**Tsingtopia测试文档**

徐栩海 马鸿鹏 卢翔 李凯文

**目录**

[1. 简介 3](#_Toc471497987)

[2. 测试方式 3](#_Toc471497988)

[2.1 白盒测试 3](#_Toc471497989)

[2.1.1 成功添加message到事件列表测试 3](#_Toc471497990)

[2.1.2 message成功触发事件测试 3](#_Toc471497991)

[2.1.3 task成功触发测试 3](#_Toc471497992)

[2.1.4 任务弹窗测试 4](#_Toc471497993)

[2.2 黑盒测试 4](#_Toc471497994)

# 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。

# 测试方式

我们的测试共分为两部分：白盒测试、黑盒测试。

## 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

### 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

### 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

### 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

### 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。

## 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。