灵棋幻境 2.0 需求文档

目录

1. 项目概述

1.1 项目名称

灵棋幻境

1.2 项目目标

基于 Qt Creator，使用 QML（界面）+C++（简单逻辑）开发一款带有简易剧情的五子棋游戏，使用 Git 协作开发，技术难度适中，功能完整且简洁，巩固 Git 协作、QML 界面设计、C++基础逻辑能力，完成一款可正常运行、交互流畅的小型桌面游戏。

1.3 面向用户

小组内部测试、小组进步学习

开发团队

云栈筑梦开发小组（共 11 人），使用 Git+Github 进行协作开发，Qt Creator 作为开发工具。

2. 项目开发环境

本项目开发环境统一，避免因环境差异导致的开发问题，所有成员需配置以下环境，新手可参考小组内已配置完成的成员步骤操作:

- 开发工具：Qt Creator 12.0 及以上
- 编程语言：QML（用于界面设计，核心使用基础组件）、C++（用于简单逻辑编写，不涉及复杂语法和设计模式）
- 版本控制：Git（基础操作：拉取、提交、推送、分支创建与合并）、Github（代码托管，创建小组仓库）
- 运行环境：Windows 10/11（64 位），确保 Qt Creator 可正常编译运行 QML+C++项目，无需额外安装其他复杂依赖
- 辅助工具：截图工具、记事本/文档工具

3. 需求

本项目核心是“简易界面剧情+基础五子棋”，兼顾开发难度和功能完整性，所有需求均围绕“简单、可实现、易协作”展开，禁止使用复杂技术，具体核心需求如下：

- 功能需求：实现基础五子棋对战、简易剧情展示、界面正常切换、基础操作反馈。
- 界面需求：界面简洁、操作直观，使用 QML 基础组件搭建，无需复杂动画和美化，确保每个界面可正常跳转。
- 技术需求：QML 负责界面布局和简单交互，C++负责五子棋核心逻辑（落子、胜负判断），不使用复杂 C++语法（如虚函数、模板、STL 复杂容器等），不使用 Qt 复杂模块。
- 协作需求：适配 Git 协作，代码结构清晰，每个功能模块独立，便于 11 人分工开发、提交代码，避免代码冲突。
- 运行需求：项目可正常编译运行，无崩溃、无报错，落子流畅，剧情展示正常，操作无明显卡顿。

4. 界面功能需求

本项目共设计 5 个核心界面，所有界面均使用 QML 基础组件（按钮、文本、矩形框、图片等）搭建，界面之间跳转逻辑清晰，操作简单，每个界面的元素、功能、交互均详细说明，新手可直接参考实现，具体如下：

4.1 启动界面（首页）

4.1.1 界面元素

- 标题：居中显示“灵棋幻境”（字体稍大，颜色自定义，无需复杂字体）
- 副标题：“黑白巧思融雅趣，方圆竞弈悟玄机”（字体略小，颜色浅于标题）
- 功能按钮（4 个，垂直排列，居中显示，按钮大小一致、样式统一）：
    - “开始游戏”按钮：位于副标题下方，点击进入剧情界面
    - “游戏规则”按钮：位于“开始游戏”按钮下方，点击进入规则界面
    - “退出游戏”按钮：位于“游戏规则”按钮下方，点击关闭整个应用程序
    - “开发人员”按钮：位于“退出游戏”按钮下方，点击显示出开发人员信息
- 背景：可使用纯色背景（如浅灰色、浅蓝色），可不添加复杂图片背景

- 底部文本：界面最下方居中显示 “© 2026 云栈筑梦 ”（小字体）

4.1.2 界面功能

- 按钮交互：点击对应按钮执行对应操作，按钮点击时有简单反馈（如颜色变深，无需动画），点击后界面正常跳转，无卡顿。
- 窗口大小：固定窗口大小（如 800x600 像素），不可拉伸、不可最大化。
- 退出功能：点击“退出游戏”按钮，直接关闭应用程序，无弹窗提示（简化操作）。

4.1.3 开发注意事项

使用 QML 的 Rectangle 组件做背景，Text 组件做标题和文本，Button 组件做功能按钮，布局使用 ColumnLayout（垂直布局）确保元素居中，无需自定义组件，代码简洁易懂。

注：若不影响其他功能元素，可使用自定义/第三方组件

4.2 剧情界面

剧情为线性剧情，内容简单，用于衔接启动界面和游戏界面，具体如下：

4.2.1 界面元素

- 剧情文本框：居中显示，占据界面上半部分（宽度 700 像素，高度 300 像素），边框为简单黑色线条，内部显示剧情文本（换行显示，字体适中）。
- 角色提示：剧情文本框左上角显示角色名称，字体加粗，与剧情文本区分开。
- 功能按钮（2 个，水平排列，居中显示，位于剧情文本框下方）：
    - “继续”按钮：点击显示下一段剧情文本，当显示到最后一段剧情时，点击进入五子棋游戏界面。
    - “跳过剧情”按钮：点击直接跳过所有剧情，进入五子棋游戏界面。
- 背景：与启动界面背景一致（纯色），保持界面统一性，无需额外添加元素。

