在 Unity 中，使用 StartCoroutine(string methodName)和 StartCoroutine(IEnumeratorroutine)都可以开启一个协程。

在 Unity 中，使用 StopCoroutine(stringmethodName)来终止该 MonoBehaviour 指定方法名的一个协同程序，使用 StopAllCoroutines()来终止所有该 MonoBehaviour 可以终止的协同程序。

协同函数代码如图 12-20 所示。

```
using UnityEngine;
using System.Collections;

public class Example12_4_5 : MonoBehaviour {

    void Start()
    {
        print("Starting " +Time.time);
        StartCoroutine(WaitAndPrint(2));
        print("Done " +Time.time);
    }

    IEnumerator WaitAndPrint(float waitTime)

    {
        yield return new WaitForSeconds(waitTime);
        print("WaitAndPrint " + Time.time);
    }

    // Update is called once per frame
    void Update () {

    }
}

```

图 12-20　协同函数

测试结果如图 12-21 所示。

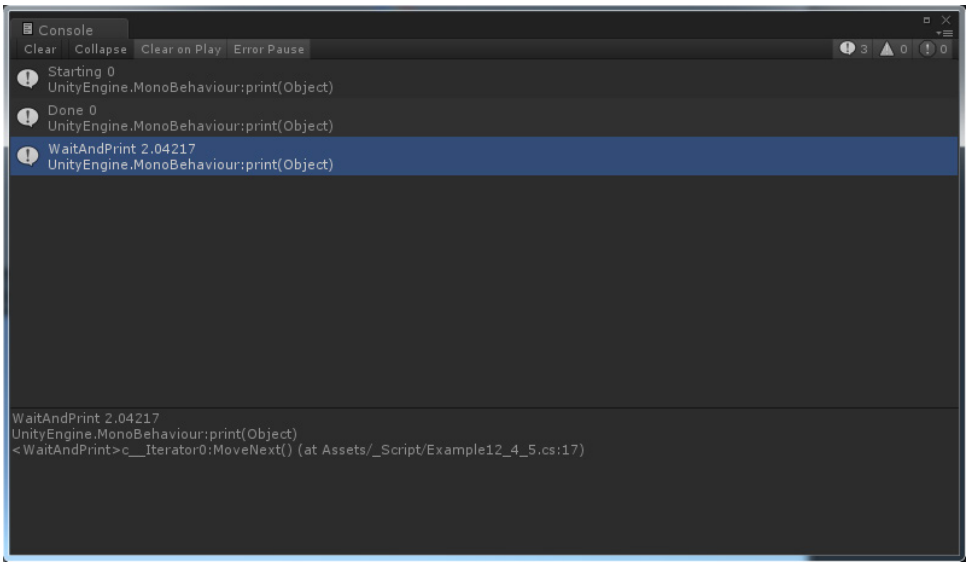

图 12-21　协同函数输出结果

12.5 拓展训练——游戏实例

本章前面 4 节分别从各个部分讲解了 C#脚本语言的开发基础，在本部分中，将通过实例编写 C#脚本来实现“旋转 Cube”的形式对 Unity C#脚本怎么应用进行综合性的训练。具体操作方式如下。

1. 启动 Unity，新建项目 12_5，保存场景为 Test 12_5。禁用 Hierarchy 视图中默认创建 Directional Light，单击 Hierarchy 菜单栏下的 Create 选项，创建一个 Cube、Plane 和 Spotlight，通过调整物体的位置和角度，然后通过给 Cube 和 Plane 添加材质，制作出一个简易的场景，如图 12-22 所示。

