專案題目：全員逃走中之寶箱解謎
小組名稱：4P
小組成員：103060010 江秉翰 103060002 關力豪
101022133 黃廷耀 103062330 游庭維

一、背景與目的

在日本熱門綜藝節目遊戲「全員逃走中」，玩家需要在一個大範圍的場地內，完成指定的任務，然而在完成任務的同時，獵人也在場地尋找玩家，追捕並進行淘汰。遊戲緊張刺激，考驗著玩家的體力與智力，如何躲避獵人的視線，將是生存的重要關鍵。然而這個遊戲需要寬闊的場地，大量的人力，以及充足的事前規劃，舉辦此一活動絕非易事。本組希望藉由移植此遊戲到電腦中，讓沒有機會參加實際活動的人，藉由電腦的虛擬世界，也能體會到「全員逃走中」的樂趣。

另外，現在公共場所的AED(自動體外心臟電擊去顫器)數量逐漸增加，在發生緊急意外時，AED能及時救回人的性命。然而AED的使用時機必須，在短短數分鐘內才能發揮效果，許多人卻不知道AED的位置，加上找尋的時間，勢必為時已晚。因此希望在玩家遊戲的同時，在清大地圖中尋找AED的位置。知道AED的位置，在實際發生意外時，才能在最短的時間，用AED進行急救。

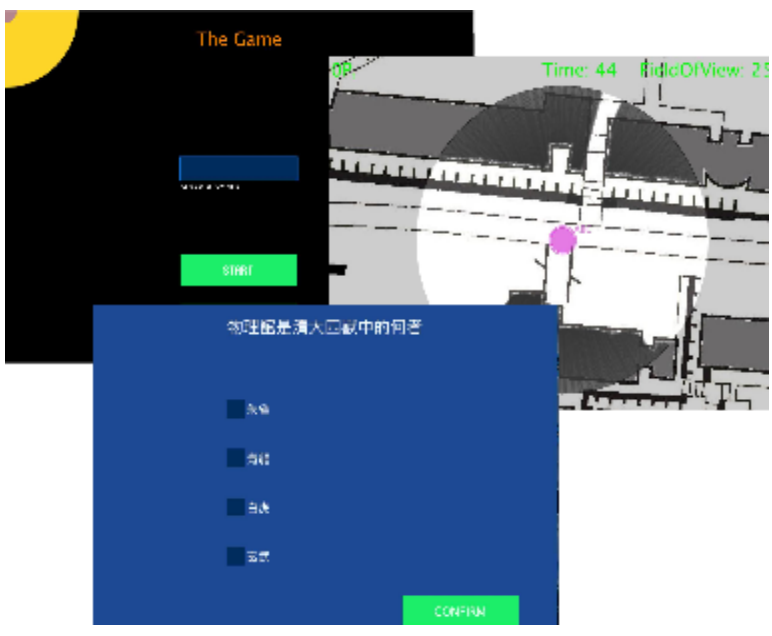
二、設計概念

<u>遊戲化設計</u>：

本遊戲為了讓大家更了解清大的校園，以清大地圖作為遊戲的地圖。玩家在遊玩時會在一張比例尺較大的地圖上活動，只看的到自己周遭的建築物和地形，按下tab鍵後就可以瀏覽清大的全圖，而玩家的會以點的形式出現在地圖上，這樣就可以更清楚的知道現在自己是在地圖(清大)的何方。

而地圖上標有數個X的記號，象徵著寶箱的所在地，而這些點並不是隨意找幾個點標出來的，我們在這裡下了一點巧思，所有的點都是清大內部AED(自動體外心臟電擊去顫器)的位置。

最後一點，當玩家接近地圖上的X時，按下空白鍵即可觸發任務，這些任務其實就是一些簡單的選擇題，而題目的內容都和清大相關。例如：校狗丁丁的本名為何等等。



三、技術細節

(一)、processing & JFrame(畫面切換)

遊戲中的畫面基本上可以區分為三個部分，遊戲的開始畫面、遊戲進行中的遊戲畫面以及任務時的答題畫面。這三個不同的畫面分別是三個不同的PApplet進

行繪製。我們將含有JFrame的Main作為Observer，並且將三個不同的畫面的PApplet當作Observable，當需要切換畫面的時候（例如：按下開始按鈕、按下空白鍵開啟任務界面），Observable就會通知作為Observer的Main，此時再將JFrame中顯示的PApplet更換，達到切換畫面的效果。

(二)、socket (client & server transmission)

在Server中，首先會有一個thread不斷等待接收client的連線，只要有client連線進來，就創一個對應的socket和client配對。另一個thread，Gamethread負責執行Server中的遊戲主程式，在這其中會對每個client都創一個thread，不斷接收client傳來的訊息，並儲存到Server的記憶體中，Server會收集所有client的資料，彙整之後broadcast給每個client，如此每個client都可以獲得其他玩家的狀態。另外，Server有控制獵人的程式，也會把每個獵人的資訊傳給client。

在client中，有一個thread會接收和傳送給Server訊息，client會把自己的資訊傳給Server，並從Server獲得其他玩家及獵人的資訊。

(三)、JSON (detailed description)

本程式的網路傳輸方式，都是把資訊放進JSON，再把JSON轉成String，用Socket傳給client或Server，接收方再把資訊取出來。透過client中的Class Transfer和Server中的Class JSON，可以把資訊轉成json String或把json String中的資訊取出。

Server傳出的資訊包括:遊戲時間，遊戲執行階段，每個玩家的位置，名子和狀態，每個獵人的位置，每個寶箱的位置和狀態。

client傳出的資訊包括:自己的位置和狀態，自己正在開的寶箱。

(四)、Hunter(AI) and Player

Player：

玩家在遊戲中以一圓點的形式呈現，每個玩家都有自己的視野範圍，只有在視野清晰的地方才可以看到任務和獵人，而玩家可以透過解任務的方式擴大視野範圍(詳情請見View及Mission)。在地圖上玩家可以透過點擊螢幕來移動，玩家的位置會隨著點擊的位置而改變，但玩家無法穿越地圖上標記為灰色的建築物(障礙物)，這邊使用到了Processing中的Ani繪製點的移動。

