清华大学云盘链接: https://cloud.tsinghua.edu.cn/library/175edf3e-7bae-4a60-872b-6c077c425f92/%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A12/
Github 链接: https://github.com/Sky-Trigger/Object-Oriented-Technology-phase2

面向对象大作业 实验报告 棋类对战平台（第二阶段）

一、设计思路与选用的设计模式

1.1 总体架构设计
系统分为三层：核心层(core)管理游戏状态、AI、用户和录像，游戏层(games)实现五子棋、围棋和黑白棋规则，界面层(ui)提供 GUI。分层分离逻辑和界面，便于维护扩展。第二阶段新增 AI 模块、录像管理、用户账户管理，保持低耦合。

1.2 设计模式应用
(1) 模板方法模式：GameEngine 基类定义游戏流程，子类实现具体规则，便于扩展新游戏如黑白棋。
(2) 工厂方法模式：create_engine()根据游戏类型创建引擎，解耦客户端代码。
(3) 策略模式：AI 类（RandomAI、ScoringAI）实现不同 AI 算法，无差异对待玩家和 AI。
(4) 观察者模式：ReplayManager 管理录像状态，GUI 监听更新。
(5) 单例模式：UserManager 管理全局用户数据。

1.3 面向对象设计原则体现
(1) 单一职责：各类职责明确，如 AI 类只负责落子，UserManager 只管账户。
(2) 开闭原则：新增黑白棋、AI 只需继承，无需修改现有代码。
(3) 里氏替换：AI 子类可替换使用，接口一致。
(4) 依赖倒置：界面通过 Controller 依赖抽象，而非具体实现。
(5) 接口隔离：精简接口，各模块按需使用。

1.4 关键设计决策
(1) AI 集成：AI 作为策略，无缝替换玩家，简化玩家-AI 对战。
(2) 录像回放：保存每步移动，支持自动和手动播放。
(3) 用户管理：本地 JSON 存储，支持游客模式。
(4) 资源管理：追踪棋子，支持黑白棋翻转。
(5) 序列化：完整保存状态，使用 JSON，支持录像和游戏存档。
(6) 异常处理：业务错误抛 ValueError，GUI 友好提示。
(7) 精简代码：移除装饰器、抽象类、CLI，专注于 GUI 和核心功能。

二、关键类和函数的简单说明

2.1 核心层(core/)

[Board 类]core/board.py
功能: 管理棋盘
主要方法: __init__(size), get/set(position), is_within_bounds(), reset(), serialize()/deserialize()
实现: 二维列表存储，边界检查，支持 JSON 序列化

[GameEngine 类]core/game_engine.py
功能: 游戏引擎基类
主要属性: board, current_player, history, captured_by_color, max_undo
主要方法: play_move()(抽象), pass_turn(), undo(), resign(), restart(), is_finished(), serialize()/deserialize(), stones_remaining(), undo_remaining()
实现: 模板方法模式，记录资源数据，完整状态保存

[GameController 类]core/controller.py
功能: 协调各组件
关键方法: start_game(), place_stone(), pass_turn(), undo(), resign(), restart(), save_replay/load_replay(), jump_to_replay(), start_replay/stop_replay(), set_ai(), make_ai_move(), register_user/login_user(), update_user_stats(), get_current_user/get_user_stats()
亮点: 统一接口，AI 集成，录像控制，用户管理，中文化显示

[Models 模块]core/models.py
功能: 数据模型和枚举
关键类: GameType, PlayerColor, Position, Move, GameResult
亮点: 枚举类型安全，Position 支持哈希，Move 记录提子

[AI 模块]core/ai.py
功能: AI 算法实现
关键类: RandomAI, ScoringAI
主要方法: get_move(board, valid_moves)
实现: 随机选择或评分函数，适用于黑白棋

[Replay 模块]core/replay.py
功能: 录像管理
关键类: ReplayManager
主要方法: start_replay(), stop_replay(), next_move(), jump_to(), get_current_board_state()
实现: 管理移动列表，支持跳转和播放

[UserManager 类]core/user_manager.py
功能: 用户账户管理
主要方法: register(), login(), login_guest(), logout(), update_stats(), get_current_user(), get_stats()
实现: JSON 本地存储，支持游客

[Persistence 模块]core/persistence.py
功能: 存档和录像保存加载
关键函数: save_game(), load_game(), save_replay(), load_replay()
亮点: 自动创建目录，UTF-8 编码，统一异常处理

2.2 游戏层(games/)

