

FOOP Final Project Report

Ticket To Ride @ NTU

資工二 B03902084 王藝霖

資工三 B02902031 李彥霆

遊戲畫面：<https://youtu.be/Ooy15G0YI64>

1. 分工：

王藝霖：GUI 設計

李彥霆：Client, Server, Game 設計

我們事前討論了程式的架構，主要可以分割為 GUI、Client、Server、Game，然而衡量這些部分的難易度，對於完全不熟悉的 GUI 設計，我們認為需要花很多心力去摸索，所以我們將兩人的工作大致上分為 GUI 與其他的程式設計，而 Debug 與 API 溝通仍然是一起討論的。而 GUI 的設計最後也如我們所預料的，佔用了幾乎一個人的資源去做到精美細緻的呈現。

2.

I. Server package:

Server 執行於任何一個可以連線的位置，處理玩家的連線與玩家在遊戲開始之前的選擇，而 Table 負責處裡玩家選擇一個桌子到遊戲開始之間的狀況。Listen 是一個獨立的 thread，專門接收新的連線。

ServerPlayer 負責處理所有與玩家的連線，是與客戶溝通的核心。

Out 本意是一個 synchronize 的 output stream，但最後在實作上回到原本的 blocking mode。Request 是與 ServerPlayer 互動的物件，任何一個與 Server 互動的 class 都需要使用。

II. Client package:

這是我們實作與 Server 對應的接口，扮演著 GUI 與遊戲之間的橋樑。不同於純粹的橋樑，在這裡我們設計了 GameController 檢驗玩家的操作，任何違反遊戲規則的操作都會在這裡被擋下來，以減輕 Server 的負擔與溝通的複雜度。GameData 是當前的遊戲資訊，作為 client 與 GUI 的溝通元件。這裡的 Out 與 Server 端一樣，因為實作上的改變使得他回到一個 ObjectOutputStream 的腳色。

III. game package:

所有的遊戲邏輯寫在 BroadGame，控制遊戲的過程。而遊戲的實體物件如車票、任務、地圖等....都是一個個獨立的 class，其中 shuffler 掌握了遊戲所需的車票與任務，經過洗牌提供隨機性與更多的挑戰性。為了方便記錄一個玩家的行為，我們新增了 ChooseAction 的 class，使得其他玩家可以很輕易地得到遊戲進行的狀況。

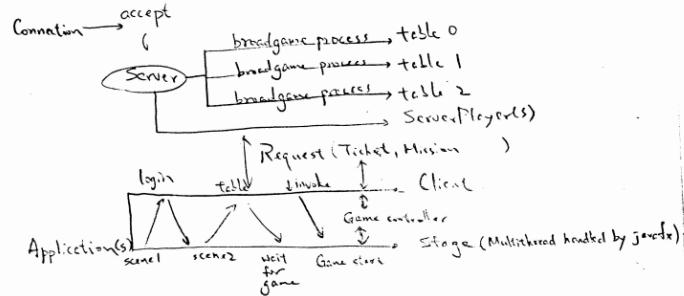
IV. GUI:

大部分都在處裡與螢幕的互動(Action event)，Class 架構跟隨著遊戲畫面一幕一幕的切換，有 Login -> ChooseTable -> mainGame 這三個 controller。因為使用 scene builder 與 css 的關係，我們有大量的 fxml 與 css 檔案而較少的 class。

V. package 之間的互動:

Server 與 client 透過 Request 溝通，Game 的主要內容是在 Server 端，他的各種小物件則是 Request 主要傳輸的對象。因為 javafx 的設計，不是 javafx application 的 thread 不能對他的內容操作，於是刷新畫面我們採用 javafx thread 對於 client 的不斷拿取最新的資料。

VI. 圖示：



3. 設計的優點：

I. Server-Client 而非單機設計的優點：

Server-Client 的設計可以有更多的彈性，以 Request 溝通使得 Client 端的設計只要符合這樣的協定，就可以連線到 Server 上遊玩。分成兩個近乎獨立的 package，使得我們在實作的時候負擔較小，進而提升寫程式的速度。

II. Out 的設計回到 blocking mode 的優點：

Server 比較好控制遊戲進度，因為這個遊戲本身就是一個一個玩家輪流詢問，其他玩家提早地回復對於遊戲本身並沒有幫助，而且這樣的 code 也比較有掌握性，在除錯階段能夠比較容易地找到臭蟲。

