Project2_Group16

▼ **Website View :**

https://jtsai1120.notion.site/Project2_Group16-01fe337c05b7472e8e8700a485b420ca?pvs=4

蔡承希、蔡辰鑫、黃士洵

關於這份報告

此次 Project2 相對 Project1 而言，我們已經非常熟絡 Qt 的各種函數概念與用法。此外，我們也已在 Project1 中已描述許多運用 Qt 函數的例子。因此此次 Project2 的報告，我們會將重點放在**如何實現特定功能的想法**，而非聚焦在類別 (*自定義類別會額外說明*)、函數的命名與使用。

特定功能的實現與想法

以下文中「**符石**」、「**珠**」皆代表「**Runestone**」，為文章流暢性，我們會將其混用。

我們在此次製作中有自訂義 `int game_status` 以紀錄遊戲狀態：
-2 ⇒ 顯示技能資料。朝面板按下左鍵就回到 1
-1 ⇒ 開啟設定畫面。若選擇投降則直接跳到狀態 5；按下回去就回到 1
0 ⇒ 角色選擇戰鬥準備階段，按下開始按鈕則會進入狀態 1
1 ⇒ 等待轉珠、轉珠、消珠，進入狀態 2。期間按下設置按鈕，會跳至狀態 -1；朝隊員按下右鍵，會進入狀態 -2。
2 ⇒ 結算倍率、傷害，並輪流攻擊。輪流完後，進入狀態 2
3 ⇒ 簡單重置狀態，準備下一回合；或是進入下一層 wave。回到狀態1
5 ⇒ 遊戲結束狀態。重新開始按鈕按下後回到 1
-

遊戲剛開始時，會從 1 開始。

▼ 轉珠

1. **如何知道選取符石了？**

若目前遊戲狀態 (`int game_status`) 是在我方回合，並且沒有被限制為無法轉珠的狀態 (`bool can_move_runestone`)，則當 Mainwindow 接收到滑鼠點擊事件時，若滑鼠 y 座標 ≥ 510，就代表滑鼠在轉珠區域選擇了 30 顆符石中的其中一顆。

> 💡 y = 510 是轉珠區開始的 y 座標

此時，為了表示已選擇符石，我們將 `bool runestone_selected` 設為 1

2. **如何知道選到哪顆符石？**

   因為符石長、寬皆為 90 pixels，因此只要將滑鼠在畫面中的 x 座標整除 90、y 座標減去 510 後再除與 90，就可以得到選中的符石所在的列與行。此時，我們將該列與行紀錄起來，並呼叫被選中的該顆符石的成員函數以達成讓選中符石黏在鼠標上的效果。

```
selected_runestone = make_pair((event->y()-510)/90, event->x()/90);
runestones[selected_runestone.first][selected_runestone.second]
        ->stick_cursor(event->x(), event->y());
```

3. **如何讓選中的符石黏在鼠標上？**

   通常符石的位置是在 90 pixels 的倍數位上，因此只要被選種的符石在被呼叫黏在鼠標的成員函數時，使其 x、y 座標跟隨滑鼠座標 (實際上為模仿原版TOS之呈現，符石會黏在鼠標左上約 10 x 10 pixels 之處) 。

   每當 Mainwindow 收到滑鼠移動事件，都會呼叫一次目前選擇的符石的黏滑鼠成員函數使其黏在滑鼠上。

4. **已知選擇符石後，如何得知開始轉珠？**

   若目前遊戲狀態 (`int game_status`) 是在我方回合，則當 Mainwindow 接收到滑鼠移動事件時，若 `bool runestone_selected` 為 1 ，即代表開始轉珠。

5. **開始轉珠後，如何讓兩顆符石交換位置？**

   由於轉珠過程 Mainwindow 會持續接收到滑鼠移動事件，因此我們在每次滑鼠移動事件函數被觸發時，重複計算 `bool runestone_swap` ：

```
runestone_swap = ((event->y()-510)/90!=selected_runestone.first
        || event->x()/90!=selected_runestone.second);
```

