Aquarium Farming Simulation System

設計文件

|  |  |
| --- | --- |
| 專案名稱 | 模擬養魚系統(AFSS) |
| 撰寫日期 | 2017/11/26 |
| 發展者 | 鄒承皓、朱冠穎、方子元、陳育祥、蔡承峰 |

**版次變更記錄**

|  |  |  |
| --- | --- | --- |
| 版次 | 變更項目 | 變更日期 |
| 0.1 | 初版 | 2017/11/26 |
| 1.1 | 模組結構有向圖、微調設計議題 | 2017/11/29 |
| 1.2 | 新增演算法設計、新增設計議題 | 2017/12/13 |
| 1.3 | 新增演算法設計、新增擺設設計、新增設備設計、新增設計議題 | 2017/12/19 |
| 1.4 | 類別圖微調、演算法設計新增(三)、介面設計微調、使用者畫面設計更新、新增擴充插件 | 2018/01/12 |
| 1.5 | 資料設計、類別圖設計微調 | 2018/01/13 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

目錄

[版次變更記錄 2](#_Toc466990441)

[1. 系統模型與架構(System Model/System Architecture) 4](#_Toc466990442)

[2. 介面需求與設計(Interface Requirement and Design) 5](#_Toc466990443)

[3. 流程設計(Process Design) 7](#_Toc466990444)

[4. 使用者畫面設計(User Interface Design) 9](#_Toc466990445)

[5. 資料設計(Data Design) 13](#_Toc466990446)

[6. 類別圖設計(Class Diagram) 16](#_Toc466990447)

[7. 實作技術(Implementation Languages and Platforms) 25](#_Toc466990448)

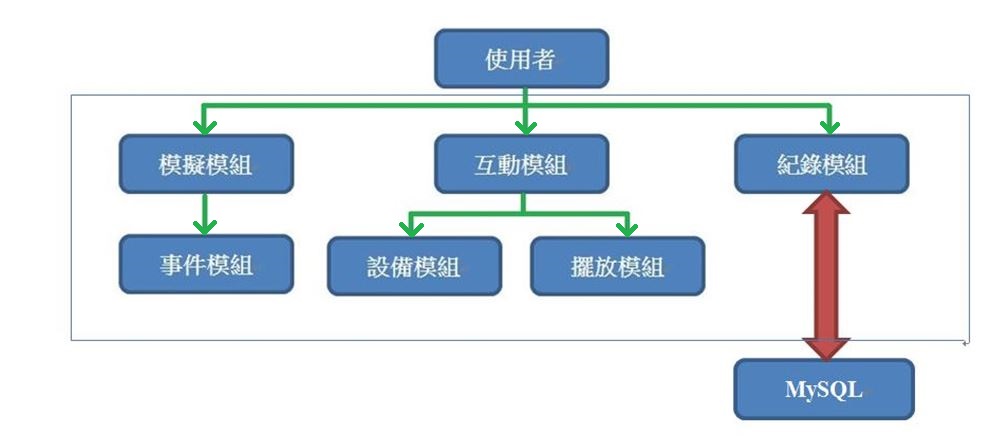
[8. 設計議題(Design Issue) 25](#_Toc466990449)

[9. 演算法設計(Algorithm Design) 3](#_Toc466990449)2

1. **系統模型與架構(System Model/System Architecture)**

本系統主要分為六個部分，分別為模擬模組、互動模組、紀錄模組、事件模組、設備模組、擺放模組。

在模擬模組，主要透過環境和事件的變化去推動整個水族箱內的變化，使用者觀看到的狀態大多都在此處；事件模組當中，因應不同時間點當下環境或是魚類狀態會產生不同影響事件，事件可能會影響魚類需要使用者做出判斷然後行動；互動模組中，主要由設備模組和擺放模組兩個模組組合而成，並且會對環境產生影響，資金的使用也會幫使用者一併紀錄。其中設備模組提供多種選擇且有不同等級並配備。擺放可以選擇不同裝飾品，自由擺放在不同位置；紀錄模組可以將時間內的資料打包儲存到後端往後做檢視，也可以調閱先前的紀錄做觀看。



1. **介面需求與設計(Interface Requirement and Design)**

|  |  |  |
| --- | --- | --- |
| **介面名稱(Interface Name)** | **介面提供者(Interface Provider)** | **介面使用者(Interface Consumer)** |
| 使用者與模擬模組的介面 | 模擬模組 | 使用者 |
| **連結方式(Connection Type)** | **輸入資料(Input Data)** | **輸出資料(Output Data)** |
| Java method call | 各項參數 | 模擬結果(畫面呈現) |
| **對應之介面需求(Interface Requirement Description)** | | |
| 使用者設定好各項參數，模擬模組將參數設定進系統並進行模擬 | | |

|  |  |  |
| --- | --- | --- |
| **介面名稱(Interface Name)** | **介面提供者(Interface Provider)** | **介面使用者(Interface Consumer)** |
| 使用者與互動模組的介面 | 互動模組 | 使用者 |
| **連結方式(Connection Type)** | **輸入資料(Input Data)** | **輸出資料(Output Data)** |
| Java method call | 需更換的設備或擺放 | 更換結果(畫面呈現) |
| **對應之介面需求(Interface Requirement Description)** | | |
| 使用者選擇欲更換的設備或是擺放，互動模組將參數進行更換並更新前端顯示 | | |

|  |  |  |
| --- | --- | --- |
| **介面名稱(Interface Name)** | **介面提供者(Interface Provider)** | **介面使用者(Interface Consumer)** |
| 使用者與紀錄模組的介面 | 紀錄模組 | 使用者 |
| **連結方式(Connection Type)** | **輸入資料(Input Data)** | **輸出資料(Output Data)** |
| Java method call | 欲查詢的日期 | 歷史紀錄 |
| **對應之介面需求(Interface Requirement Description)** | | |
| 使用者輸入想查詢的日期，紀錄模組將歷史紀錄顯示供使用者查閱 | | |