III. 使用 ObjectOutputStream 的優點：

考慮傳輸資料時，主要有 java 提供的 ObjectOutputStream 與 Json，Json 的使用範圍較廣泛，然而 ObjectOutputStream 的操作更為容易。在考慮 client 與 server 都是 java 的實作下，我們選擇 ObjectOutputStream。只要 implements Serializable 的物件，都可以直接傳輸。這帶給我們很大的便利，也使得程式簡單好讀。

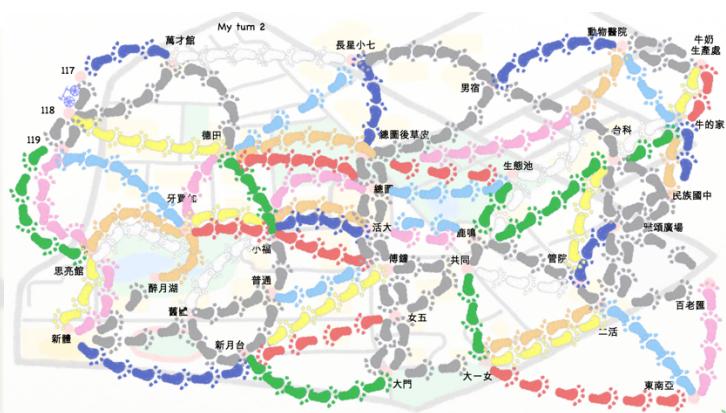
IV. javafx 的優點：

能夠引入 css 語法美化介面，將按鈕的質感大幅度提升，透過 Scene Builder 能夠較容易的排版，在畫面風格呈現使得我們能夠一貫我們的塗鴉風格。

4. 設計的缺點：

- I. 一開始思考不夠周全，導致原本設定的 API 不敷使用，一直修改，導致有些區塊的程式碼如補丁般，附加在原本的設計裡。又受限於分散的架構，有時想到的改寫方法，會導致過於巨大的結構改變代價。
- II. FXML 的語法導致宣告的變數無法是結構化的資料，我們有大量的 Button00, Button01... 的變數來管理我們大量的按鈕。儘管我們有時可以自己寫一些腳本產生這些程式碼，這依然無法改變程式碼過長而可讀性大幅下降的問題。
- III. 設計缺陷之改善方法：花更多的時間做事前的討論，訂下 SPEC，讓寫程式的思考在動手之前就有了藍圖，讓程式更具有結構性。另外，更深度的瞭解 javafx，尋找可能替代的寫法，讓程式更加可讀。

5. 遊戲玩法



A. 緣起

台大，是我們每天生活的地方，熟悉的台大地名，瞭若指掌的台大地形。玩膩了歐洲版美洲版的 Ticket To Ride 嗎？那麼台大版是你最棒的新選擇。台大版是個不但充滿回憶，又充滿驚喜的最新版本。

我們在尋找桌遊的過程中，看了 google 圖片搜尋中成千上萬的桌遊照片。在某次機緣巧合之下，赫然發現美洲地圖和台大地圖有某種程度的相似，我們便決定了這次的 project 的主題是 Ticket To Ride，將原本的美洲地圖，以幾乎不改變道路長度的前提下，改繪製成台大的地圖。並且任務沿用美洲版原本的設計，將美洲城市對應到台大地點。

在繪製地圖的過程中，我們驚訝地發現原來台大還有這麼多我們不知道的秘境，值得我們去探訪。還有許多地點的相對位置，我們也更加清楚。常玩台大版 Ticket To Ride，從此近路小達人的稱號就非你莫屬啦。

我們所決定的遊戲風格為，輕鬆的塗鴉風。相對於鬧區的吵雜擁擠，台大是個非常悠閒清新的地方。因此我們決定以「漫步台大」的意象，為忙碌的日子增添一點調劑。

B. 遊戲簡介

本遊戲的主要目標為，使用手中僅有的車票，盡量的探訪各個地點。

獲得分數有兩種方法：一種是走過即有的分數，走越多路，則獲得越多分數，另一種是達成任務卡的要求。任務分為短程和長程任務，長程任務較難達成，但能得到的分數則較多。短程則相反。在我們的遊戲實做中，長短程任務是在同一個任務卡堆中的。