   也就是在計算目前滑鼠 x、y 座標經轉換成行與列後是否完全與選擇的符石相同。若完全相同，則 `bool runestone_swap` 為 0；反之，滑鼠所在行或列任一與選擇的符石不同，則 `bool runestone_swap` 為 1。

   當 `bool runestone_swap` 為 1 時，代表滑鼠 swap 到另一顆符石了，因此進行以下動作：

   - 播放「交換符石」的音效
   - 偵測是否碰到風化珠、燃燒珠等，並進行相應動作 (e.g. 扣血、停止轉珠等等)，在下方的章節會進行詳細說明
   - 將原本選擇的符石歸位 (原本黏在滑鼠上，歸位後會移回原本的行列上)
   - 將選擇的符石改為目前滑鼠所在區域的符石

- 若還未開始計時，又 `bool runestone_selected` 為 1，就代表開始轉珠了，因此進行轉珠計時器的歸零後並開始計時。

6. **計時 10 秒是如何實現的？**

   我們利用 `QTimer` 創建了一個每 *1ms* 觸發一次的計時器，並將其的觸發槽設定成一個簡單的 lambda 函數：每當被觸發，就將 `int ms_elapsed` 加一，再呼叫 `update()` （即 `Mainwindow::paintEvent()` 函數）

   > 💡 雖然 `Mainwindow::paintEvent()` 函數到最後並沒有發揮它該有的渲染功能，但我們乾脆的讓這個函數做為倒數時間條更新的引用母函數（詳細在下方）。

   在 `Mainwindow::paintEvent()` 函數中，我們偵測 int ms_elapsed 是否超過 10 * 1000（即 10 秒），若有則達成第二種結束轉珠的方式（下方詳述）。

7. **轉珠時，是否會有滑鼠滑出轉珠區的情形？**

   是的，並且我們發現一旦滑出轉珠區，程式就會立即崩潰，因此有在邊界上做偵測防治。至於如何實現，因為如果單純使用滑鼠事件的 x、y 座標，此座標是相對程式視窗的相對座標，我們猜測當滑鼠滑超出程式視窗時，滑鼠事件的座標可能會無法順利傳出，因此導致崩潰。

   因此，我們改成：

   1. 使用滑鼠事件的 **globalX**、**globalY** 座標，此座標即為相對螢幕畫面之絕對座標，使程式在滑鼠移出視窗時不會崩潰。
   2. 偵測當滑鼠距轉珠區上下左右四個邊小於固定值的話，就將滑鼠利用 `QCursor::setPos` 函數將滑鼠強制移回視窗畫面內的位置，使其不會超出視窗畫面。

8. **何時結束轉珠？**

   結束轉珠目前有兩種方式：

   1. 滑鼠主動放開（由 Mainwindow 的滑鼠放開事件偵測得知）
   2. 10 秒時間到（如 6. 所述，由 `Mainwindow::paintEvent()` 偵測得知）

   不論是哪種結束方式，接著都會進行一系列的設定動作：

   - 將計時器暫停
   - 設定轉珠狀態為停止轉珠
   - 倒數時間條轉回血量條
   - 目前選擇的符石回到行列原位

   最後，再呼叫：消珠、落珠階段主函數 `Mainwindow::combo_count_and_drop()`

▼ 消珠 & 落珠

1. **消珠 & 落珠的階段是如何運作的？何時消珠、何時落珠？**

   控制目前消珠 & 落珠狀態的主函數為 `Mainwindow::combo_count_and_drop()`，由結束轉珠的兩種方式其中一種呼叫，並且會標記為首次呼叫。

   此函數被呼叫時，流程大致如下：

   - 若為首次被呼叫，代表剛轉珠結束，因此會重置 Combo 數，並直接進到 Combo 計算的函數 `Mainwindow::combo_count()`。

- 若非首次被呼叫，代表有天降 (也有可能沒有)，因此要先清除前一次消珠時生成的背景發光特效 `light_halo_vfxs` ，再進入 Combo 計算的函數 `Mainwindow::combo_count()` 。