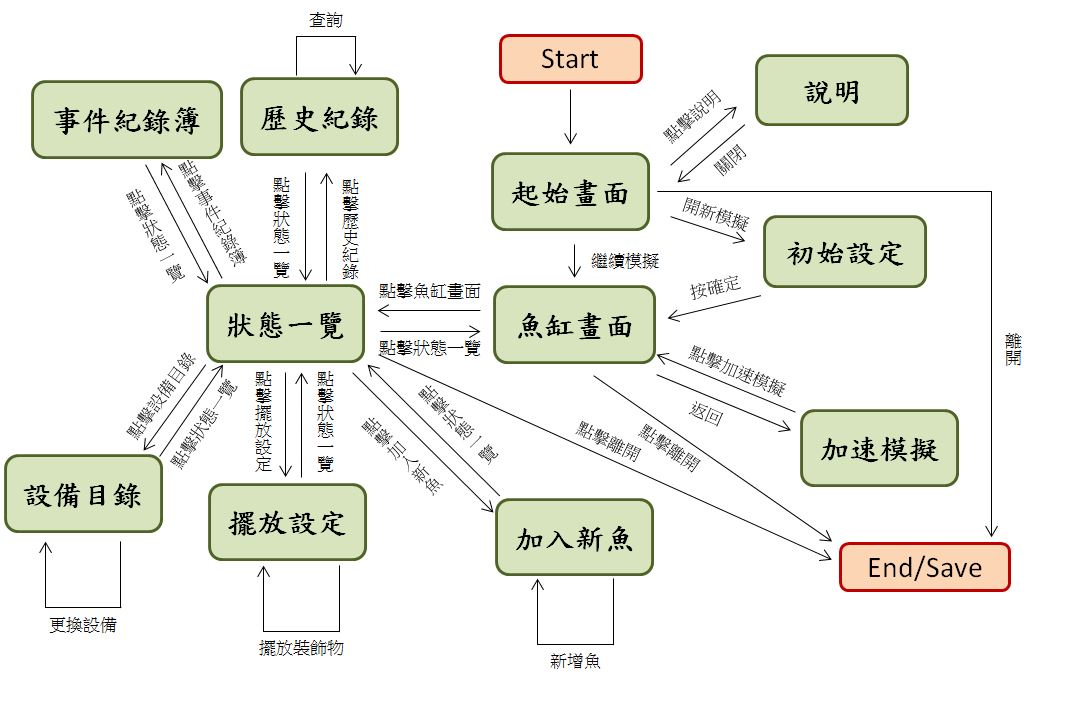
|  |  |  |
| --- | --- | --- |
| **介面名稱(Interface Name)** | **介面提供者(Interface Provider)** | **介面使用者(Interface Consumer)** |
| 模擬模組與事件模組的介面 | 事件模組 | 模擬模組 |
| **連結方式(Connection Type)** | **輸入資料(Input Data)** | **輸出資料(Output Data)** |
| Java method call | 魚的資料 | 發生的事件 |
| **對應之介面需求(Interface Requirement Description)** | | |
| 模擬模組傳遞魚現在的狀況給事件模組判斷事件是否會發生 | | |

|  |  |  |
| --- | --- | --- |
| **介面名稱(Interface Name)** | **介面提供者(Interface Provider)** | **介面使用者(Interface Consumer)** |
| 模擬模組與設備模組的介面 | 設備模組 | 模擬模組 |
| **連結方式(Connection Type)** | **輸入資料(Input Data)** | **輸出資料(Output Data)** |
| Java method call | 想更換的設備 | 設備對環境影響 |
| **對應之介面需求(Interface Requirement Description)** | | |
| 互動模組傳遞使用者想更換的設備給模擬模組，設備模組將設備更新完後告訴模擬模組更新完成，模擬模組再將前端畫面更新並做參數影響 | | |

|  |  |  |
| --- | --- | --- |
| **介面名稱(Interface Name)** | **介面提供者(Interface Provider)** | **介面使用者(Interface Consumer)** |
| 模擬模組與擺放模組的介面 | 擺設模組 | 模擬模組 |
| **連結方式(Connection Type)** | **輸入資料(Input Data)** | **輸出資料(Output Data)** |
| Java method call | 想更換的擺設 | 擺設對模擬影響 |
| **對應之介面需求(Interface Requirement Description)** | | |
| 擺放模組傳遞使用者想更換的擺設給模擬模組，擺放模組將擺設更新完後告訴模擬模組更新完成，模擬模組再將前端畫面更新並做參數影響 | | |

|  |  |  |
| --- | --- | --- |
| **介面名稱(Interface Name)** | **介面提供者(Interface Provider)** | **介面使用者(Interface Consumer)** |
| 紀錄模組與MySQL模組的介面 | MySQL模組 | 紀錄模組 |
| **連結方式(Connection Type)** | **輸入資料(Input Data)** | **輸出資料(Output Data)** |
| JDBC | MySQL指令 | 對應指令所存入的資料 |
| **對應之介面需求(Interface Requirement Description)** | | |
| 紀錄模組將魚的狀況及事件資料存入MySQL模組，MySQL模組根據紀錄模組所發出之MySQL指令進行資料的存取 | | |

1. **流程設計(Process Design)**



狀態敘述(State description)