遊戲中還有一項規定，車票和道路必須是相同的顏色，才能通過。由於這項規則的加入，使得遊戲不再只是純粹比誰運氣好的遊戲，而是必須靠策略、察言觀色、小心下手才能獲得勝利。如此一來，遊戲更加的富趣味性。

C. 遊戲執行

使用 make 指令編譯

使用 make runServer 指令執行 Server (預設 port 為 5410，若要指定加 PORT=XXXX)

使用 make runClient 指令執行 Client (預設 address 為 127.0.0.1，若要指定加 SERVER=XXXX PORT=XXXX)

D. 連線部分

輸入 ID 後，進入選桌畫面，在加入遊戲桌之前系統會持續更新遊戲桌資訊。五人即開始遊戲。



例如此畫面中，ID 為 1~5 的 player 正在等待系統開始遊戲。(Join 在點選後更新成 Wait)

E. 主遊戲部份

(1) 在遊戲開始時，每個玩家手上有 4 張 tickets、步數剩餘 45 步



(2) 每一輪，每個玩家可以選擇以下一種動作

- a. 抽兩張 tickets (牌面上或牌堆中)
- b. 抽任務卡 (最多 5 張)
- c. 消耗 tickets，走某條道路



(3) 計分方式：

- a. 每種長度的道路有分別的分數，走過即可獲得。

長度	1	2	3	4	5	6
分數	1	2	4	7	15	21

- b. 每張任務卡有自己的分數 (任務卡下方)，若達成，即可得到該分數。

(4) 遊戲結束條件：

任何一個 player 的剩餘步數 $<= 2$ ，則下一輪到此 player 時遊戲結束

6.

我們使用了 javafx 來設計 GUI 介面 (with fxml files)，我們的遊戲所用到的圖均自行設計繪製，並且使用 css 檔案設定樣式。遊戲主地圖上的每條道路上的每一個腳印都是一個 Button，因此可以點選道路上任何的腳印。

選擇要走的路之後，原本為腳印形狀的道路，就變成腳踏車，而且顏色是玩家的代表顏色。

例如：(代表色為藍色的 player)