4.2.2 界面功能

- 剧情展示：默认显示第一段剧情文本，点击“继续”按钮，依次显示后续剧情（共 3-4 段，每段文本简短，不超过 50 字），剧情文本无滚动效果，直接替换显示。
- 跳转功能：最后一段剧情点击“继续”，或任意时段点击“跳过剧情”，均跳转至五子

棋游戏界面，跳转过程无延迟。

- 按钮反馈：与启动界面按钮反馈一致，点击时颜色变深，无其他复杂交互。

4.2.3 剧情内容

剧情共 4 段，文本简洁，要贴合五子棋主题，具体如下：

1. 【神秘人】：欢迎来到灵棋幻境！击败你的对手，洞悉棋境隐秘
2. 【小棋童】：规则很简单，黑白棋子轮流落子，先将五颗棋子连成一条直线（横、竖、斜均可），便可获胜
3. 【小棋童】：棋境里有很多强大的存在，对弈时你要小心
4. 【神秘人】：祝你好运，再见

4.2.4 开发注意事项

使用 QML 的 TextEdit 或 Text 组件做剧情文本框，Button 组件做按钮，布局使用 ColumnLayout+RowLayout（垂直+水平），剧情文本使用数组存储，点击“继续”时切换数组索引，实现文本替换，可以不用复杂逻辑。

4.3 游戏界面

本界面是项目核心，实现五子棋对战功能，逻辑简单，操作直观，避免复杂算法，具体如下：

4.3.1 界面元素

- 棋盘：居中显示，使用 15x15 的网格，网格为黑色线条，棋盘背景为浅木色（使用纯色模拟，可不用图片），棋盘大小固定（如 500x500 像素）。
- 棋子：黑白两色圆形棋子（纯色填充，无纹理），黑色棋子由玩家控制，白色棋子由电脑控制，棋子大小适配棋盘格子（略小于格子，避免重叠）。
- 状态提示区：位于棋盘上方，居中显示，包含 2 个文本：
    - 回合提示：“当前回合：玩家（黑棋）”或“当前回合：电脑（白棋）”，字体加粗，颜色区分（玩家回合显示黑色，电脑回合显示红色）。
    - 胜负提示：默认为空，当出现胜负时，显示“玩家获胜！”或“电脑获胜！”，颜色为红色，字体稍大。
- 功能按钮（3 个，垂直排列，位于棋盘右侧，按钮大小一致）：

- “悔棋”按钮：点击悔棋（仅允许玩家悔棋 1 步，电脑不悔棋），悔棋后恢复上一步棋盘状态，回合切换回玩家。
- “重新开始”按钮：点击重置棋盘（清空所有棋子），重置回合为玩家，清空胜负提示，重新开始对战。
- “返回主菜单”按钮：点击关闭当前游戏界面，返回启动界面。

- 背景：与前两个界面一致（纯色），保持统一性。

4.3.2 界面功能

4.3.2.1 落子功能

- 玩家落子：点击棋盘上的空白格子，落下黑色棋子，落子后不可移动，落子成功后，回合切换为电脑，状态提示区更新回合信息。
- 电脑落子：玩家落子后，电脑自动在空白格子中落子，落子后回合切换为玩家，状态提示区更新回合信息。
- 落子限制：不可在已有棋子的格子落子，点击已有棋子的格子无反应，无提示。

4.3.2.2 胜负判断功能

- 判断时机：玩家和电脑每次落子后，仅判断当前落子位置的横、竖、左斜、右斜四个方向，是否有连续 5 颗相同颜色的棋子（简化判断逻辑，无需遍历整个棋盘）。
- 胜负结果：
  - 若玩家落子后，形成 5 子连线，显示“玩家获胜！”，停止游戏（玩家和电脑均不可再落子）。
  - 若电脑落子后，形成 5 子连线，显示“电脑获胜！”，停止游戏（玩家和电脑均不可再落子）。
- 平局处理：暂不实现平局。

4.3.2.3 其他功能

- 悔棋功能：仅玩家可悔棋，点击“悔棋”按钮，删除玩家上一步落下的棋子，回合切换回玩家，若玩家未落子（刚启动游戏），点击悔棋无反应。
- 重新开始功能：点击后，清空棋盘所有棋子，重置回合为玩家，清空胜负提示，游戏恢复初始状态，可重新开始对战。
- 返回主菜单功能：点击后，直接关闭游戏界面，返回启动界面，当前游戏进度不保存。

