测试技术说明

我们使用了Rust自带的测试框架，它支持对每个模块分别定义单元测试和对整个应用定义集成测试。同时，Rust的测试工具也支持对测试限时和运行基准测试。

1 测试环境

- **服务器环境**: GitHub CI，Azure虚拟机，7G可用内存，Intel Xeon单核
- **系统环境**: Windows 2022，macOS，Ubuntu 22.04
- **应用环境**: 云原神，云崩坏星穹铁道

开发测试计划

1 系统控制子系统

测试项	描述	测试结果
键盘和鼠标控制	能否正确调用系统API控制鼠标键盘	通过
管理员权限获取	能否获取对应系统的管理员权限	通过
窗口焦点管理	能否将焦点置于窗口，并将窗口置于前台	通过
窗口分辨率管理	能否正确获取和设置窗口的分辨率	通过

2 交互逻辑子系统

测试项	描述	测试结果
坐标读取	能否正确读取对应位置的物品	通过
扫描逻辑设计	能否正确按照顺序扫描物品	通过

3 OCR集成

测试项	描述	测试结果
专用OCR模型	OCR模型的运行结果是否正确	通过
通用OCR模型	OCR模型的运行结果是否正确	通过

4 通用导出

描述: 能否根据给定的数据导出成通用的JSON格式

测试结果: 通过

5 截图

测试项	描述	测试结果
坐标获取	正确根据窗口分辨率获取应该截图的坐标	通过
截图实现	正确截取对应位置的窗口内容	通过

发行测试计划

测试项	描述	测试结果
发行测试	能否在仓库提交时正确生成对应的发布内容，包括可执行文件和改动日志	通过

详细说明:

1. **发布内容生成**: 确保在每次代码提交到主分支时，自动触发发布流程，生成包括可执行文件和更新日志在内的完整发布包。
2. **版本控制**: 确认发布版本号的自动递增和正确记录，确保每个发布版本都有唯一标识。
3. **自动部署**: 验证发布包能自动部署到指定的环境，并正确运行，确保发布流程的自动化和可靠性。
4. **发布日志**: 检查生成的发布日志是否包含所有关键更改和更新说明，以便用户了解新版本的改进和修复。

用户测试计划

测试项	描述	测试结果
运行测试	测试程序能否运行并返回正确结果	通过

详细说明:

1. **功能验证**: 确保所有核心功能在用户实际使用场景中都能正常运行，并返回预期的结果。
2. **边缘测试**: 测试分析特殊情况下应用的反馈，识别和修复潜在的问题和缺陷。
3. **兼容性测试**: 在不同操作系统和硬件配置下测试应用程序，确保其兼容性和稳定性。
4. **使用文档**: 验证用户手册和使用文档的准确性和易用性，确保用户能够顺利安装和使用程序。

测试设计

开发阶段，我们的所有测试都在固定的几个游戏版本上进行，以固定环境，保证我们的功能开发和逻辑修复确实修复了问题；同时，使用多个版本同时测试也强制了我们的功能开发能够满足游戏更新带来的兼容性需求。

详细说明:

1. **固定版本测试**: 选择若干个代表性的游戏版本，确保在这些版本上的测试覆盖率和准确性。这有助于识别特定版本中的问题，并验证修复的有效性。
2. **多版本兼容性测试**: 在不同的游戏版本上运行相同的测试用例，确保新功能和修复能够兼容不同版本的游戏更新。这有助于提高应用程序的适应性，减少版本更新带来的潜在问题。
3. **环境一致性**: 确保测试环境的一致性，包括操作系统版本、硬件配置和依赖库版本，保证测试结果的可靠性和可重复性。
4. **自动化测试**: 使用自动化测试框架和工具，编写和执行单元测试、集成测试和回归测试，提高测试效率和覆盖率，减少人工测试的误差和成本。
5. **测试数据管理**: 创建和维护测试数据集，确保测试数据的真实性和代表性，以便准确评估功能和性能。
6. **测试报告和分析**: 定期生成和分析测试报告，跟踪测试进度和结果，及时识别和解决问题，不断优化测试策略和方法。

缺陷跟踪

在开发和测试过程中，我们使用JIRA进行缺陷跟踪。

- 每个缺陷都被分配一个唯一的标识，并记录缺陷的详细描述、发现时间、发现人、优先级、修复状态等信息。
- 每个缺陷的处理流程包括发现、确认、修复、验证和关闭五个步骤。
- 通过JIRA的看板视图，我们可以直观地了解缺陷的处理进度和当前状态，确保每个缺陷都得到及时有效的处理。

质量保证方法

为了保证软件的质量，我们采取了以下几种方法：

1. **代码审查**: 在代码提交之前，必须经过团队成员的审查，确保代码质量和逻辑的正确性。
2. **持续集成**: 每次代码提交后，自动触发构建和测试，及时发现并修复问题。
3. **自动化测试**: 使用Rust的测试框架，编写单元测试和集成测试，保证代码的功能和稳定性。
4. **回归测试**: 在每次版本发布前，进行全面的回归测试，确保新功能的引入不会影响已有功能的正常运行。
5. **性能测试**: 定期进行性能测试，监测系统的性能指标，及时优化和改进。

通过以上方法，我们能够有效地保证软件的质量，降低缺陷率，提高用户满意度。