# 網路程設期末專題 - 重力四子棋

###### tags: 網路程式設計

## 1. 題目、組別

* 題目：重力四子棋
* 成員：1102914 龔弋筠、1102916 游雅碩
* [口頭報告影片](https://youtu.be/w9rfuLpxZEw)

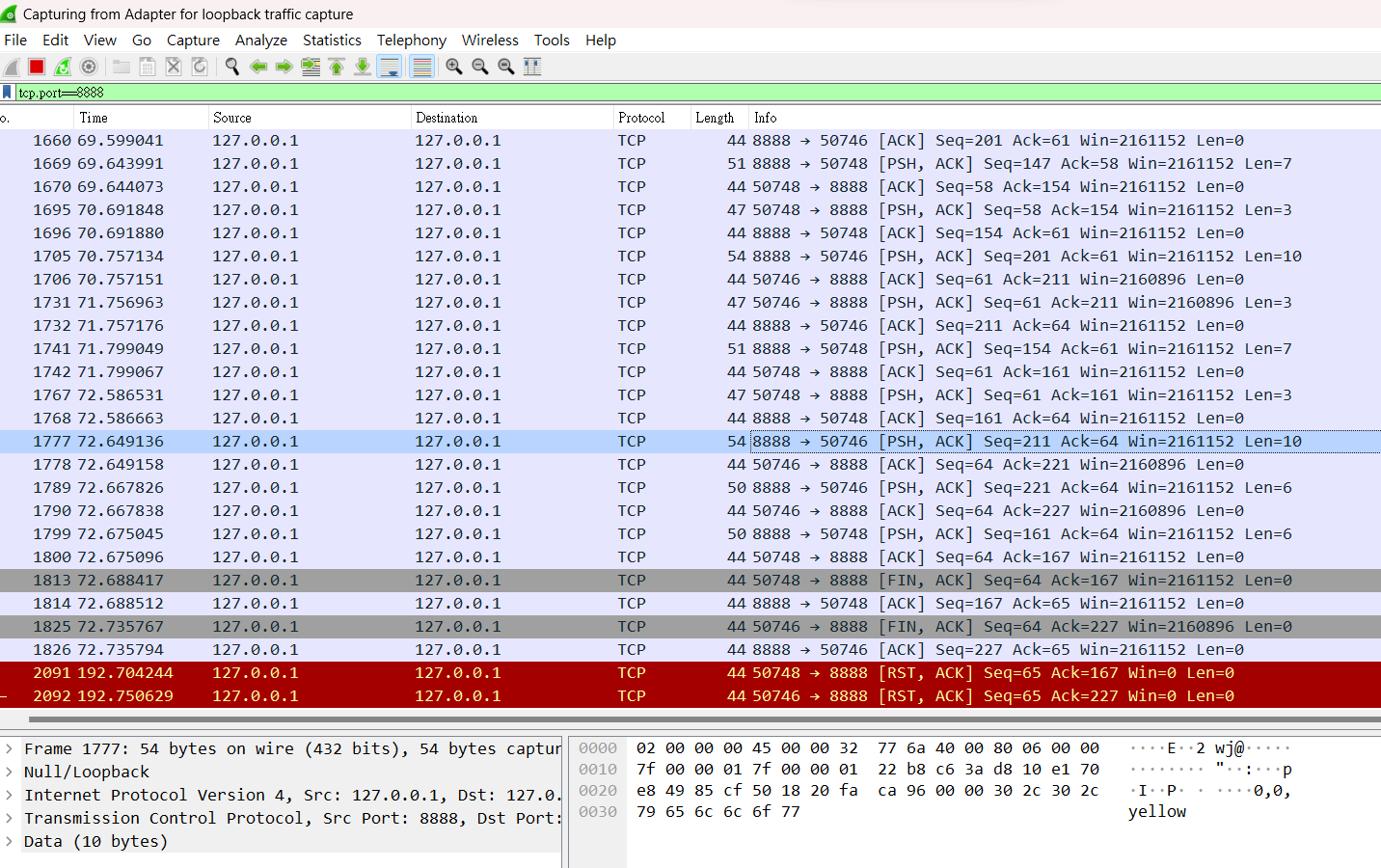
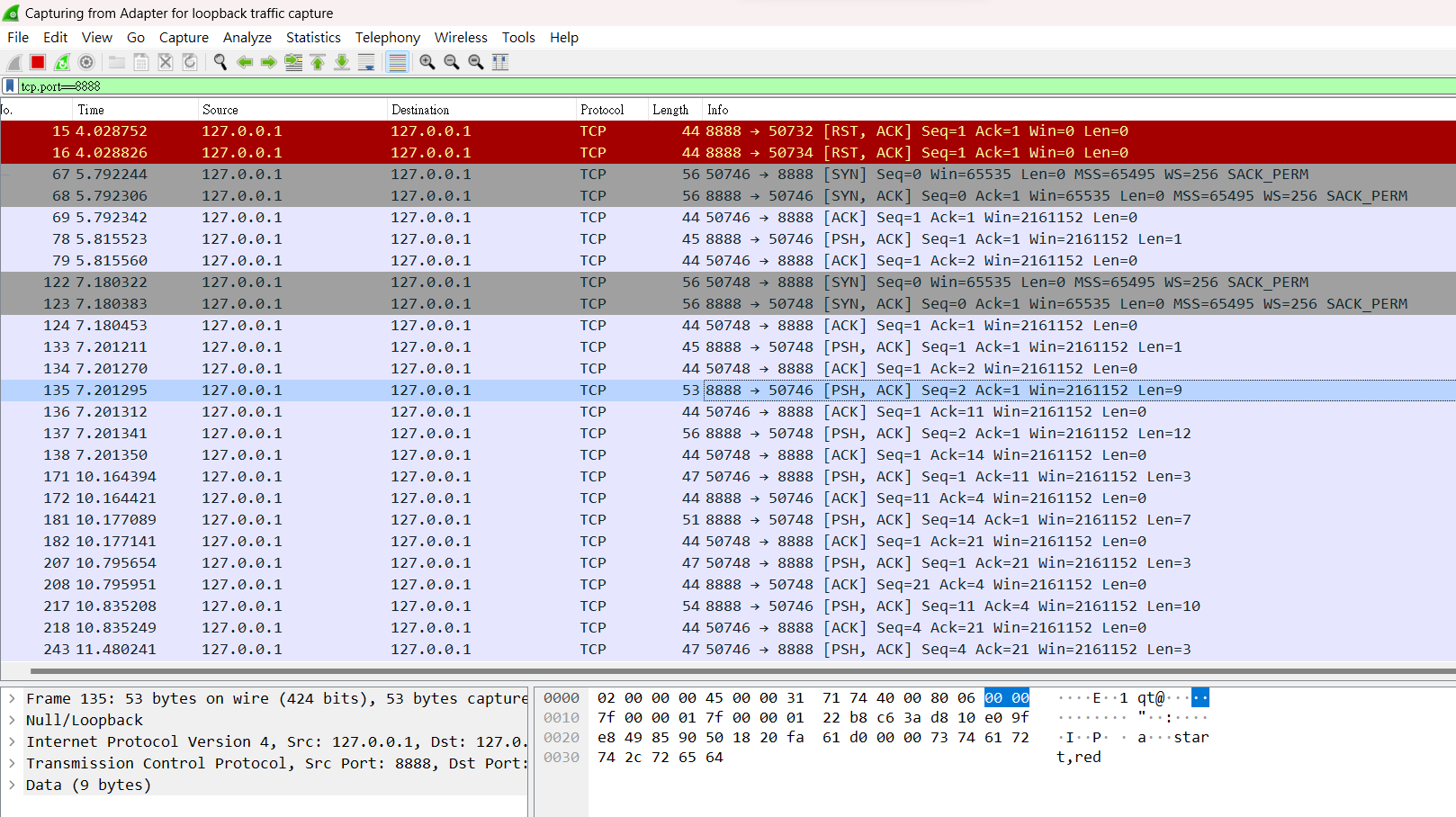
## 2. 作品介紹

