三维峡谷探险迷宫

软件配置与运维文档

目录

1 简介

1.1 项目概述

本设计中的 3D 迷宫游戏主要通过二维矩阵所表示的迷宫图，来构建三维立体场景。我们为系统设计了一架摄像机，来模拟用户在迷宫中的第一视角，用户可以通过鼠标、键盘控制自己在 3D 迷宫中的移动。游戏也很好地设计了碰撞检测，给玩家以一种身临其境的感觉。游戏中设置了各种各样的元素，包括可以翻阅的障碍围墙，岩浆陷阱，以及得分机关。用户可以很好地在游戏的操作过程中与这些元素进行交互。游戏的场景设计耶十分用心，峡谷与地板的贴图都是份精细，此外还有藤蔓的动态纹理。我们完成了天空盒的设计并且引入了自己的光照模型，在天空盒动态移动的时候，光照阴影也会随之移动。总体来说，游戏界面精细美观，可操作性和交互性强。

1.2 文档概述

本文档主要关于软件的配置核运维指南.

2 系统架构

系统架构如下图, 详参考系统架构文档.

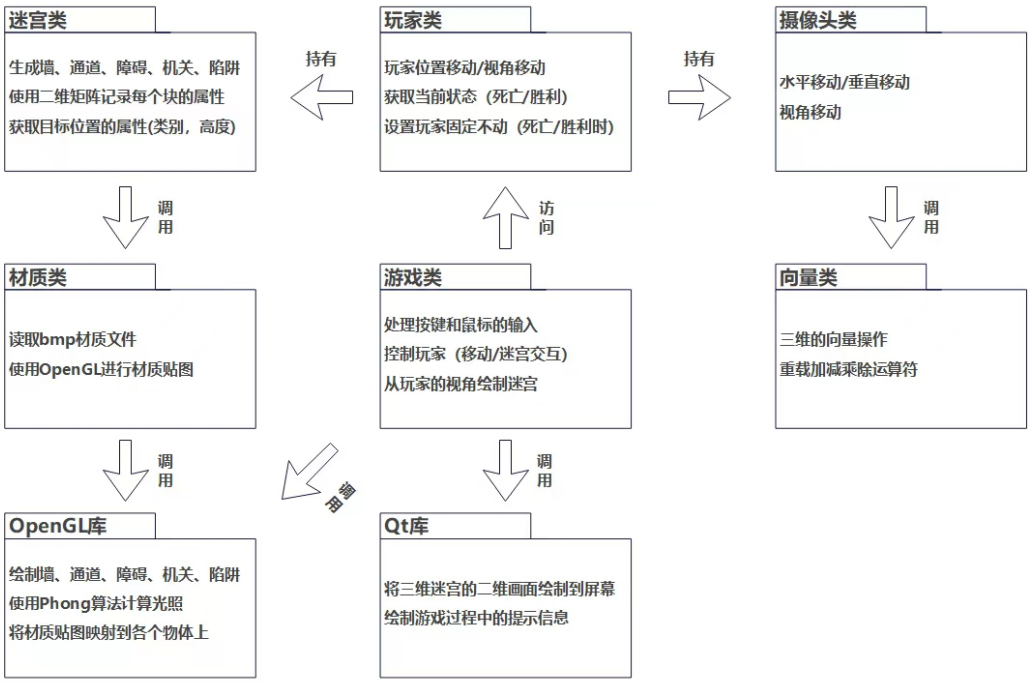


图 1: 系统架构图

3 环境要求

3.1 硬件要求

- **图形处理器（GPU）**:
  - 支持 OpenGL 3.3 或更高版本的图形卡。现代的集成显卡（如 Intel HD Graphics）和独立显卡（如 NVIDIA 或 AMD）一般都支持.
- **处理器（CPU）**:
  - 该软件可以在所有常见的处理器上流畅运行, 如果你的处理器型号特殊请咨询处理器厂商.
- **内存（RAM）**:
  - 至少 2GB，推荐 8GB 及以上以保证开发环境和运行时的流畅性.
- **存储**:
  - 至少 512MB 的储存空间.
- **显示器**:
  - 支持至少 1080p 分辨率的显示器.

