多人五子棋

Introduction

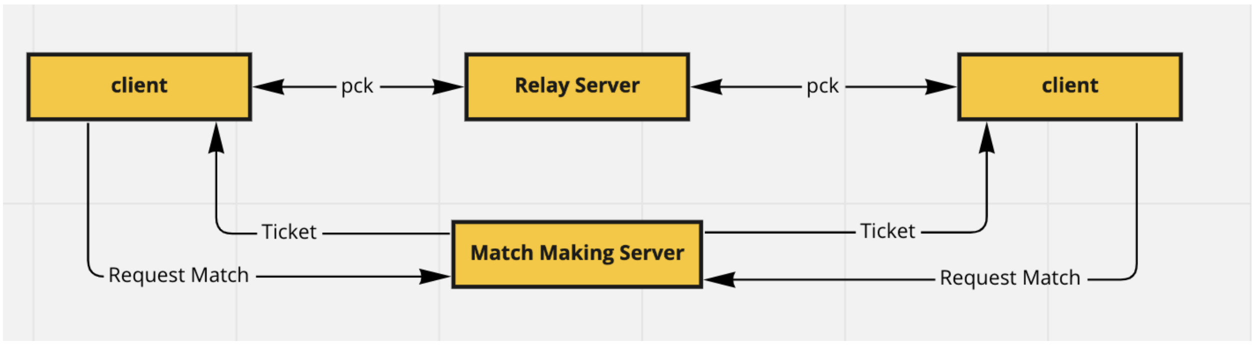
我想運用學到的東西，做出一個多人遊戲，而考慮到我像要著重於網路連線部分，我選擇用規則簡單的五子棋作為遊戲。這份報告我使用socket.io作為我的後端，運用Match making server & relay server讓玩家可以互相連線。

System Architecture

Here we brief describe the architecture of frontend(client) and backend(server)

在實作方面，我將遊戲邏輯跑在client app上面，而server只提供Match making 讓玩家可以互相找到對手，和relay讓我家可以用server當作中間人來溝通，這樣設計理念是想要簡化server要跑的邏輯，我想試試看將client責任變大，隨然目前無法有效防止玩家作弊，但我目前也不想著重於安全性方面問題。

[Diagram of communication within different node in system]



```
client: game app that user interact with.
RelayServer: relay data between clients
MatchMakingServer: matching two client together
  1. get Request Match from client
  2. Perform match making
  3. when find a match create ticket, ticket will contain information of the match
     which therefor let two client can connect to each other
```

Match Making Server & Relay Server

這裡我描述我Server的程式中每個物件所做的工作

▼ Component description of it job

Socket io Server:

1. start socket server, listen for client connection
2. emit socket relate event

Player Manager:

1. managing player
2. a player is like a encapsulation of a socket connection, once a client sent it player data, we will create a player object
3. emit player relate event

MatchMaker:

