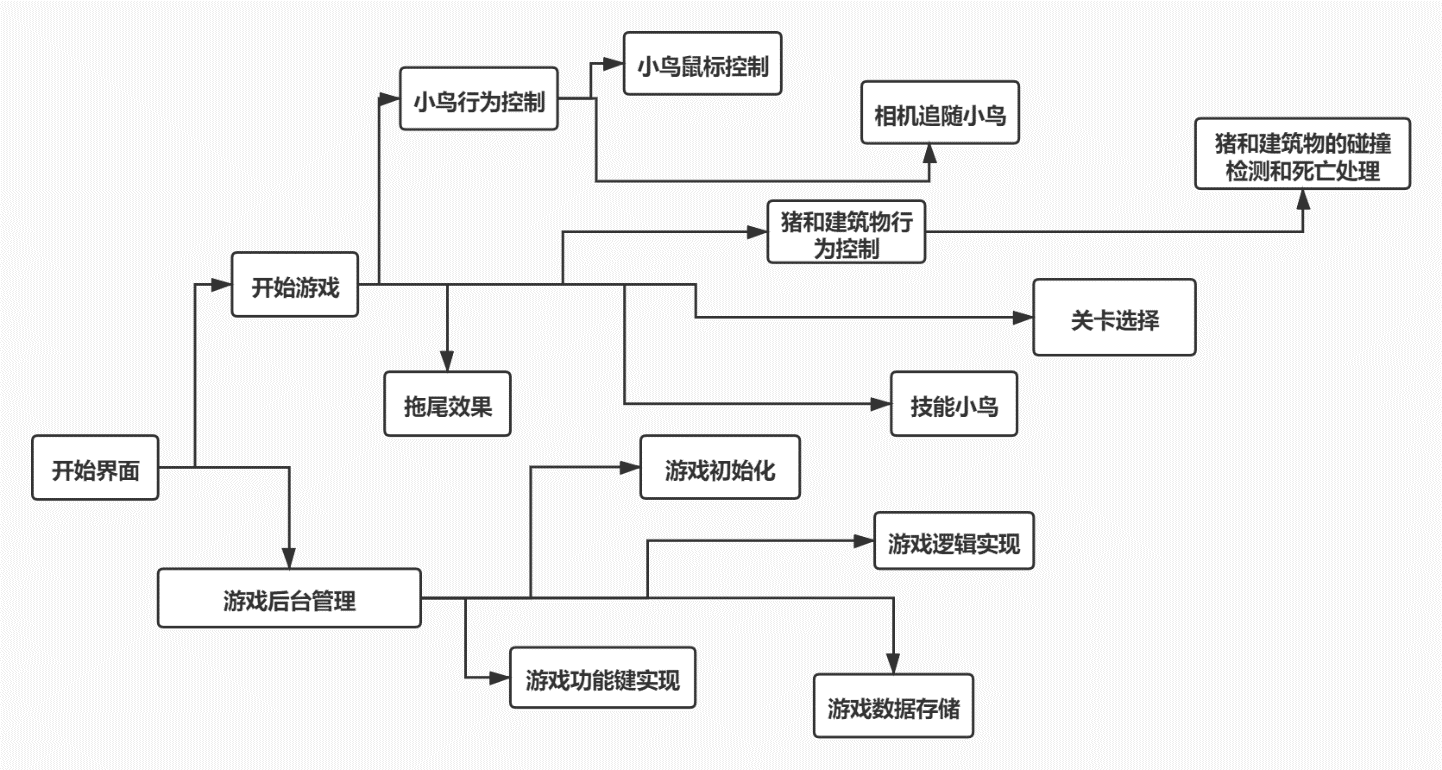
**AngryBirds(Seasons) 软件设计说明书**

**20373623 软件学院 周恩申**

1. **总体设计**
   1. **模块组成**



* 1. **模块结构说明**

游戏分为可供玩家操控的游戏模块和不可被玩家操控的游戏后台管理板块组成，这两个模块在游戏软件打开的时候，就开始一同工作，密不可分。

1. **程序描述**
   1. **游戏后台管理模块**
      1. **功能**

实现游戏的全自动化管理，让游戏在不被外界干扰的情况下自行运转和调整。

* + 1. **输入项目**