### 靈感來源與參考

### 遊戲規則

* 兩client端連入後開始遊戲。
* 一方為red棋，一方為yellow棋。
* 點擊其中一行，棋會落於該行當前的最底層。
* 雙方輪流，一人一次下棋機會，其中以先連入的client端優先下棋。
* 當有任一顏色的四個棋子連成一線，雙方client斷線並停止GUI介面更新。

## 3. 作品執行過程說明(配合畫面)

<https://clipchamp.com/watch/4cg8eGUPoHr>

## 4. 作品架構(1對1，client對server，還是1對多)

1個server對多client  
配合遊戲規則，client最多只能連入兩個。

## 5. 作品的功能(包含特色、困難、挑戰、花最多精力)

* 接收滑鼠點擊位置（綁定）
* 判斷該行的最底可下棋位置
* 傳遞落棋座標及顏色 (ex: client1->server, server->client2)
* 接收座標、顏色並更新於GUI介面
* 雙方輪流操作
* 判斷勝負
* 結束遊戲後停止GUI介面更新（取消綁定滑鼠事件）

## 6. 程式碼

### Server

import socket

import threading

class GameState: #遊戲狀態

WAITING = 0

PLAYING = 1

GAME\_OVER = 2

class ClientThread(threading.Thread):

def \_\_init\_\_(self, client\_socket, game):

threading.Thread.\_\_init\_\_(self)

self.client\_socket = client\_socket

self.game = game

def run(self):

try:

while True:

data = self.client\_socket.recv(1024).decode() # 接收client端訊息

#print(f"Received data: {data}")

if not data: # 沒有接收到訊息

#break 這邊若使用break會結束整個run()函式

continue

if data == "start": # 收到開始的訊息

self.game.start\_game()

else: # 收到對手下棋位置

row, col = map(int, data.split(","))

self.game.process\_opponent\_move(row, col) # 更新並傳送棋子的動作

finally:

self.client\_socket.close()

class FourInARowGame():

def \_\_init\_\_(self):

self.clients = [] #存客戶端連接

self.game\_state = GameState.WAITING

self.board = [[None] \* 7 for \_ in range(6)]

self.current\_player = "red"

def add\_client(self, client\_socket):

if self.game\_state != GameState.WAITING or len(self.clients) >= 2:

client\_socket.send("0".encode()) # 遊戲已經開始or滿人

client\_socket.close()

return

else:

client\_socket.send("1".encode()) # 加入遊戲

self.clients.append(client\_socket)

if len(self.clients) == 2:

self.start\_game()

def start\_game(self):

self.game\_state = GameState.PLAYING

self.current\_player = "red"

self.clients[0].send("start,red".encode())

self.clients[1].send("start,yellow".encode())

def process\_opponent\_move(self, row, col): # 更新並傳送棋子的動作

if self.game\_state != GameState.PLAYING:

print("OK self.game\_state != GameState.PLAYING")

return

#print(f"對手的操作：{row},{col}")

self.board[row][col] = self.current\_player # 棋盤位置設定player的顏色

move = f"{row},{col},{self.current\_player}" # 格式化

print(move)

'''

for client in self.clients:

print("OK for client in self.clients:")

client.send(move.encode())

'''

for i, client in enumerate(self.clients):

if self.current\_player == "red" and i == 1:

client.send(move.encode()) # 將紅棋位置傳給黃方(client[1])

elif self.current\_player == "yellow" and i == 0:

client.send(move.encode()) # 將黃棋位置傳給紅方(client[0])

self.check\_game\_over() # 檢查遊戲是否結束

self.switch\_player() # 交換玩家

def switch\_player(self):

if self.current\_player == "red":

self.current\_player = "yellow"

else:

self.current\_player = "red"

def check\_game\_over(self):

# win?

for row in range(6):

for col in range(4): # 檢查橫向

if self.board[row][col] is not None and self.board[row][col] == self.board[row][col+1] == self.board[row][col+2] == self.board[row][col+3]:

self.game\_state = GameState.GAME\_OVER

self.send\_game\_over\_message(f"{self.board[row][col]} wins!")

return

for row in range(3): # 檢查直行

for col in range(7):

if self.board[row][col] is not None and self.board[row][col] == self.board[row+1][col] == self.board[row+2][col] == self.board[row+3][col]:

self.game\_state = GameState.GAME\_OVER

self.send\_game\_over\_message(f"{self.board[row][col]} wins!")

return

for row in range(3): # 檢查斜線

for col in range(4):

if self.board[row][col] is not None and self.board[row][col] == self.board[row+1][col+1] == self.board[row+2][col+2] == self.board[row+3][col+3]:

self.game\_state = GameState.GAME\_OVER

self.send\_game\_over\_message(f"{self.board[row][col]} wins!")

return

for row in range(3, 6): # 檢查斜線

for col in range(4):

if self.board[row][col] is not None and self.board[row][col] == self.board[row-1][col+1] == self.board[row-2][col+2] == self.board[row-3][col+3]:

self.game\_state = GameState.GAME\_OVER

self.send\_game\_over\_message(f"{self.board[row][col]} wins!")