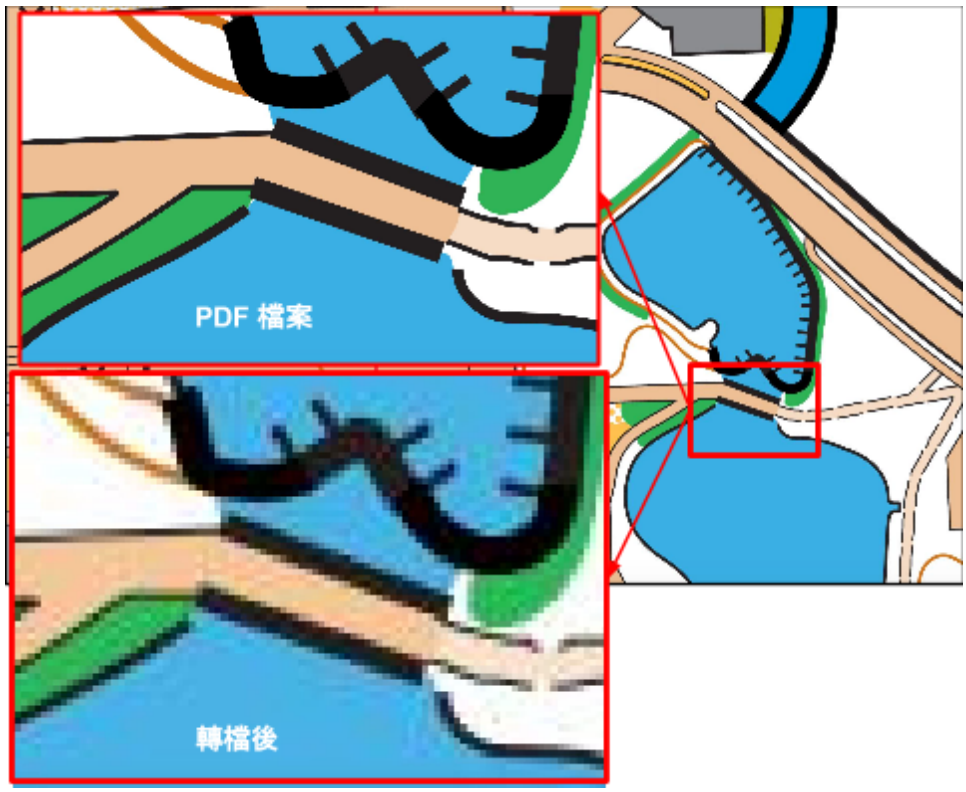
Hunter：

Hunter為AI控制，總數為15個，一開始會先分配到地圖上的各個地方並以較慢的速度遊走，Hunter的移動和Player相同，一樣透過Ani來達成。計算出下一個將要移動到的點則是透過java.util中的Random函數，隨機跑出0,1,2,3來決定要走向北、東、南或西。在走之前會先判斷路徑上是否有障礙物，如果有障礙物的話就換一條走。

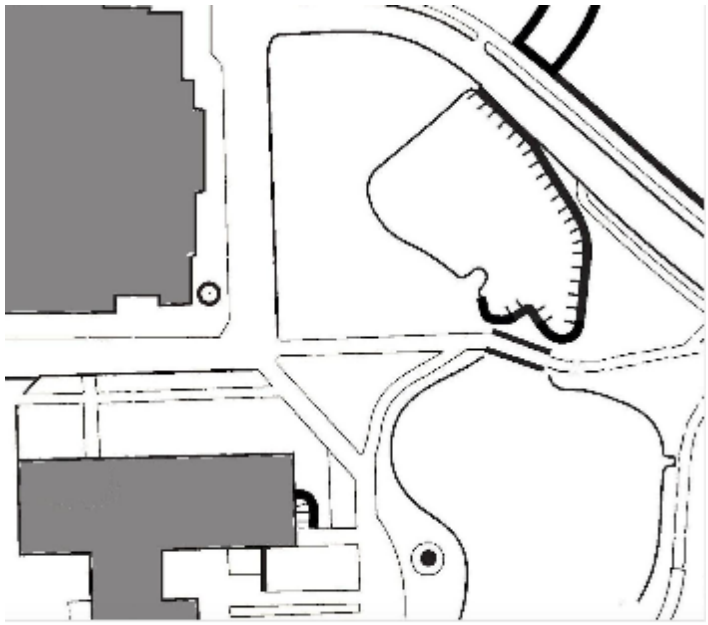
另外會隨時監看獵人的視野範圍內是否有玩家出現，如果有就開啟狩獵模式，移動速度加快，並且以玩家目前所在的點作為下一個移動的目標，並且設定若追了五次還是沒有接觸到玩家，就放棄狩獵。

(五)、Map and Picture editing

遊戲中所使用的清大地圖來自體育課定向越野所使用的地圖，原本的格式為PDF，經過轉檔將地圖輸出為.jpg檔的時候，若是採用較小的尺寸，地圖的解析度會明顯的降低，原本的直線會產生顆粒，由於我們希望玩家在遊戲的過程中也能夠慢慢熟悉清大的校園，我們希望圖片在放大之後能夠維持清晰的畫面，因此遊戲中使用相當大的圖片，jpg圖片大小為9890x9770。