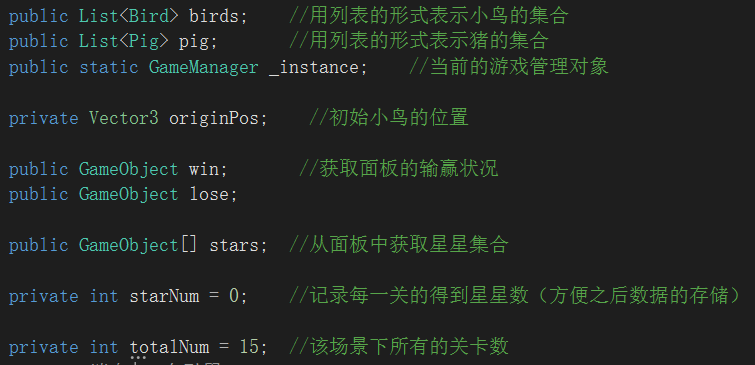
鼠标点击按钮

* + 1. **输出项目**

游戏按逻辑正常进行；游戏页面跳转；游戏功能键按键反馈；界面更新；

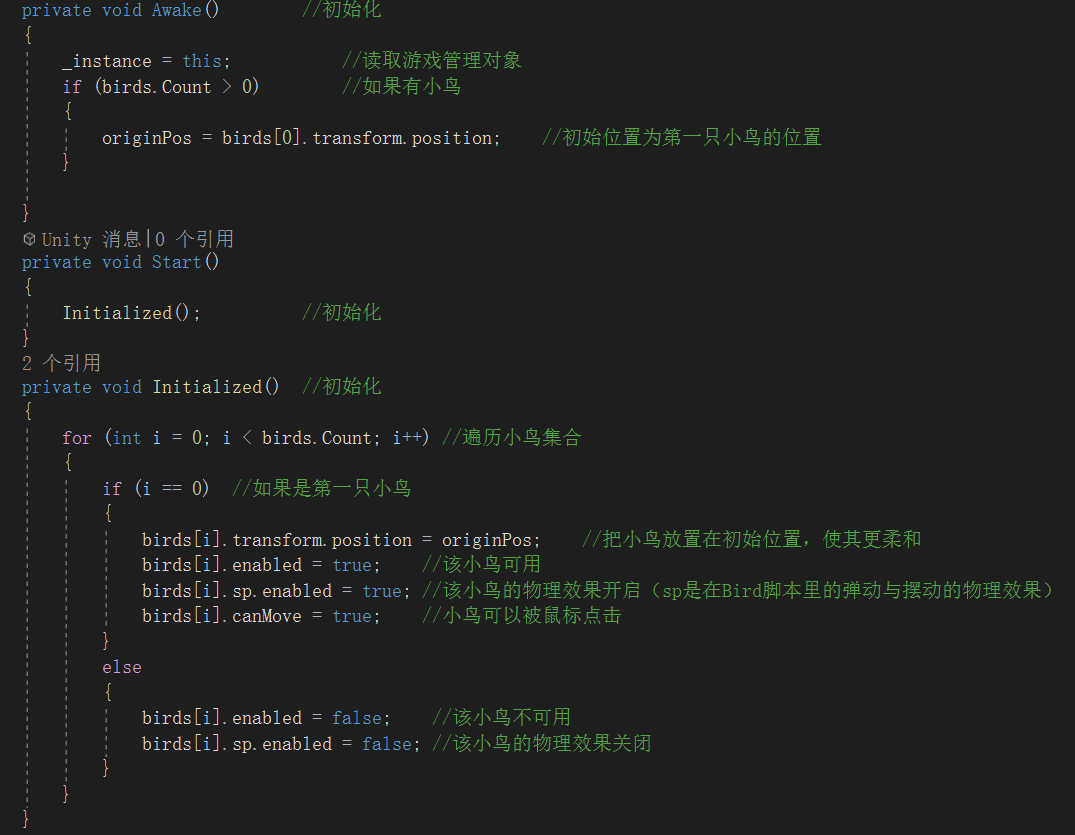
* + 1. **算法**

1. 游戏的初始化操作
2. 游戏逻辑的实现
3. 游戏功能键的设置
4. 游戏的数据存储
   * 1. **代码**



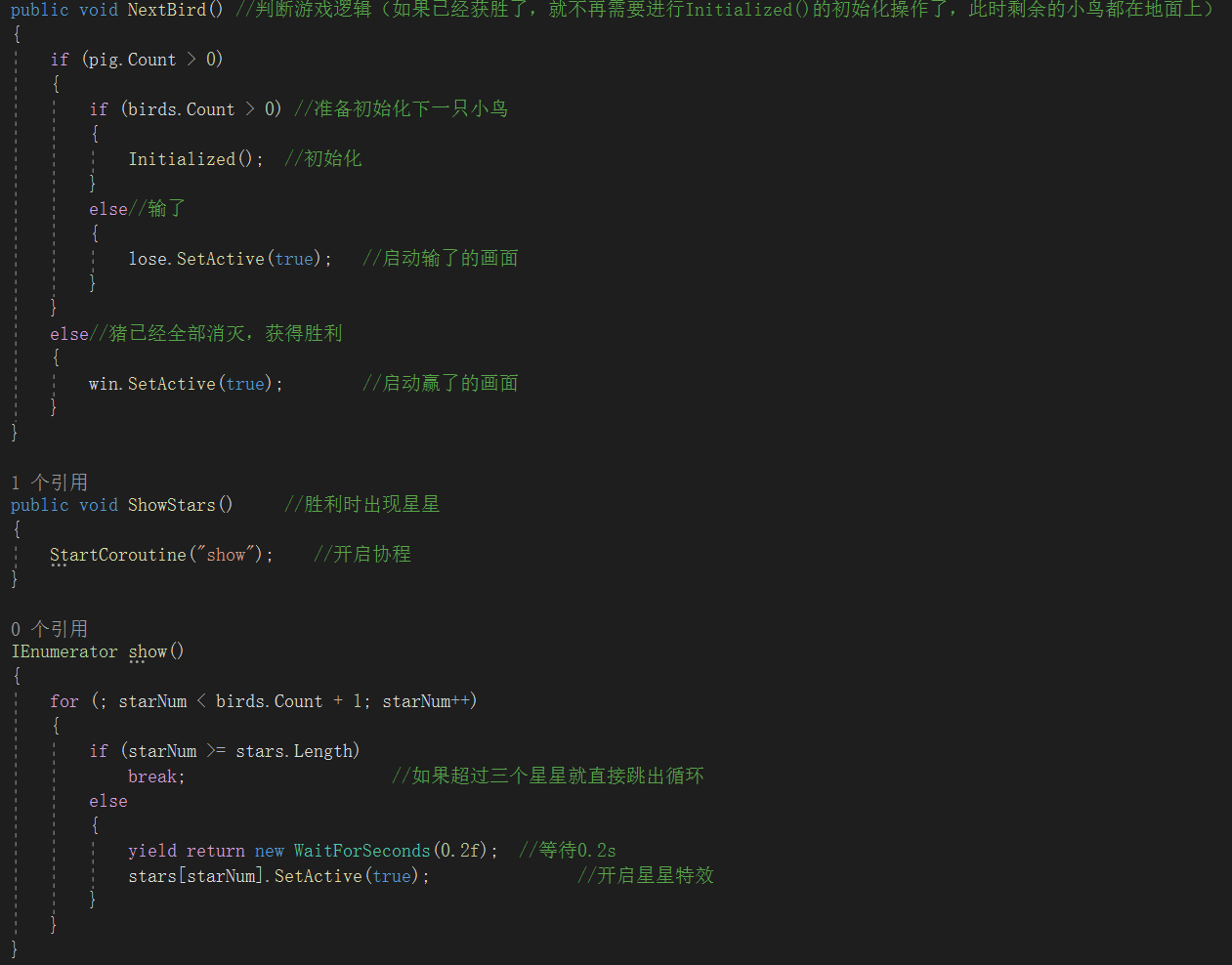
GameManager.cs变量申明

下面是游戏的初始化：



GameManager.cs初始化

下面是具体游戏逻辑实现：