[GomokuEngine 类]games/gomoku.py
功能: 五子棋引擎
主要方法: play_move(), _check_five_in_row(), _count_in_direction(), _undo_internal()
判断输赢: 检查四个方向连子≥5

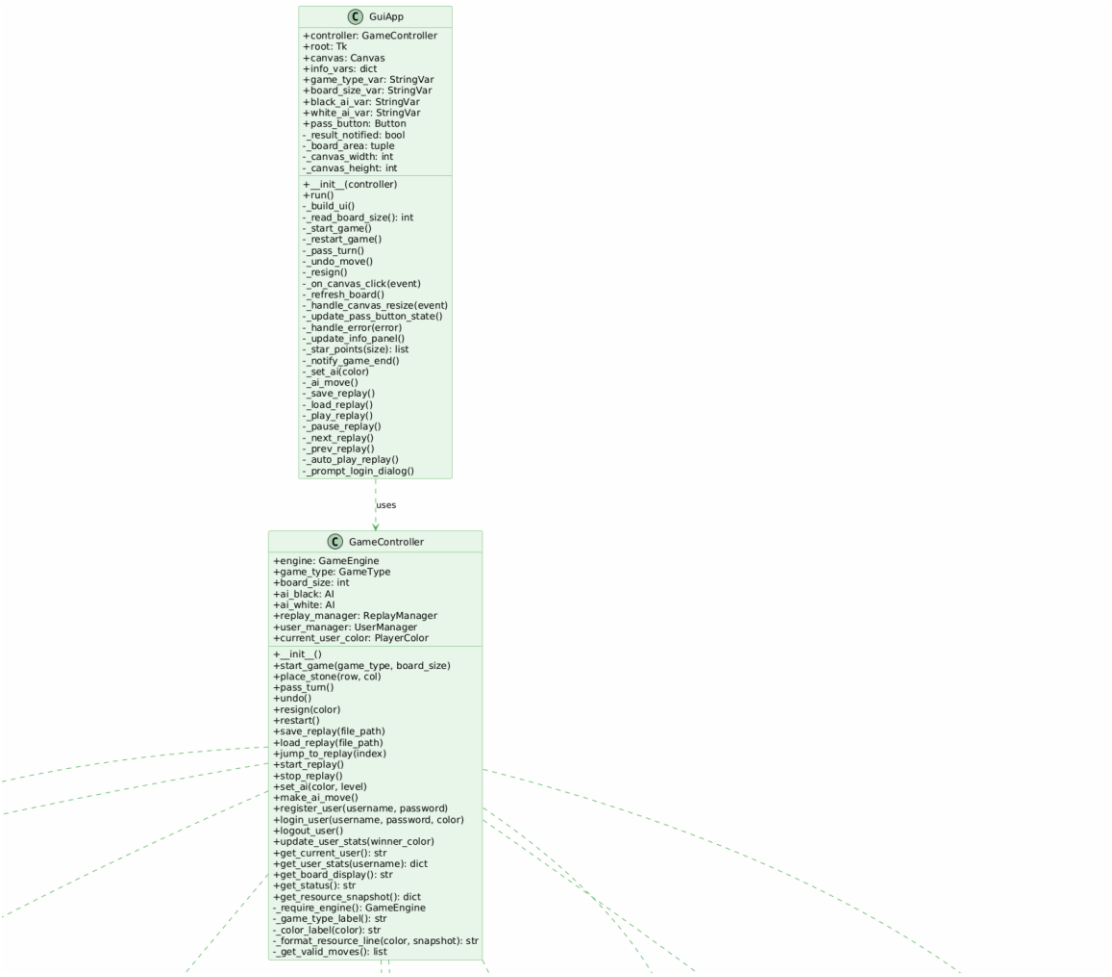
[GoEngine 类]games/go.py
功能: 围棋引擎
主要方法: play_move(), pass_turn(), _capture_adjacent_groups(), _has_liberty(), _collect_group(), _score_game(), _compute_territory(), _undo_internal()
实现: 栈遍历棋组，提子判断，禁入点检测，数子+提子+地盘算分