4.3.3 开发注意事项

- 棋盘：使用 QML 的 Grid 组件搭建 15x15 网格，每个格子使用 Rectangle 组件，设置黑色边框，背景为浅木色。
- 棋子：使用 QML 的 Ellipse 组件，黑色和白色填充，落子后添加到棋盘对应格子中，使用数组存储棋盘状态。
- 逻辑分工：QML 负责棋盘、棋子的显示和点击交互（获取点击的格子位置），C++负责落子逻辑、胜负判断、悔棋逻辑。
- 电脑 AI：简单落子即可，遍历二维数组，找到空白格子，随机选择一个位置落子，可以不用复杂的 AI 算法。

4.4 游戏规则界面

用于展示五子棋基础规则，文本简洁，无复杂交互，具体如下：

4.4.1 界面元素

- 标题：居中显示“游戏规则”（字体稍大，与启动界面标题样式一致）。
- 规则文本框：居中显示，位于标题下方，宽度 700 像素，高度 400 像素，边框为黑色线条，内部显示规则文本（换行显示，字体适中，分点列出，清晰易懂）。
- “返回主菜单”按钮：居中显示，位于规则文本框下方，按钮样式与启动界面一致，点击返回启动界面。
- 背景：界面风格不能相差太多。

4.4.2 界面功能

- 规则展示：规则文本固定，无需修改，分点列出，清晰易懂，适合新手查看。
- 跳转功能：点击“返回主菜单”按钮，直接返回启动界面，无其他交互。

4.4.3 规则内容

1. 游戏双方：玩家控制黑棋，电脑控制白棋，玩家先落子，双方轮流落子。
2. 落子规则：点击棋盘空白格子即可落子，落子后不可移动、不可更改。
3. 获胜规则：先将五颗相同颜色的棋子连成一条直线（横、竖、斜均可），即为获胜。
4. 特殊操作：玩家可点击“悔棋”按钮，悔棋 1 步（仅玩家可悔棋）；点击“重新开始”按钮，可重置游戏。

4.4.4 开发注意事项

使用 QML 的 Text 组件做规则文本，Button 组件做返回按钮，布局使用 ColumnLayout，确保元素居中，代码简洁。

4.5 胜负结果界面

游戏结束（玩家或电脑获胜）后，弹出该界面，展示结果，提供简单操作，简化逻辑，具体如下：

4.5.1 界面元素

- 结果提示：居中显示，字体较大，颜色红色，显示“玩家获胜！”或“电脑获胜！”。
- 提示文本：结果提示下方居中显示，字体适中，颜色自定义，显示“恭喜你！”（玩家获胜）或“再试一次吧！”（电脑获胜）。
- 功能按钮（2 个，水平排列，居中显示，位于提示文本下方，样式与其他界面一致）：
    - “再来一局”按钮：点击关闭结果界面，重置棋盘，重新开始游戏（回合为玩家，无剧情）。
    - “返回主菜单”按钮：点击关闭结果界面，返回启动界面。
- 背景：与其他界面一致（纯色），保持统一性。

4.5.2 界面功能

- 结果显示：游戏结束后自动弹出该界面，动态显示获胜方（根据 C++传递的结果，显示对应文本）。
- 跳转功能：
    - 点击“再来一局”，关闭结果界面，棋盘重置，回合为玩家，可直接开始落子。
    - 点击“返回主菜单”，关闭结果界面，返回启动界面。
- 界面弹出：游戏结束后自动弹出，无需玩家点击，弹出后游戏界面不可操作（简化逻辑）。

4.5.3 开发注意事项

使用 QML 的 Window 或 Rectangle 组件做弹出界面，Text 组件显示结果和提示文本，Button 组件做功能按钮，通过 C++传递的获胜结果（如 1 表示玩家获胜，2 表示电脑

获胜），动态修改结果提示文本，无需复杂的弹窗动画。

5. 剧情需求

剧情暂不要求开发，不增加开发难度，具体要求如下：

- 剧情数量：固定 4 段（详见 4.2.3），无分支、无选择，线性展示，无需动态生成剧情。
- 剧情触发：在启动界面点击“开始游戏”后触发，进入剧情界面，展示剧情，跳过剧情后，后续游戏（再来一局等）不再触发剧情。
- 剧情交互：支持“继续”和“跳过”，可选其他交互（如点击角色、剧情选项等），简化开发。
- 文本要求：剧情文本简洁，语言通俗易懂，贴合灵棋幻境主题。

