程式設計

期末專案書面報告

Group N

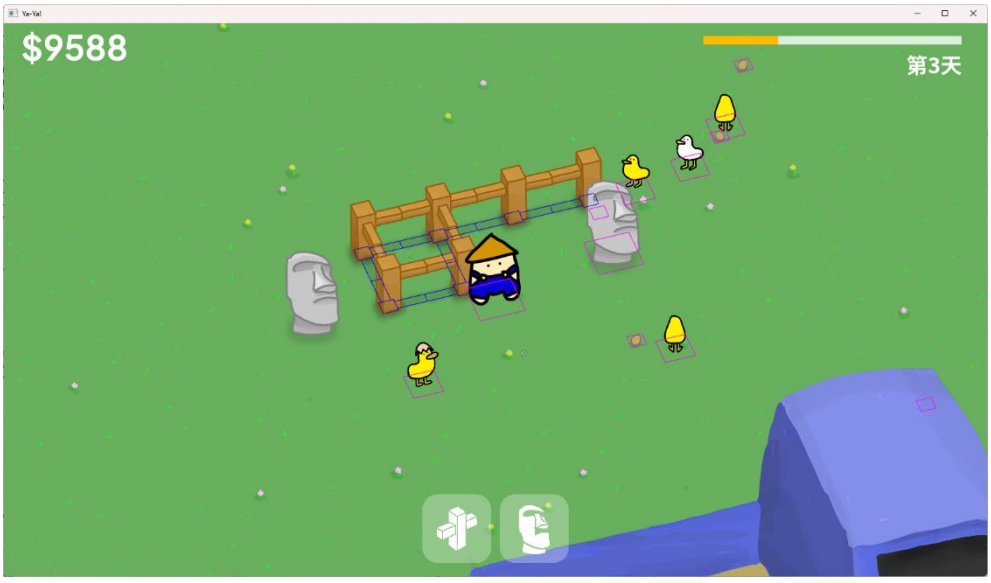
B11705003 蔡炘誼

B11705030 嚴邦華

B11705034 蔡逸芃

B11705049 林宏澤

B11705060 盧沛宏



一、遊戲主題

我們以其中一名組員最喜愛的動物（鴨子）進行發想，製作了模擬「鴨子農場」的療癒遊戲。玩家將扮演農夫，以滑鼠點擊方式拾起鴨蛋、將其收納入蛋盒中，並把欲銷售的鴨子和蛋盒放入卡車。遊戲中的一天結束後，卡車會將所載運的物品銷售完畢。銷貨收入可以用來購買如摩艾像、柵欄等建材，以拖曳方式決定建築地點、點綴玩家的農場。玩家可以隨時關閉遊戲，並在主畫面以「回奶奶家農場」選項恢復進度，或選擇「捨棄奶奶的農場」進行新遊戲。

二、分工

※ **盧沛宏：**遊戲系統設計與實作、進度及分工協調

※ **蔡逸芃：**存檔實作、書面報告編撰

※ **蔡炘誼：**遊戲主題發想、角色繪製、美術設計

※ **顏邦華：**音效及角色設計、書面報告編撰

※ **林宏澤：**書面報告編撰

三、系統設計

1. 系統架構簡介

遊戲的主體架構可以分成若干部分：遊戲角色、遊戲地圖、音效、物件與使用者介面渲染、遊戲控制、以及序列化存檔。可以將標頭檔分類為下表：

遊戲角色	entity, player, egg, eggcarton, truck
遊戲地圖	map
渲染	model, camera, graphics, ui
遊戲控制	action, timer, controls, neighborsfinder, game, gamecontroller
存檔	saveUtilities, serialization, serializationExtended
其他	audio, debugger, events, localization, names, utilities