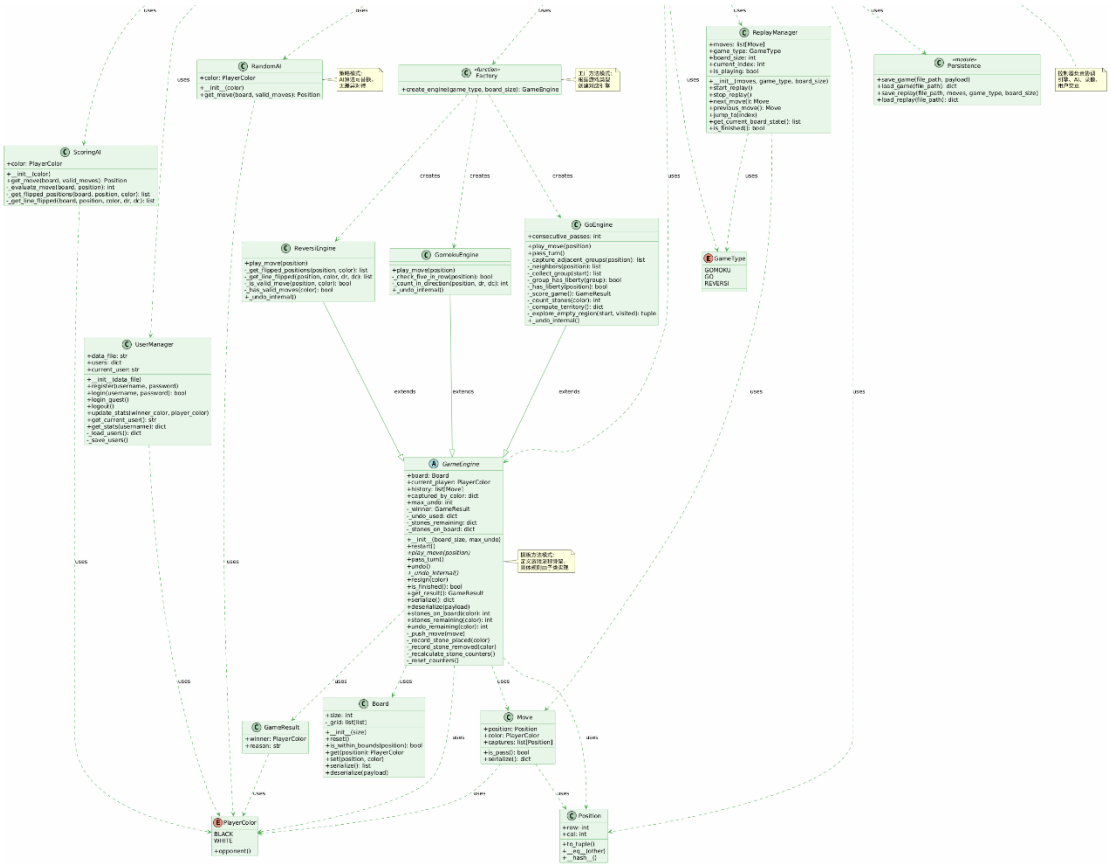
[ReversiEngine 类]games/reversi.py
功能: 黑白棋引擎
主要方法: play_move(), _get_flipped_positions(), _get_line_flipped(), _is_valid_move(), _has_valid_moves(), _undo_internal()
实现: 翻转对手棋子，检查合法位置，8x8 棋盘

2.3 界面层(ui/)

[GuiApp 类]ui/gui.py
功能: 图形界面
主要方法: __init__(), _build_ui(), _start_game(), _on_canvas_click(), _refresh_board(), _update_info_panel(), _notify_game_end(), _set_ai(), _ai_move(), _play_replay/_pause_replay(), _next_replay/_prev_replay(), _prompt_login_dialog()
界面: 控制面板（AI 设置、录像控制）、信息面板（用户战绩）、棋盘画布、登录对话框
特点: 响应式布局，自动星位，彩色显示，结束弹窗，AI 集成，录像播放

三、UML 类图


GuiApp
+controller: GameController
+root: Tk
+canvas: Canvas
+info_vars: dict
+game_type_var: StringVar
+board_size_var: StringVar
+black_ai_var: StringVar
+white_ai_var: StringVar
+pass_button: Button
-_result_notified: bool
-_board_area: tuple
-_canvas_width: int
-_canvas_height: int
+__init__(controller)
+run()
-_build_ui()
-_read_board_size(): int
-_start_game()
-_restart_game()
-_pass_turn()
-_undo_move()
-_resign()
-_on_canvas_click(event)
-_refresh_board()
-_handle_canvas_resize(event)
-_update_pass_button_state()
-_handle_error(error)
-_update_info_panel()
-_star_points(size): list
-_notify_game_end()
-_set_ai(color)
-_ai_move()
-_save_replay()
-_load_replay()
-_play_replay()
-_pause_replay()
-_next_replay()
-_prev_replay()
-_auto_play_replay()
-_prompt_login_dialog()
uses
GameController
+engine: GameEngine
+game_type: GameType
+board_size: int
+ai_black: AI
+ai_white: AI
+replay_manager: ReplayManager
+user_manager: UserManager
+current_user_color: PlayerColor
+__init__()
+start_game(game_type, board_size)
+place_stone(row, col)
+pass_turn()
+undo()
+resign(color)
+restart()
+save_replay(file_path)
+load_replay(file_path)
+jump_to_replay(index)
+start_replay()
+stop_replay()
+set_ai(color, level)
+make_ai_move()
+register_user(username, password)
+login_user(username, password, color)
+logout_user()
+update_user_stats(winner_color)
+get_current_user(): str
+get_user_stats(username): dict
+get_board_display(): str
+get_status(): str
+get_resource_snapshot(): dict
-_require_engine(): GameEngine
-_game_type_label(): str
-_color_label(color): str
-_format_resource_line(color, snapshot): str
-_get_valid_moves(): list



