



```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  //玩家控制脚本
6  //负责控制玩家的移动 检测碰撞 统计游戏分数
7  public enum Pos//位置枚举
8  {
9      Left,
10     Right,
11     Middle
12 }
13 public class PlayerControl : MonoBehaviour {
14     public Transform left;
15     public Transform right;
16     public Transform middle;
17
18     private Pos currentPos;
19     private int score;
20
21     void Start () {
22         transform.position = middle.position;
23         currentPos = Pos.Middle;
24     }
```

```
25
26 void Update () {
27
28     if (Input.GetKeyDown(KeyCode.A))
29     {
30         if (currentPos == Pos.Middle)
31         {
32             transform.position = left.position;
33             currentPos = Pos.Left;
34         }
35         else if (currentPos == Pos.Right)
36         {
37             transform.position = middle.position;
38             currentPos = Pos.Middle;
39         }
40     }
41     else if (Input.GetKeyDown(KeyCode.D))
42     {
43         if (currentPos == Pos.Middle)
44         {
45             transform.position = right.position;
46             currentPos = Pos.Right;
47         } else if (currentPos == Pos.Left)
48         {
49             transform.position = middle.position;
50             currentPos = Pos.Middle;
51         }
52     }
53 }
54
55 void OnTriggerEnter(Collider collider)
56 {
57     if (collider.gameObject.CompareTag("Item"))
58     {
59         //立即销毁
60         Destroy(collider.gameObject);
61         score++;
62         Debug.Log("分数: " + score);
63     }
64 }
65 }
66
```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 //游戏控制脚本
6 //负责控制在不同位置随机生成道具 控制生成的速度
7 public class GameControl : MonoBehaviour {
8     private GameObject[] points;
9     private float time;
10    private int itemCount;
11    private float speed = 3.0f;
12    void Start () {
13        points=GameObject.FindGameObjectsWithTag("Point");
14    }
15
16    void Update () {
17        time += Time.deltaTime;
18        if (time > 0.5f)
19        {
20            time = 0f;
21            itemCount++;
22
23            string path = "Item" + Random.Range(1, 4).ToString();
24
25            GameObject
26            obj=GameObject.Instantiate(Resources.Load(path)) as GameObject;
27
28            obj.transform.position = points[Random.Range(0,
29            points.Length)].transform.position;
30
31            if (itemCount % 5 == 0)
32            {
33                speed += 2.0f;
34            }
35
36            obj.GetComponent<ItemMove>().speed = speed;
37        }
38    }
39 }
```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 //道具移动脚本
6 //负责控制道具向前方移动
7 public class ItemMove : MonoBehaviour {
8     //道具移动速度
9     public float speed;
10    void Update () {
11        transform.Translate(Vector3.forward * speed * Time.deltaTime);
12    }
13 }
```

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 //死亡区域的控制脚本
6 //负责记录所有Miss的道具个数
7 public class MissControl : MonoBehaviour {
8     private int missCount;
9     void OnTriggerEnter(Collider collider)
10    {
11        if (collider.gameObject.CompareTag("Item"))
12        {
13            Destroy(collider.gameObject);
14            missCount++;
15            Debug.Log("Miss+ " + missCount);
16        }
17    }
18 }
```

## Camera 摄像机



### ClearFlags 清除标识

决定屏幕中哪些部分数据会被清除 一般多台摄像机存在时使用 控制不同的渲染对象

SkyBox 天空盒

在屏幕的空白处显示当前摄像机的天空盒

SolidColor

当没有天空盒的情况下 将默认显示此处设置的背景颜色

### DepthOnly

相机深度 用于多摄像机同时存在 渲染多个叠加效果 例如 画中画

DontClear

不清除颜色和缓存

每一帧效果都会叠加在下一帧上

### CullingMask

剔除遮罩 根据对象的层(layer)来控制渲染的对象

### Projection 投影方式

分为正交和透视模式

perspective 透视 近大远小 完全透视的方式

FieldofView 视野范围

Orthographic 正交 没有透视感 均匀渲染所有对象的方式

Size 视野大小

ClippingPlanes 裁剪平面

近平面和远平面 决定了我们摄像机能照射到的世界大小

ViewPort Rect 视图矩形

4个值 (0~1)

X 水平起始点

Y 垂直起始点

W 宽度

H 高度

Depth 深度

用于控制摄像机的渲染顺序 数值大的摄像机将渲染在数值小的摄像机之上

RenderingPath 渲染路径

UsePlayerSetting 使用PlayerSetting的全局设置

顶点光照 Vertex Lit 将所有的对象作为顶点光照来进行渲染

正向渲染 Forward 顶点光照

延时光照 DeferredLighting 多次渲染 buffer数据

HDR优化 高动态光照渲染

NSAA优化 抗锯齿

Unity3D的四种坐标系

## 1 WorldSpace (世界坐标系)

transform.position 这就是世界坐标系

## 2 ScreenSpace (屏幕坐标系)

屏幕坐标系以像素定义

左下角为0 0点

右上角为Screen.Width Screen.Height

