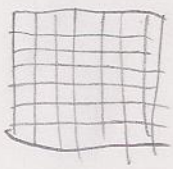
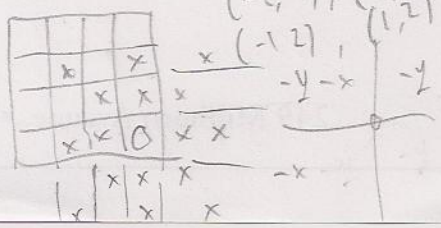
Init → Get move$_1$ → Update → Get move$_2$
Update

Game AI$_1$ AI$_2$

Game starts
→ Inits game state
→ Forks AI$_1$, AI$_2$
→ Sends game state → Read Input
Choose Turn
→ Read command ← Output command
→ update game state
→ Send game state → Read Input
Choose Turn
→ Read Command ← Output command
→ update game state
→ Game over?
↓ yes
Finished
no

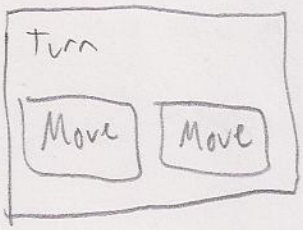


(-1,-2), (1,-2)
(-2,-1), (2,-1)
(0,0)
(-2,1), (2,1)
(-1,2), (1,2)



investorsHome
mortgage

Jerry Jacob
973.477.8508
gjacob@myinvestorsbank.com



Move
- Vector2
- Errors (list of strings)

Turn
- Moves (2)
- Errors (list of strings)
- Stone/Stone ID

Board
- Stones []
- width } Vector2 dimensions
- height
- getStone(id) ✓
- getStone(vector2) ✓

Game Rules
- transitionBoard(Game, Board, Turn): Board
- validateTurn(Game, Board, Turn): Turn (with errors)

Game(GameRules, Board)
- turnNumber ✓
- whosTurn ✓
- takeTurn(turn): list of error strings
- gameRules ✓
- board ✓

Stone
- Color
- ID

AIController
- sendGameState(process, game)
- getNextMove(process)

249 Millburn Avenue • Millburn, NJ • 07041

Gameplay

Browser sends request
- Turn + SID
Server looks up SID, gets gamestate
Validate turn, if errors send response with them
Transition board
Send new board to AI, get turn response
Validate turn
Transition board
Save game state w/ SID
Respond w/ new game state

Load site

Browser sends index request
Server responds w/ index + JS
JS:
- If SID in URL, request gamestate
- Else request new SID & gamestate

Request

↳ NodeJS → (JSON) → Engine.jar → (JSON) → NodeJS
Turn
or Turn Board

↳ (JSON) → AI.jar → (JSON) → Engine.jar → (JSON) → NodeJS
Board
Turn
Board
↓
Response

// pseudocode

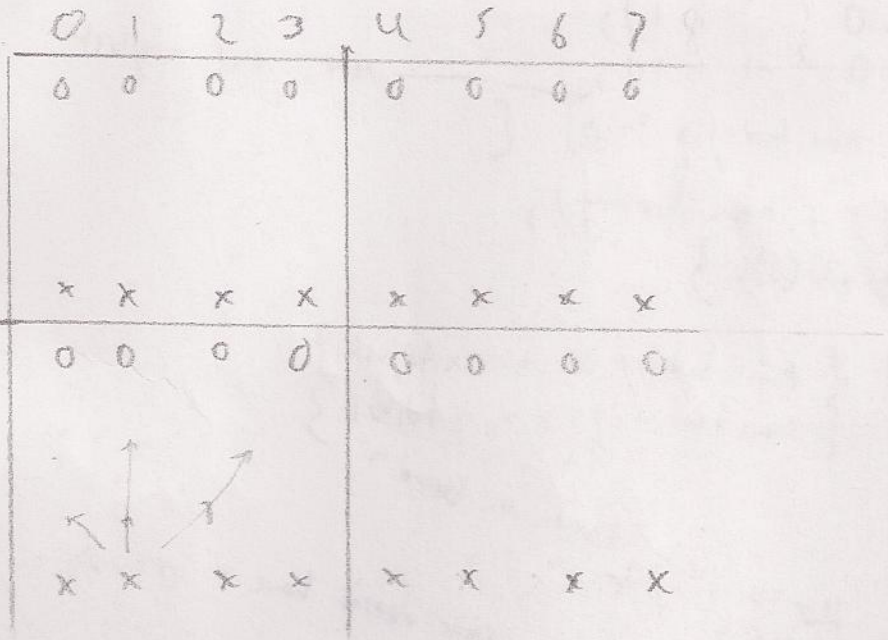
```
List<Stone) getStonesIntersected(board, move) {
    int xAdjust = (move.heading.x >= 0 ? -1 : 1);
    int yAdjust = (move.heading.y >= 0 ? -1 : 1);
                                        <- new stones arraylist
    while ( move.heading.x != 0 || move.heading.y != 0) {
      s = board.getStone( move.origin + move.heading);
      if (s != null) { stones.add(s); }

      if ( move.heading.x != 0) { move.heading.x += xAdjust }
      if (move.heading.y != 0) { move.heading.y += yAdjust }
    }
    return stones;
}
```

maybe this is just "transition board"

partially

stone doesn't have a position

```
Vector2 getStonesNewPositionAfterMove (board, stone, move) {
    Vector2 step = new Vector2 (clamp(move.heading.x, -1, 1), ... )
    // calculate first stones new position
    // if on top, move other stone same direction
    // if not done, repeat
    Vector2 intermediatePos = new Vector2( move.origin);
    loop {
        intermediatePos = intermediatePos + step;
        if ( board.getStone(intermediatePos) != null) {
            board.moveStone (intermediatePos + step)   // could handle "border crossing"
        }
        if (intermediatePos == move.origin + move.heading) {
            break
        }
    }
}
```



Number of possible opening moves: from bottom left: 16

From a given first (passive) move, active options: 4, 6, 8

Messages

```
{
  "type": "gamestate"
  "payload": {
    "board": "xxxx ..... oooo",
    "turn": "black",
    "turnNumber": }
  }
}
```

```
{
  "type": "turn"
  "payload": {
    "passive": {
      "origin": {
        "x": 1,
        "y": 2
      }, "heading": { ... },
    },
    "aggressive": {
      :
    }
  }
}
```