6. 技术限制

为避免遇到难以解决的技术问题，本项目严格限制以下技术，所有成员必须遵守，禁止使用过于复杂技术，具体如下：

- QML 限制：尽量使用基础组件（Rectangle、Text、Button、Ellipse、Grid、ColumnLayout、RowLayout），尽量不使用自定义组件、复杂动画（如 Rotation、Scale 等）、Qt Quick Controls 2 中的复杂组件（如 ListView、TableView 等），尽量不使用图片资源。
- C++限制：仅使用基础语法（变量、循环、条件判断、二维数组），尽量不使用复杂语法（STL 复杂容器如 vector、map 等），尽量不使用 Qt 复杂模块（网络、数据库、多线程等），C++仅负责简单逻辑，不参与界面绘制。
- 逻辑限制：不实现复杂功能（如存档读档、网络对战、多难度 AI、平局判断、棋子动画等），所有逻辑追求“简单可实现”，优先保证功能正常运行，不追求极致性能和体验。
- Git 协作限制：仅使用基础 Git 操作（拉取、提交、推送、创建个人功能分支、合并分支），禁止直接向主分支提交代码，每个成员负责一个界面或一个功能模块，提交代码时添加清晰注释，避免代码冲突。

7. 团队协作说明

为便于小组使用 Git 协作开发，明确分工和协作流程，避免混乱，具体如下：

7.1 分工建议（仅参考）

- 1 人：负责项目初始化：创建 Qt 项目、配置 Git 仓库、提交初始代码，供其他成员拉取。
- 2 人：负责启动界面：1 人写 QML 界面布局，1 人写按钮交互逻辑，配合完成。
- 2 人：负责剧情界面：1 人写 QML 界面布局，1 人写剧情文本切换逻辑，配合完成。
- 3 人：负责五子棋游戏界面：1 人写 QML 棋盘、棋子布局，1 人写 C++落子和悔棋逻辑，1 人写 C++胜负判断逻辑，配合完成。
- 1 人：负责游戏规则界面：写 QML 界面布局和文本展示，独立完成。
- 1 人：负责胜负结果界面：写 QML 界面布局和动态结果显示，独立完成。
- 1 人：负责测试和整合：测试所有界面和功能，提交 Bug，协助整合 Github 仓库所有模块，确保项目可正常运行。

7.2 Git 协作流程

1. 组长创建 Github 仓库，初始化 Qt 项目（选择 QML Application，添加基础文件），提交初始代码，设置仓库规则权限。
2. 每个成员从 Github 仓库拉取初始代码，创建个人功能分支（分支命名要清晰简介）。
3. 成员在个人分支上开发自己负责的模块，开发过程中定期拉取主分支代码（避免冲突），完成后提交代码（添加清晰注释），推送至个人分支。
4. 成员提交 Pull Request（PR），请求整合负责人合并个人分支到主分支，整合负责人审核代码，审核通过后合并分支。
5. 合并后，所有成员拉取主分支最新代码，继续开发或测试，避免使用过时代码。

7.3 代码规范

- 变量命名：使用简单易懂的名称（如 startButton、chessBoard、blackChess），不使用复杂缩写，QML 变量和 C++变量区分开（QML 变量加 qml 前缀，C++变量不加）。
- 代码注释：关键代码添加注释（落子逻辑、胜负判断逻辑），注释简洁易懂，说明代码功能，便于其他成员查看和修改。
- 文件结构：按界面和功能划分文件（如 startwindow.qml、storywindow.qml、gamewindow.qml、chesslogic.h、chesslogic.cpp），文件名称清晰，便于查找。

8. 验收标准

项目完成后，对照以下标准验收，确保功能正常，符合需求，具体如下：

- 界面验收：5 个核心界面均可正常显示，元素完整，无缺失，界面之间跳转正常（点击按钮可正常跳转，无卡顿、无报错）。
- 剧情验收：剧情可正常展示，点击“继续”可切换文本，点击“跳过剧情”可直接进入游戏，剧情无错乱、无缺失。
- 游戏验收：
    - 落子正常：玩家可点击空白格子落黑棋，电脑可自动落白棋，不可在已有棋子的格子落子。
    - 胜负判断正常：玩家和电脑落子后，可正确判断胜负，显示对应结果，游戏停止。
    - 辅助功能正常：悔棋、重新开始、返回主菜单等按钮可正常使用，功能符合需求。
- 技术验收：未使用禁止的复杂技术，代码可正常编译运行，无崩溃、无报错，Git 协作流程正确，代码结构清晰。
- 运行验收：项目可在 Windows 10/11 上正常运行，落子流畅，无明显卡顿，操作直观，适合新手使用。

9. 补充说明

- 本需求文档针对学习目的设计，所有功能和技术均以“简单、可实现”为核心，若有成员遇到技术问题，要协作解决，优先使用基础方法实现，不追求复杂效果。
- 项目开发过程中，可根据小组实际情况，在不增加技术难度的前提下，调整 qml 界面样式（如颜色、字体大小），但不可修改核心功能和界面结构。
- Git 协作过程中，若出现代码冲突，及时联系组长和相关成员，共同解决，避免擅自修改他人代码。