return

# 平手?

is\_full = all(all(cell is not None for cell in row) for row in self.board)

if is\_full:

self.game\_state = GameState.GAME\_OVER

self.send\_game\_over\_message("平手")

def send\_game\_over\_message(self, message):

for client\_socket in self.clients:

print(message)

client\_socket.send(message.encode())

class Server:

def \_\_init\_\_(self):

self.server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

self.server\_socket.bind(("127.0.0.1", 8888))

self.server\_socket.listen(2)

self.game = FourInARowGame()

def start(self):

print("等待客戶端連接...")

while True:

client\_socket, address = self.server\_socket.accept()

print("客戶端連接成功")

self.game.add\_client(client\_socket)

# 啟動ClientThread

client\_thread = ClientThread(client\_socket, self.game)

client\_thread.start()

server = Server()

server.start()

### Client

import socket

import threading

import tkinter as tk

from tkinter import scrolledtext

def send\_move(server\_socket, move): # 傳送移動位置

server\_socket.send(move.encode())

def receive\_move(server\_socket): # 接收移動資訊

data = server\_socket.recv(1024).decode()

return data

class GameUI:

def \_\_init\_\_(self, root, network\_thread):

self.root = root

self.network\_thread = network\_thread

self.root.resizable(width=False, height=False) # 長寬不可調整

self.canvas = tk.Canvas(self.root, width=700, height=600, bg="#0072E3")

# 右側文字介面

frame = tk.Frame(self.root)

frame.pack(side=tk.RIGHT, fill=tk.Y) # 靠右，填滿Y軸

scrollbar = tk.Scrollbar(frame) # 滾動軸

scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

self.text\_box = scrolledtext.ScrolledText(frame, width=20, height=5, wrap=tk.WORD, yscrollcommand=scrollbar.set)

self.text\_box.insert(tk.END, "遊戲規則: \n玩家輪流下棋，先紅色後黃色。\n"

"輪到當前玩家時，需點選一行。\n"

"棋子將落到所選行中的最低空格。\n"

"目標將自己的四個棋子在水平、垂直或對角線方向連成一直線。\n\n"

"遊戲狀態: \n")

self.text\_box.pack(side=tk.LEFT, fill=tk.Y)

scrollbar.config(command=self.text\_box.yview)

# 設定框中的字體和大小

self.text\_box.configure(font=("微軟正黑體", 20))

self.board = [["white"] \* 7 for \_ in range(6)]

self.canvas.pack()

# 綁定滑鼠左鍵點擊事件

self.canvas.bind("<Button-1>", self.on\_click)

self.can\_click = False # 預設為不可點擊

self.draw\_board()

def update\_text(self, text):

self.text\_box.insert(tk.END, text + "\n")

self.text\_box.see(tk.END)

def draw\_board(self):

self.canvas.delete("all")

for row in range(6):

for col in range(7):

x1 = col \* 100

y1 = row \* 100

x2 = x1 + 100

y2 = y1 + 100

fill\_color = self.board[row][col] # 獲取對應位置的顏色

# 在該位置繪製圓形棋子顏色

self.canvas.create\_oval(x1 + 10, y1 + 10, x2 - 10, y2 - 10, fill=fill\_color)

def update\_board(self, row, col, color):

self.board[row][col] = color

self.draw\_board()

def show\_winner\_message(self, winner): # 勝負結果文字方塊

# 繪製矩形色塊 (左上x, 左上y, 右下x, 右下y, 顏色)

self.canvas.create\_rectangle(200, 250, 500, 350, fill="black")

# 文字位置, 內容, 字體及大小, 顏色

self.canvas.create\_text(350, 300, text= winner, font=("微軟正黑體", 30), fill="white")

def on\_click(self, event):

if not self.network\_thread.game\_running or not self.can\_click:

return

col = event.x // 100 # 透過點擊位置計算col值

color = network\_thread.player # 取得當前player顏色

# 由該行尚未落棋的最下層開始填棋子

for row in range(5, -1, -1):

if self.board[row][col] == "white": # 白色 尚未有棋子

self.board[row][col] = color # 該位置填上顏色

self.draw\_board() # 更新到畫面上

move = f"{row},{col}"

#print(move)

send\_move(self.network\_thread.server\_socket, move)

self.can\_click = False # 點擊一次後，當前為不可點擊

break

class NetworkThread(threading.Thread):

def \_\_init\_\_(self):

threading.Thread.\_\_init\_\_(self)

self.server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

self.server\_socket.connect(("127.0.0.1", 8888))

self.game\_running = False

winning\_color = None

def run(self):

while True:

data = receive\_move(self.server\_socket)

if data == "1":

print("加入遊戲成功。")

game\_ui.update\_text("加入遊戲成功。\n等待對手加入\n")

continue

# 收到遊戲開始訊息

elif data == "start,red":

print("遊戲開始！")

self.game\_running = True

self.player = "red" # 初始化玩家顏色

game\_ui.update\_text("遊戲開始！\n你是紅色\n\n由你開始下棋!")