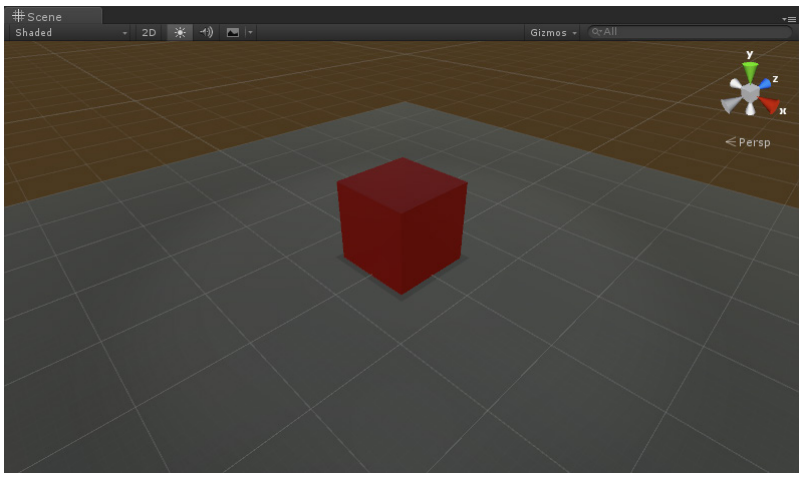

图 12-22　创建好的简易场景

2. 在 Assets 目录下，创建文件夹（单击 Create→Folder 选项），用于存放游戏的各种资源：材质、脚本、关卡等。

3. 双击进入_Scripts 文件夹，创建一个名为 CubeRotate 的 C#脚本（如果脚本名称和内部的类名称不同，一定要更正），如图 12-23 所示。

4. 因为我们要控制 Cube 的旋转和颜色变换，所以把脚本 CubeRotate 拖放到场景的方块上（或者先选择方块，将脚本拖到方块的属性栏），调整好相机位置。

5. 双击打开脚本，在脚本中加入鼠标相关函数，这里需要用到 OnMouseOver、OnMouseExit、OnMouseDown（此类特殊函数不会有智能拼写出现），如图 12-24 所示。

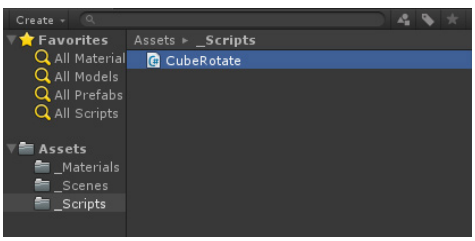

图 12-23　创建 C#脚本 BoxRotate

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class CubeRotate : MonoBehaviour {
5
6     void Start () {
7
8     }
9
10    void Update () {
11
12    }
13
14    void OnMouseOver()
15    {
16
17    }
18    void OnMouseExit()
19    {
20
21    }
22    void OnMouseDoWn()
23    {
24
25    }
26
27 }
28
```

图 12-24　在脚本 BoxRotate 中添加鼠标函数

6. 当鼠标光标移动到物体上时，物体材质色彩变为黄色，同时将一个初始值为假的布尔变量 1 的值取真；当鼠标光标离开后，物体材质色彩还原，布尔变量 1 为假；当按下鼠标左键，且布尔变量 1 的值为真，布尔变量 2 的值为真，如图 12-25 所示。

OnMouse 函数都是执行一次的函数，因此不能将与动画有关的控制函数放于其内执行，通常会使用布尔值开关来控制 Update 函数中的动画函数。

```
using UnityEngine;
using System.Collections;

public class CubeRotate : MonoBehaviour {

    private bool bCube1 = false;
    private bool bCube2 = false;
    private Color OldColor;

    void Start () {
        //获取物体的原始颜色
        OldColor = this.gameObject.GetComponent<Renderer> ().material.color;
    }

    void Update () {

    }

    void OnMouseOver()
    {
        this.gameObject.GetComponent<Renderer> ().material.color = Color.yellow;
        bCube1 = true;
    }
    void OnMouseExit()
    {
        this.gameObject.GetComponent<Renderer> ().material.color = OldColor;
        bCube1 = false;
    }
    void OnMouseDoWn()
    {
        if(bCube1)
        {
            bCube2 = true;
        }
    }

}


```

图 12-25　丰富鼠标函数

7. 当鼠标键按下，且 bCube2 为真时，Cube 转动。因为 Cube 转动是持续性的，所以把脚本写在 Update 函数里面，在 Update 函数里面实现 Cube 转动，如图 12-26 所示。

8. 因为脚本 CubeRotate 只作用于其所依附的物体，想要控制 Spotlight，我们有两个选择：第一，创建新的脚本；第二，使用查找物体函数。很显然，我们没必要为这么简单的功能加入一个新的脚本。因此使用函数 GameObjec.Find()获取 Spotlight 的强度属性，如图 12-27 所示。

9. UGUI 的使用。要在游戏视图显示各种 UI 信息，需要使用 UI 组件。使用 UGUI，首先需要在 Hierarchy 中创建 UI 组件，这里使用 Text 组件，如图 12-28 所示。

