棋类对战平台大作业报告

# 小组成员：

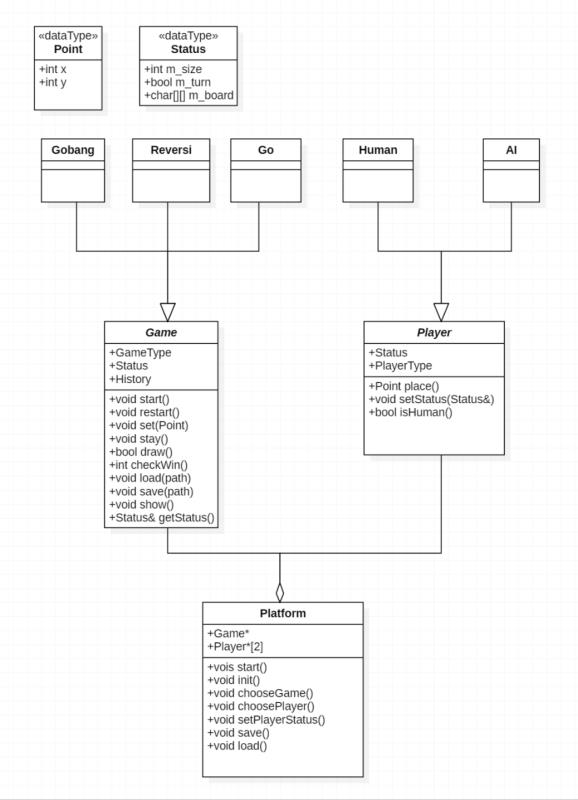
计科40洪鑫焰 2014011378，计科40魏子豪 2014012149

# 主要工作：

我们小组实现了一个完整的棋类对战平台，支持8x8黑白棋、9x9五子棋和围棋。支持存档读档和读档后继续悔棋的功能。另外实现了alpha-beta剪枝的基于搜索的五子棋AI。代码可在<https://github.com/hxywtc/OOPProject.git>中获取。

# 基本功能接口设计：

我们的基本功能接口设计如下图，



我们的对战平台由一个Game类的实例和两个Player类的实例构成。这有点类似工厂模式。Game类和Player类是两个虚基类。Game类可以派生出Gobang（五子棋），Reversi（黑白棋）和Go（围棋）三类游戏派生类，在相同的基类接口的基础上分别实现不同游戏的具体功能和行为。Player类可以派生出Human、AI两个派生类分别提供人和AI的操作接口，这么做的原因比较自然，因为不管是Human还是AI它们都只需要对于给定的游戏局面给出一个落子的决策就行了。