至於為何是大致，是因為此主函數還有包含關於遊戲狀態的狀態機控制，詳細會寫在下方遊戲流程中。

2. **如何偵測 3 顆以上相連符石、計算 Combo 數？**

   在函數：`Mainwindow::combo_count()` 中，

   我們創建了一個專門計算符石相連的物件名為 `combo_counter` 。我們首先將當下的轉珠盤面 `runestones` 傳入 `Combo_counter::count()`，等到計算完後再將結果作為回傳值傳出。

   我們的相連符石的計算不只包含 3、4、5 相連這三種，而是 3 ≤ 符合規定的相連數目 ≤ 30 都可以被計算出來，其核心思路如下：

   1. 先尋找水平、垂直兩個方向的 3 顆同色直線相連的組合
   2. 從第一個發現的組合開始，一一與其他組合比對，只要兩個組合發生：
      a. 任一位置顏色相同且重疊
      b. 組合 A 的某位置與組合 B 的某位置相鄰，且顏色皆相同

      即代表此兩個組合可以融合 (merge)，因此將其融合後，刪除前述兩個舊的小組合，再把這個新的大組合放入所有組合當中，重複再去跟其他組合作比對
   3. 直到所有組合都已經無法跟任一其他組合融合，即計算完畢。回傳所有組合。

   此時 combo 數即是該回傳的組合的 vector size；`Mainwindow::combo_count()` 在收到這些答案後，就會接著呼叫 `Mainwindow::combo_eliminate()` 進行消珠。

3. **如何進行消珠？如何有時間差的消不同組珠？**

   我們創建了一個 QTimer 的計時器，將觸發時間設為 390 ms，即可有時間差的進行消珠。我們建立了一個計算當前消到哪組的變數 `int cur_pair_num`，並在每次計時器被觸發時選擇 `Mainwindow::combo_count()` 回傳答案的第 `int cur_pair_num` 組珠進行以下動作：

   - Combo 數 + 1
   - 將要被消除的珠的顏色改為 empty
   - 播放 Combo 數相對的消珠音效 (隨 Combo 數越高，聲音頻率越高)
   - 新增背景發光特效 `light_halo_vfxs`
   - 檢查是否消心珠、進行回血
   - `int cur_pair_num` + 1

   一直持續到 `int cur_pair_num` 等於 `Mainwindow::combo_count()` 回傳答案的 vector size，即代表消完當前盤面所有珠。此時，我們就暫停計時器，並呼叫落珠函數 `Mainwindow::drop_detect()`。

4. **落珠的動畫是如何實現的？如何讓落珠顏色隨機產生？**

   我們首先去找轉珠盤 6 個行 (Column) 中每個行最下方的 empty 珠 (即被消掉了)，因為落珠最後要落到該位置。整行都沒有 empty 珠，則最下方的 empty 珠即為 -1 (不在轉珠盤面上)。

   再來，我們建立了一個用來展示落珠動畫的暫時盤面，再去數每行最下方 empty 珠開始往上檢查：

   - 若檢查到的珠有色，則生成一個暫時珠到相同行列位置，用來展示該珠的落珠動畫
   - 若檢查到的珠無色 (empty)，則利用函數 `Mainwindow::random_runestone_color()` 生成一個隨機顏色，並依此生成一個暫時珠到轉珠盤面同行上方開始堆疊落珠。

> 💡 `Mainwindow::random_runestone_color()` 的隨機方法我們並非採用 `srand(time(NULL))`，而是使用 C++ <random> 函數庫的 `randome_device` 物件生成隨機種子，再將種子傳入梅森旋轉數生成器 `mt19937` 物件，最後再將機率分布設為常態分佈 `uniform_int_distribution<>` 物件，以達到隨機的效果。