```
using UnityEngine;
using System.Collections;

public class CubeRotate : MonoBehaviour {

    private bool bCube1 = false;
    private bool bCube2 = false;
    private Color OldColor;

    void Start () {
        //获取物体的原始颜色
        OldColor = this.gameObject.GetComponent<Renderer> ().material.color;
    }

    void Update () {
        if(bCube2)
        {
            this.gameObject.transform.Rotate(Vector3.up*Time.deltaTime*200);
        }
    }

    void OnMouseOver()
    {
        this.gameObject.GetComponent<Renderer> ().material.color = Color.yellow;
        bCube1 = true;
    }
    void OnMouseExit()
    {
        this.gameObject.GetComponent<Renderer> ().material.color = OldColor;
        bCube1 = false;
    }
    void OnMouseDown()
    {
        if(bCube1)
        {
            bCube2 = true;
        }
    }

}

```

图 12-26　在 Update 函数里实现 Cube 转动

```
using UnityEngine;
using System.Collections;

public class CubeRotate : MonoBehaviour {

    private bool bCube1 = false;
    private bool bCube2 = false;
    private Color OldColor;

    void Start () {
        //获取物体的原始颜色
        OldColor = this.gameObject.GetComponent<Renderer> ().material.color;
        GameObject.Find ("Spotlight").GetComponent<Light> ().intensity = 1.0F;
    }

    void Update () {
        if(bCube2)
        {
            this.gameObject.transform.Rotate(Vector3.up*Time.deltaTime*200);
            if(GameObject.Find ("Spotlight").GetComponent<Light> ().intensity <8.0F)
            {
                GameObject.Find ("Spotlight").GetComponent<Light> ().intensity +=0.05F;
            }
        }
    }

```

图 12-27　更改 Spotlight 的强度

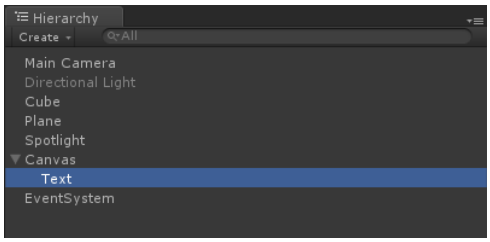

图 12-28　添加 Text 组件

10. 使用 UGUI 组件必须在 C#脚本中添加 UI 的命名空间，这样我们才能引用。当 bCube2 的值为真时，Text 组件显示“Cube 正在旋转中…”，所以把脚本写在 Update 的 if 语句里面，如图 12-29 所示。

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

public class CubeRotate : MonoBehaviour {

    private bool bCube1 = false;
    private bool bCube2 = false;
    private Color OldColor;

    void Start () {
        //获取物体的原始颜色
        OldColor = this.gameObject.GetComponent<Renderer> ().material.color;
        GameObject.Find ("Spotlight").GetComponent<Light> ().intensity = 1.0F;
    }

    void Update () {
        if(bCube2)
        {
            this.gameObject.transform.Rotate(Vector3.up*Time.deltaTime*200);
            GameObject.Find ("Text").GetComponent<Text> ().text = "Cube正在旋转...";

            if(GameObject.Find ("Spotlight").GetComponent<Light> ().intensity <8.0F)
            {
                GameObject.Find ("Spotlight").GetComponent<Light> ().intensity +=0.05F;
            }
        }
    }

```

图 12-29　添加控制 Text 显示的脚本

11. 单击“Play”按钮，进行游戏测试，如图 12-30 所示。

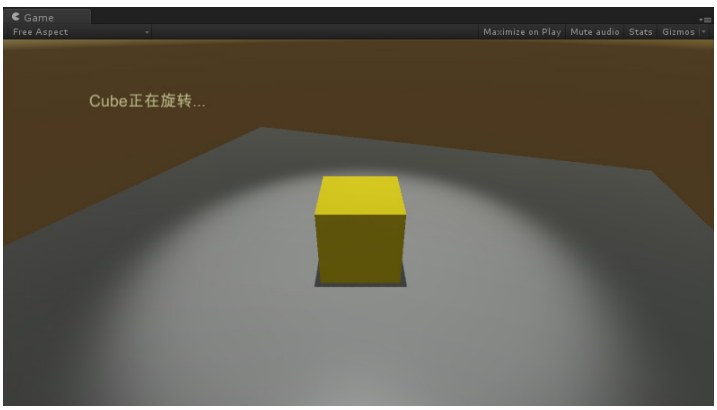

图 12-30　单击“Play”按钮测试游戏