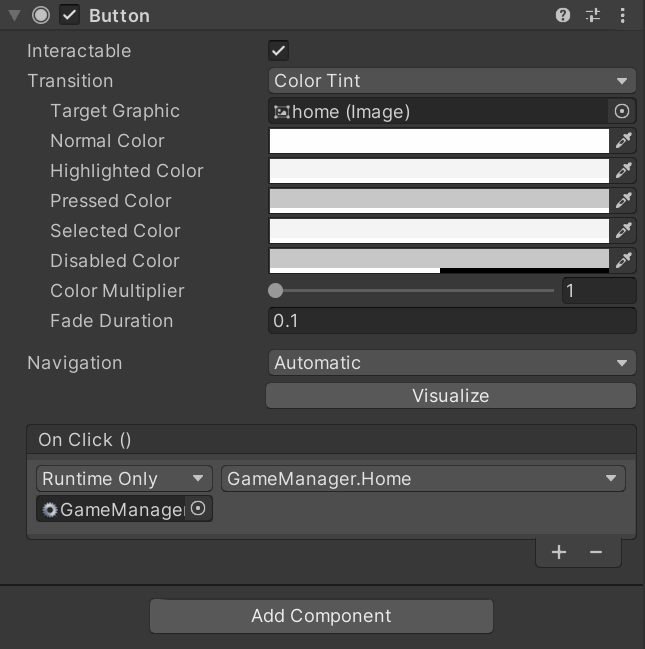


GameManager.cs游戏逻辑

下图是游戏功能键的设置：

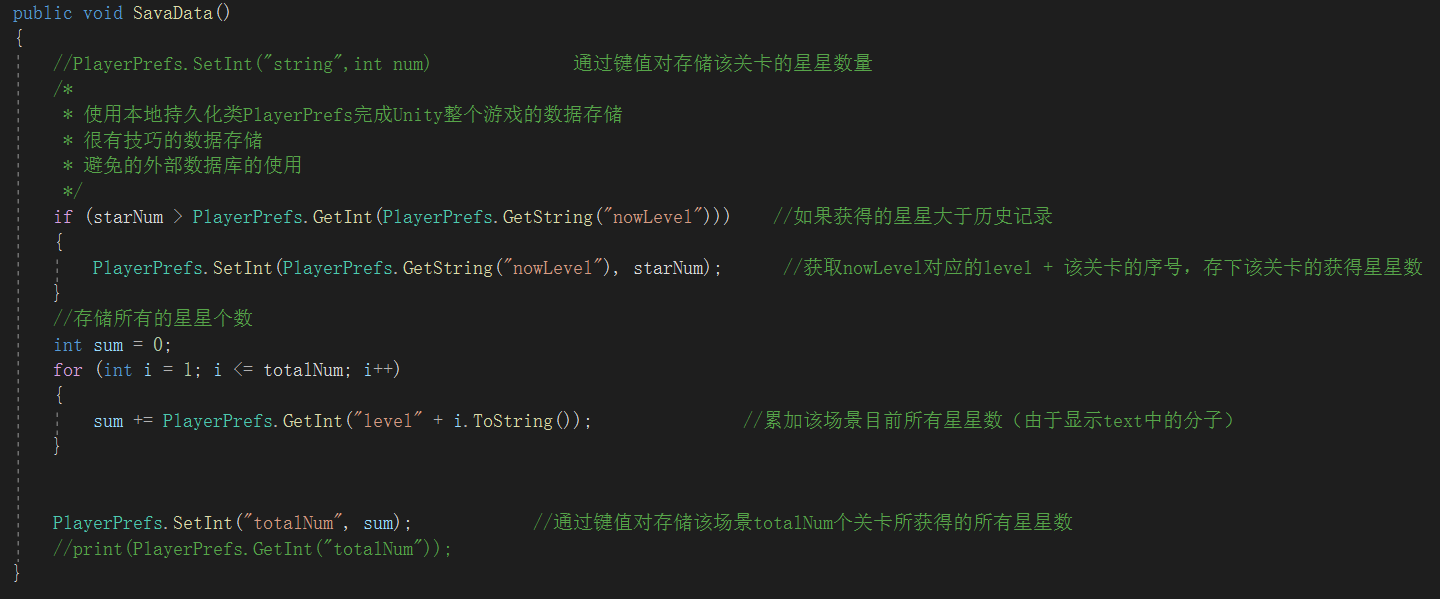


GameManager.cs功能键设置



按钮添加GameManager.cs功能键函数

下图是游戏数据存储：



GameManager.cs游戏数据存储

* 1. **游戏模块**
     1. **功能**

1. 控制小鸟的物理组件
2. 控制相机追随小鸟移动
3. 控制猪和建筑物的物理组件
4. 控制猪和建筑物的受伤或死亡状态
5. 用鼠标控制小鸟技能的释放
6. 控制场景与关卡的开启和关闭
   * 1. **输入项目**

