**溢出棋 Overflow Chess**

**规则介绍：**

游戏人数2-4人。每人拥有一种标识色（红黄蓝绿）。

棋盘为7\*7（或其它尺寸）的方形棋盘。

分为两个阶段，放置阶段和溢出阶段。

放置阶段：

首先清空棋盘（置为白色方块），然后每个玩家轮流在空白处放置一个自己颜色的方块，直到棋盘放满。

注意：这里提供边长为5\*5至7\*7的正方形棋盘，这三种棋盘在除以2-4时最多余1，此时引入一个规则：棋盘最中间的方块置黑色，任何玩家无法放置。

溢出阶段：

玩家每回合从3种操作选一个：

1. 推动方块：选择一行或一列方块，从一个方向推动一格，被推到棋盘外的方块会“溢出”消失，而另一端填充一个黑色方块。
2. 放置阻挡方块：选择一个黑色方块，将其替换为自己的阻挡方块。阻挡方块所在的行和列无法被推动。每个玩家最多放置一个阻挡方块。
3. 移动阻挡方块：将自己的阻挡方块移动到另一个黑色方块的位置。原位置填充黑色方块。

游戏的目标是尽可能使其他玩家颜色的方块“溢出”。失去所有颜色方块的玩家被淘汰，最后存活的玩家获胜。

注意：

1. 阻挡方块不算颜色方块，不能用于判断存活。
2. 阻挡方块一旦放下，无法主动收回。
3. 当玩家被淘汰时，他的阻挡方块消失并替换为黑色方块。

此外，该游戏在2回合内禁止出现重复局面，即你无法“撤销”上一位玩家的操作。

**代码架构：**

使用python和pygame

1. 棋盘类chess\_board

成员变量：

1. 当前棋盘状态state：7\*7数组，每个元素是一位int，含义如下：

0：黑色方块

1 ~ 4：玩家1-4的颜色方块

-1 ~ -4：玩家1-4的阻挡方块

100：白色方块（放置阶段）

1. 状态栈state\_stack：储存从开局以来所有状态，每步同样为7\*7数组。

成员函数：

1. 放置颜色方块place\_colour（x，y ，colour）

colour表示玩家颜色编号（1-4）

（放置阶段）在对应x，y坐标处放置颜色为colour的方块。被放置的位置必须是白色方块（100）。

1. 推动方块push（direction，n）：

direction表示推动的方向，0 ~ 3分别表示上下左右

n表示列编号或行编号（从0开始）

根据方向判定是行还是列，例如（0，3）表示向上推动第4列。

如果有阻挡方块则失败。

1. 放置阻挡方块place\_ barrier（x，y ，colour）

先将当前状态压入队列。

在对应x，y坐标处放置玩家为colour的阻挡方块。首先检测放置位置是否为黑色方块（0），然后调用delete\_ barrier函数确保删除原来的阻挡方块。

1. 删除阻挡方块delete\_ barrier（colour）

遍历棋盘数组，如果有该玩家的阻挡方块，将其替换为黑色。

1. 重置棋盘reset（）

将棋盘重置为全白色，但中央方块为黑色。

同时清空状态栈。

1. 判断存活judge\_alive（colour）

遍历棋盘数组，判断一名玩家是否存活。如果存活返回true。

1. 判断放置阶段完成place\_phase\_finish（）

遍历棋盘数组，判断所有白色方块是否都放置完成。

1. 储存一步状态save\_state（）

将当前state压入状态栈。

1. 撤销一步状态undo\_state（）

将状态栈栈顶写回当前state，随后弹栈一步。

1. 避免重复局面no\_repetition（step）

step：搜索的范围

判断当前state是否和状态栈中前step步相同。若相同，将栈顶写回当前state并返回false。否则返回true。

1. 获取棋盘数组get\_chess\_board（）
2. 游戏UI界面类game\_UI

成员变量：

1. 棋盘实例。
2. 玩家状态数组player [4] ，每个元素是一位int，含义如下：

-1：未参与游戏

0：淘汰

1：存活

1. 当前玩家编号current\_player\_id（0-3，不是1-4）
2. 当前阶段current\_phase（0-1分别表示放置、溢出阶段）
3. 当前回合turn\_count

成员函数：

1. 渲染：render（）

根据当前棋盘状态、玩家状态与当前玩家编号渲染游戏界面。

注意，除了渲染7\*7的棋盘外，还要在外围渲染一圈，以放置“推动”按钮（表现为灰色，中间绘有朝向棋盘的三角形）

在棋盘的右侧，显示玩家状态、当前操作玩家、当前阶段。

同时，绘制撤销和重启按钮。

1. 游戏主循环game\_loop（）

每个回合，先调用save\_state储存当前状态。

若为放置阶段：

For循环使用pygame检测键盘或鼠标点击，直到完成一次成功操作：

如果按下撤销按钮，调用undo。

如果按下重启按钮，先进入重启状态，下回合开始时调用start。

如果点击到白色方块，则在对应位置放置颜色方块。

随后检测放置阶段是否结束。结束则进入溢出阶段。

若为溢出阶段：

For循环使用pygame检测键盘或鼠标点击，直到完成一次成功操作：

如果按下撤销或重启按钮，操作同上。

如果点击到推动按钮，则执行一次推动操作。

如果点击到黑色方块，则在对应位置执行放置阻挡方块的操作。

最后调用no\_repetition处理重复局面。

随后检测所有玩家存活状态，若有玩家被淘汰，修改其状态，删除其阻挡方块。若有玩家获胜，游戏结束。

最后切换下一名玩家作为当前操作玩家。

调用render渲染画面。

1. 撤销操作undo（）:

调用undo\_state，并返回上一个玩家的回合

1. 重启游戏start（）：

绘制按钮以读入玩家人数，和棋盘大小。

随后reset棋盘，进入放置阶段，修改玩家状态，并设置第一位玩家为当前玩家回合。

1. 主循环main（）：

实例化game\_UI，进入主循环。