以下，我們將逐一介紹重要部分的系統設計。

2. 遊戲角色

由於各角色共同擁有的成員及成員函式皆相似，我們實作了 base class Entity。以下為 Entity 之重要成員及成員函式，以及其功用：

id	不同實體的名稱字串，用以在 std::map<std::string, Entity>中做為鍵值，以及作為讀檔時物件類型的參考。鴨子會取名叫 duck$name，以此類推。
type	每個實體的類型（列舉）。在 Entity 類別中，每個物件都是 ENTITY，但在 Duck 等 Entity 的 derived class 中，類型會是 DUCK 等。作為執行動作和存檔的參考。
inventory	Entity 指標的陣列，表示該實體所擁有的其他實體。例如鴨子在下蛋前，會先將「蛋」物件的指標存放在鴨子的 inventory 中。
ownedBy	Entity 指標。表至該實體被其他實體擁有。例如鴨子在下蛋前，「蛋」的 ownedBy 會是鴨子的指標。
entityTimer	Timer 物件。用來計算經過多少時間、要移動多少，避免不同機器幀率不同造成移動速度不一致。
model	Graphics::Quad 的陣列，儲存該實體的模型。
collideBox	collideBox 物件，處理實體的碰撞。
neighborsFindMyTile	整數數對，在處理碰撞時儲存實體所在的「區塊」。
position	coord 物件（基本上是浮點數數對），儲存位置。
heading	實體面向的方向。
runAction( )	虛擬函式，引數為 Action 物件的參考，以及一 Action 陣列，在遊戲控制會多加討論。子類別會實作該實體應該做的動作。
pushQuads( )	載入實體的 model 並透過 Graphics 渲染模型。
update( )	用來更新實體的狀態。若子類別有自定義更新的需要，可以在虛擬函式 customUpdate( )實作。
setInventoryProps( )	更新該實體每個 inventory 中物件的狀態。

而在 Entity 的子類別：Duck, Egg, EggCarton, 以及 Truck 中，有時會新增一些成員，如 growStage, genderIsMale, fertilized 等等。

3. 遊戲地圖

遊戲地圖可以分為三個類別：Tile, Chunk, 以及 Map。我們將地圖分成許多小格（遊玩時可以選取的地圖格皆為 Tile 的 instance），每個 Tile 皆有自己的座標位置、亂樹種子（讓草地有變化性）、及 CollideBox 物件（如摩艾像等地圖物件，需要設定碰撞）。我們以 TileType 列舉來分辨 Tile 是哪一種地圖物件（草地 GRASS，摩艾像 MOAI 等等），同樣以 pushQuads 函式來載入地圖物件的 model 並渲染，並以 setTileType 函式，讓玩家花錢建築時，能更新地圖物件的類型，並以 neighborUpdate 來檢查該格周圍有沒有連帶要更新的地圖格。

此外，我們將地圖分成 16x16 的 Chunk，每個 Chunk 皆有自己的 Tile 陣列。Map 中再利用 std::map<整數數對, Chunk> 來儲存整個遊戲地圖。使用 getTile( )函式就能存取地圖中特定 Chunk 的特定 Tile&，並呼叫 setTileType 等函式。將地圖小格分成 Chunk 來儲存，主要是考量存檔的方便性，先儲存一個區塊內的小格、再一起儲存 16 個區塊，比起逐格儲存及讀檔，來得有系統許多。

4. 遊戲控制

首先，我們以表格簡介 Action, NeiborsFinder, Game, 以及 GameController 的重要成員及函式。

Action::time	Timer 物件，決定 Action 哪個時間點後應該被執行。
Action::entity	Entity 指標，決定 Action 對哪一個實體執行。
Action::command	列舉各種行動，如 ON_CREATION(Entity 被創建時會被執行的行動）、DUCK_LOOP_WANDER 等等。
Action::argBool 等	bool（或其他類別）陣列。在執行該行動時可能用到的其他參數。如執行 DUCK_LOOP_LAY_EGGS 時，需要將指定鴨子的指標為 argEntity，以便把蛋的 ownedBy 設為鴨子的指標。
Action::operator<( )	用以比較和其他 Action 物件的執行先後順序，以及供 std::priority_queue 使用。使用與 time 的時間差做比較。