|  |  |
| --- | --- |
| Name | 起始畫面 |
| Definition | 提供選擇開新模擬、繼續模擬、說明、離開等功能 |

|  |  |
| --- | --- |
| Name | 魚缸畫面 |
| Definition | 提供動畫畫面、狀態一覽、加速模擬、進階互動等功能 |

|  |  |
| --- | --- |
| Name | 狀態一覽 |
| Definition | 觀看魚類狀態、環境狀態，功能選擇列表 |

|  |  |
| --- | --- |
| Name | 歷史紀錄 |
| Definition | 查詢並查看歷史上特定時間點的資料 |

|  |  |
| --- | --- |
| Name | 事件紀錄簿 |
| Definition | 紀錄當天所有已發生的事件 |

|  |  |
| --- | --- |
| Name | 設備目錄 |
| Definition | 提供裝備供使用者挑選使用 |

|  |  |
| --- | --- |
| Name | 擺放設定 |
| Definition | 選擇裝飾物放置到不同位置，購買不同裝飾物 |

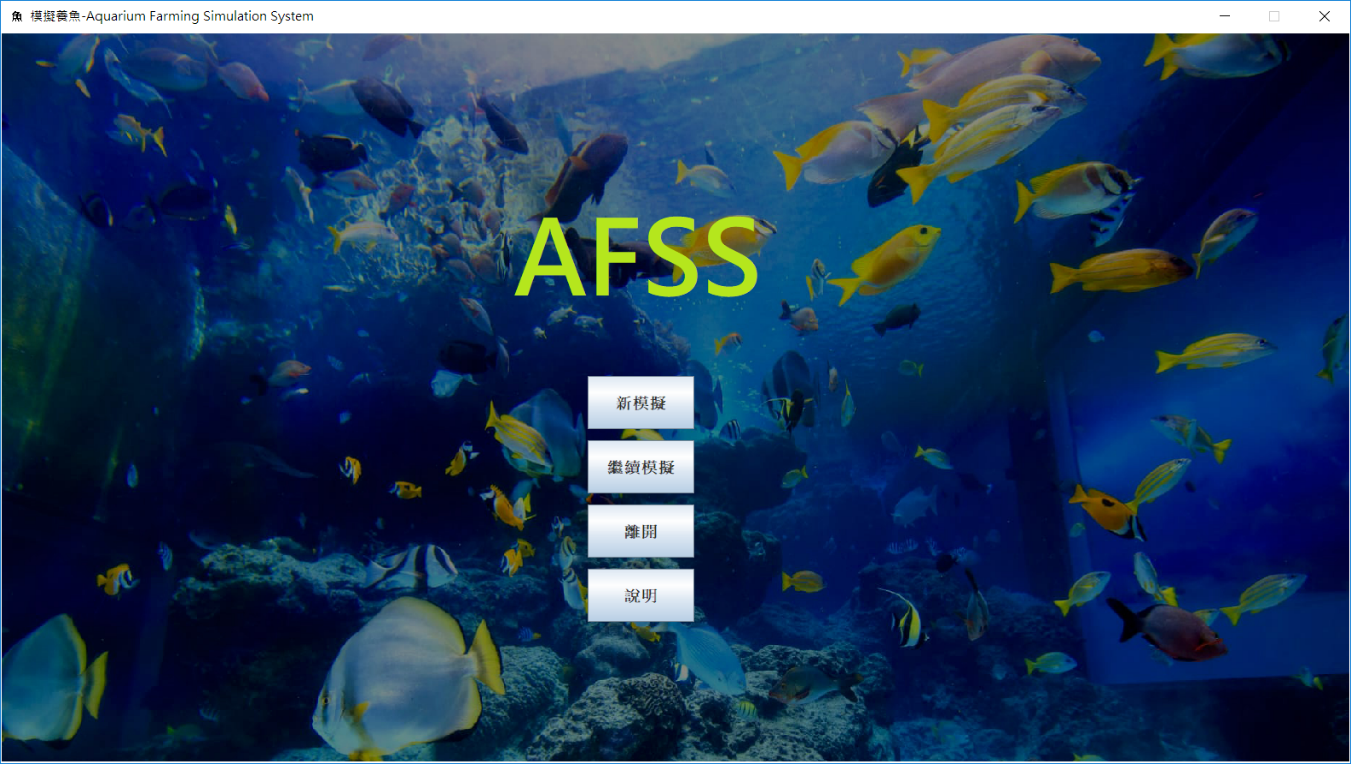
|  |  |
| --- | --- |
| Name | 加入新魚 |
| Definition | 加入新的魚類、選擇數量 |

|  |  |
| --- | --- |
| Name | 說明 |
| Definition | 告知使用者此程式使用的方法及規則 |

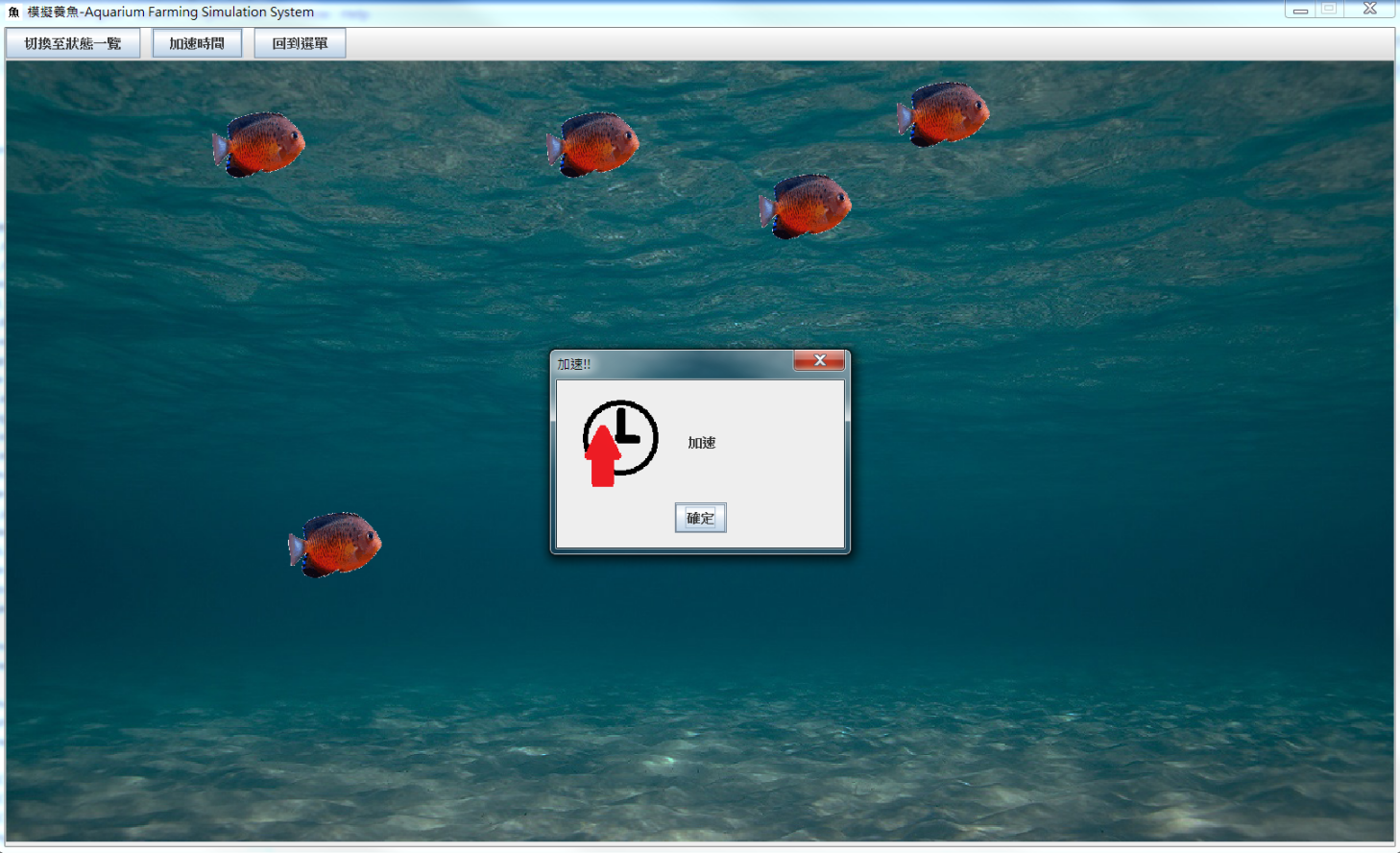
|  |  |
| --- | --- |
| Name | 初始設定 |
| Definition | 第一次使用魚缸做環境基礎的設定並套用 |