```
Debug.Log(Screen.width);
```

```
Debug.Log(Screen.height);
```

Z值是以相机在世界单位的Z值来衡量

获取鼠标在屏幕中的位置(屏幕坐标)

```
1 void Update () {  
2     Debug.Log(Input.mousePosition);  
3 }
```

## 3 ViewPort Space 视口坐标系

左下角为0 0点 右上角为1 1点

## 4 绘制GUI界面的坐标系 UI Space

屏幕左上角为0 0点 右下角为Screen.Width Screen.Height

## 坐标系之间的转换

```
1 //世界坐标系 转 屏幕坐标系  
2 Camera.main.WorldToScreenPoint(transform.position);  
3  
4 //屏幕坐标系 转 视口坐标系  
5 Camera.main.ScreenToViewportPoint(Input.GetTouch(0).position);  
6  
7 //视口坐标系 转 屏幕坐标系  
8 Camera.main.ViewportToScreenPoint(new Vector3(0, 1, 0));
```

```
9
10 //视口坐标系 转 世界坐标系
11 Camera.main.ViewportToWorldPoint(new Vector3(0, 1, 0));
```

## 射线 Ray

射线是在三维世界 从一个点沿着一个方向发射的一条无限长或固定的线

射线在发射的轨迹上 一旦与接受碰撞的游戏对象发生碰撞 将停止发射

射线可以用来实现子弹击中目标 鼠标点击物体等功能

## 射线的重要参数

origin 在世界坐标下 射线的的起始点

direction 射线的方向

hitInfo 射线碰撞后返回的碰撞信息

distance 射线的长度

layermask 层

## 判断是否点击到一个游戏物体

```
1         if (Input.GetMouseButtonDown(0))
2         {
3             Ray ray=
Camera.main.ScreenPointToRay(Input.mousePosition);
4
5             //存储碰撞信息的对象
6             RaycastHit hit;
7
8             Debug.DrawLine(ray.origin, ray.direction * 100,
Color.yellow);
9
10            Physics.Raycast(ray, out hit, 100);
11
12            //hit.collider为空 说明没有和任何物体产生碰撞
```



```

13         if (hit.collider == null)
14             return;
15
16         if (hit.collider.CompareTag("Player"))
17         {
18             Debug.Log("点击到了Cube");
19         }
20     }
21

```

## 一个物体往前方发射线

```

1         //实例化一条射线
2         Ray ray = new Ray();
3         //设置射线起始位置
4         ray.origin = transform.position;
5         //设置射线方向
6         ray.direction = transform.forward;
7
8         RaycastHit hit;
9         Physics.Raycast(ray, out hit, 50);
10        Debug.DrawLine(ray.origin, ray.direction * 50, Color.yellow);

```

## RayCastHit