當生成完所有落珠動畫的暫時珠後，我們也用堆疊的方式去計算每個暫時珠最終會落到哪個位置，再去呼叫每個暫時珠的落珠動畫成員函數。

> 💡 落珠動畫採用 QT animation，並且有設置平滑曲線讓落珠過程不會很突兀

5. **落珠完若又有相連組珠，要如何再次回到消珠階段？**

   我們根據落珠最大的動畫移動距離計算出需要多少落珠時間，因此建立一個 QTimer 的 Single Shot 等到所有落珠都落完之後再觸發，內容如下：

   - 將暫時珠物件刪除
   - 將正式盤面改為暫時珠的顏色
   - 呼叫消珠階段主函數 `Mainwindow::combo_count_and_drop()` ，並將是否為首次呼叫標記為 false ⇒ 可再次回到消珠階段檢查是否有天降

▼ 血量條 / 倒數時間條

我們將血量條、倒數時間條設在同一個物件裡面，其中改變長度的方式是計算血量、倒數時間的百分比乘上總寬度設為新的寬度。

▼ 場景技能

> 💡 `runestone` 的底下有 `int status` 用以紀錄符石狀態
> 0 ⇒ 正常
> 1 ⇒ 燃燒
> 2 ⇒ 風化

燃燒

`vector<pair<int,int>> burning` 紀錄燃燒位置， `bool burn_road` 紀錄此回合是否有燃燒路徑。

正常燃燒：

- 每回合結束時會檢查配戴該技能的敵人還活著，並啟動或解除
- 燃燒的位置會推進 `burning` 中，確保每次啟動時都會將指定位置的 `runestone status` 轉為 1
- 轉珠放手、combo結算完成，都將移除燃燒效果，並重新填充 (避免跟燃燒軌跡打架)

燃燒軌跡：

- 每回合結束時會檢查配戴該技能的敵人還活著，並啟動或解除

傷害計算：

- 在移動期間，檢查交換的 `runestone status` 是否為 1。若是，會將此 `status` 傳遞 (做出動畫)，並將累積的傷害算在 `int harm` 底下，作為傷害值的顯示。

- 若 `bool burn_road` = `true`，則傳遞時，並不會將原本位置的符石 `status` 還原，就能實現燃燒路徑的效果。

風化

風化：

- 每回合結束時會檢查配戴該技能的敵人還活著，並啟動或解除
- 隨機選取位置時會確保指定位置是乾淨 (`status` 為 0) 的符石

停止轉珠：

- 在移動期間，檢查交換的 `runestone status` 是否為 2。若是，則交換位置時，即強制跳出移動狀態，並扣除 `hp`。

combo 盾

攻擊無效：

- 在角色攻擊時，檢查此關卡是否配有此技能
- 若有，則檢查 `combo` 是否達要求
- 若 `combo` 沒有達要求，攻擊傷害 = 0，且敵人 atk 乘上差異數量

先攻

在回合結束階段中，若進入指定層，指定敵人 CD 設定為 0 即可實現。

▼ 角色

- 15個角色+空欄外觀，用 vector 對應 ID 紀錄
- 分別有 vector 對應 ID 紀錄各角色 atk、hp、heal、CD 等基礎值
- 顯示攻擊數值用的 `QLabel`

> 💡 雖然角色 ID 在設計的時候，是由 1 開始數，但為了配合程式由 0 開始的特性，語法中使用的 ID 會差 1。

換角色

對應函式： `void Charac_slot:: change_charac (int is_lead, bool is_basic)`

- 被呼叫時，會將當前 ID + 1，數到最後再從頭
- 對應 ID 讀出 外觀、atk、hp、heal、CD vector紀錄的數值，並將其顯示
- is_basic 用來判斷是否為 basic mode，若是則顯示的數值為固定 (hp = 2000、atk = 1、heal = 5)

由於只有在選擇角色的階段才會呼叫此函式，因此開始遊戲之後，這些資料都不會再改變。

攻擊

攻擊流程：

- 使用vector survive存取仍然存活的敵人，在攻擊階段透過隨機抽取survive中的敵人選擇其為攻擊對象

- 透過屬性克制計算出最後的傷害值
- 根據單體傷害或群體傷害，將傷害顯現，並將被攻擊的敵人扣除相對應的血量
- 判斷敵人是否死亡

傷害/回復計算：

皆為四捨五入值。

- `basic` (mission 1)
  - 攻擊力 = combo x 消除對應屬性顆數 (x 屬性克制)
  - 回復力 = 5 x combo x 消除對應屬性顆數
- `!basic` (mission 2)
  - 攻擊力 = 攻擊力基值 x combo x 消除對應屬性顆數 x 0.5 × 0.001 (x 屬性克制) (x 技能/隊伍技能額外倍率)
  - 回復力 = 回復力基值 x combo x 消除心顆數 x 0.5