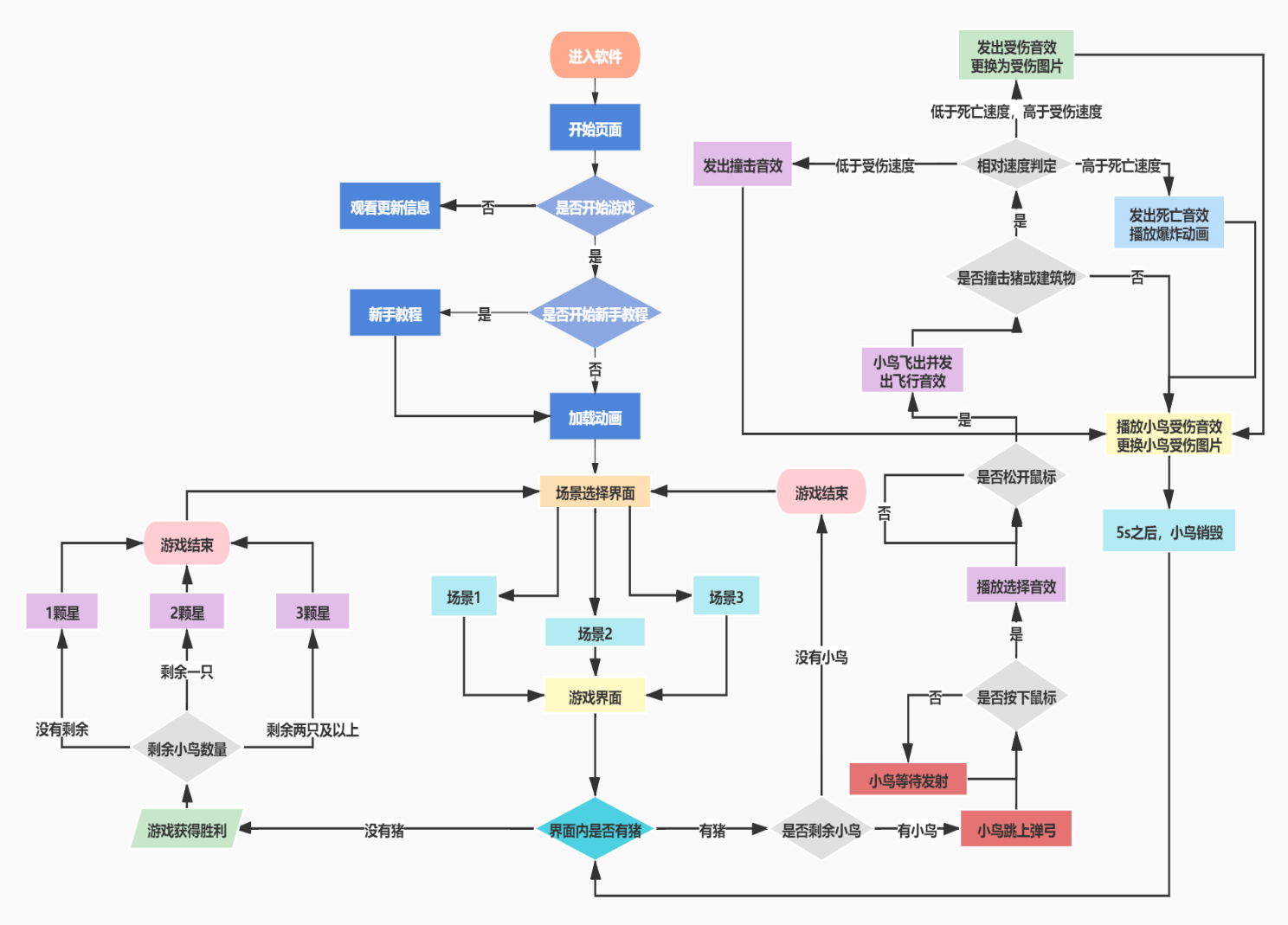
鼠标的点击和拖拽

* + 1. **输出项目**

小鸟位置的改变；小鸟图片的更改；猪和建筑物位置的改变；猪和建筑物的图片更改；得分方式的反馈；按钮反馈；音效反馈

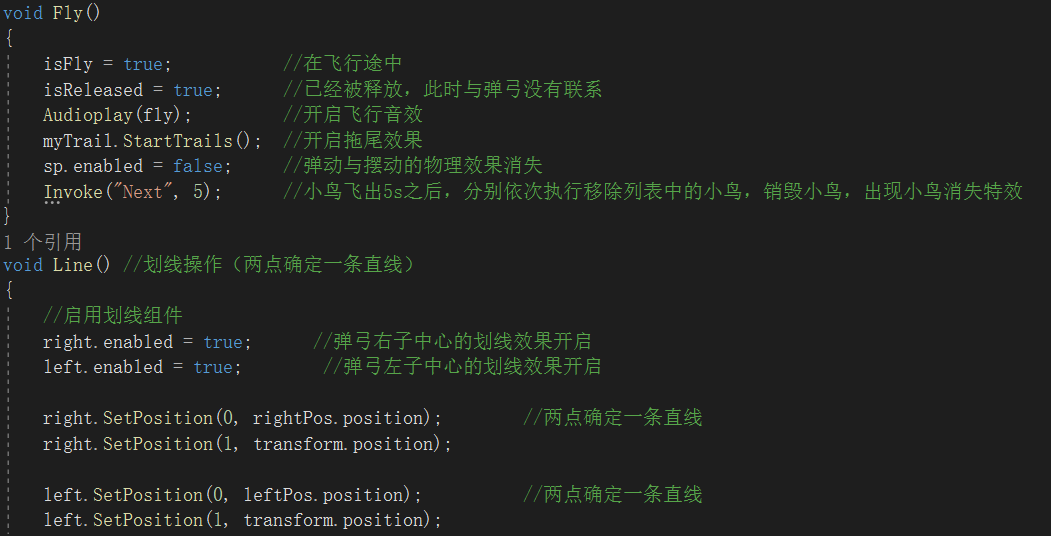
* + 1. **算法**

1. 采用OnMouseDown()和OnMouseUp()来控制鼠标行为
2. 运用动量守恒定律与collision.relativeVelocity()碰撞组件完成碰撞检测
3. 采用前驱法完成对场景和关卡的开启判断
4. 利用Bird.cs中的虚方法ShowSkill()，方便进行重载和继承，完成小鸟技能的实现
   * 1. **程序逻辑**



