Tsingtopia 测试文档

徐栩海 马鸿鹏 卢翔 李凯文

目录

1. 简介

本文档对改工程制作过程中的测试方式做一非常简要的总结。Unity 的特性使得我们的测试相对困难，间接。

2. 测试方式

我们的测试共分为两部分：白盒测试、黑盒测试。

2.1 白盒测试

由于 Unity 引擎的特性和 Unity 开发依赖 Unity 本身的编辑器和自身引擎绑定得比较紧密，我们的白盒测试多数选择对特定模块的测试脚本，一起存放在 Assets/Scripts 目录下的 TestingUnit 目录下

2.1.1 成功添加 message 到事件列表测试

- MessageTest.cs
- 调试过程中解决 GUI 部分按钮点击未能触发指定事件的问题
- 解决方法：在 GUI 初始化时重新注册

2.1.2 message 成功触发事件测试

- MessageTriggerTest.cs
- 解决场景切换时，由于 MessageManager 再次被初始化而破坏单件模式，导致事件触发失败 bug
- 解决方法：添加了单件模式中再次实例检查的，由于 Unity 引擎的特性，我们在初始化后如果发现已有 MessageManager 则将新产生的 MessageManager 实例摧毁保证单件

2.1.3 task 成功触发测试

- taskTest.cs
- 解决 collider 碰撞和 task 导入错位的 bug
- 解决方法：阅读 C# List 在开始插入一个空的 task

2.1.4 任务弹窗测试

- genericModalWindow.cs
- 测试弹窗组件可以正常响应

[注：]

这些白盒测试是在学习软件工程测试之后保留下来的部分测试，实际开发过程前期，我们在开发每个模块时基本都会写简单的白盒测试脚本，然而由于对测试理解上的偏差，我们把许多测试认为是冗余代码删除，这件事情是也是我们本次软件开发的一个经验教训。

2.2 黑盒测试

借助虚拟输入进行自动黑盒测试。

脚本随机产生鼠标游走位置，鼠标点击操作，方向键按击操作。目的为模仿用户进行键鼠输入。同时在游戏脚本中存留尽可能多的容错管道，一旦某处发生意料之外的错误，即通过管道进行报告错误，辅助 debug。

自动测试脚本一旦运行，需要手动关闭。

脚本具体内容详见软工网络平台上的 assignment4。

自动测试结果：

- 发现一处模型构建漏洞，两 collider 之间存在缝隙，玩家可以通过。
  解决方式：增加 collider 进行封堵
- 发现一处 GUI 点击误触发
  解决方式：修改 GUI 渲染图层顺序。