屬性克制：

- 透過存取在charac_slot中每個角色的屬性，比較欲攻擊敵人的屬性，將原本計算出的傷害乘上特定倍率，即為最終的傷害

傷害值顯示：

- 在charac_slot中damage_text被用作顯示該角色的攻擊數值，在攻擊階段會顯示在被攻擊敵人的周圍，並在0.5秒後消失
- 若情況為群體攻擊，則不會以敵人周遭的白色數值作為攻擊數值的顯現，會將字體放大，並改為紅色字體，顯示在敵人中間正上方，文字一樣會在0.5秒後消失

技能 (效果皆在 mission 2 中才會發動)

對應函式：主要在`void MainWindow::` ***`mousePressEvent`*** `(QMouseEvent *event)` 底下

- 在`game_status == 1`時，若點擊角色框位置，將會觸發檢查`CD`是否為 0，若是，則發動相應技能，並呼叫`void Charac_slot::` **`CD_reset`** `()` 把`CD`重置，此函式還會同時把開啟技能的角色 ID 寫進`skill_power`來做進一步檢驗及發動其他效果；若非，顯示剩餘回合。
- 隊長技能只判斷第一個角色，不需要雙隊長。

不同效果的技能大致的實現方式：

▼ **轉換、還原版面：**

利用`void Runestone::` **`change_color`** `(QString clr, int get_status)` 更改符石的屬性/狀態。

▼ **倍攻、屬性克制：**

在`charac_slot`和`mainwindow`底下都有變數紀錄開啟的技能，結算時會讀取並判斷是否需要倍率。

▼ **吸收傷害：**

`mainwindow`底下有變數紀錄開啟的技能，結算攻擊時會讀取並判斷是否需要減傷。

▼ **追擊：**

`charac_slot`底下變數`hit_more`紀錄追擊次數，結算攻擊時會讀取並判斷要追擊幾次。

▼ **燃燒回復：**

`mainwindow`底下有變數紀錄開啟的技能，觸碰燃燒時會判斷將`harm`量值倒轉。

以下是所有角色的技能：

▼ **ID 1 ~ 5，神魔內建主角**

主動技能：

- CD 5
- 還原 (消除風化效果) 並轉換固定 10combo 版面
- 自身攻擊克制任何屬性

隊長技能：

- 全隊 atk 2倍
- 進場 CD 減至 0

▼ **ID 6/ID 13**

- CD 6
- 一回合內atk 3倍，消除燃燒軌跡效果
- 轉換固定版面

隊長技能：

- 暗水木互相兼具屬性效果/水火光暗互相兼具屬性效果

▼ **ID 7/ID 11**

- CD 4
- 一回合內，根據消除的心符石數量向上提升 atk
- 轉換固定版面

隊長技能：

- 心兼具所有屬性效果
- 回合結束時，將最左排轉換成心符石

▼ **ID 8/ID 14**

- CD 6
- 可以任意移動符石而不會發動消除 (排珠)
- 左右兩旁隊員CD -1
- 自身攻擊克制任何屬性

隊長技能：

- 每次消除8粒相連的符石時，延遲 CD 最短的敵人 1回合，combo 數 +8

▼ **ID 9/ID 15**

- CD 7
- combo 數 2倍，自身額外追擊兩次自身攻擊
- 此效果持續到自身沒有發動攻擊

隊長技能：

- 自身進場 CD 減至 0
- 永久吸收右方成員的攻擊，並加入自身攻擊

▼ **ID 10/ID 12**

- CD 8
- 一回合內全隊atk 2倍
- 燃燒傷害轉回血
- 敵方攻擊無效

隊長技能：

- 全隊 CD 永久縮減 1回合
- 所有隊員攻擊力，跟隨全隊最高傷害 (技能倍率都可以吃到，但屬性克制會另外計算)