3.2 软件要求

- **操作系统**:
  - Windows 10 或更高版本，Linux（内核版本 2.6.32 或更新, 否则可能不支持 OpenGL3.3），macOS 10.15 及以上版本.
- **开发工具和库**:
  - **Qt**:
    - 安装 Qt Creator IDE 及 Qt 库（推荐使用 Qt 5.15 或更高版本）。
    - 可以通过Qt下载。
  - **OpenGL**:
    - 安装相关的开发工具包（SDK），大多数操作系统自带 OpenGL 支持.
    - 对于 Windows，可以通过安装适合的图形驱动程序获得最新的 OpenGL 支持.

- **编译器**:
  - GCC（对于 Linux），Clang（对于 macOS），MSVC（对于 Windows）等.
  - 推荐使用 Qt Creator 中集成的编译器.
- **其他工具**:
  - CMake（用于构建和管理项目）.
  - IDE (推荐使用 Visual Studio2022).

3.3 依赖库

- **GLEW**（OpenGL Extension Wrangler Library）:
  - 提供跨平台的 OpenGL 扩展功能加载。
- **GLFW** 或 **SDL2**:
  - 用于创建窗口和处理输入（如果不使用 Qt 的窗口管理功能）。
- **GLM**:
  - 提供 OpenGL 的数学库，方便处理向量和矩阵运算。

4 安装与部署

4.1 安装步骤

- **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:

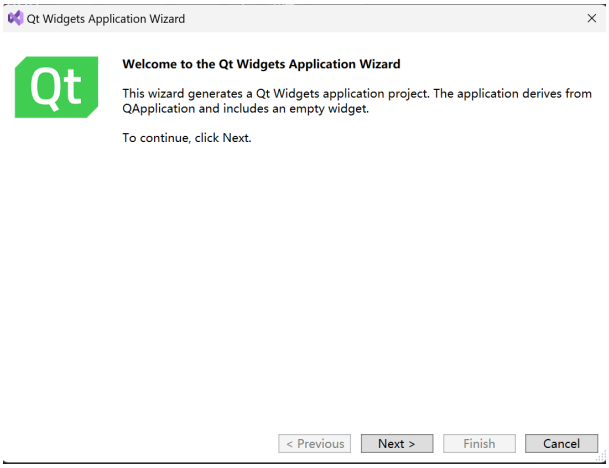


图 2: Qt Wizard

  - 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/<version>/mingw_64/bin/qmake.exe

- **安装 OpenGL 相关库**:
  - 使用包管理器安装（如 `apt`、`brew` 或 `vcpkg`）。

- **创建 Qt 和 OpenGL 集成项目**:
  - 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

5 运行与维护

5.1 启动与停止

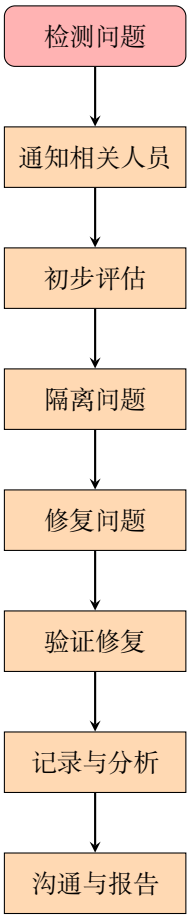
如果你正确配置了上述环境, 你只需点击 Visual Studio 的本地 Windows 调试器按钮, 运行/关闭生成的.exe 文件即可启动/停止该软件.