|  |  |
| --- | --- |
| Name | 加速模擬 |
| Definition | 選擇加速跳躍時間、加速期間忽略事件 |

1. **使用者畫面設計(User Interface Design)**
   1. 選擇模擬/說明



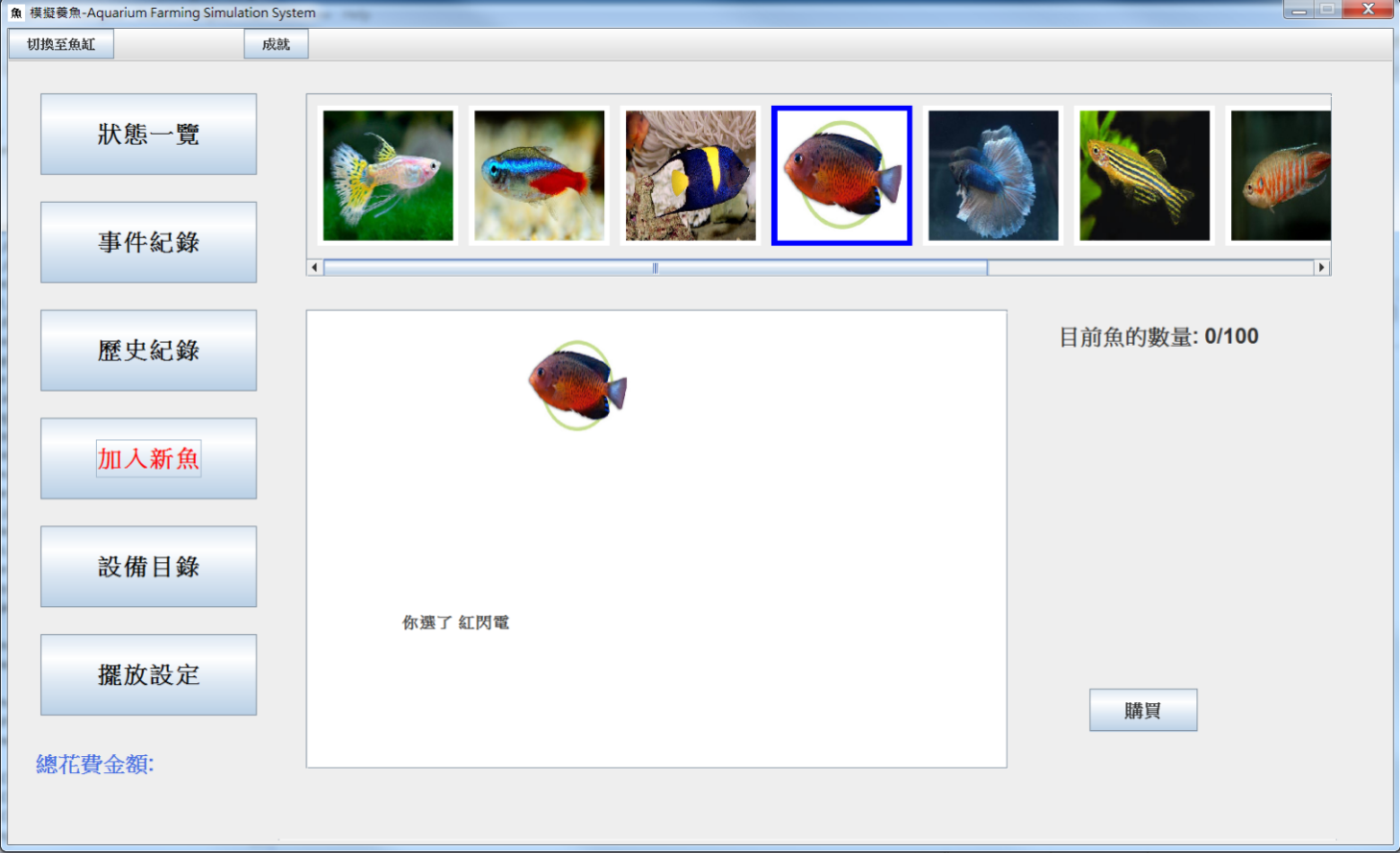
* 1. 魚缸呈現畫面



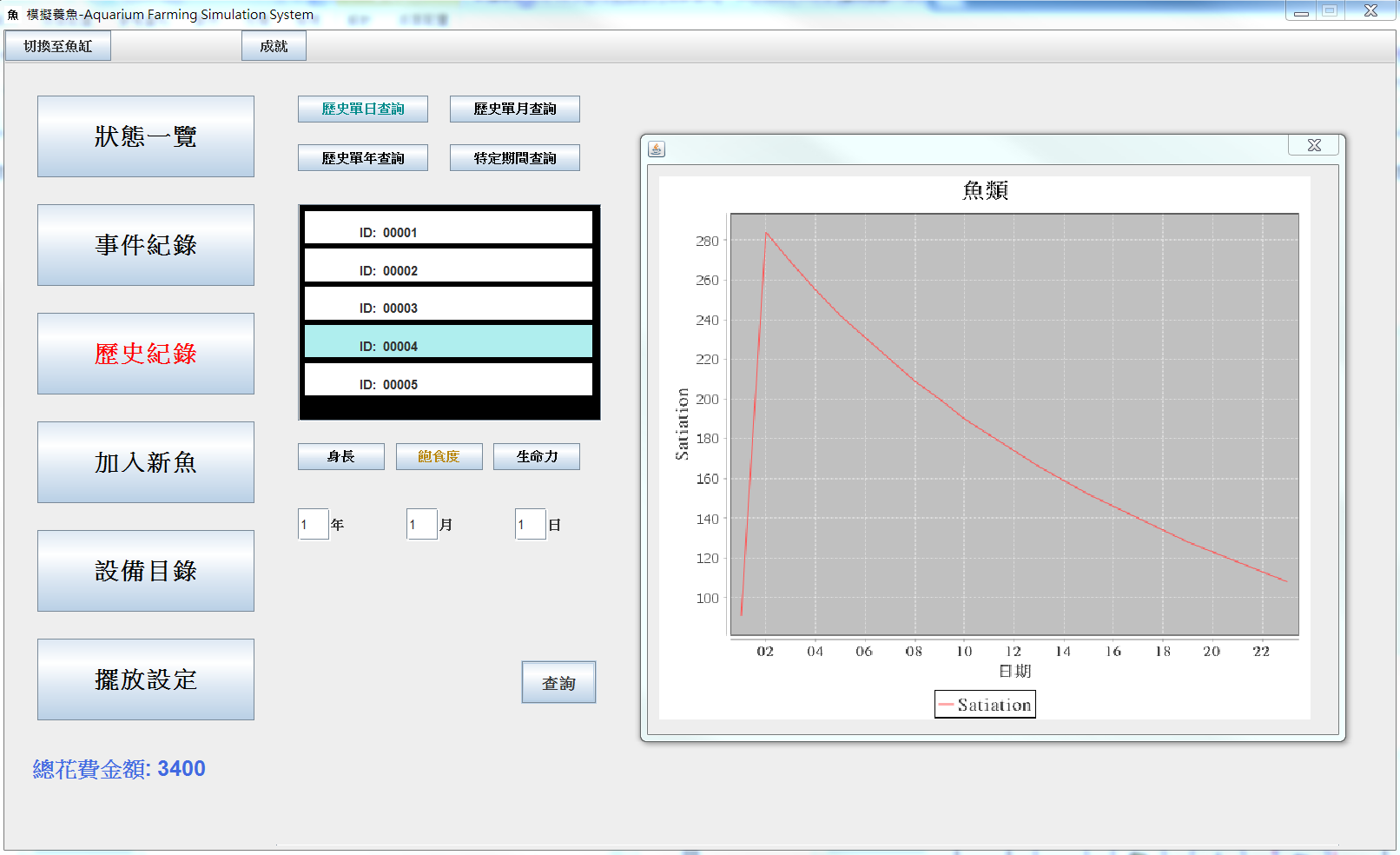
* 1. 查看魚缸數據



* 1. 加入新的魚



* 1. 查詢歷史紀錄



* 1. 觀看所有事件



* 1. 對裝飾物做擺設



* 1. 升級設備



1. **資料設計(Data Design)**

**資料庫schema**

**Fish表格**

CREATE TABLE ‘Fish’ (

‘Fish\_ID’ varchar (20) NOT NULL CHARACTER SET utf8mb4,

Fish\_CName’ varchar (20) NOT NULL CHARACTER SET utf8mb4,

‘Fish\_Length’ DOUBLE(10) NOT NULL,

‘Fish\_Satiation’int(10) NOT NULL,

‘Fish\_Age’ varchar(10) NOT NULL,

‘Fish\_Lively’ int(10) NOT NULL,

‘ Fish\_status ’ VARCHAR(20) CHARACTER SET utf8mb4,

‘Fish\_Healthy’ VARCHAR(20) CHARACTER SET utf8mb4,

‘Date\_ID’ varchar(5) NOT NULL,

PRIMARY KEY (‘Fish\_ID’),

FOREIGN KEY (Date\_ID) REFERENCES Date(Date\_ID),

);

[Fish\_ID] 魚的代碼

[Fish\_CName] 中文魚的名字

[Fish\_Length] 魚的長度(公分)

[Fish\_Satiation] 魚的飢餓度

[Fish\_Age] 魚的年齡 (以小時計算)

[Fish\_hurt] 魚的受傷程度

[Fish\_status] 魚的狀態

[Fish\_Healthy] 魚的健康狀態

[Date\_ID] 日期的代碼

|  |  |  |
| --- | --- | --- |
| Fish\_ID | 1 | 2 |
| Fish\_CName | 小丑魚 | 泰國鬥魚 |
| Fish\_Length | 10.0 | 12.2 |
| Fish\_Satiation | 50 | 60 |
| Fish\_Age | 150 | 50 |