```

1     Debug.Log(hit.rigidbody);
2     Debug.Log(hit.collider);
3     //碰撞器到射线起点的距离
4     Debug.Log(hit.distance);
5     //碰撞点
6     Debug.Log(hit.point);
7     //法线
8     Debug.Log(hit.normal);

```

## 点击地面控制人物移动

```

1 public class Text : MonoBehaviour {
2
3     private Vector3 target;
4     private bool move;
5
6     void Update()
7     {
8         if (Input.GetMouseButtonDown(0))
9         {
10             Ray ray =
Camera.main.ScreenPointToRay(Input.mousePosition);
11
12             RaycastHit hit;
13
14             if (Physics.Raycast(ray, out hit, 100))
15             {
16                 if (hit.collider.CompareTag("terrain"))
17                 {
18                     target = new
Vector3(hit.point.x,transform.position.y,hit.point.z);
19                     move = true;
20                 }
21             }
22         }
23
24         if (move && Vector3.Distance(transform.position, target) >
0.1f)
25         {
26             transform.LookAt(target);
27             transform.Translate(Vector3.forward * 3.0f *
Time.deltaTime);
28         }
29         else
30             move = false;
31     }
32 }

```

拖拽3D世界的物体

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Text : MonoBehaviour {
6
7      bool getObj;
8      Vector3 startPoint;
9      void Update()
10     {
11         if (Input.GetMouseButtonDown(0))
12         {
13             MouseButtonDown();
14         }
15
16         if (Input.GetMouseButton(0)&&getObj)
17         {
18             MouseButton();
19         }
20         if (Input.GetMouseButtonUp(0))
21         {
22             getObj = false;
23             transform.position = startPoint;
24         }
25     }
26
27
28     void MouseButton()
29     {
30         UnityEngine.Ray ray =
Camera.main.ScreenPointToRay(Input.mousePosition);
31         RaycastHit hit;
32         Physics.Raycast(ray, out hit, 100);
33         if (hit.collider == null)
34             return;
35
36         if (hit.collider.CompareTag("Player"))
37         {
38             transform.position = new Vector3(hit.point.x, hit.point.y,
transform.position.z);
39         }

```

```

40     }
41
42     void MouseButtonDown()
43     {
44         UnityEngine.Ray ray =
Camera.main.ScreenPointToRay(Input.mousePosition);
45         RaycastHit hit;
46         Physics.Raycast(ray, out hit, 100);
47         if (hit.collider == null)
48             return;
49
50         if (hit.collider.CompareTag("Player"))
51         {
52             getObj = true;
53             startPoint = transform.position;
54         }
55     }
56 }

```

## 两个物体发射线并渲染

```

1  public class Text : MonoBehaviour {
2
3      public Transform sphere;
4      private LineRenderer line;
5
6      void Start()
7      {
8          line = GetComponent<LineRenderer>();
9      }
10     void Update()
11     {
12         Ray ray = new Ray();
13         ray.origin = transform.position;
14         ray.direction = sphere.position - transform.position;
15         ray.direction.Normalize();
16
17         RaycastHit hit;
18         Physics.Raycast(ray, out hit, 10f);
19         Debug.DrawRay(ray.origin, ray.direction*100f, Color.yellow);

```

```

20
21     line.positionCount = 2;
22     line.SetPosition(0, transform.position);
23     line.SetPosition(1, sphere.position);
24 }
25 }

```

## 使用Layer过滤碰撞信息

```

1     if (Input.GetMouseButtonDown(0))
2     {
3         Ray ray =
Camera.main.ScreenPointToRay(Input.mousePosition);
4
5         RaycastHit hit;
6
7         //使用Player层来过滤碰撞 只检测Player层的碰撞 过滤其他所有层
的碰撞
8         //不能值传递Player层的编号 而是传递Layer的值 使用左移运算
9         Physics.Raycast(ray, out hit,
100, 1<<LayerMask.NameToLayer("Player"));
10
11         if (hit.collider == null)
12             return;
13
14         Debug.Log(hit.transform.name);
15     }

```

## \*射线点击 实现人物的爬坡