* + 1. **代码**





Bird.cs鼠标控制

下图是相机追随小鸟移动的部分：

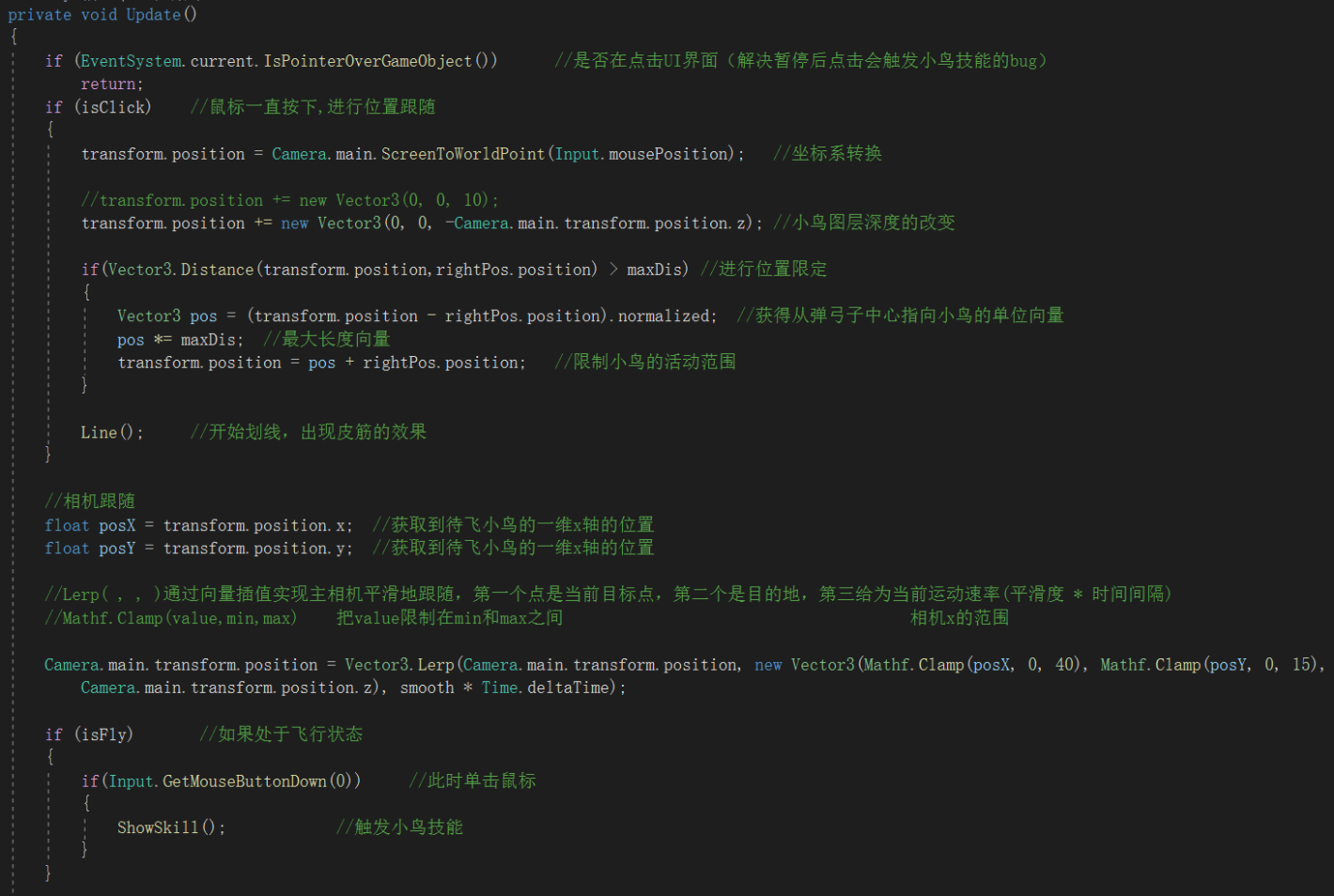


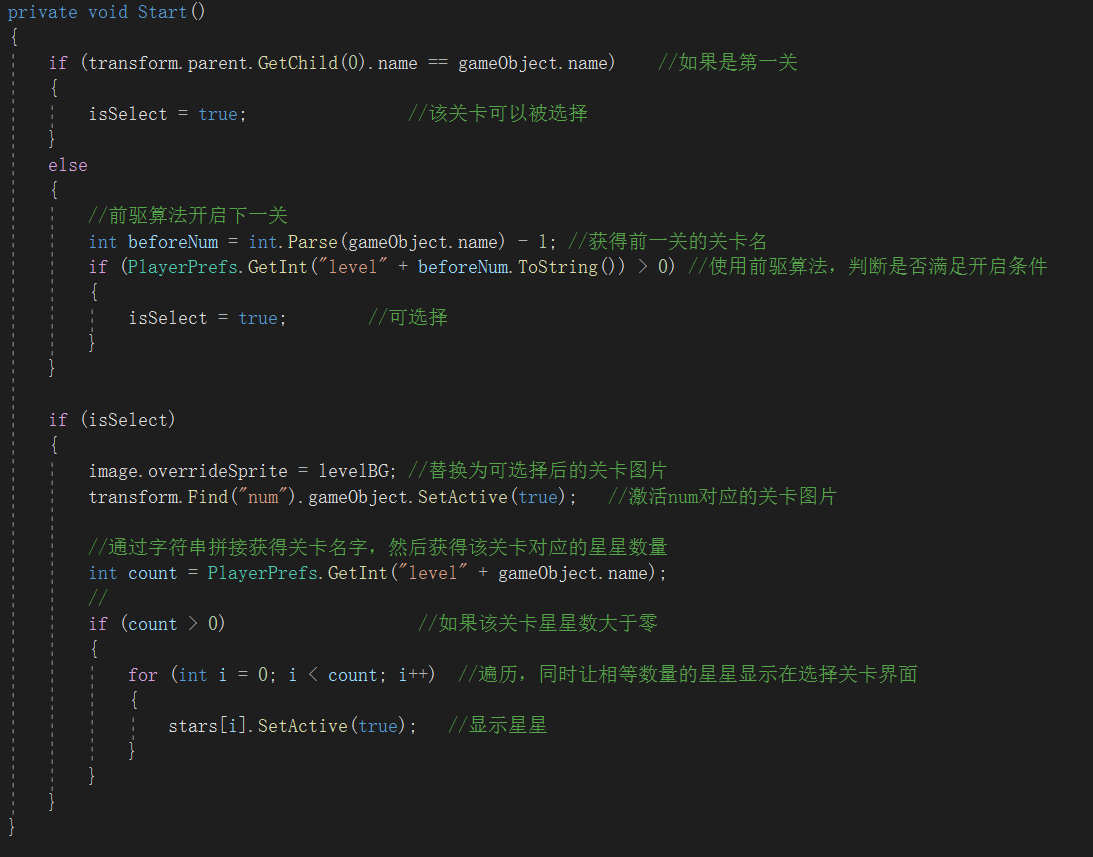
图5.9 Bird.cs相机移动

下图是碰撞检测和死亡处理部分：

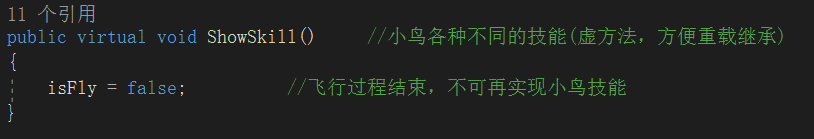


Pig.cs碰撞检测和死亡处理

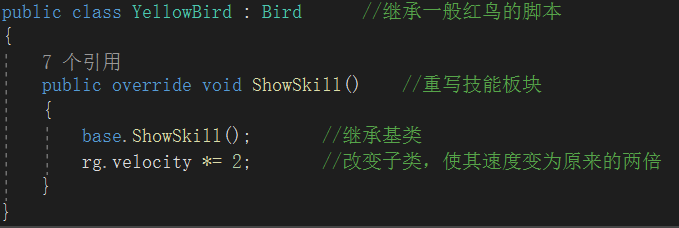
下图是场景与关卡开启部分：



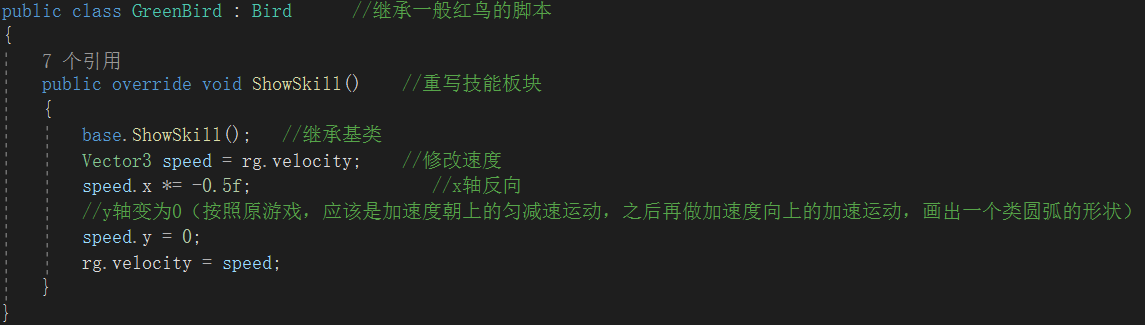
LevelSelect.cs场景与关卡开启



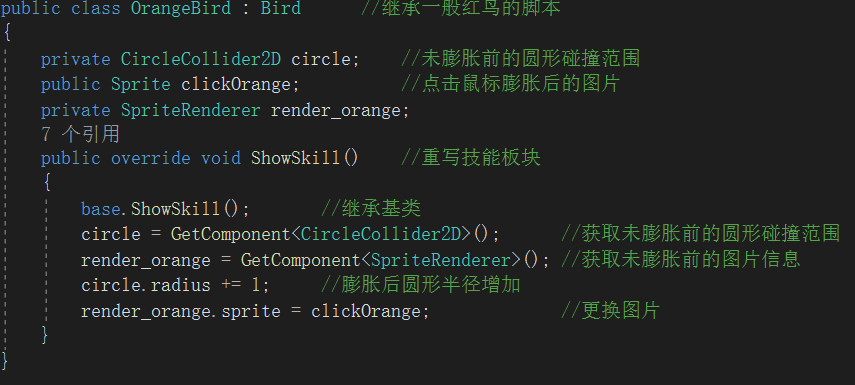
Bird.cs中ShowSkill()的声明



