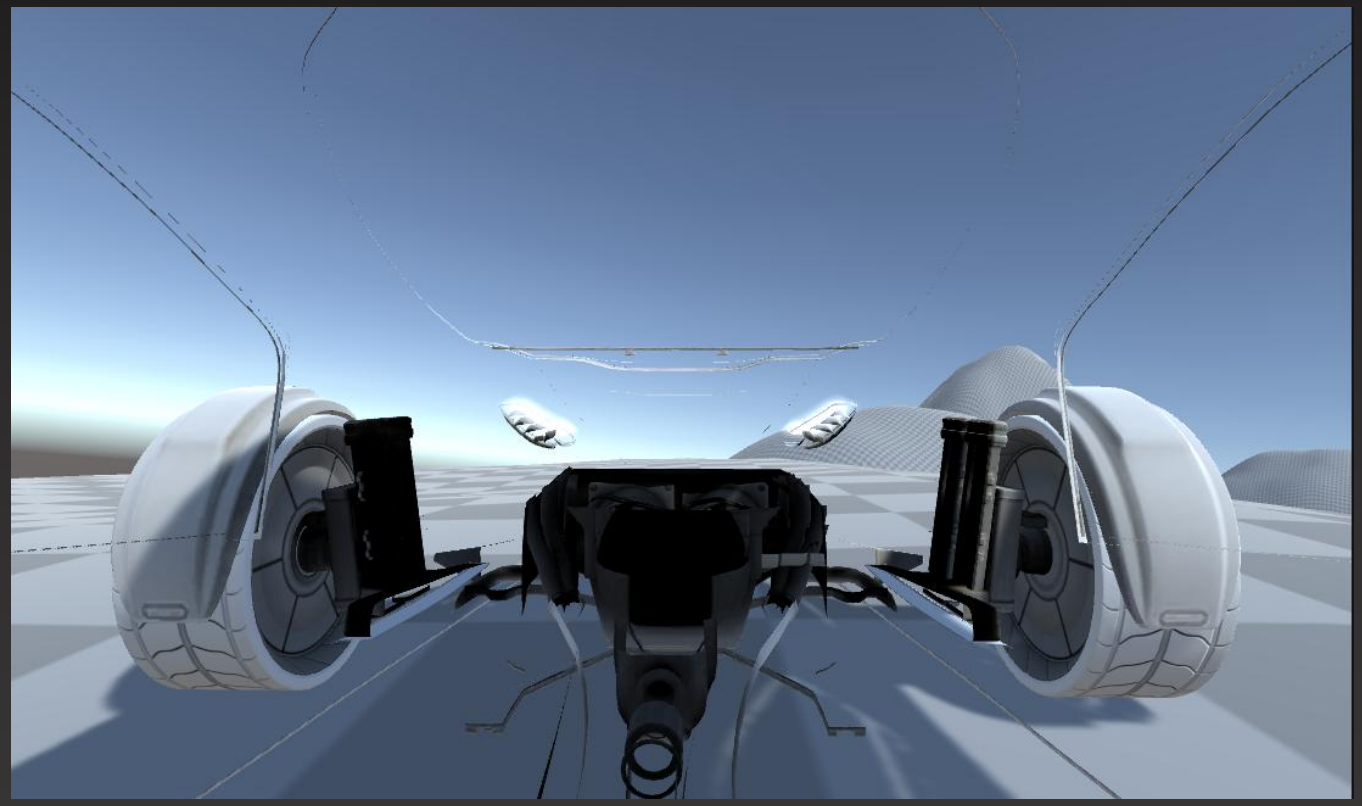


位置方法：Translate();
角度方法：Rotate();

代码部分

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class moveBox : MonoBehaviour
{
    float offsetX, offsetZ;
    int moveSpeed = 10;
    int rotateSpeed = 80;
    void Start()
   {   }

    void Update()
    {
        offsetX = Input.GetAxis("Horizontal") * Time.deltaTime * rotateSpeed;
        offsetZ = Input.GetAxis("Vertical") * Time.deltaTime * moveSpeed;
        transform.Translate(0f, 0f, offsetZ);
        if (offsetZ != 0) { transform.Rotate(0f, offsetX, 0f); }
    }
}
```

思考题：



如何让车轮转动？

1.向前，正转；
2.向后，反转。

思考题解答