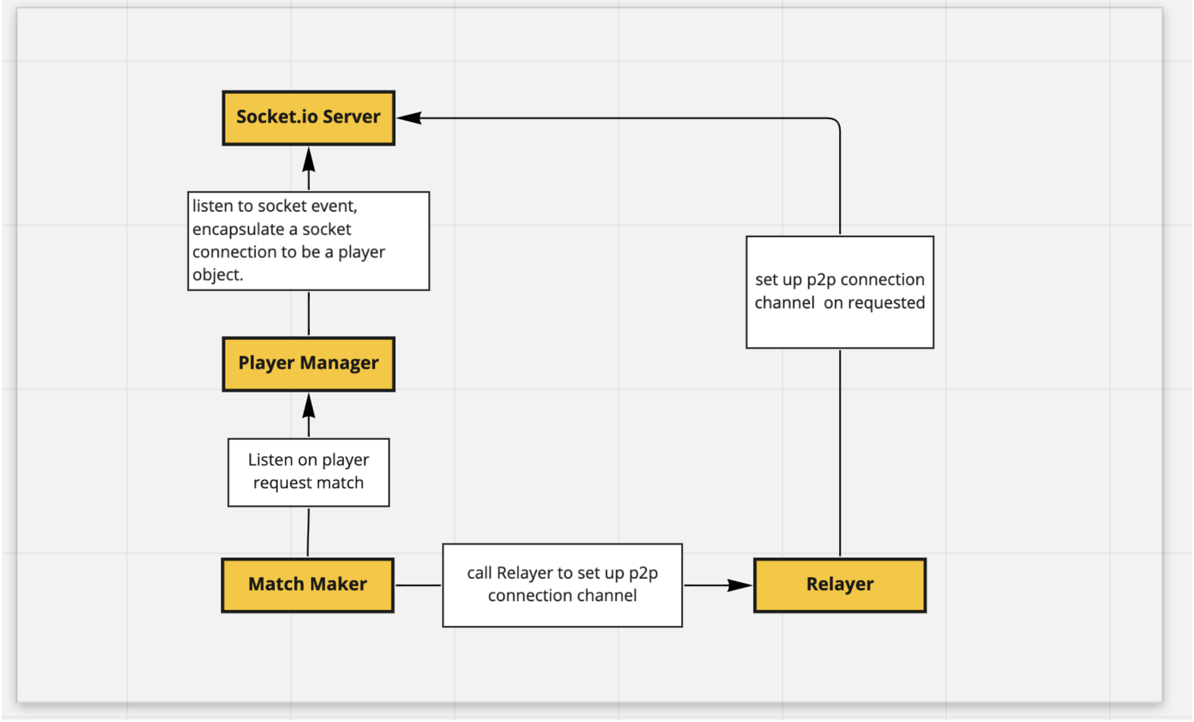
1. listen on player request for match
2. perform match making
3. When Match:
    a. calling relayer to set up connection channel for players
    b. Send ticket to matched players

Relayer:

1. Set up p2p connection/communication channel for clients

▼ Dependency & Architecture

Match Making Server & Relay Server Architecture


Socket.io Server
listen to socket event, encapsulate a socket connection to be a player object.
Player Manager
Listen on player request match
Match Maker
call Relayer to set up p2p connection channel
Relayer
set up p2p connection channel on requested

▼ Communication Protocol

Communication Protocol

Match Making Server and Relay Server share same endpoint address, client can connection then by sock.io protocol.

dev endpoint: http://127.0.0.1:3000

Match Making Server Protocol

- using socket.io on(event name, args) method

event name	Bound To	Field Name	Field Type
requestMatch	Sever	[no field]	
ticket	client	ticketPck	TicketPck

▼ Interface TicketPck:

```
export interface TicketPck{
    p2pConnectMethod: string;
    MethodSpecificData: any;
}
```

Relay Server Protocol

- using socket.io on(event name, args) method
- (Note: relay channel will auto set up once match, hence client don’t have to request for a relay channel, client can send relay data right after get ticket)

event name	Bound To	Field Name	Field Type
relayData	Sever	playLoad	bytes

Client

這裡我描述我client程式中每個物件的工作

▼ Components & description of their main job

▼ **GameManager**

- Description: Control the App(Game), the root class of whole architecture, it control all other managers.
- Methods:
    - StartGame(): start the game bundle of go game and ui.

▼ UIManger

- Description: Control UI
- Method:
    - HideAllPage()
    - ShowStartPage()
    - ShowGameOverPage()

▼ GoManager

- Description: control the board by mouse, and compute go game logic.
- Methods:

    - StartGoGame(): start go game.(only have control of board and go game logic)
    - EndGoGame(): end go game.(only have control of board and go game logic)
- Event:
    - `GoGameEndEvent` : emit when have a winner.
        - param: `StoneType winner`

▼ MouseDataProvider

- Description: provide mouse data.
- Parameters:
    - `Vector2 MouseWOrldPosition`
    - `GameObject MouseHit`

▼ Board

- Description: interface to control the board.
- Methods:

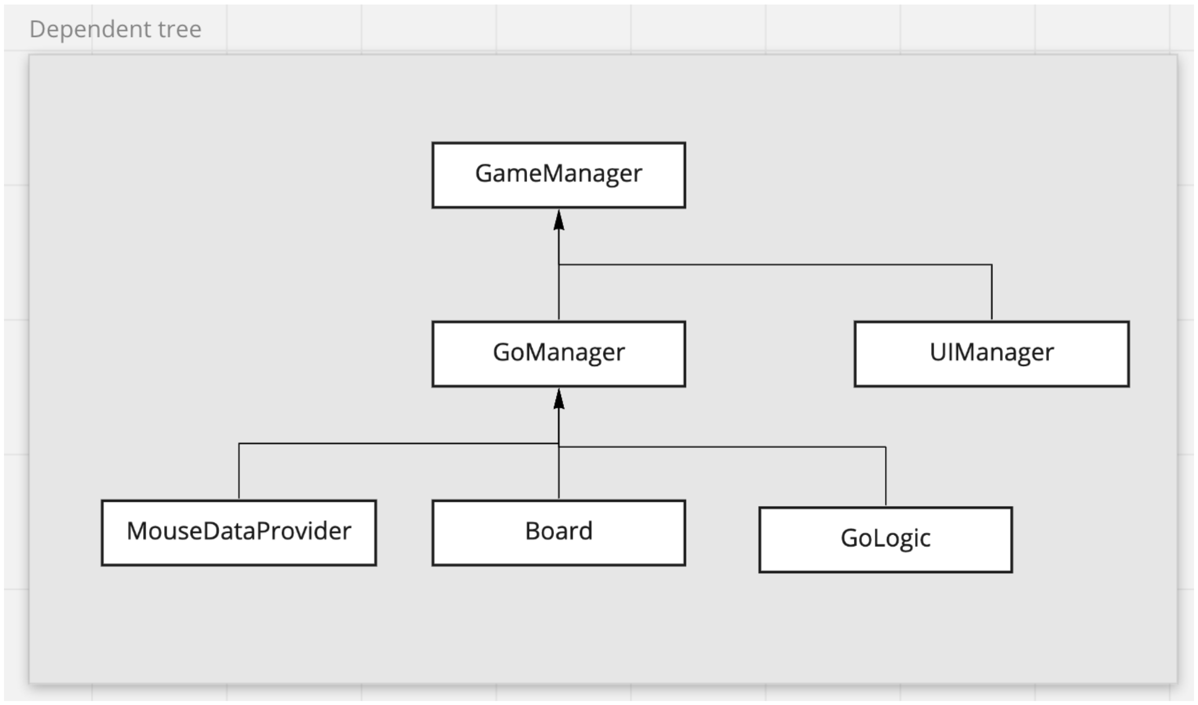
```
void Clear();
void PlaceStone(StoneType stoneType, int2 xy);
int2 NearestIntersectionIndex(Vector2 position);
```

▼ GoLogic

- Description: go game logic computer, main job is let you place stone and it will tell you whether have a winner.
- Methods:
    - PlaceStone(stone type, index): let you place stone.
    - IsIntersectionOccupied: check intersection is occupied or not.
    - clear board(): clear the board
- Events:

        - WinEvent: emit when there is a winner.
            - param: `StoneType winner`

▼ Dependency Tree



▼ Go Game Protocol

Protocol

- Definition: go game clients connect each other establish a reliable communication. Clients send packets to each other, the packets meaning is define by it packets name, the data is transfer in Json format.

HandShaking

Pck Name	Field Name	Field Type
HandShake	SenderId	string

PlaceStone

Pck Name	Field Name	Field Type	Note
PlaceStone	StoneType	int	0 → white, 1 → black
	X	int	0~18
	Y	int	0~18

Github

https://github.com/easonyu0203/csharp-program-class-go-game

Presentation

csharp-class-hw2-presentation

https://youtu.be/4O5fcG3BQjI

Reference

https://socket.io/