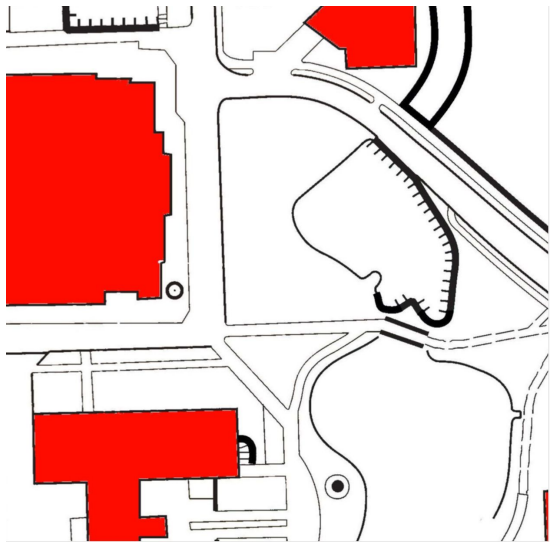


選定圖片大小之後，利用自己另外寫的Java程式，藉由判別每個像素的RGB範圍，將圖片上的其他色彩轉成黑白，僅保留原圖建築物的灰色和黑色區塊，以期讓畫面更加簡潔。



黑白色的遊戲地圖

為了製造出玩家視野無法穿過建築物的效果，我們利用和地圖大小相同的二維陣列，並將無法穿過的灰色建築物部分在陣列上標註，方法和上述轉為黑白的圖相同。然而由於黑色及灰色的的RGB範圍太過接近，導致圖中道路旁的細線也會被誤認為無法通過的區域，因此我們又另外製作了一張地圖，利用修圖軟體將建築物塗成較容易辨識的紅色。在遊戲一開始時，我們將紅色的地圖讀入，將紅色的部分標註在對應的二維陣列上，之後再另外讀入遊戲中所使用的黑白遊戲地圖進行遊戲。



由於讀圖相當耗費時間，我們原本打算將二維的陣列預先產生之後再serialize儲存成檔案，方便遊戲開始之後讀入，不過實際產生出來的.ser檔案實在太大，超過github能夠上傳的檔案大小上限，因此最後並沒有採用。

(六)、View

在View中，會處理各種資訊後並顯示在畫面上，像是各位玩家和獵人的位置、任務的狀態、得分和遊戲時間等等。在顯示時分成兩種畫面，一個是大地圖，另一個是區域地圖。

在處理大地圖時相對較為簡單，先將整張地圖畫上去，再畫上每個任務，接著呼叫出每個玩家的位置和ID，將位置換算成座標並根據ID畫上不同顏色的圓，此外為了方便玩家判斷其他玩家是否陣亡了，我們在左邊加上了一個玩家名字的清單，若死亡則在名字上畫一條橫線。

小地圖的部分則較為麻煩，為了能讓地圖隨著玩家移動而改變顯示範圍，每次移動後就要從大地圖擷取出一小塊地圖並畫上去，然後根據玩家的位置是否在邊界決定是玩家移動還是地圖移動。另外由於我們有設計玩家的視野和建築物的遮蔽效果，因此在畫上獵人和其他玩家時，要先判斷是否在視野內，但最麻煩的是為了讓玩家能辨認出可視範圍，我們打算在不可視的部分鋪上一層灰色的陰影，原先是打算一個像素一個像素畫上去，但這樣的時間複雜度太高了，會導致運行太慢，後來決定先畫上一整塊方形陰影蓋住整個視窗，再把可視區域挖出一塊空白，但很不幸的是似乎是因為我們Processing版本沒有完整支援挖空的Method，一直沒辦法順利使用，最後幾度微調陰影的透明度才勉強達到可接受的程度。最後為了讓圓形視野可以模擬光的直進性，我們把視野切成360度分開掃描，每個角度由近到遠偵測該點是否有建築物，若有則在這點開始畫上一條黑色的陰影蓋住後面的區域，可惜的是360度還是不夠細緻，當半徑太大時會出現沒有遮蓋到的部分，因此我們將陰影的線條加粗，但隨之而來的缺點便是線條重疊的部分陰影看起來會較黑，這部分也是透過透明度調整成勉強可接受的結果。

(七)、Mission

遊戲中的任務分成兩個部分，Client 的部分負責顯示題目，並且確認答案的正確性，並且將結果回傳給 Server，Server 的部分則根據每個 Client 回傳的結果，讓開過的寶箱開始倒數計時，並在倒數計時結束時，將寶箱重設為可開啟的狀態。

有關 Client 的部分，任務的問題會在遊戲開始時讀取文字檔中的題目和地圖上每個寶箱的位置，建構一個題號對應到選項的 HashMap。隨機選取 Map 中任一題號，即可產生任務所需的題目。當玩家回答正確時，便會回傳寶箱的編號給 Server。

有關 Server 的部分，由於 Server 並不需要回答任務的問題，因此 Server 僅會讀取地圖上寶箱的位置，當 Server 接收到回傳的寶箱編號，則會將對應的寶箱利用新的 thread 開始倒數計時，並在倒數結束時，再將寶箱重設。

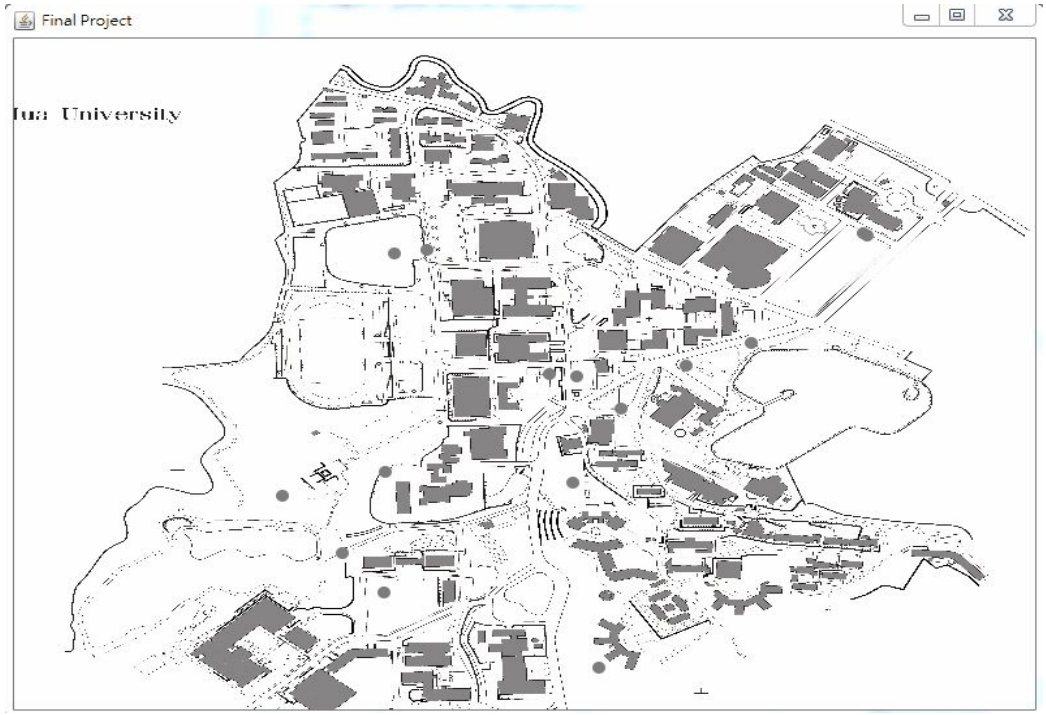
在寫這部分時，遇到的問題是在讀取中文題目的時候，有因為沒有字型而無法顯示，且在檔案的第一行會有無法解讀的亂碼，後來我們才瞭解在讀入時必須指定編碼的格式才會運作正常。