NeighborsFinder:: update( )	我們發現以實體數量平方暴力處理碰撞，明顯影響遊戲幀率，因此使用 NeighborsFinder 的 update 函式維護每個實體的 neighborsFinderMyTile。如此一來，處理碰撞就只要考慮同區塊內的實體，減少延遲。
Game::重要成員	每個 Game 皆包含一個 GameController, NeighborsFinder, Map, 以及 UserInterface 的實體。
Game::actionList	Action 的優先權佇列，儲存所有尚待執行的 Action 物件。
Game::entities	std::map<std::string, Entity>。儲存所有遊戲中的實體，方便進行渲染或動作。注意到實作中其實是將 Entity 的子類別的指標存入，是一種多型的應用。
Game::update( )	對所有實體進行 update( ) 以及 setInventoryProps( )。
Game::processCollisions ( )	處理所有實體與地圖物件的碰撞。此處會呼叫 NeighborsFinder 來「逐區塊」處理。
Game::runActions( )	將現在時間與 actionList 中最頂端的 Action 進行比較，並對目標實體呼叫 runAction( )。注意到指標是指向 Entity 的子類別，不會呼叫到 Entity 本身的 runAction。
Game::pushAction( )	新增待執行的行動。
Game::insertEntity( )	新增實體至 entities，並以該實體為目標實體插入 ON_CREATION 行動。
Game::render( ) Game::renderMap( )	呼叫所有實體以及地圖格的 pushQuads( )，在螢幕上的相對位置渲染模型。
Game::save( ) Game::load( )	讀檔以及存檔。
GameController:: update( )	根據 Timer 的時間更新遊戲狀態，如進入一天開始的畫面、一天結束的畫面、或遊玩畫面等。

儲存這個遊戲所有物件與物件指標的類別，就是 Game 這個大類別。在 yaya.cpp（主程式）中，創建 SFML 的視窗、載入各種字體與圖片後，就會創建 Game 的 instance，進入遊戲的 main loop。根據我們類別的設計與分工，我們的主程式結構可以簡單表示如下：

```
Game game  // Game instance
while window is open:
    handle SFML events  // player controls, whether to load game...
    set Camera

    game.update()
    game.render()  // render entities and game map
    game.ui.renderUI()
    game.ui.renderOverlay()

    display window
game.save()
```

其中 update() 的架構如下：

```
function game.update():
    move player by its velocity
    neighborsFinder.update()
    processCollisions()

    for all entities e in entites map:
        e.update()
    runActions()
    for all entities e in entites map:
        e.setInventoryProps()
```

我們的系統設計以 update 包辦遊戲內幾乎所有事項，接著以 GameController 配合 Timer 來創造遊戲從開始畫面到日循環的流程以及控制 UI 的顯示。其中，我們只要自定義各個 Entity 子類別的 runAction()，他們就能透過 update 自行運作。如鴨子散步（DUCK_LOOP_WANDER），我們會在 Duck::runAction() 處理 ON_CREATION 的部分，對那隻鴨子插入 DUCK_LOOP_WANDER 的 Action，之後，每次那隻鴨子進行 DUCK_LOOP_WANDER，該行動會再次在未來時間點插入 DUCK_LOOP_WANDER。其他週期性行動，諸如生蛋、交配等等，皆為同樣的機制。至於鴨子從蛋中孵出來等等牽涉到 inventory 的行動，也有 ENTITY_OWN_BY 等等 Action 可以幫我們處理。在 main loop 中，我們只需要在 handle events 的地方處理玩家的行動（controls.h），無需處理實體。

5. 存檔

存檔與讀檔，不外乎就是將物件與變數轉換成字串，再將其轉換回變數或字串。我們會在欲存檔的類別中加入 properties 物件，其中包含「指向想要儲存的成員的指標」以及「存檔 ID」。接著，參考網路上的資料，我們會動態展開迴圈，對 properties 中不同類別的不同物件「遞迴地」做序列化處理。

序列化與「反」序列化，是使用 template function 的方式實作，分為 C++的 primitive-type，std 中的資料型態（如 vector, map 等），以及自定義型態。如果遇到物件，就照指定規則轉成字串，或從字串轉換回物件。若遇到指標，就要先行檢查指標指向的物件是否被創建過了。序列化時，我們會儲存指標指向物件的記憶體位置，反序列化時，只要以一個 std::map 檢查該「舊」記憶體位置的物件是否被創建過即可。

儲存檔案的格式，有點仿照 html，以<tag></tag>方式區分各成員，相較 iostream 的空格分隔不方便許多，也容易有巢狀標籤問題。未來若有機會重訪存檔的時做，不妨可以好好利用 iostream。

四、心得感想

■ **盧沛宏：**

大概跟預想的差不多，最後還是很難讓大家都寫到均勻的量。最好的分工體驗絕對是蔡逸芃了，他是寫 serialization 的部分，也就是存讀檔。我們知道這個部分不會是現場 demo 的重點，就算 demo 了應該也不會「感覺多了不起」。當天的報告現場的感覺是... 好像其實大家還是感覺得出來這不是很簡單的事情，所以我想我們還是成功了吧，技術展示的部分。由於 serialization 是分開的檔案，class 專屬的程式碼也分得很開，所以我們的分工幾乎完全沒有問題。而且我遊戲裡面加入什麼，他就序列化什麼，因為他把他的系統寫得很 general，所以地圖的序列化在前一晚才寫完，最後也沒有任何問題。