| Fish\_Lively | 60 | 30 |
| Fish\_status | DYING | ALIFE |
| Fish\_Healthy | HEALTH | HEALTH |
| Date\_id | 1 | 1 |

**Cost表格**

CREATE TABLE ‘ Cost’ (

’Cost\_ID’ varchar(5) NOT NULL,

‘Cost\_Thing\_Name’ nvarchar(20) NOT NULL,

‘Cost\_Money’ int(8) NOT NULL,

‘Date\_ID’ varchar(5) NOT NULL,

PRIMARY KEY (‘Cost \_ID’),

FOREIGN KEY (Date\_ID) REFERENCES Date(Date\_ID),

)

[Cost\_ID] 花費金額的代碼

[Cost\_Thing\_Name] 花費物品的名字

[Cost\_Money] 花費物品的金額

[Date\_ID] 日期的代碼

|  |  |  |  |
| --- | --- | --- | --- |
| Cost\_ID | Cost\_Thing\_Name | Cost\_Money | Date\_ID |
| 1 | Fish\_food | 100 | 1 |
| 2 | Fish\_tank | 300 | 5 |

**Date表格**

CREATE TABLE ‘Date’ (

’Date\_ID’ varchar(5) NOT NULL,

’Date\_time ’ VARCHAR(20) NOT NULL,

PRIMARY KEY (‘Date \_ID’),

)

[Date\_ID] 日期的代碼

[Date\_time] 年月日小時

|  |  |
| --- | --- |
| Date\_ID | Date\_time |
| 1 | 0001/01/01 00 |
| 2 | 0005/12/30 22 |

**Event表格**

CREATE TABLE ‘Event’ (

’Event\_ID’ varchar(5) NOT NULL,

’Event\_Type’ varchar(20) NOT NULL,

‘Event\_Description’ varchar(80) NOT NULL,

‘Date\_ID’ varchar(5) NOT NULL,

PRIMARY KEY (‘Date \_ID’),

FOREIGN KEY (Date\_ID) REFERENCES Date(Date\_ID),

FOREIGN KEY (Fish\_ID) REFERENCES Date(Fish\_ID),

)