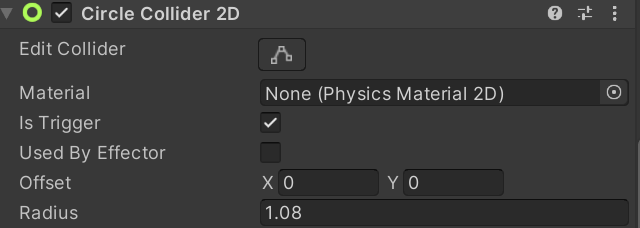
YellowBird.cs中ShowSkill()的重写



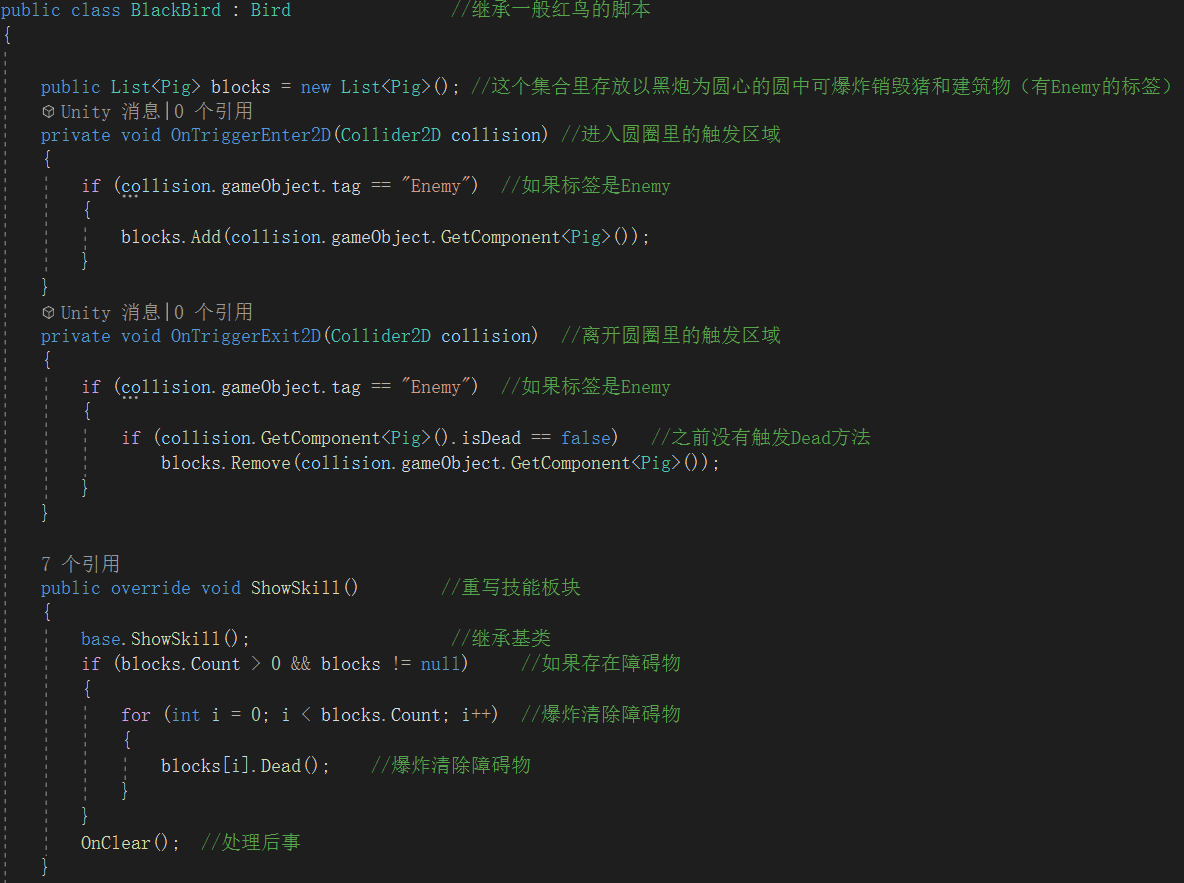
GreenBird.cs中ShowSkill()的重写



OrangeBird.cs中ShowSkill()的重写

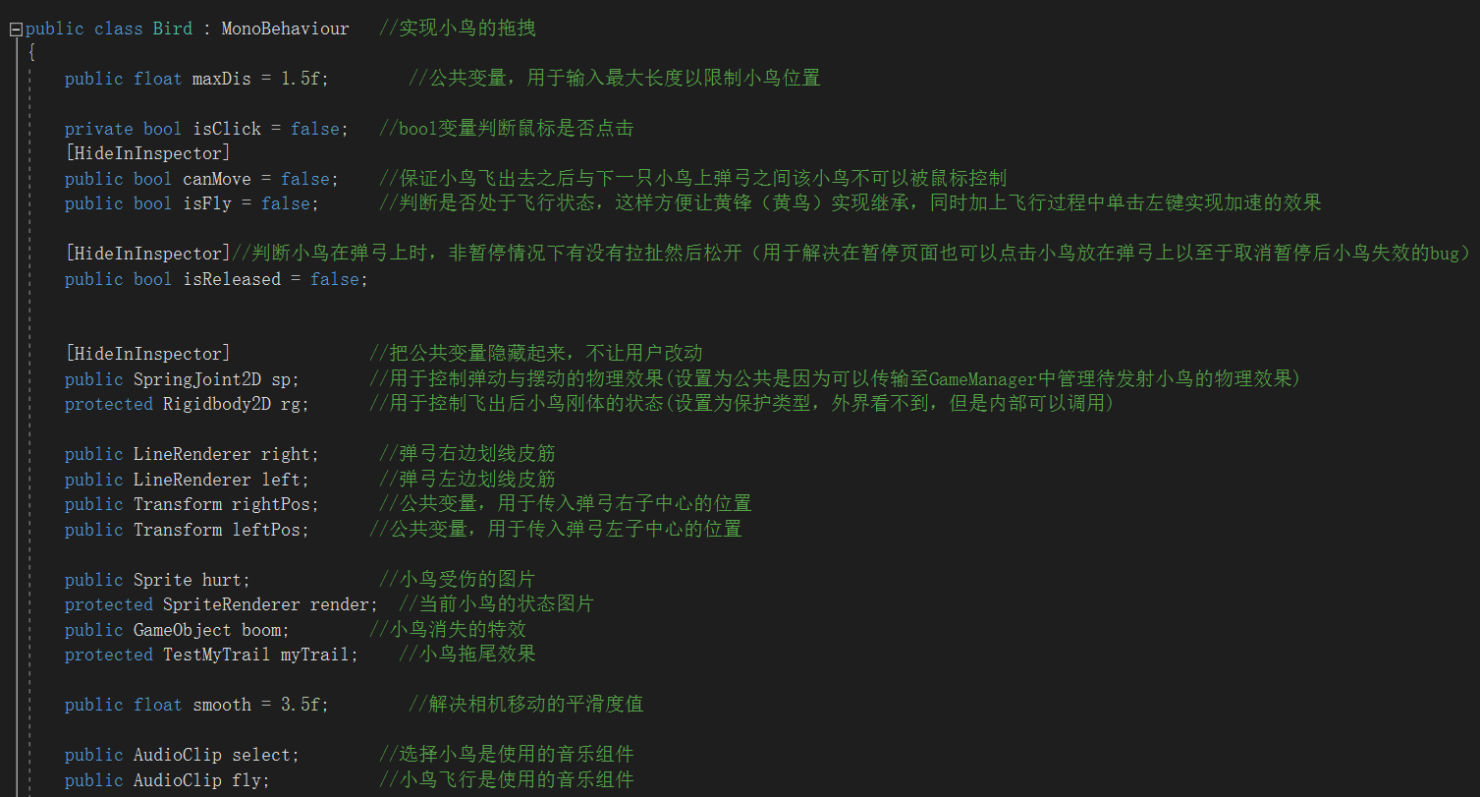


Circle Collider 2D的注意事项

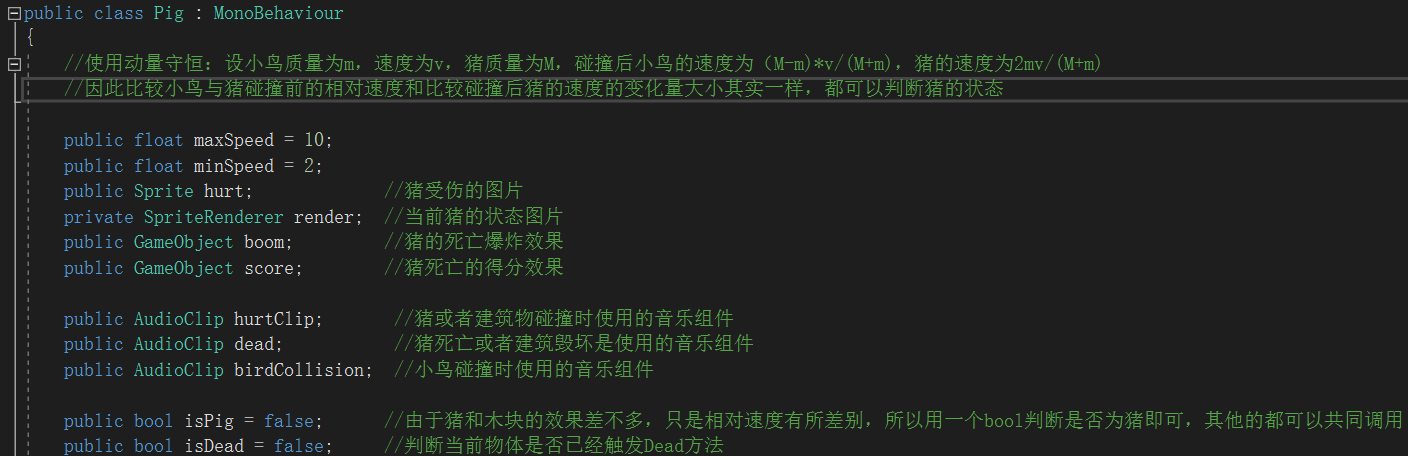


BlackBird.cs中ShowSkill()的重写

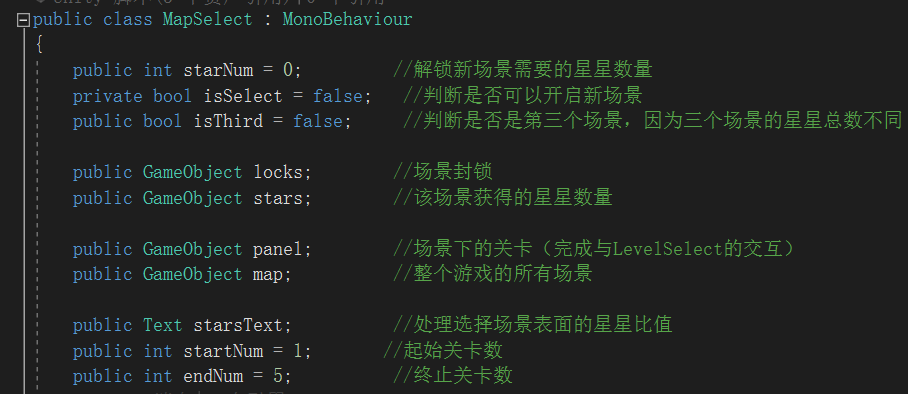
1. **系统数据结构设计**
   1. **逻辑结构设计**
2. 小鸟状态