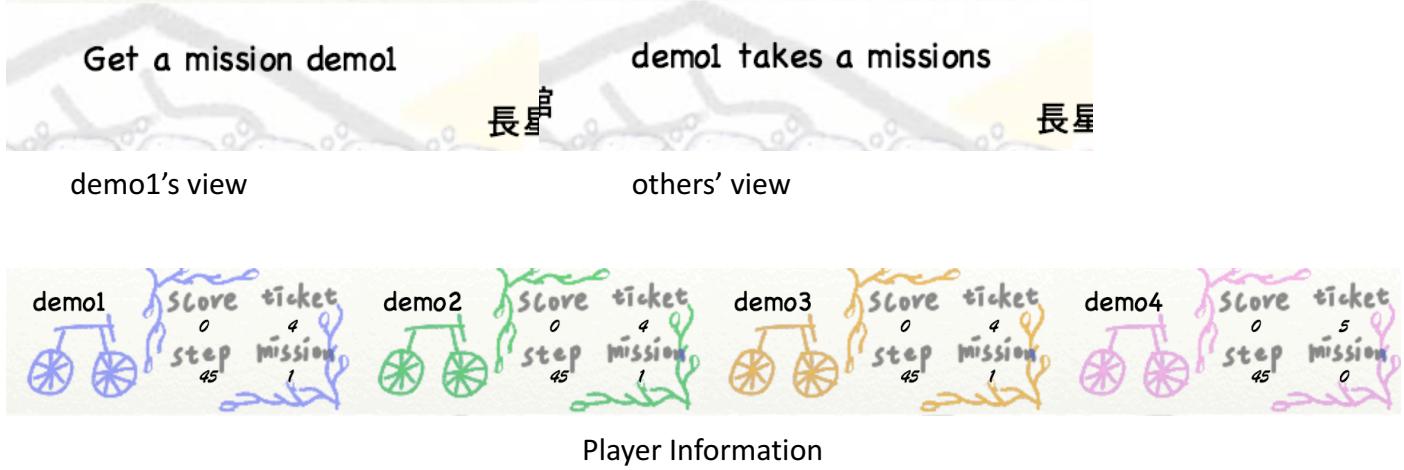


每個場景均有一個相對應的 Action Controller，分別是：

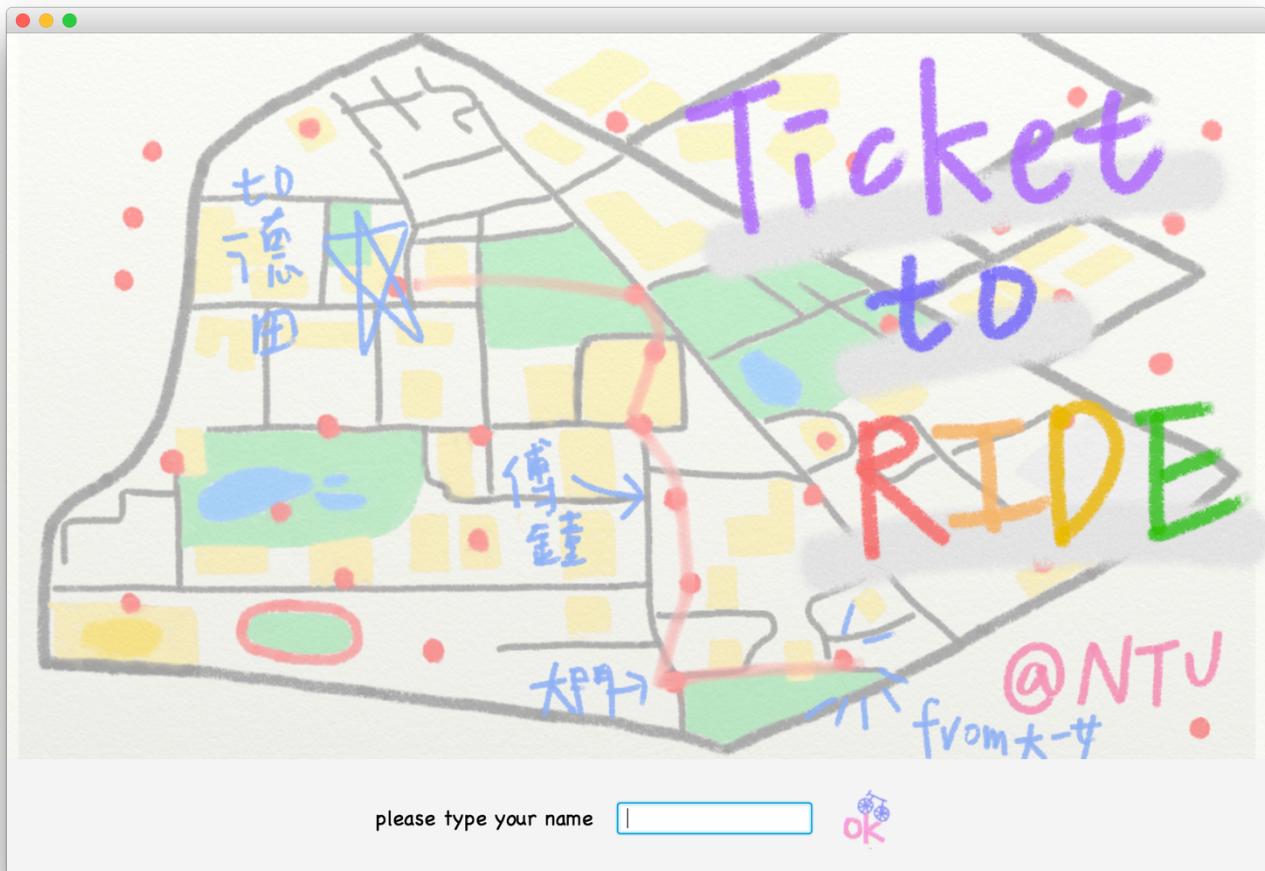
LoginController / ChooseTableController / ScoreBoardController

每個 class 都有 myinit method，用來初始化 player 的設定值等。以及大量的 @FXML instances 用來控制圖形介面的顯示。還有其最主要的功能，作為 button 等控制元件的 Action Listener。

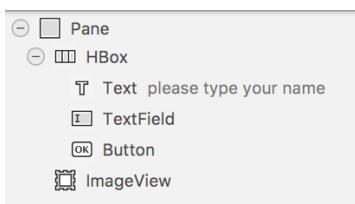
由於在 javafx 中並沒有自動更新畫面的設計，但是我們是連線的遊戲，因此我們使用了 OnMouseMoved Action 來觸發更新畫面。有任何一個 player 做了任何動作，其他 player 也可以即時收到資訊，並更新圖形介面。(左側的牌堆、各個玩家的計分板也會同步更新)