5.2 日志管理

所有日志信息均在终端输出。

5.3 故障处理流程

在生产环境中遇到故障时，我们遵循以下流程以确保快速响应和恢复系统：



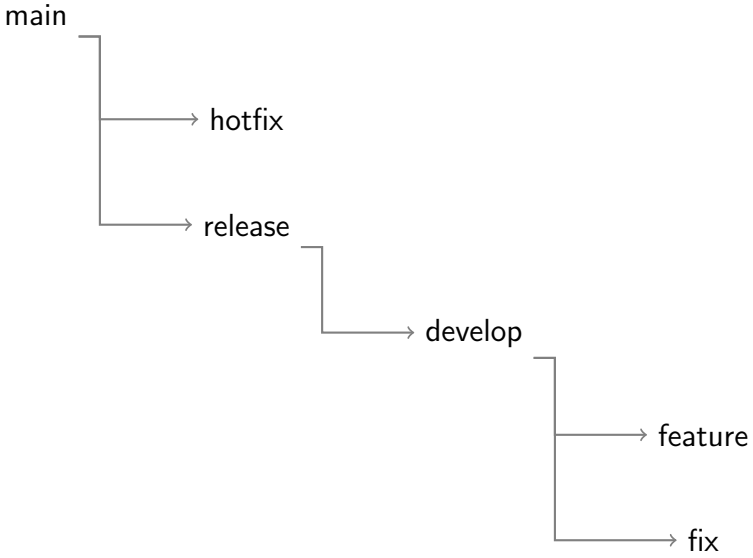
1. **检测问题**：通过监控系统或用户报告发现问题，确定问题的范围和影响。
2. **通知相关人员**：通过邮件、电话或即时通讯工具通知团队成员。启动应急响应团队，包括开发人员、运维人员和项目经理。
3. **初步评估**：分析问题症状，初步确定问题原因。决定是否需要立即采取措施（如暂停某些服务或功能）。
4. **隔离问题**：尽量将问题隔离，减少对其他系统或服务的影响。如有必要，回滚最近的更改或切换到备用系统。
5. **修复问题**：根据初步评估结果，制定并实施修复计划，可能包括代码修复、配置修改、重启服务等。
6. **验证修复**：确认问题已解决，系统恢复正常运行。进行全面系统测试，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。

5.4 联系人员

角色	姓名	联系方式
开发负责人	李英骏	liyj323@mail2.sysu.edu.cn

6 版本控制

我们采用如下的分支控制, 在 Git 中进行版本管理.



- 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。

7 持续集成

7.1 工具和服务

- 持续集成服务：Travis CI
- 构建工具：CMake
- 测试框架：Google Test
- 代码质量检查：Cppcheck, Clang-Tidy

7.2 CI 配置

以下是使用 Travis CI 的配置文件 '.travis.yml'：

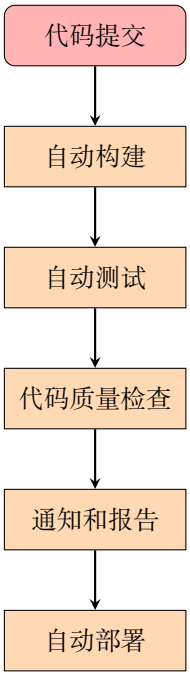
```
language: cpp
os:
  - linux
  - windows

compiler:
  - gcc
  - msvc

before_install:
  - if [[ "$TRAVIS_OS_NAME" == "windows" ]]; then choco install qt5; fi

script:
  - mkdir build
  - cd build
  - cmake ..
  - cmake --build .
  - ctest
```

7.3 持续集成流程



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。

8 代码质量保证

8.1 静态代码分析

使用 Cppcheck 和 Clang-Tidy 进行静态代码分析，确保代码质量。

8.2 单元测试

使用 Google Test 编写和运行单元测试，确保每个模块的正确性。

8.3 代码审查

通过 Pull Request 进行代码审查，确保代码的质量和一致性。

9 参考资料

1. OpenGL: OpenGL
2. Qt5::Qt5