(八)、Music

音樂處理上我們用到Processing中的Minim來處理，我們設定了背景音樂、滑鼠點擊音效、答題音效(正確&錯誤)和撞到建築物的音效等等。

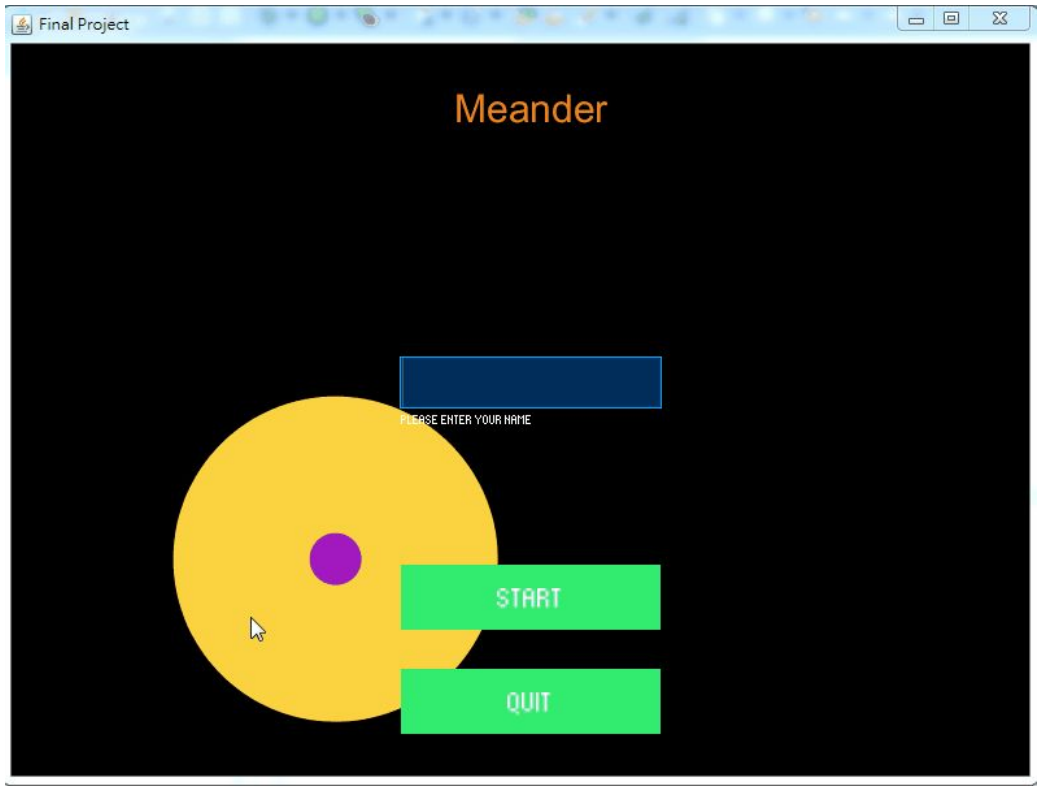
四、實作成果

1. 首先，開啟Sever後，會在地圖上產生獵人們，圖上的灰色小圓圈為獵人的位置。

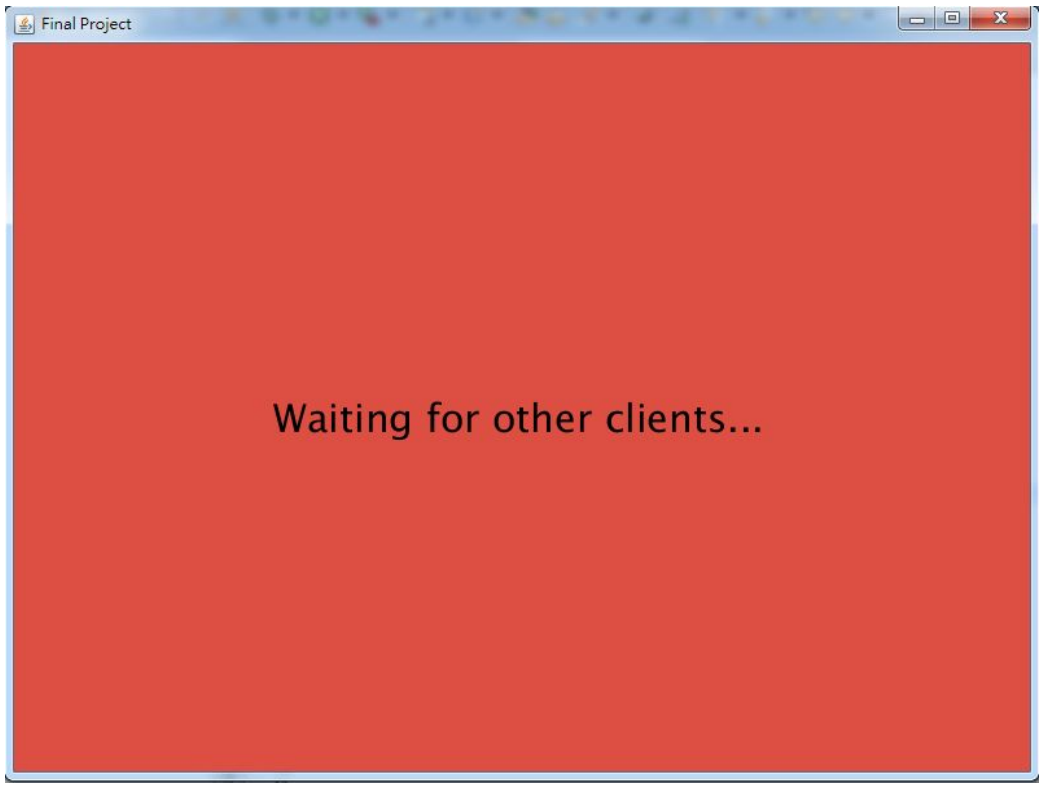