[Event\_ID] 事件的代碼

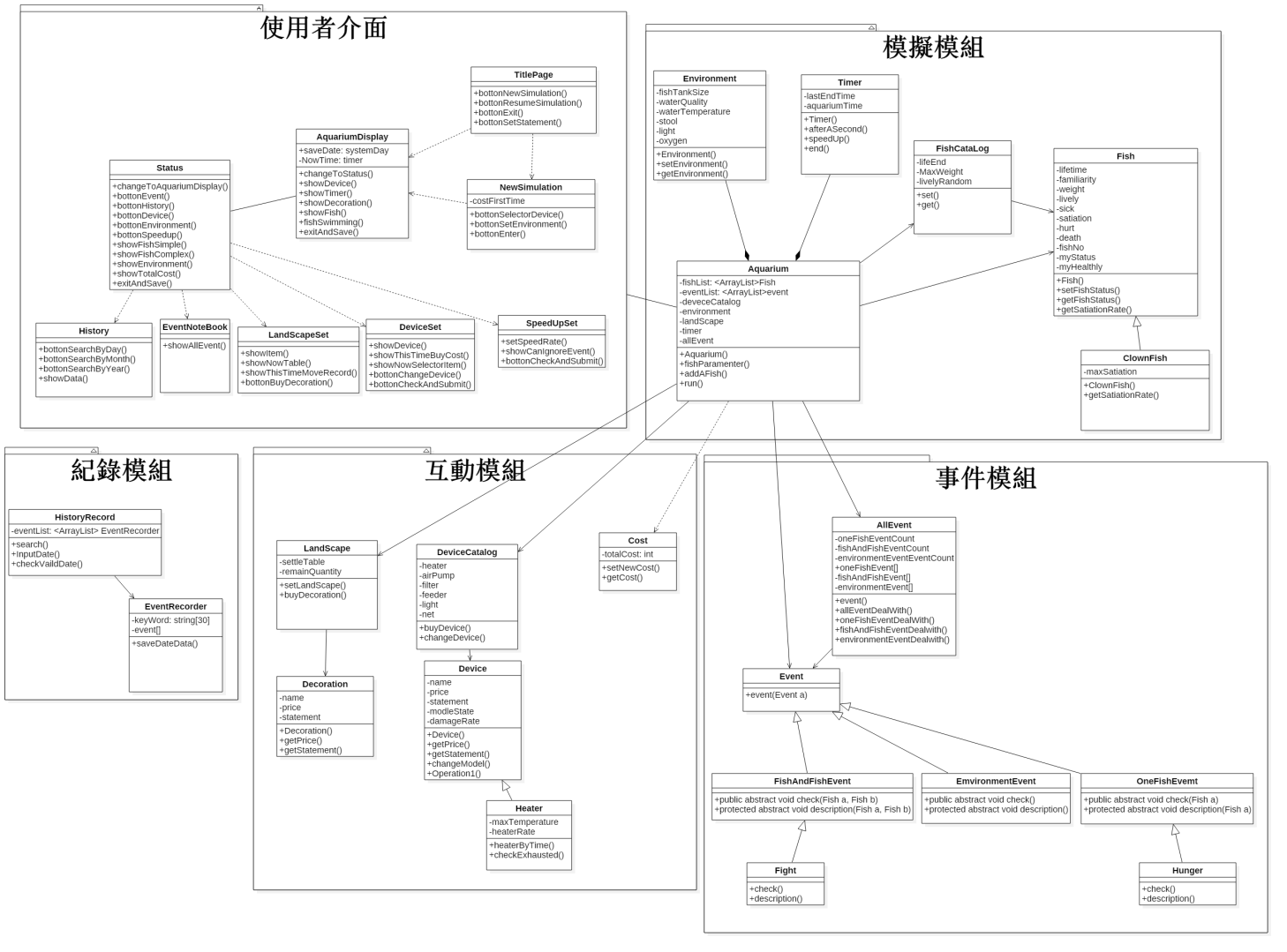
[Event\_Type] 事件的種類

[Event\_Description] 事件的描述

[Date\_ID] 日期的代碼

|  |  |  |  |
| --- | --- | --- | --- |
| Event\_ID | Event\_Type | Event\_Description | Date\_ID |
| 1 | fight | Fish A fight with B | 2 |
| 2 | death | Fish A died | 5 |

1. **類別圖設計(Class Diagram)**



**模擬模組**

|  |  |
| --- | --- |
| **Class Name** | Fish |
| **Module** | 模擬模組 |
| **Description** | 魚的類別，裡面包括了一般魚會擁有的所有參數，此類別為虛構類別(父類別)其他不同種類的魚會繼承此類別。 |
| **Relationship with other classes** | Aquarium, FishCatalog, FreshwaterFish, OceanFish |
| **Attributes** | FishStatus, FishHealthly, FishMove, totalFish, fishNO, lifeTime, lifeEnd, familiarity, maxWeight, maxSatiation, weight, lively, sick, satiation, hurt, death, nowPosition[], goalPosition[],feedArray |
| **Traceability with other use case** | 新增魚 |

|  |  |
| --- | --- |
| **Class Name** | BettaSplendens |
| **Module** | 模擬模組 |
| **Description** | 繼承魚的類別，一個小丑魚的物件、裡面記錄了屬於這種類魚有的數值 |
| **Relationship with other classes** | Fish |
| **Attributes** | maxSatiation |
| **Traceability with other use case** | 無 |

|  |  |
| --- | --- |
| **Class Name** | Environment |
| **Module** | 模擬模組 |
| **Description** | 環境類別，記錄所有環境的變數 |
| **Relationship with other classes** | Aquarium |
| **Attributes** | Water, fishTankSize, fishTankXYZSize[], waterQuality, waterTemperature, stool, stoolXY, feedXY, oxygen |
| **Traceability with other use case** | 建立新的模擬、選擇魚缸、設定環境 |

|  |  |
| --- | --- |
| **Class Name** | Timer |
| **Module** | 模擬模組 |
| **Description** | 時間類別，負責記錄魚缸總模擬的時間和離開程式時的時間，並且會隨著時間增加總模擬時間的值和回傳加速模擬所需加速的小時數 |
| **Relationship with other classes** | Aquarium |
| **Attributes** | lastEndTime, aquariumTime, speedUpTime |
| **Traceability with other use case** | 繼續模擬、加速模擬 |

|  |  |
| --- | --- |
| **Class Name** | FishCatalog |
| **Module** | 模擬模組 |
| **Description** | 魚的圖鑑類別，記錄了不同種類的魚它的基礎數值和最大數值(身長、年齡) |
| **Relationship with other classes** | Fish、Aquarium |
| **Attributes** | EatType, fishName, lifeEnd, lifeEndRange, initLively, maxLength, maxLengthRange, initFight, eating, suitableTemperature, statement |
| **Traceability with other use case** | 無 |

|  |  |
| --- | --- |
| **Class Name** | Aquarium |
| **Module** | 模擬模組 |
| **Description** | 水族箱類別，記錄水族箱中的狀態和裡面有的魚，並記錄當天發生過的事件和模擬的運作 |
| **Relationship with other classes** | Environment, Timer, Fish, FishCatalog, AllEvent, Cost, DeviceCatalog、LandScape, Cost\_data, Date\_data,Move,Feed |
| **Attributes** | fishList、eventList、deviceCatalog、environment、landScape、timer、  allEvent |
| **Traceability with other use case** | 顯示魚缸畫面、更換擺設、加速模擬、新增魚，顯示魚缸狀態 |

**事件模組**

|  |  |
| --- | --- |
| **Class Name** | Event |
| **Module** | 事件模組 |
| **Description** | 事件類別，對事件的描述之類的，為虛構類別(子類別會繼承他) |
| **Relationship with other classes** | Aquarium, Event\_data, Timer |
| **Attributes** | 無 |
| **Traceability with other use case** | 無 |

|  |  |
| --- | --- |
| **Class Name** | AllEvent |
| **Module** | 事件模組 |
| **Description** | 事件圖鑑的類別，記載了每一個事件，並且有提供一次判斷所有事件是否有發生過的method |
| **Relationship with other classes** | Event、Aquarium |
| **Attributes** | oneFishEvent、fishAndFishEvent、environmentEvent |
| **Traceability with other use case** | 加速模擬 |

|  |  |
| --- | --- |
| **Class Name** | FishAndFishEvent |
| **Module** | 事件模組 |
| **Description** | 魚和魚之間的事件類別，繼承事件類別，為父類別，其子類別都為魚和魚之間發生的事件(打架、搶食…) |
| **Relationship with other classes** | Event |
| **Attributes** | 無 |
| **Traceability with other use case** | 無 |

|  |  |
| --- | --- |
| **Class Name** | EnvironmentEvent |
| **Module** | 事件模組 |
| **Description** | 環境的事件類別，繼承事件類別，為父類別，其子類別都為環境會發生的事件(水質太糟、水溫過低…) |
| **Relationship with other classes** | Event |
| **Attributes** | 無 |
| **Traceability with other use case** | 無 |

|  |  |
| --- | --- |
| **Class Name** | OneFishEvent |
| **Module** | 事件模組 |
| **Description** | 單隻魚的事件類別，繼承事件類別，為父類別，其子類別都為單隻魚會發生的事件(飢餓、死亡…) |
| **Relationship with other classes** | Event |
| **Attributes** | 無 |
| **Traceability with other use case** | 無 |

|  |  |
| --- | --- |
| **Class Name** | Fight |
| **Module** | 事件模組 |
| **Description** | 打架事件類別，繼承魚和魚之間的類別，會檢查事件是否發生，發生的話會對事件進行描述和處理 |
| **Relationship with other classes** | FishAndFishEvent |
| **Attributes** | 無 |
| **Traceability with other use case** | 無 |

|  |  |
| --- | --- |
| **Class Name** | Satiation |
| **Module** | 事件模組 |
| **Description** | 飢餓事件類別，繼承單隻魚之間的類別，會檢查事件是否發生，發生的話會對事件進行描述和處理 |
| **Relationship with other classes** | OneFishEvent |
| **Attributes** | 無 |
| **Traceability with other use case** | 無 |

