System Component Interaction Diagram

```
┌─────────────┐              ┌──────────────┐   ┌───────────┐
│  Client AIs │◄────────────►│  Strategies  │◄──│ Game Tree │
└─────────────┘              └──────────────┘   └───────────┘
   ▲      ▲                         ▲
   │ TCP  │ HTTP                    │ Internal
   │      │                        
   ▼      ▼                         
┌────────────────────┐     ┌──────────────┐
│ Tournament Servers │────►│ In House AIs │
└────────────────────┘     └──────────────┘
   ▲                              
   │                              
   ▼                              
┌─────────────────┐        ┌────────────┐
│  Game Referees  │───────►│ Game Rules │
└─────────────────┘        └────────────┘
   ▲                              
   │                              
   ▼                              
┌─────────────┐
│ Game State  │
└─────────────┘
   ▲
   │
   ▼
┌─────────────┐
│ Game Board  │
└─────────────┘
```

TCP

HTTP
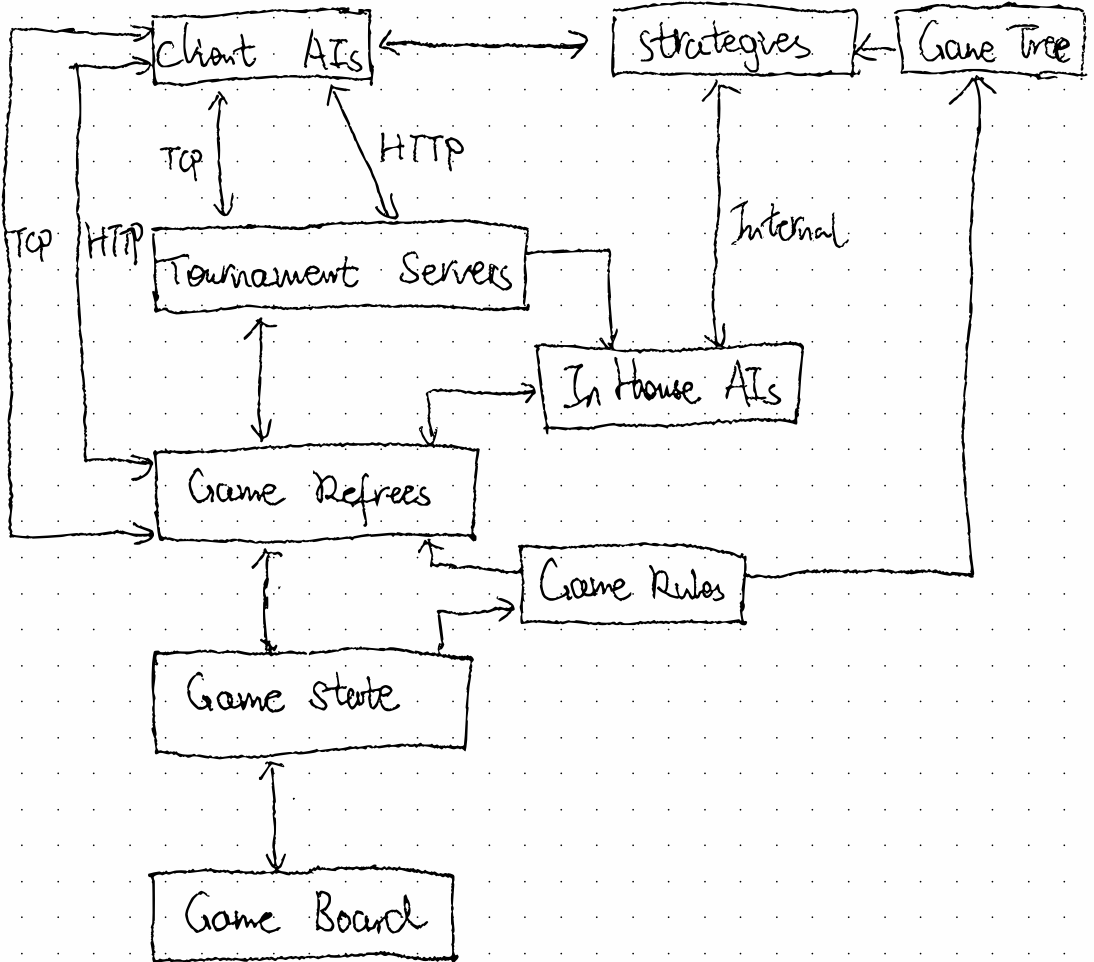
System Class UML Diagram

Client
+ conc : server proxy
+ agent : Player

Server Proxy
+ connect() : bool
+ disconnect() : bool
+ send(message) : bool
+ recieve() : message

TCP Server Proxy
+ IP : str
+ Port : int

HTTP Server Proxy
+ base_url : str
+ port : int
+ endpoint : str

Observer

Player

Local Player
+ strat : Strategy

Client Proxy
+ send(message) : bool
+ recieve() : message

TCP Client Proxy
+ conc : socket

HTTP Client Proxy
+ conc :

Remote Player
+ conc : Client Proxy
+ Strat : Strategy

Observer (Interface)

+ player_colors(List[str])
+ action_update (str, action)

+ tournament_start()

+ tournament_result (Bool)

+ game_start()
+ game_result (Bool)

Player (Interface)

+ get_action(): action

+ playing_as (color)

Tournament Manager

+ game_type: str

+ referees: List[Referee]

+ observers: List[Observer]

+ players: List[Player]

+ run_tournament(): (int, int)

Referee

+ game_state: GameState

+ game_tree: GameTree

+ observers: List[Observer]

+ players: List[Player]

+ run_game(): tuple(List[players])

GameState (Interface)

+ apply_action(): bool

+ current_player(): str

+ game_over(): bool

+ remove_player(): GameState

+ game_winners(): List[str]

+ serialized(): Map()

+ valid_actions(): List[Action]

GameTree

+ state: GameState

+ children: Map(action, GameTree)

+ get_children(): Map(action, Cache)

+ get_current_state(): GameState

+ apply_action(action): bool

+ apply_function(function): Map(action, X)

```
┌──────────────────────┐
│      Game State       │
└──────────────────────┘
```

```
┌────────────────────────────┐
│   MultiAgentGameState       │
├────────────────────────────┤
│ + turn_queue: Queue(str)    │
│ + board : Board             │
│ + Scores : Map(str,  )      │
└────────────────────────────┘
```

```
┌────────────────────────────┐
│   SingleAgent GameState     │
├────────────────────────────┤
│ + board: Board             │
│ + Scores : int             │
└────────────────────────────┘
```

```
┌──────────────────────────────────────────────┐
│ Board                                         │
├──────────────────────────────────────────────┤
│ + valid_from_position (str): List [Posn]      │
│ + reachable_ Positions (from_Posn): List[Posn]│
│ + Serialize() : Map()                         │
└──────────────────────────────────────────────┘
```

```
┌────────────┐   ┌──────────────┐   ┌─────────────────────┐
│ FishBoard  │   │  ChessBoard   │   │ MarbleSolitareBoard │
└────────────┘   └──────────────┘   └─────────────────────┘
```