2. 遊戲開始畫面。於中間的文字框輸入名字後即可開始遊戲。此外有個小設計，背景的黃色圓圈可以想像成手電筒，會隨著滑鼠移動，有機率在視窗上照出隱

藏著的獵人(H黑點)。

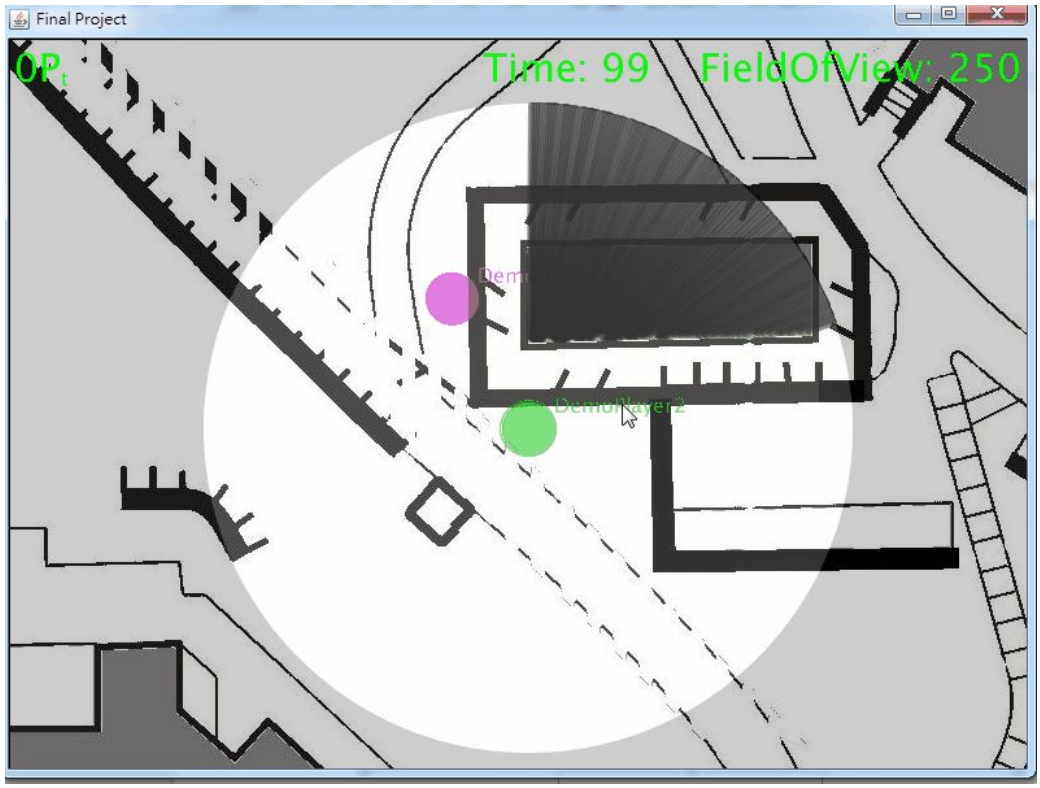


3. 遊戲開始後，若連線的玩家數量小於預設值便進入等待畫面。

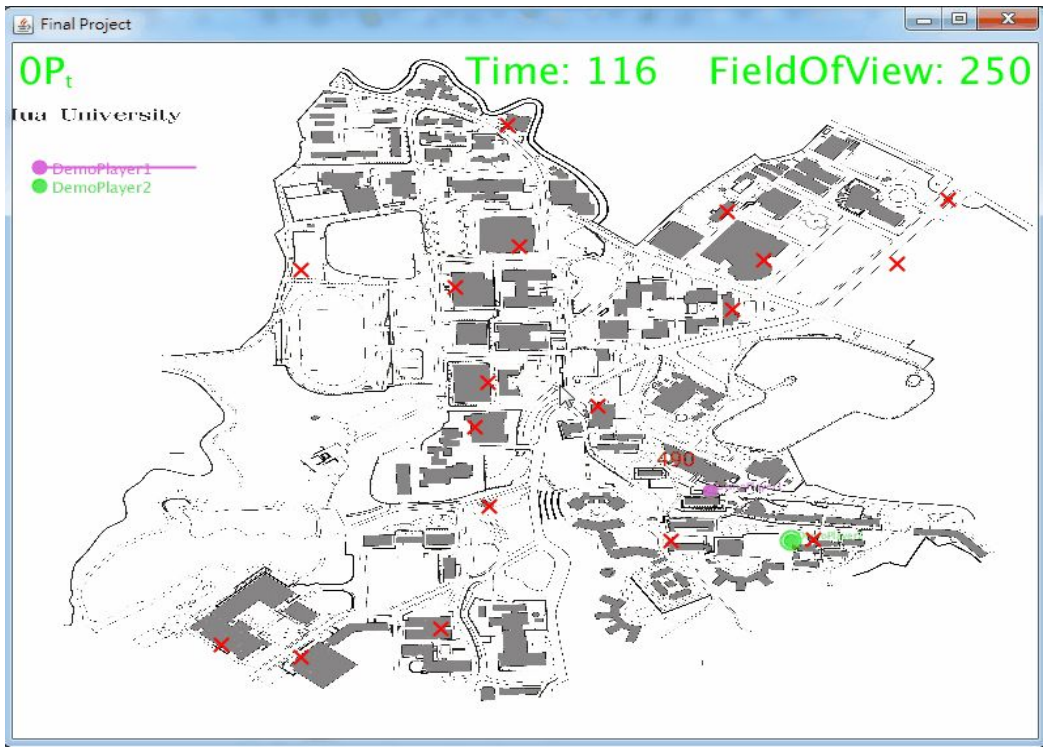


4. 遊戲畫面。左上角為回答題目的得分，右上角分別顯示時間和目前玩家的視野大小，灰色部分即為不可視區域，在視野範圍內可以看見其他玩家或獵人，但