**互動模組**

|  |  |
| --- | --- |
| **Class Name** | Cost |
| **Module** | 互動模組 |
| **Description** | 紀錄所有已使用的金錢 |
| **Relationship with other classes** | Aquariums |
| **Attributes** | totalCost |
| **Traceability with other use case** | 更換擺設 |

|  |  |
| --- | --- |
| **Class Name** | LandScape |
| **Module** | 互動模組 |
| **Description** | 將裝飾物放置到對應位置產生造景 |
| **Relationship with other classes** | Decoration、Aquarium、Cost、Cost\_data、Date\_data |
| **Attributes** | Table、stoneQuantity、cobbleStoneQuantity、prismStoneQuantity、shenmuQuantity、coralQuantity、waterPlantQuantity |
| **Traceability with other use case** | 更換擺設 |

|  |  |
| --- | --- |
| **Class Name** | Decoration |
| **Module** | 互動模組 |
| **Description** | 裝飾物的基礎參數和使用設定 |
| **Relationship with other classes** | LandScape、Coral、Shenmu、Stone、WaterPlant |
| **Attributes** | Name, price, statement,sizex,sizey,damageRateNow |
| **Traceability with other use case** | 更換擺設 |

|  |  |
| --- | --- |
| **Class Name** | DeviceCatalog |
| **Module** | 互動模組 |
| **Description** | 設備總目錄，做所有設備的初始化和檢閱及使用 |
| **Relationship with other classes** | Aquarium, Device, ChangeWater, Feeder, Filter, Inflator, FishNet, FlashLight, Heater, Cost\_data, Date\_data |
| **Attributes** | feeder[], filter[], inflator[], flashLight[], heater[],fishnet, changeWater |
| **Traceability with other use case** | 更換設備 |

|  |  |
| --- | --- |
| **Class Name** | Device |
| **Module** | 互動模組 |
| **Description** | 設備的基礎參數和使用設定 |
| **Relationship with other classes** | Aquarium, DeviceCatalog, Feeder, Filter, Heater, Inflator, FlashLight, FishNet, ChangeWater |
| **Attributes** | Name, price, statement, modelState, damageRateNow, operateModel, countTime, haveBuy |
| **Traceability with other use case** | 更換設備 |

|  |  |
| --- | --- |
| **Class Name** | Heater |
| **Module** | 互動模組 |
| **Description** | 對魚缸內部環境的溫度加熱 |
| **Relationship with other classes** | Device |
| **Attributes** | maxTemperature, heaterRate |
| **Traceability with other use case** | 無 |

**紀錄模組**

|  |  |
| --- | --- |
| **Class Name** | Connect\_data |
| **Module** | 紀錄模組 |
| **Description** | 建立和資料庫的連線 |
| **Relationship with other classes** | Cost\_data, Date\_data, Event\_data, Fish\_data, IO,Draw\_line |
| **Attributes** | Con, stat, rs, pst |
| **Traceability with other use case** | 歷史紀錄 |

|  |  |
| --- | --- |
| **Class Name** | Date\_data |
| **Module** | 紀錄模組 |
| **Description** | 紀錄時間ID到資料庫內，把每個單位時間命名 |
| **Relationship with other classes** | Aquarium |
| **Attributes** | 無 |
| **Traceability with other use case** | 歷史紀錄 |

|  |  |
| --- | --- |
| **Class Name** | Event\_data |
| **Module** | 紀錄模組 |
| **Description** | 紀錄事件到資料庫內，根據Timer給予每個事件一個特定ID， |
| **Relationship with other classes** | Aquarium |
| **Attributes** | 無 |
| **Traceability with other use case** | 歷史紀錄 |

|  |  |
| --- | --- |
| **Class Name** | Draw\_line |
| **Module** | 紀錄模組 |
| **Description** | 將資料庫內的資料繪製成一份圖表呈現給使用者做觀察 |
| **Relationship with other classes** | Aquarium |
| **Attributes** | 無 |
| **Traceability with other use case** | 歷史紀錄 |

1. **實作技術(Implementation Languages and Platforms)**

系統實作平台選擇在個人電腦端上執行，主要用應用程式的方式使用，且不需連接到網際網路就可以使用。應用程式架構主要分成兩部分，呈現使用到JAVA Swing去做出框架及選項提供給使用者去做選擇和操作，也做到魚缸畫面的動態呈現讓使用者體驗更貼近真實；利用Jfreechart繪製出圖表讓紀錄層面更加透徹；Java Clip在後台加上音效使之更為生動。平台後端有使用到MySQL去做資料庫儲存，且利用折線圖的方式呈現給使用者端。

1. **設計議題(Design Issue)**

**議題一**

討論日期：2017/10/13

問題定義：採用「回合制」還是「即時制」的模擬方法?

解決方案：

方案A：

以回合制的方式來呈現，使用者做完一輪處理動作後(餵魚、新增設備…)後進行一輪的模擬後才在准許使用者進行下一輪的操作。

方案B：

採用即時模擬的方式，模擬會在每個單位時間都模擬一次，使用者可以隨時地執行操作來影響接下來的模擬結果。

最終定案：方案B

定案理由：採用回合制的方式雖然簡單，但非常地不貼近模擬這件事情，為達成

模擬的效果所以採用即時的模擬，這樣的話才更貼近現實。

**議題二**

討論日期：2017/11/9

問題定義：是否加入生殖系統

解決方案：

方案A：

加入，當同種公魚和母魚相處一段時間後，有機率會觸發生殖事件

方案B：

不加入，就算同種魚的公母魚長期相處在一起也不會發生交配事件。

最終定案：方案B

定案理由：生殖系統會造成影響的變數太多了，而且生出來的小魚需要進行額外的照顧和培養，除了不好實作外，還要將小魚隔離飼養這件事等於多開了一個魚缸來模擬，就不符合原本一缸模擬的狀況了，需要新增魚的話就直接利用購買的方式來添加就好。 生殖系統整體來說成本太高。

**議題三**

討論日期：2017/11/9

問題定義：魚缸要用2D畫面來呈現還是3D畫面？

解決方案：

方案A：

使用2D，Z軸的部分利用圖片的縮小和一點Y軸的配置來實現

方案B：

使用3D，找到適合的API來輔助寫出3D畫面的魚缸

最終定案：方案A

定案理由：使用2D加上設計(圖層、圖片放大縮小、Y軸的輔助)，不需要再額外尋找API來實現3D畫面，類似2D橫向卷軸模式。這個程式主要還是側重在模擬魚的生活情形，後端方面是比較核心實作的區塊。

**議題四**

討論日期：2017/11/21

問題定義：關於魚的成長是呈「線性成長」還是「亂數成長」?