```
public class moveBox : MonoBehaviour
{
    float offsetX, offsetZ,wheelSpeed;
    int moveSpeed = 10;
    int rotateSpeed = 80;
    GameObject leftWheel,rightWheel;
    void Start()
    {
        leftWheel=GameObject.Find("SkyCarWheelFrontLeft");
        rightWheel=GameObject.Find("SkyCarWheelFrontRight");
    }

    void Update()
    {
        offsetX = Input.GetAxis("H") * Time.deltaTime * rotateSpeed;
        offsetZ = Input.GetAxis("V") * Time.deltaTime * moveSpeed;
        wheelSpeed=Input.GetAxis("V")*5;
        transform.Translate(0f, 0f, offsetZ);
        if (offsetZ != 0) { transform.Rotate(0f, offsetX, 0f); }

        leftWheel.transform.Rotate(wheelSpeed,0f,0f);
        rightWheel.transform.Rotate(wheelSpeed,0f,0f);
    }
}
```

获取对象

GameObject．Find(对象名)

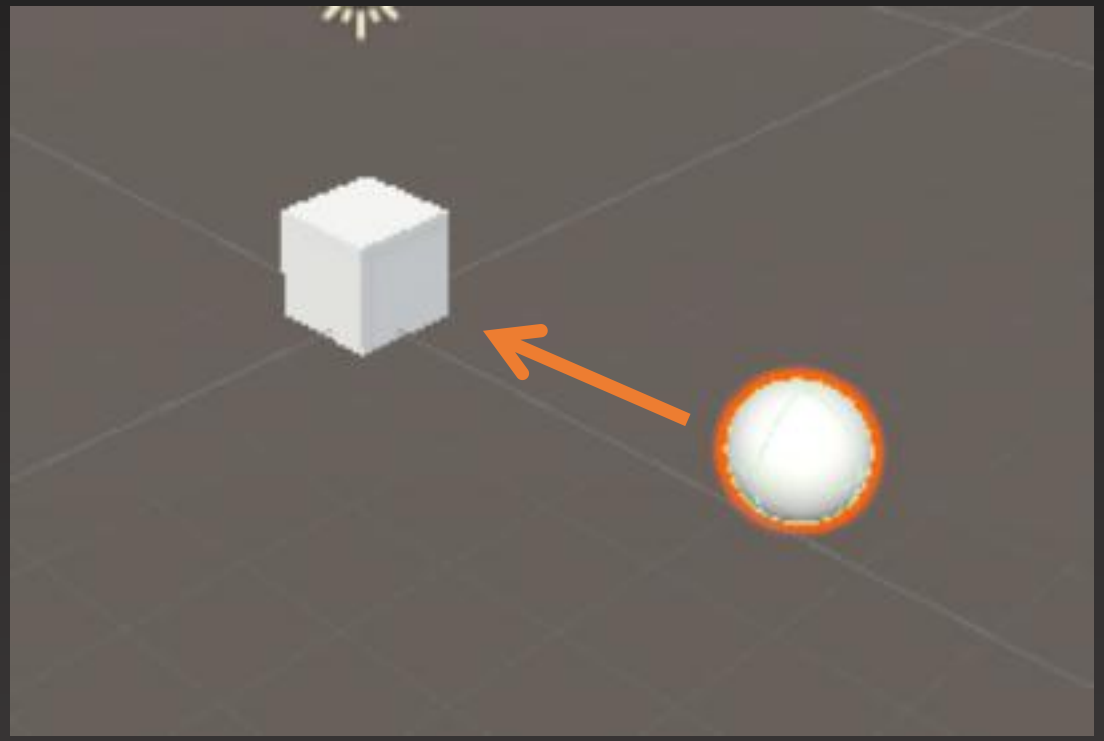
案例：缓进跟踪

功能要求：

1、WASD键控制立方体前后左右移动；

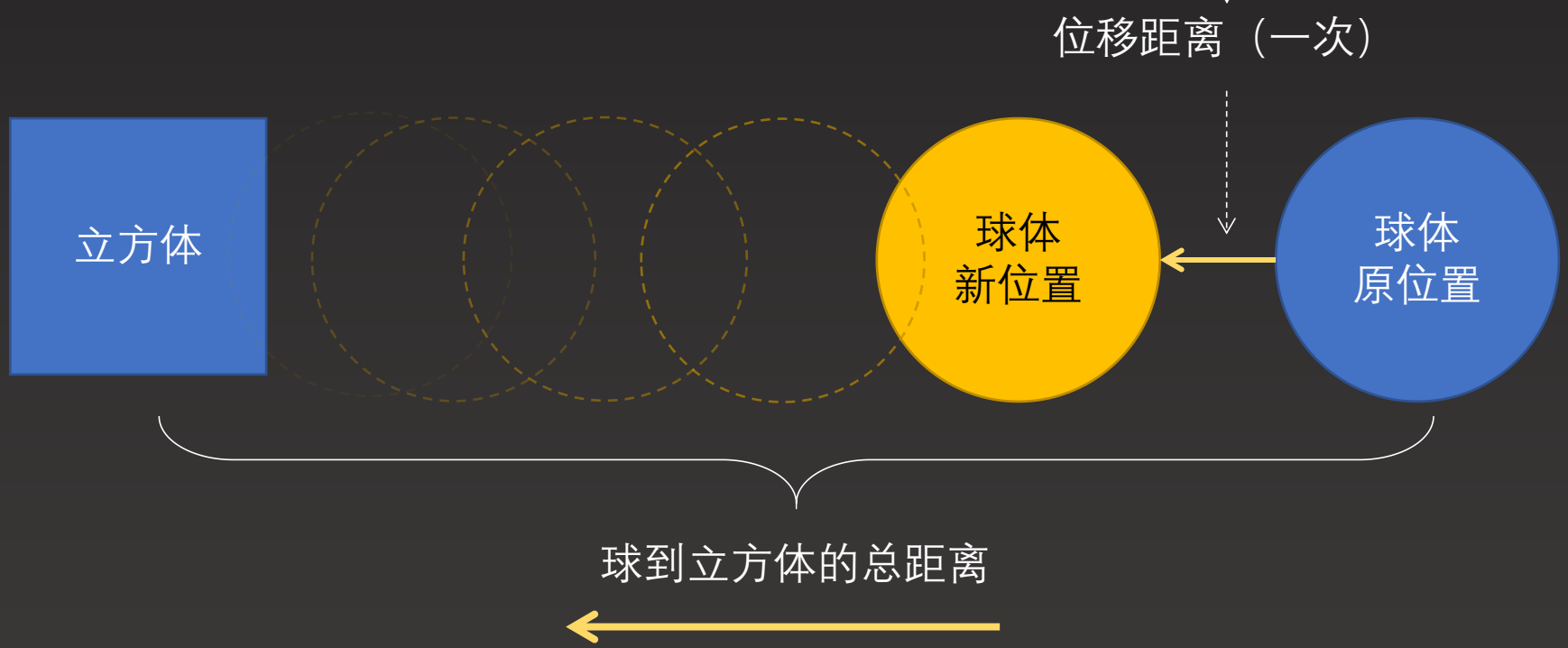
2、立方体移动后，球体缓慢跟近；

3、球体自动吸附到立方体位置。



核心算法：

球.新位置=球.原位置+总距离/常量



代码参考

立方体代码：