我们首先来说一下Game虚基类，改类描述了了所有不同的棋类游戏所需要的所有接口，其中：

void start() —— 一局新游戏的游戏开始。

void set(Point \_point) —— 当前回合落子。

bool draw() —— 悔棋（每次悔一个回合，两手棋）。

void stay() —— 当前回合停一手。

void restart() —— 重新开始一局新游戏。

int checkWin() —— 检查当前状态下是否已经分出胜负（否，player1胜利，player2胜利）。

void load(const string &\_path) —— 存档。

void save(const string &\_path) —— 读档。

void show() —— 显示当前棋盘。

Status& getStatus() —— 返回当前游戏状态（棋盘大小、状态，回合状 态）。

bool canPlace() —— 当前玩家是否能落子（黑白可能无法落子）。

bool canPlace(int x, int y) —— 当前玩家是否能在当前位置落子。

所有三种棋的基本功能都能够通过实现该接口来完成。当然为了实现这些功能我们需要一些成员变量来记录一些状态。其中：

GameType m\_type —— 当前游戏类别（REVERSI, GO, GOBANG）。

Status m\_status —— 当前游戏状态（棋盘大小、状态，回合状态）。

vector<Status> m\_history —— 当前游戏的顺序历史状态。用于存档读档。

对于Player虚基类来说，相对简单。其中：

Point place() —— 当前回合落子决策。

void setStatus(const Status& \_status) —— 设置当前游戏状态。

bool isHuman() —— 是否是人类。

为了实现这些功能，我们需要如下成员变量来记录状态：

Status m\_status —— 当前游戏状态。

PlayerType m\_type —— 当前玩家类别（COMPUTER, HUMAN）。

为了综合Game和Player，我们接下来描述Platform类。主要成员变量为：

Game\* game —— 一个游戏基类指针。

Player\* player[2] —— 两个玩家基类指针。

主要成员函数为：

void start() —— 平台启动，唯一的入口。

void init() —— 初始化，包括通过交互确定游戏和玩家。

void chooseGame() —— 确定游戏。

void choosePlayer(int id) —— 确定第id个玩家。

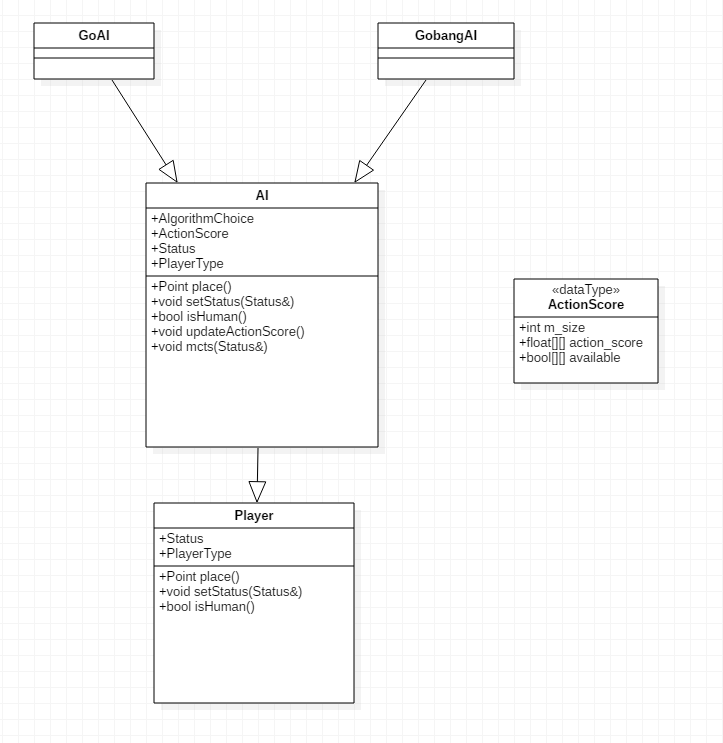
void setPlayerStatus() —— 将游戏状态传给玩家。

void save() —— 存档。

void load() —— 读档。

主要的运作流程为，平台启动后，先通过init初始化，通过和用户交互确定游戏类型和玩家类型。然后对于每一手，如果当前玩家可以落子，对于AI玩家直接调用落子接口的到落子方案后让game落子即可，对于人类玩家，会有1. PLACE，2. RETRACT，3. RESTART，4. SAVE，5. LOAD，6. EXIT一共6个交互选项。每一手结束后先判断游戏是否结束。不结束就更新玩家的游戏状态继续游戏。

# AI功能接口设计：



我们写了AI玩家基类，继承了Player类，同时其具有纯虚函数Point place()。不同的棋类函数只需要实现Point place()即可，即对于当前棋局的局面返回下一个下棋点。

在AI基类的基础上，我们定义了GobangAI类作为五子棋AI的接口，GobangAI通过place成员方法去下棋，每次在place函数中，我们通过遍历所有未下棋的点来进行树搜索，evaluation函数则接受一个当前棋盘局面，并且给出对于当前情形的估值。我们通过极大极小搜索来对最优的局面进行搜索，经过测试我们将树的搜索深度大约设置为3，可以做到AI几乎瞬间对棋局进行应答，同时我们对于我们的搜索树进行alpha-beta剪枝处理，用来对其进行优化，加快AI的速度。