软件工程化说明文档

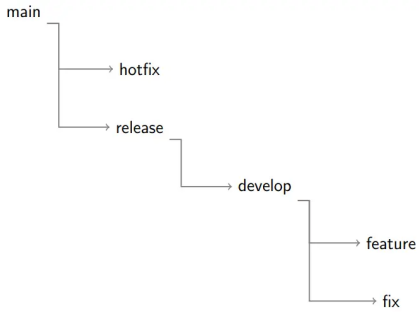
1.项目概述

- **项目名称**：基于 `OpenGL` 的三维峡谷探险迷宫设计。
- **项目描述**：使用 `OpenGL` 和 `C++` 开发的一个三维峡谷探险迷宫游戏，玩家通过控制角色在迷宫中移动来找到出口。
- **开发环境**：
  - 编程语言： `C++`
  - 开发工具： `Qt Creator` 、 `Visual Studio 2022`
  - 其他工具： `OpenGL`

2.软件工程化手段

- **需求分析**：
  - **功能需求**：
    - 玩家控制角色进行移动
    - 迷宫生成算法
    - 碰撞检测
    - 摄像机视角调整
  - **非功能需求**：
    - 高效的渲染性能
    - 良好的用户体验等

- **系统设计**：
  - **架构设计**：采用模块化设计，主要有以下模块
    - 游戏逻辑
    - 渲染引擎
    - 输入处理
  - **详细设计**：
    - `camera` 模块：处理摄像机视角和移动
    - `maze` 模块：生成迷宫和处理迷宫逻辑
    - `player` 模块：控制玩家的移动以及交互
    - `myglwidget` 模块：负责 `OpenGL` 渲染
    - `textures` 模块：加载和管理纹理
- **编码规范**：
  - **命名规则**：类名采用大驼峰命名法，变量名和函数名采用小驼峰命名法。
  - **注释要求**：每个类和每个函数均有简要描述，较为复杂的逻辑有详细注释。
- **版本控制**：
  我们使用Git进行版本控制, 分支策略如下:



• main 分支：主分支，存放稳定的生产代码。• develop 分支：开发分支，存放最新的开发代码。

• feature 分支：从 develop 分支创建，用于开发新功能。命名规则为 feature/功能描述。

• fix 分支：从 develop 分支创建，用于修复 Bug。命名规则为 fix/bug 描述。

• release 分支：从 develop 分支创建，用于准备发布版本。命名规则为 release/版本号。

• hotfix 分支：从 main 分支创建，用于紧急修复生产环境的问题。命名规则为 hotfix/问题描述。

- **测试策略：**
  - **软件测试：**进行了全面的软件测试策略来确保游戏的质量。包括功能测试、性能测试、软件性能测试等多个方面。采取人工测试的方式进行，分为组内成员测试以及路人随机测试。过程中进行了多次、多角度、多方面的测试，以提高测试结果准确性。测试结果如下：
    - **产品稳定性：**经过在不同性能设备上的连续运行、进行大量有意义/无意义键盘输入，游戏均可正常运行，没有出现故障与崩溃。
    - **功能完整性：** 游戏运行后，玩家会置身于一个峡谷之中，峡谷为一个迷宫。玩家需要根据右下角的小地图确定终点和自身位置以及探索路线。游戏场景部分，经过测试，迷宫地图完整、实际地形与小地图一致、贴图纹理正常显示、可互动地形交互正常、可正常通关。以下为部分游戏运行截图：

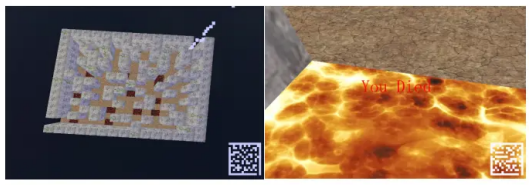


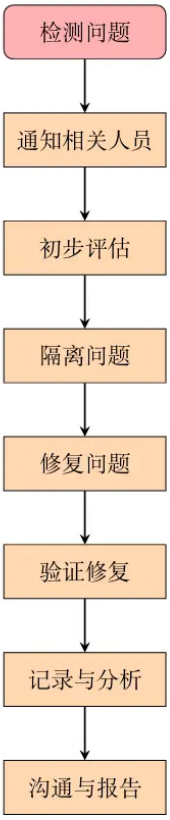
图 1-1 实际地图与小地图

图 1-2 可互动地形交互正常（该图为触碰岩浆后游戏结束）

图 1-3 贴图纹理正常显示(岩壁、泥土地面以及冷凝的岩浆)

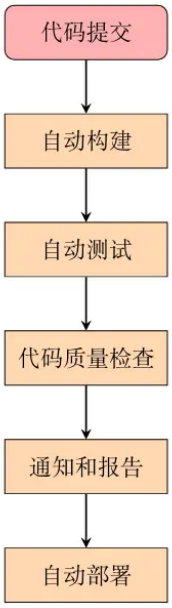
图 1-4 到达终点后可正常通关

    - **性能测试：**经过在不同设备上的尝试运行，软件性能良好，相应速度快，资源占用较合理。游戏内运行不卡顿，帧率在15-30帧左右。
- **项目管理：**
  - **日志管理:** 所有日志信息均在终端输出。
  - **故障处理:** 流程在生产环境中遇到故障时，按照以下流程进行故障处理:



1. **检测问题**：通过监控系统或用户报告发现问题，确定问题的范围和影响。
2. **通知相关人员**：通过邮件、电话或即时通讯工具通知团队成员。启动应急响应团队，包括开发人员、运维人员和项目经理。
3. **初步评估**：分析问题症状，初步确定问题原因。决定是否需要立即采取措施（如暂停某些服务或功能）。
4. **隔离问题**：尽量将问题隔离，减少对其他系统或服务的影响。如有必要，回滚最近的更改或切换到备用系统。
5. **修复问题**：根据初步评估结果，制定并实施修复计划，可能包括代码修复、配置修改、重启服务等。
6. **验证修复**：确认问题已解决，系统恢复正常运行。进行全面系统测试，确保没有引入新问题。
7. **记录与分析**：记录问题详细信息，包括时间、原因、处理过程和结果。分析问题根本原因，制定预防措施，避免类似问题再次发生。
8. **沟通与报告**：向相关人员报告问题处理和系统恢复情况。在团队内部分享经验，提高应急响应能力。

- **持续集成与部署：**
  - **持续集成:**



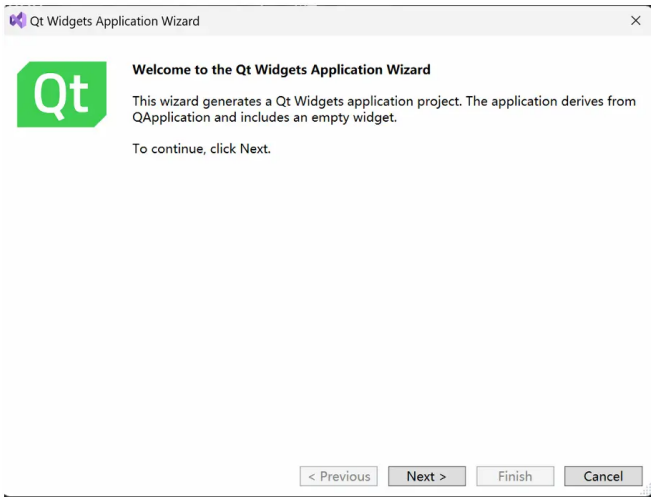
1. **代码提交**：开发人员将代码提交到 Git 仓库，触发 CI 流程。
2. **自动构建**：CI 服务检测到代码变化，自动拉取最新代码并构建项目。
3. **自动测试**：构建完成后，自动运行所有单元测试和集成测试。
4. **代码质量检查**：运行代码静态分析工具，检查代码质量。
5. **通知和报告**：CI 服务将构建和测试结果通知开发团队。
6. **自动部署**：在所有测试通过后，自动部署改进后的代码。

  - **部署:**
    - 安装Qt和设置环境：

      — 通过Qt下载符合Visual Studio 版本的 Qt Installer.

      — 关闭所有Visual Studio 相关的进程, 运行 Qt Installer.

      — 重新运行Visual Studio, 创建一个 Qt 相关的项目. 如果你是初次创建 Qt 项 目, 会自动运行下图的向导:



一 根据向导提示配置Qt的路径即可. 对Windows用户,路径通常是C://Qt//mingw_64/bin/qmake.exe

- 安装OpenGL 相关库：一 使用包管理器安装（如 apt、brew或 vcpkg）。 • 创建Qt和OpenGL 集成项目：一 配置你的visual studio, 在 visual studio 中安装合适版本的Qt VS Tools.
- 一个可行的版本是: * Visual Studio 2019 * Qt 5.14.2 * Qt VS Tools2.4.3
- 如果你正确配置了上述环境,你只需点击VisualStudio 的本地 Windows 调试器按钮,运行生成的.exe 文件即可启动软件.

3.项目实施

- **项目计划**：
  - 大体上分为4个阶段：需求设计与分析、开发、测试与优化、相关文档的撰写（需要撰写团队报告的同学补充一下4个阶段的时间）。
    - 需求设计与分析：
    - 开发：
    - 测试与优化：
    - 相关文档的撰写：
- **任务分配**：（由撰写团队报告的同学撰写一下任务分工，最好是分阶段的）
- **进度跟踪**：
  - 通过召开线上会议确认项目选题以及项目的任务分工。
  - 后续通过召开线上会议以及使用腾讯文档进行任务进度的追踪。

4.总结与展望

- **项目成果**：
  - 成功实现了一个基于 `OpenGL` 的三维峡谷迷宫游戏，具有较为良好的用户体验和性能。
  - 项目文档完善，代码易于维护和扩展。
- **展望**：
  - 可以尝试增加更多的迷宫生成算法，进一步提高游戏的多样性和挑战性。
  - 后续可以尝试引入多人模式，增加游戏的互动性。
  - 优化渲染功能，进一步提高游戏流畅性。