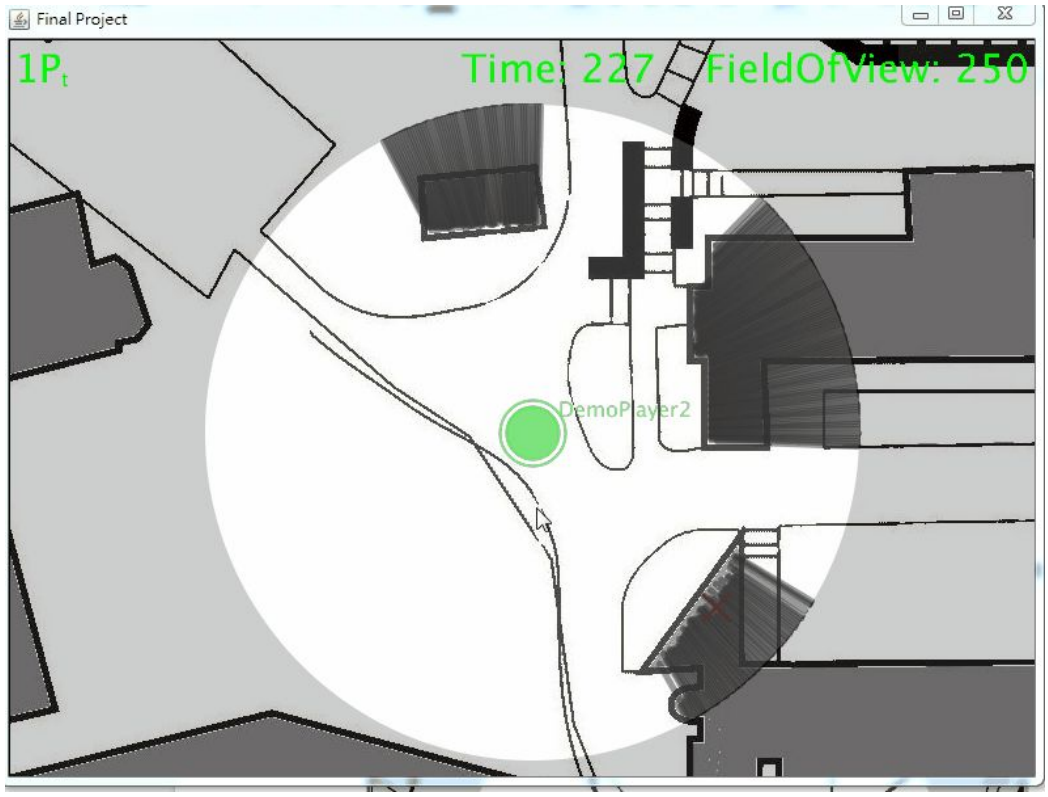
建築物後面會產生陰影擋住其他人。



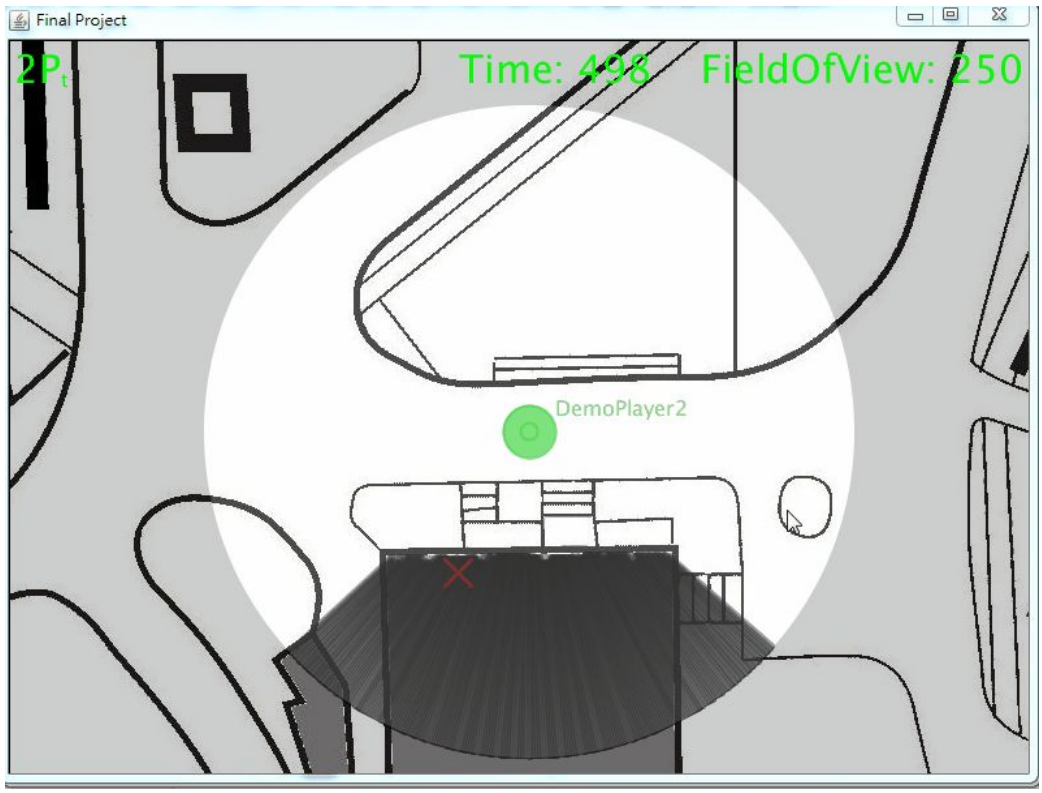
5. 大地圖，在遊戲畫面中按下 Tab 鍵會顯示。左上角會顯示玩家們的顏色與名字，畫上一條橫線代表該玩家已死亡，另外，地圖上的紅色叉叉代表任務的位置，當玩家靠近到答題範圍內會產生圓形波來提示，若該任務已被完成，則改為顯示刷新的倒數秒數。