所以遊戲到最後就從無到有，大致上花了我兩個禮拜吧，時間實在是有點趕，但是一大原因也是因為我們為了"不存在的"效能問題，寫到一半把所有東西都從字串存取碼改成指標。然後我自己也花太多時間在美術跟一些不太重要的功能，導致最後介面巨醜、遊戲內容欠缺。不過我其實也不後悔，因為我腦中的想像大致上都完成了，比方說蛋盒可以分格子個別操作，或

是有商店、可以放置物品在地圖上也是差點因為時間的關係被砍掉，但是最後一晚還是趕出來的東西（然後我們超認真的蔡逸芃接續我，把序列化熬夜寫完，寫到隔天五點的樣子吧）。

原本以為我們的美術總監蔡炘誼會弄出像貓咪那組品質的東西，但因為她最後只用筆電觸控板畫，所以好像也還好。不過那個用跳的來解決腳不會動的方法，是我在做這個專案的時候沒想到的。還有嚴邦華也弄了音樂與音效的部分，但是因為效果不是很好，當場沒有演示。

總之，這個專案還是花了有點太多時間，而且一開始的各種想像也與現實落差有點大。不過這樣的完成度從我兩個禮拜前開始寫就有預期了。我覺得我算滿認真的吧？雖然還是不會報告，又要被大家嘖了。

■ **蔡逸芃：**

這次製作程式設計期末專案，很開心能做出這麼可愛的小遊戲，也第二次見識到 Jerry（盧沛宏）的厲害了 xD，但心中卻感觸良多。

在微積分和其他科目岌岌可危的期末，寫期末專案固然開心，可是時不時給我帶來不踏實感。明明其他科目比較不擅長，或許花更多心思在其他科目，會有比較高的邊際效益，那為何要費盡心思弄好期末專案呢？

或許，這就是我學資訊的動機吧。做出東西，然後自嗨。

主要負責的存檔部分，實在遇到非常非常多挫折。正如老師上課提醒，template 實際操作上還蠻麻煩的，ambiguous template specialization 等問題，都要靠相對新（外星文）的語法解決（如 std::is_pointer<>, std:: remove_reference<>...），網路資料的 integer_sequence 等等更是花了好一會才搞懂。想到以前高中時學會 C++11 的 iterator 等酷東西就在那邊自嗨，現在看到 C++20 的玄學一直碰壁，真是愚蠢。

然而，這次寫專案也強迫我踏出舒適圈、多加了解 C++的模板以及新語法，相信對未來一定有所助益。感謝組員辛苦繪畫、設計其他各種功能，感謝老師提供期末專案的機會，也感謝我自己沒有被這些挫折搞瘋。

■ **蔡炘誼：**

這次期末專案我們的主題很有趣，需要結合程式、繪圖、動畫，而我是負責繪圖的部分。我認爲除了自己的圖案精細程度可再加強外，若時間允許也可以多畫一些 NPC 或是建築，豐富農場內部。這次因為時間關係以及初期人物設計的疏忽，導致所有角色都只能以跳躍的方式前

進。雖然也很可愛，但希望再有機會時我能做出更擬真精細的圖樣，改善使用者體驗。程式部分則要感謝我的組員們日以繼夜，還要教我如何使用 GitHub 並應付我上傳時造成的問題，真的辛苦了！我也會用寒假時間好好增強自己的程式能力，希望未來能在編寫程式碼的部分做出更多貢獻。

- **嚴邦華：**

  這次期末專案對我而言很有挑戰性，分工也非常的麻煩。我負責的部分主要是音效及音樂本身的製作、音效以及各種控制的程式碼。我覺得除了音效以外，還可以再增進 UI 的介面、debug 等。另外這次的專案也讓我對於 GitHub 的使用更熟練，我也會持續增進我的程式能力，期望在未來更有貢獻。

- **林宏澤：**

  非常謝謝組員們。這次專案，我學到了一些 GitHub 的東西，非常實用。特別是用 GitHub 協作可以很好地查閱和建立程式碼之間的關係。大家寫的各種程式也讓我學習到了很多。