game\_ui.can\_click = True # 剛開始紅方先進行下棋動作

continue

elif data == "start,yellow":

print("遊戲開始！")

self.game\_running = True

self.player = "yellow" # 初始化玩家顏色

game\_ui.update\_text("遊戲開始！\n你是黃色\n\n等待對手下棋...")

continue

# 收到遊戲結果

elif "平手" in data:

game\_ui.update\_text("遊戲結束！\n你們平手")

game\_ui.show\_winner\_message(data)

game\_ui.canvas.unbind("<Button-1>") # 取消綁定點擊事件

#game\_ui.draw\_board.stop()

break

elif "wins" in data:

game\_ui.update\_text("遊戲結束！\n" + winning\_color + " win!")

game\_ui.show\_winner\_message(data)

game\_ui.canvas.unbind("<Button-1>") # 取消綁定點擊事件

#game\_ui.draw\_board.stop()

break

# 接收對手操作的位置跟顏色

row, col, player = map(str, data.split(","))

row, col = int(row), int(col)

game\_ui.board[row][col] = player

# 新的移動訊息

updated\_move = f"{row},{col},{player}"

print("Updated move:", updated\_move)

#print(f"對手的操作：{data}")

#row, col, full\_color = map(int, data.split(","))

game\_ui.update\_board(row, col, player)

#game\_ui.update\_text("對手下棋位置為:"+f"{col+1},{row+1}")

game\_ui.update\_text("輪到你下棋囉!!!")

# 交換玩家點擊權限

if game\_ui.can\_click:

game\_ui.can\_click = False

else:

game\_ui.can\_click = True

self.server\_socket.close()

root = tk.Tk()

network\_thread = NetworkThread()

game\_ui = GameUI(root, network\_thread)

network\_thread.start()

root.mainloop()

## 7. 作品使用到課程所教的技術，以及其他自學的技術

### 課程所學

* socket 的 send、receive
* Thread
* Tkinter 文字框

### 自學

* 綁定事件與函式  
  bind("<Button-1>", self.on\_click)
* 將要傳送的多個訊息格式化再送出  
  f"{row},{col}, {self.current\_player}"
* 將接收到的data分割出多個訊息  
  row, col, player = map(str, data.split(","))
* 繪製圓（橢圓）形  
  create\_oval(左上x, 左上y, 右下x, 右下y, fill=color)

## 8. 人員分工與時間分配

### 1102914 龔弋筠

* 提供遊戲功能實現想法
* 遊戲細部功能（輪流操作、結束遊戲後停止UI介面更新）
* debug
* 報告文件撰寫
* 報告影片後半段（說明程式碼、作品使用之技術）
* 時間：兩個下午+一個晚上

### 1102916 游雅碩

* 遊戲框架
* 遊戲細部功能（判斷輸贏）
* 調整GUI介面
* debug
* 作品執行過程影片
* 報告影片前半段（介紹作品架構、特色等）
* 兩個下午、晚上

## 9. 兩份心得

### 1102914 龔弋筠

因為課程中的程式主要著重於C++語言的撰寫，因此這是第一次親自寫出python的client server程式。幸好python的可讀性很高，遇到不會的部分也可以透過上網查資料或是詢問chatGPT而很容易就能學會。  
一大挑戰是學了一學期的網路程設才第一次**結合GUI介面**，一開始不禁懷疑：要做出遊戲是可行的嗎？？ 謝謝老師事先提供了Tkinter教學，否則若少了GUI介面的呈現，有很多類型的遊戲都會被侷限而無法實作出來。  
總之，感謝隊友的合作，包括一起想主題、一步一步實現遊戲功能、debug，甚至一起找不到bug，最終能做出符合我們預期的作品實在很感動～

### 1102916 游雅碩

在之前很少用python來實作TCP Socket Server/Client網路通訊程式，較常用C語言來寫作業與練習，不過老師介紹的製作GUI的方法是使用python裡的tkinter來製作，所以我們就選擇使用python來完成這次的專題。  
在寫程式碼過程中有遇到一些問題，原本以為chatgpt會說出哪裡錯誤、如何更改等，而花了很久跟他溝通，結果蠻常得到一些無用的答案，或許使用wireshark來看封包如何傳送，會更快些吧。