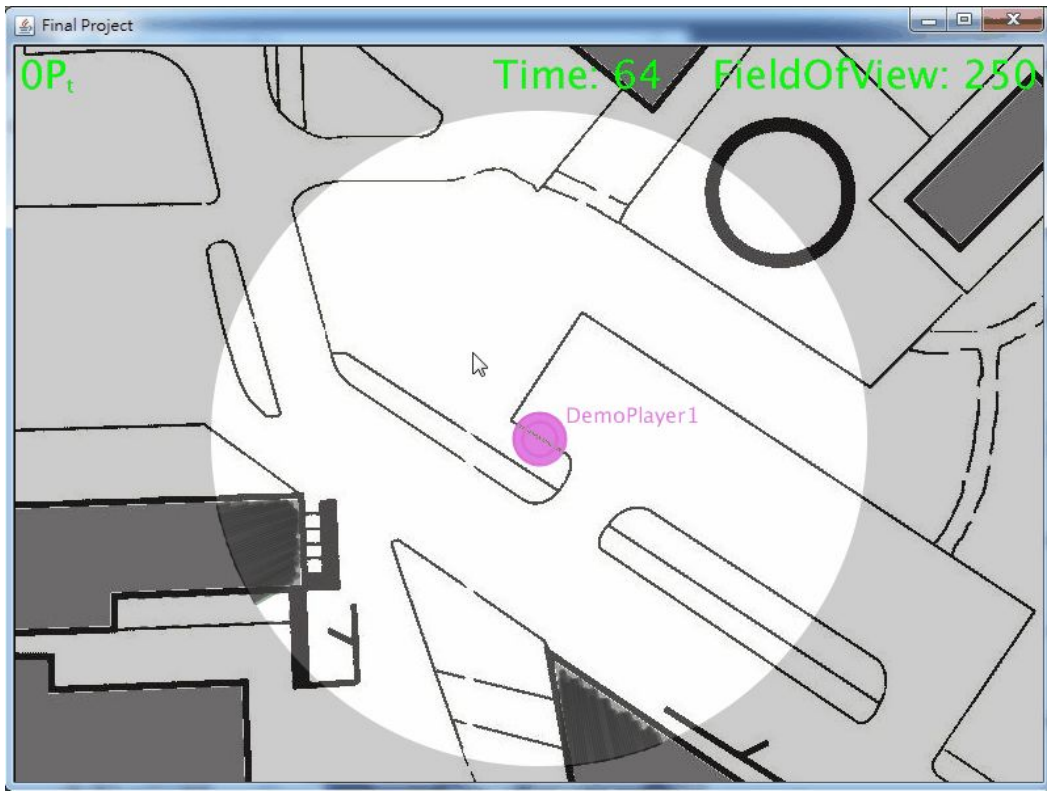
6. 答題畫面，選擇答案後按下確認，對錯會分別有各自的音效。



7. 視野擴張，當答對題數達到3的倍數時，視野就會增加，玩家的可見範圍增加，可以提早看到獵人。

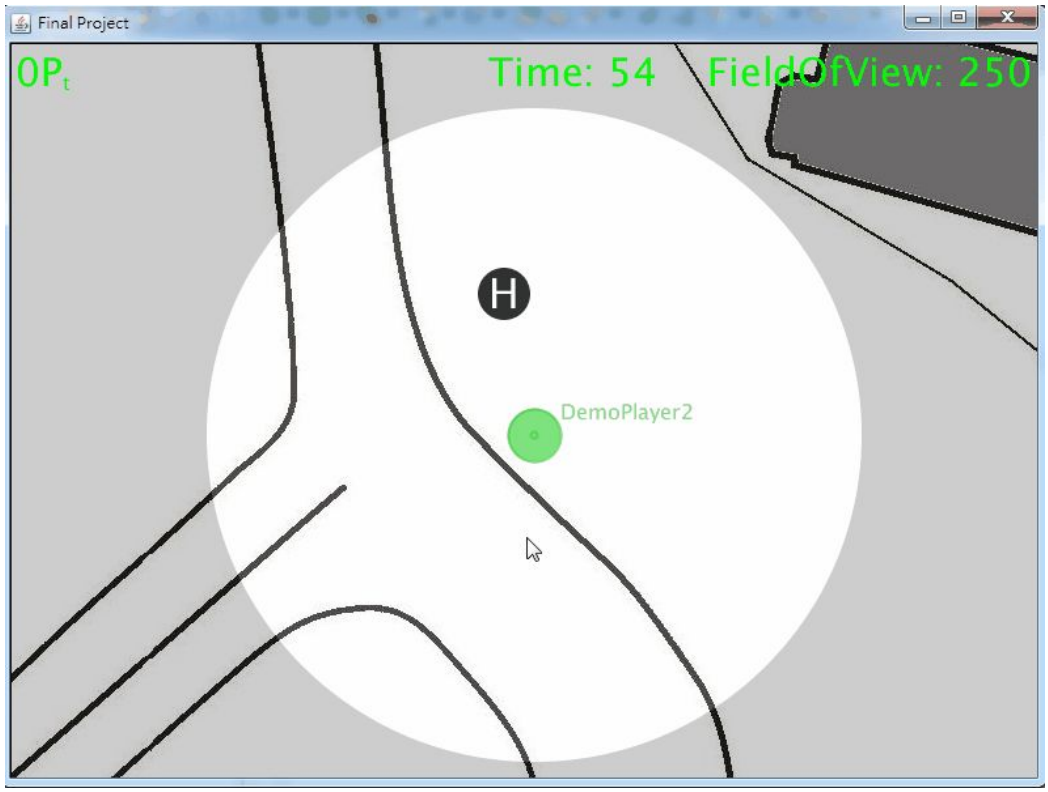


8. 獵人(黑色H圓圈)，當靠近到一定範圍內，獵人便會自動追蹤，追一段時間才會放棄。

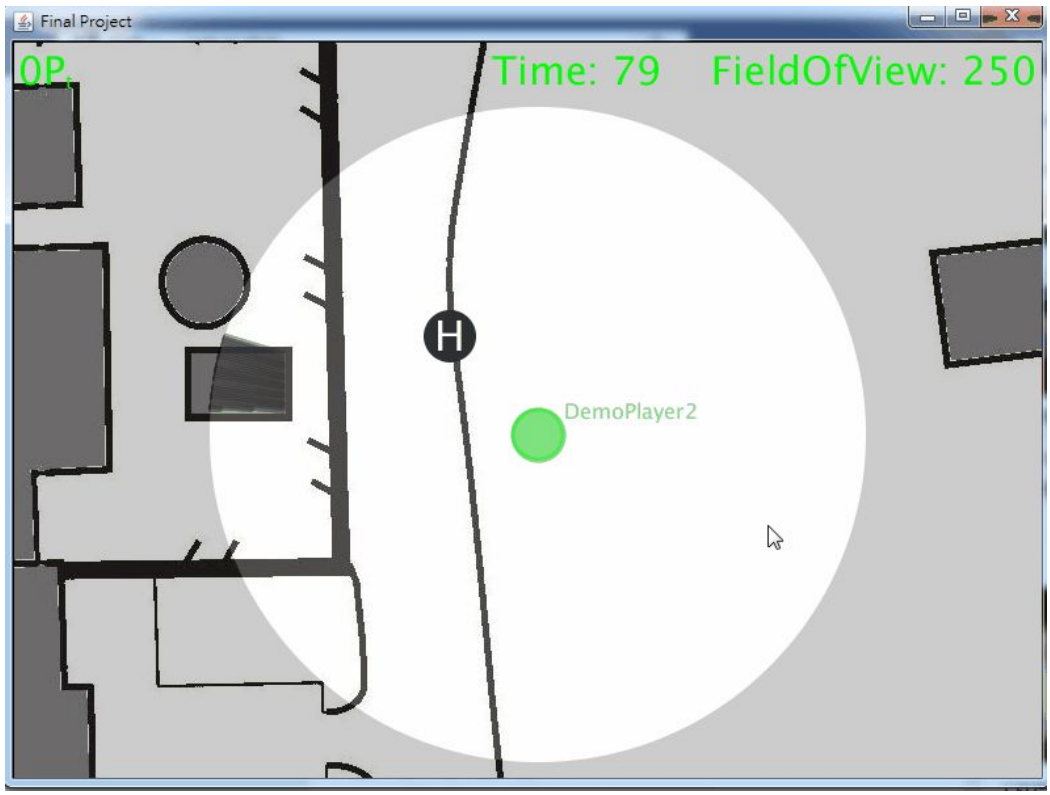


9. 死亡狀態，當被獵人殺掉後，會進入幽靈模式，可以自由穿越牆壁移動，觀察其他玩家和獵人，此時沒有視野的限制，獵人也不會追幽靈，但同時，幽靈模

式的玩家無法被其他玩家看到，其他玩家只能看到他死亡的位置。



10. 當所有玩家都被殺掉後便進入結算畫面，顯示該次遊戲的生存時間。



五、遇到困難

(一)、Hunter AI

一開始設定讓獵人在地圖上隨意遊走時，會發生兩個問題，第一個是走超過邊界，第二個是一職撞牆，或卡在死胡同裡出不來。於是在判斷路徑的時候我多寫了兩條判斷他將要移動的點是不是在地圖之內，還有路徑上是不是有障礙物。

(二)、Minim

我們使用AudioPlayer的型別來撥放音樂但遇上當播放了一次之後就無法再播放的問題，去網路上找了資料才知道在重複播放前要先call rewind()。

(三)、Github conflict

在寫網路連線時，因為原本連線架構Socket和JSON是不同人寫的，寫連線架構的人為了符合他寫的架構(需要傳輸的內容等)，改動了JSON encode和decode的內容，卻沒有通知其他人，寫JSON的人也為了其他原因，改了JSON的內容。之後在merge的時出現很大的conflict，最後其中一個人把他的branch關掉，創一個新的才修好。

(四)、資料傳輸