四、测试用例与程序测试过程的输入输出

4.1 五子棋功能测试（继承阶段一）

[测试用例 1: 正常对局]
操作: GUI 选择五子棋 15 路，开始，落子至五连。
期望: 弹出胜者对话框。
实际: ✓ 符合预期

[测试用例 2: 悔棋]
操作: 落子后悔棋。
期望: 棋盘恢复，悔棋余量减少。
实际: ✓ 符合预期

4.2 围棋功能测试（继承阶段一）

[测试用例 3: 提子]
操作: 落子形成包围。
期望: 对手棋子被提。
实际: ✓ 符合预期

[测试用例 4: 虚手胜负]
操作: 连续虚手。

期望: 计算得分，显示胜者。
实际: ✓ 符合预期

4.3 黑白棋功能测试

[测试用例 5: 正常对局]
操作: GUI 选择黑白棋 8 路，开始，落子翻转对手棋子。
期望: 棋盘正确翻转，显示棋子数量。
实际: ✓ 符合预期

[测试用例 6: 合法落子检查]
操作: 尝试落子非法位置。
期望: 提示非法，无效。
实际: ✓ 符合预期

[测试用例 7: 结束判定]
操作: 棋盘填满或无合法落子。
期望: 显示胜者和分数。
实际: ✓ 符合预期

4.4 AI 功能测试

[测试用例 8: 随机 AI]
操作: 设置黑方 AI 等级 1，开始对战。
期望: AI 随机落子，游戏正常进行。
实际: ✓ 符合预期

[测试用例 9: 评分 AI]
操作: 设置白方 AI 等级 2。
期望: AI 选择高分位置，战胜随机 AI。
实际: ✓ 符合预期

[测试用例 10: AI-AI 对战]
操作: 设置双方 AI。
期望: 自动对战，显示结果。
实际: ✓ 符合预期

4.5 录像与回放测试

[测试用例 11: 播放录像]
操作: 进行几步对局，点击播放录像。
期望: 从当前游戏历史播放，每秒自动播放一步，至结束弹出提示。
实际: ✓ 符合预期

[测试用例 12: 手动控制]
操作: 暂停、下一步、上一步。
期望: 正确跳转棋盘状态。
实际: ✓ 符合预期

4.6 用户账户管理测试

[测试用例 13: 注册登录]
操作: 启动游戏，弹出登录，注册新用户，选择角色登录。
期望: 登录成功，信息面板显示用户名。
实际: ✓ 符合预期

[测试用例 14: 游客模式]
操作: 选择游客登录。
期望: 无需密码，显示游客。
实际: ✓ 符合预期

[测试用例 15: 战绩更新]
操作: 登录用户完成对局。
期望: 战绩更新，显示对战场次和胜场。
实际: ✓ 符合预期

4.7 GUI 界面测试

[测试用例 16: 信息面板显示]
操作: 登录用户，开始游戏。
期望: 信息面板显示用户名、战绩、AI 状态、资源。
实际: ✓ 符合预期

[测试用例 17: 响应式布局]
操作: 调整窗口大小。
期望: 棋盘居中缩放。
实际: ✓ 符合预期

4.8 异常处理测试

[测试用例 18: AI 设置]
操作: 设置无效 AI 等级。
期望: 提示错误。
实际: ✓ 符合预期

[测试用例 19: 录像操作]
操作: 无录像时播放。
期望: 提示请先加载。

实际: ✓ 符合预期

4.9 性能测试

[测试用例 20: 大棋盘黑白棋]
操作: 8x8 黑白棋，AI 对战。
期望: 响应流畅。
实际: ✓ 符合预期