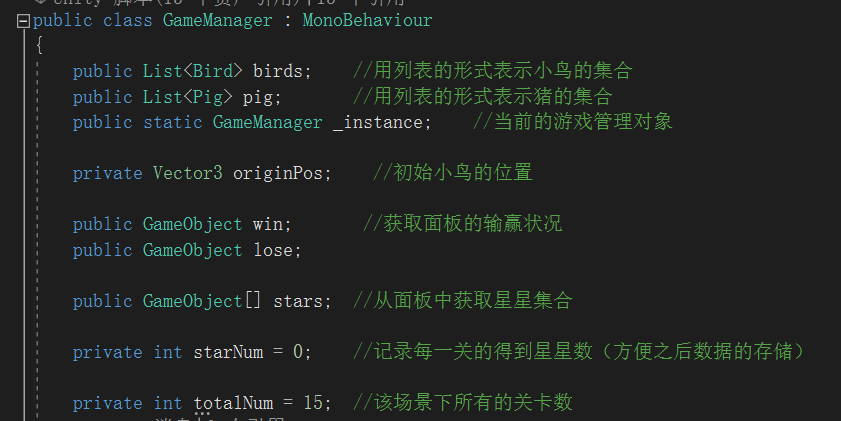
1. 猪和建筑物状态



1. 场景与关卡选择



1. 游戏后台管理



* 1. **逻辑结构设计**

此游戏并未涉及到负载的数据结构，唯一重点在于场景与关卡的存储以及过关数据的保留和场景与关卡之间的跳转。

由于Unity的本地持久化类PlayerPrefs包含类似于Hash一样的键值对存储，所以合理利用内置的数据结构可以高效的完成设计目的。

**存储：**本地持久化类PlayerPrefs在存储方面有两个方法。

1. PlayerPrefs.SetInt("string",int num)

通过键值对存储该关卡的星星数量，具体方式为PlayerPrefs在内部存储空间申请一个空间，并命名为“string”，然后在该空间内部存储整形数据num，表示该关卡的星星数量。

1. PlayerPrefs.SetString("nowLevel", levelNum);

通过键值对存储玩家需要打开的关卡编号，具体方式为PlayerPrefs在内部存储空间申请一个空间，并命名为“nowLevel”，然后在该空间内部存储字符串类型数据levelNum，表示玩家需要打开的关卡编号。

**访问**：本地持久化类PlayerPrefs在获取方面有两个方法。

1. PlayerPrefs.GetInt(string);

通过键值对获取该关卡的星星数量，具体方式为PlayerPrefs在内部存储空间访问命名为“string”的存储空间，然后取出在该空间内部存储整形数据num。

1. PlayerPrefs. GetString("nowLevel");

通过键值对获取玩家需要打开的关卡编号，具体方式为PlayerPrefs在内部存储空间访问命名为“nowLevel”的存储单元，然后获取在该空间内部存储字符串类型数据levelNum。

下面为具体使用案例：