▼ 敵人

基礎設置

1. **初始化設定**

   設置了一個指向Enemy的vector，大小為3，用來改變場上的敵人出現情況。根據當下level的不同，會出現與之對應的敵人排列。

   - level 為 1
     - 出現第一關的三個敵人，由左至右排列分別代表vector的第0、1、2項
     - 根據已經建立好的hp_list以及attack_list分配該敵人的攻擊力以及生命值
   - level 為 2
     - 出現第二關的三個敵人，由左至右排列分別代表vector的第0、1、2項
     - 根據已經建立好的hp_list以及attack_list分配該敵人的攻擊力以及生命值
   - level 為 3
     - 由於第三關只有一個敵人，保留vector的第一項並將圖片設製成該敵人的模樣，並將其他兩項移出畫面外
     - 攻擊力以及血量一樣照著建立好的list分配

2. **敵人死亡**

   - 當敵人被攻擊後血量小於等於0之後，則將enemy中的dead設定為true，並將該角色移出畫面
   - 在攻擊階段，隨機攻擊的實現是透過隨機抽取存活著的敵人進行攻擊，因此當敵人被判定為死亡後，須將其從倖存的敵人清單中移除
   - 敵人死亡後，需一併將其對應的血條移出畫面之外

3. **level 更新**

   - 當在場上所有敵人的`bool dead`都為true時，則進行下一階段的關卡
   - 將level變數加一，透過呼叫`Enemy::show()`將下一階段的enemy圖片呈現，併出現在相對應的位置，同時使用`Enemy_hp::reset`將血條長度復原，歸位
   - 將所有enemy相關數值都重新設定，包括血量、攻擊力、CD、屬性

CD

1. 建立一個 cd_list，透過當下level以及enemy的索引值改變初始值
2. 透過`cd_reset`這個function賦值
3. 在每次玩家攻擊結束之後減一，cd變為0時攻擊，一攻擊完即重新恢復成原本的值
4. cd 文字的顯示
    - 和enemy的顯示方式雷同，一樣建立三個cd_text的label，將值設定成`"CD"+該enemy當前的cd值`，在enemy cd減少時會一併更新。

血量條

1. 根據敵人及角色攻擊時的屬性不同，會產生相互克制的結果，因此建立一個vector紀錄敵人的屬性
2. 依照屬性可以做為血量調顏色的改變依據(土屬性為黃色，火屬性為紅色...)
3. 顏色更新
    - 在敵人重新更新時，透過回傳進來的level以及enemy以`setRgb()`改變血量條的顏色
4. 長度更新
    - 透過回傳進來的傷害，除上敵人原本的血量值得到比例，再以總長度乘以比例得到承受傷害過後的血量條長度，透過`setFixedWidth()`改變長度

▼ 攻擊特效

一般攻擊

- 一般攻擊使用的是白色圓點作為子彈從角色欄為起點射向欲攻擊的敵人，初始化階段透過設定一個指向Bullet的vector進行操作，此vector大小為6，亦即將六個角色各分配一個子彈
- 玩家攻擊前，先將所有的子彈隱藏，再將所有的子彈歸位，確保所有的子彈起點皆為各自對應的角色欄
- 當角色攻擊對象已確定時，透過獲取該敵人的位置，先將子彈顯示，再透過`Bullet::shoot`函式，以傳入終點的方式，使用`QPropertyAnimation`實現子彈位移的過程

群體攻擊

- 當角色方面的`all_atk`為true，代表將要進行全體攻擊，此時則不以子彈作為攻擊特效，改以火焰燃燒作為攻擊特效
- 火焰燃燒特效是以將火焰的圖檔放置於敵人周遭實現，並將該圖檔的透明程度設定為0.5
- 若為群體攻擊，則不需產生動畫，可在確認為群體攻擊之後，將火焰圖檔顯示，並在玩家攻擊階段結束，進入敵人攻擊階段前再隱藏即可實現該功能