目前本程式傳輸資料的作法是client只傳自己的資訊給Server，另一種做法是傳送所有玩家的資料，只改動自己的欄位，其他玩家的資訊不要動，如此可以讓傳送和接收的格式是一樣的，在encode和decode JSON時也可以重複利用同樣的程式。但是實際做了之後，發現會出現Server和client同時改動檔案的問題，造成該改的資訊改完又被覆蓋掉了，如果要採用該作法，需要另外一種控制傳輸時間先後程式。所以現在的做法是，改動檔案只在Server進行，Server根據client的id和所傳過來的資訊，改動對應的欄位。

(五)、陰影

原本是打算直接使用Processing內建的blend()，來達到挖空的效果，結果貌似因為版本不是最新版本所以有些Method不完整，但這時候才更新又太晚很多東西要改，而且又不能採用原本的方法(一個像素一個像素處理)，時間複雜度太高了，最後只好配合不完整的blend()勉強透過調整顏色透明度來達到想要的效果。

(六)、圓形視野

原本想在掃描建築物陰影時同時偵測其他玩家和獵人的位置，結果沒想到切割成360度掃描還是不夠細緻，半徑較遠時會出現沒有掃描到的座標，若是獵人剛好在

這個座標，那就不會處理到。但若切割成720個0.5度，運行又太慢，最後只好把建築物的陰影跟獵人、玩家的顯示分開處理，雖然code比起來較冗長，但時間複雜度反而會較低。

六、分工

游庭維	關力豪	黃廷耀	江秉翰
* 圖片及地圖處理 * JSON初期處理 * 遊戲畫面切換 * 解任務模式 * 任務題目發想	* Server端處理 * Client端處理 * JSON後期處理 * 任務題目發想	* 畫面顯示 * 玩家視野設計 * 玩家移動設計 * 將任務繪至地圖	* Hunter AI設計 * 音效處理 * 上台報告

七、參考資料

(一)、Stack Overflow
(二)、Minim
(三)、Processing Reference