```
public class boxMove : MonoBehaviour
{
    public bool moving=false;
    float xoffset,zoffset;
    int speed=10;
    void Start()
    {


    }
    void Update()
    {
       zoffset=Input.GetAxis("Vertical")*Time.deltaTime*speed;
       xoffset=Input.GetAxis("Horizontal")*Time.deltaTime*speed;

       this.transform.position+=new Vector3(xoffset,0f,zoffset);
       if(zoffset!=0 || xoffset!=0)
       { moving=true;}
       else
       { moving=false;}

    }

}
```

球体代码：

```
public class sphere : MonoBehaviour
{
    GameObject box;
    boxMove bm;
    Vector3 offset, d;
    void Start()
    {
        box = GameObject.Find("Cube");
        bm=box.GetComponent<boxMove>();

    }

    void Update()
    {
        d = box.transform.position - this.transform.position;

        if (d.magnitude > 0.15f || bm.moving)
        {
            this.transform.position += d / 100;
        }
        else
        {
            this.transform.position=box.transform.position;
        }
    }
}
```

刚体的移动（力）

基础：Rigidbody组件

使用对象受物理力学系统影响，具有力学特性。

常用项目：
Mass：质量；
Drag：阻力；
Angular Drag：角阻力；
Use Gravity：受重力影响；
Is Kinematic：是否为运动学物体。

刚体操作的基础代码

1、定义刚体对象（变量）。

2、获取刚体组件。

3、对刚体施加力。

```
Rigidbody rb;
void Start()
{
    rb = this.GetComponent<Rigidbody>();
}
void Update()
{
    if (Input.GetKey(KeyCode.LeftArrow))
    { rb.AddForce(Vector3.Up * 10f); }

 }
```

刚体对象常用属性/方法

属性/方法	功能
AddForce()	施加力（世界坐标系）
AddRelativeForce()	施加力（局部坐标系）
AddTorque()	施加扭力（世界坐标系）
AddRelativeTorque()	施加扭力（局部坐标系）
AddForceAtPosition()	在指定位置施加力
Mass	设置对象质量
Drag	设置对象阻力
angularDrag	设置对象角阻力
centerOfMass	设置对象重中点位置
velocity	对象移动速度
angularVelocity	对象角速度

案例：撞球游戏

黄色球为母球，将其它白色球撞出绿色球台。

功能要求：

1、WASD键控制施加力的方向并击发球；

2、数字键控制力的大小。