02 执行“资源>创建>C#脚本”菜单命令创建一个脚本，并命名为BulletControl，然后挂载到Bullet物体上。编写的脚本代码如下。

输入代码

```
using UnityEngine;

public class BulletControl : MonoBehaviour
{
  void Start()
  {
    //5s后自动销毁
    Destroy(gameObject, 10f);
    //设定一个速度
    GetComponent<Rigidbody>().velocity = Vector3.forward * 50;
  }

  private void OnTriggerEnter(Collider other)
  {
    //触发后销毁
    Destroy(gameObject);
  }
}
```

03 将Bullet拖曳到“项目”面板中生成预制件，然后删除场景中的Bullet。接着继续完善PlayerControl的脚本，编写的代码如下，游戏的运行情况如图5-68所示。

输入代码

```
using UnityEngine;

public class PlayerControl : MonoBehaviour
{
  //子弹预制件，这里关联创建好的Bullet预制件
  public GameObject BulletPre;
  //子弹发射位置
  private Transform firePoint;

  void Start()
  {
    firePoint = transform.Find("FirePoint");
  }

  void Update()
  {
    //横向飞行
    //获取水平轴数值
    float horizontal = Input.GetAxis("Horizontal");
    //如果不为0，证明我们按下了左右键
    if (horizontal != 0)
    {
      //移动
        transform.position -= Vector3.left * 10f * Time.deltaTime *
horizontal;
      //这里我们设定一个范围，限制飞机不能无限远移动
      if (transform.position.x < -10 || transform.position.x > 10)
      {
        //超出范围，复原位置
          transform.position += Vector3.left * 10f * Time.deltaTime *
horizontal;
      }
    }
    //纵向飞行，这里为了使代码清晰，我们将横向飞行和纵向飞行分开写
    //获取垂直轴数值
    float vertical = Input.GetAxis("Vertical");
    //如果不为0，证明我们按下了上或下方向键
    if (vertical != 0)
    {
      //移动
      transform.position += Vector3.up * 10f * Time.deltaTime * vertical;
      //同样设定一个垂直移动的范围
      if (transform.position.y < -10 || transform.position.y > 10)
      {
        //超出范围，复原位置
          transform.position -= Vector3.up * 10f * Time.deltaTime *
vertical;
      }
    }

    //按下空格键发射子弹
    if (Input.GetKeyDown(KeyCode.Space))
    {
      Instantiate(BulletPre, firePoint.position, firePoint.rotation);
    }
  }

  //如果碰到敌人
  private void OnCollisionEnter(Collision collision)
  {
    //游戏结束
    Debug.Log("游戏结束");
    //游戏停止，这里我们停止游戏，在后期学会UI设计后，
//可尝试自行制作一个游戏结束画面
    Time.timeScale = 0;
  }
}
```