(1) 登入畫面：



javaFX 結構：

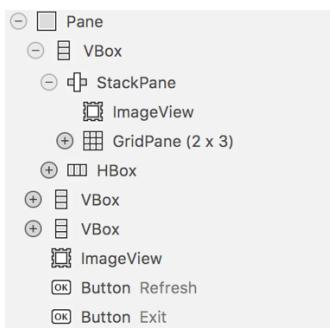


畫面下方使用 HBox，因此可以依序加入文字輸入框及按鈕，讓 HBox 自動調整位置。

(2) 選桌畫面：

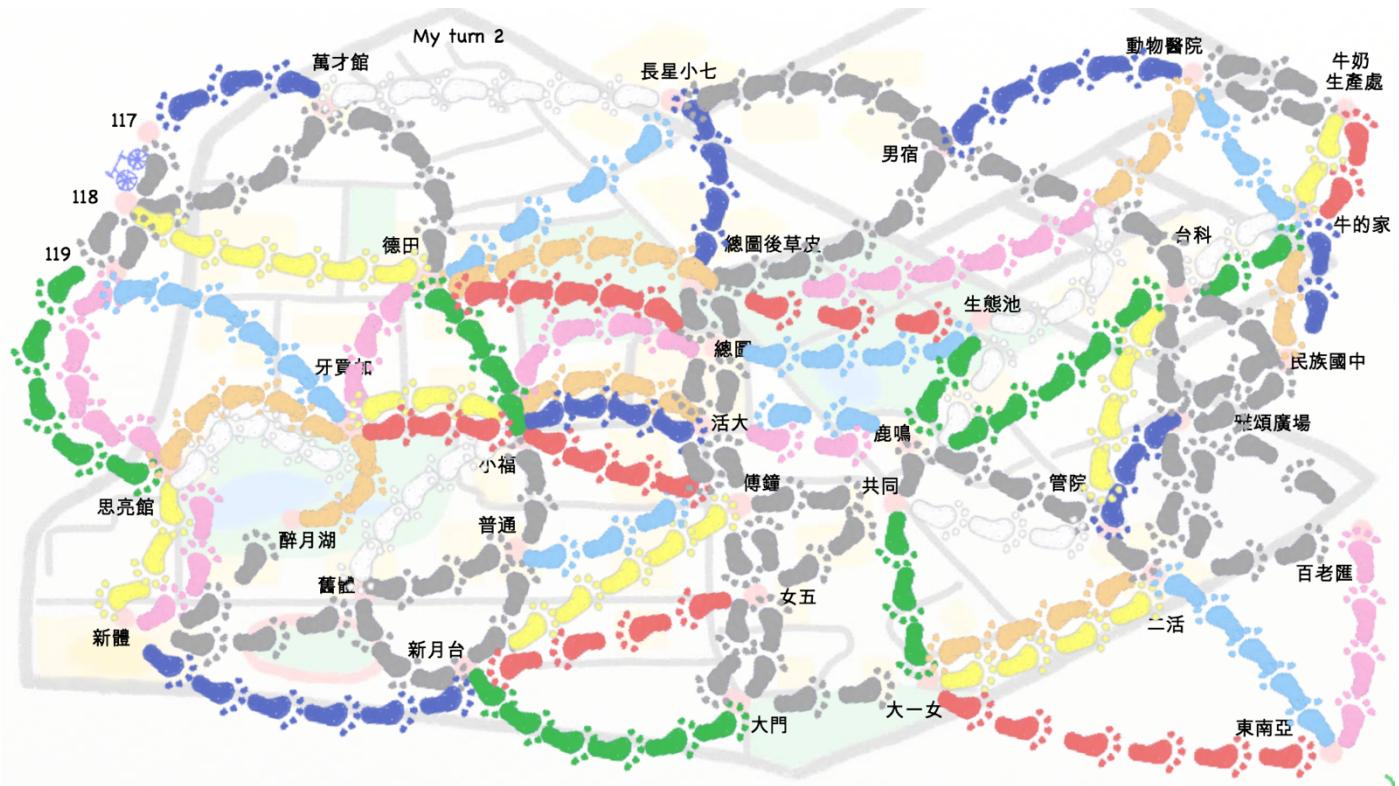


JavaFX 結構：



使用 GridPane，可以使 Text (players' name) 整齊排列。

(3) 遊戲主畫面：



javaFX 結構：