解決方案：

方案A：

使用線性成長來修改參數。

方案B：

使用亂數去決定一個區間的數值來修改參數

最終定案：方案B

定案理由：若使用線性成長會有嚴重的不真實感且失去模擬的本意。使用亂數來進行參數的修改的話，會令參數的成長更加隨機和增加不確定性，這會使程式感覺上來更加地像現實模擬的感覺(真實情況也應該會是如此 ex:一隻魚吃的飼料量不會每次都一樣，會依情況不同有不同的進食量)。

**議題五**

討論日期：2017/11/21

問題定義：資料庫使用「本地存取」還是「網路遠端存取」?

解決方案：

方案A：

使用本地端存取資料

方案B：

資料庫連上網路, 達成遠端存取

最終定案：方案A

定案理由：系統最初設定在個人電腦端不須網路的平台，依現階段的實作方面，如果使用網路來存取資料庫怕會影響整體的程式效率(比如回傳資料的速度並不合預期所要求的)，所以採取本地端的資料庫連線增加穩定性和便利性。

**議題六**

討論日期：2017/11/21

問題定義：前端是否需要物件化?

解決方案：

方案A：

將前端頁面全部寫在一個類別之中

方案B：

將每個前端的頁面都寫成一個類別，並最後把全部以一個類別(main)來實作

最終定案：方案B

定案理由：將全部寫在同一個類別的話會讓一個類別內太過雜亂，以至於連接後端和資料庫時會造成實作上的困難，還會造成維護和修改的困難，因此決定用物件化來解決這些問題也增加未來維護和擴充的可行性。

**議題七**

討論日期：2017/11/21

問題定義：物件的劃分和使用

解決方案：

方案A：

將魚種特別的參數(最大體長、最大壽命…)放在該「魚種」的類別中

方案B：

將魚種特別的參數(最大體長、最大壽命…)放在「魚」的類別中

方案C：

在寫一個類別叫做魚種圖鑑來儲存魚種特別的參數

最終定案：方案C

定案理由：方案A和B的話都會有使用上的矛盾，A(水族箱類別需同時知道魚種的類別和魚的類別因為這些參數只有魚種類別才會知道)

B(同一種魚種可以設定不同的最大壽命和體長，這樣就不像是這個魚種的特別參數，而是這個魚的特別參數)

因此使用方案C，額外建立一個類別來記錄這些資料，需要時直接從該類別提取資料來使用的邏輯比較說得通(透過圖鑑來查詢最大壽命和身長)。

**議題八**

討論日期：2017/10/13

問題定義：影響參數挑選(考慮外在因素)

解決方案：

方案A：

考慮加入很多的環境變數，讓模擬貼近百分之百的真實(ex:水蒸發、外部溫度)

方案B：

僅挑選關鍵環境變數做設計和撰寫

最終定案：方案B

定案理由：整個模擬要貼近真實需要有很多看的到的和看不到的變因，但是要追究有太多變因可以加進去了，但是我們取捨掉比較不重要的(影響環境較不明顯的變因)，只加入較關鍵的核心參數，如此一來不失模擬性也不會讓製作過程產生太多困難。

**議題九**

討論日期：2017/11/9

問題定義：朝向「遊戲模式」還是「模擬模式」的方向製作?

解決方案：

方案A：

遊戲模式偏重使用者的互動性也會需要比較多前端技術，後端實際推算模擬部分不會深度講究

方案B：

模擬模式偏重後端關係運算式制定，需要比較多設計，也會有比較多的議題需要討論(ex:事件互相影響、魚類之間的爭鬥)。

最終定案：方案B

定案理由：我們想要朝模擬方面發展，所以偏重於後端事件關係的設計和運算，但是我們又不想讓使用者失去前端的使用體驗，所以我們雖偏重於方案B但還是會將遊戲模式中使用者便利的介面設計加入，增加使用者的方便性及互動性。

**議題十**

討論日期：2017/11/21

問題定義：「加速模擬」功能使用後處理的方式

解決方案：

方案A：

使用者選擇加速一段時間碰到事件後(使用者選擇的時間尚未結束)，完全停止「加速模擬」並讓使用者可以處理事件，處理事件完之後不再繼續執行加速模擬剩餘的時間

方案B：

使用者選擇加速時間碰到事件後(使用者選擇的時間尚未結束)，暫停「加速模擬讓使用者處理當次事件，事件處理完後繼續執行「加速模擬」尚未結束之時間，循環至時間結束

最終定案：方案A

定案理由：選擇停止後不再繼續執行，相對於另外一種方式有更多讓使用者觀察變化的空間，假如繼續執行完剩餘時間會讓使用者沒辦法獲得最多的觀察體驗。所以選擇碰到事件停止，但也增加略過選項的功能讓使用者做勾選，可以勾選不想處理的事件。

**議題十一**

討論日期：2017/11/21

問題定義：系統關閉後是否繼續執行後端計算?

解決方案：

方案A：

系統關閉後應用程式也關閉，紀錄系統關閉時間，下次開啟時用開啟時間和上次關閉時間做推算，且一樣會做模擬，比例尺依照現實時間做推進。後端計算對於使用者來說是關閉的，但是實際上是有在運行的。

方案B：

系統關閉後應用程式也關閉，紀錄系統內timer，下次開啟時從上次關閉的時間再開始運行，關閉後對使用者來說不會做後端計算。關閉應用程式只需紀錄一個參數。

最終定案：方案A

定案理由： 選擇關閉後後端也進行推算貼近真實狀況，而且timer也依照現實時間1:1推進。類似使用者離開不再魚缸旁觀測的時間，不過我們會提醒使用者在離開的期間內發生哪些事情，如此一來使用者可以體驗不再魚缸的模擬實際情況也能知會離開時事件發生的順序列表。

**議題十二**

討論日期：2017/11/21

問題定義：資料庫存取是否用connection pool 的連接方式來做設計???(說明:簡單來說就

是作業系統的Thread Pool)

其優點就是:降低與資料庫連線時間及效率

解決方案：

方案A:

用，對未來擴大程式規模發展成多人連線使用時有幫助

方案B:

不用，對應目前程式開發使用設定者為一人，並無多人連線使用的問題，故無此需要

最終定案:方案B

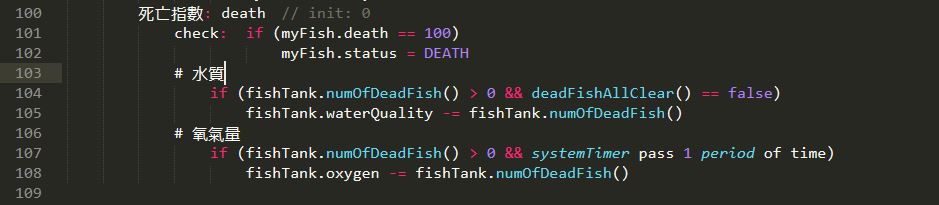
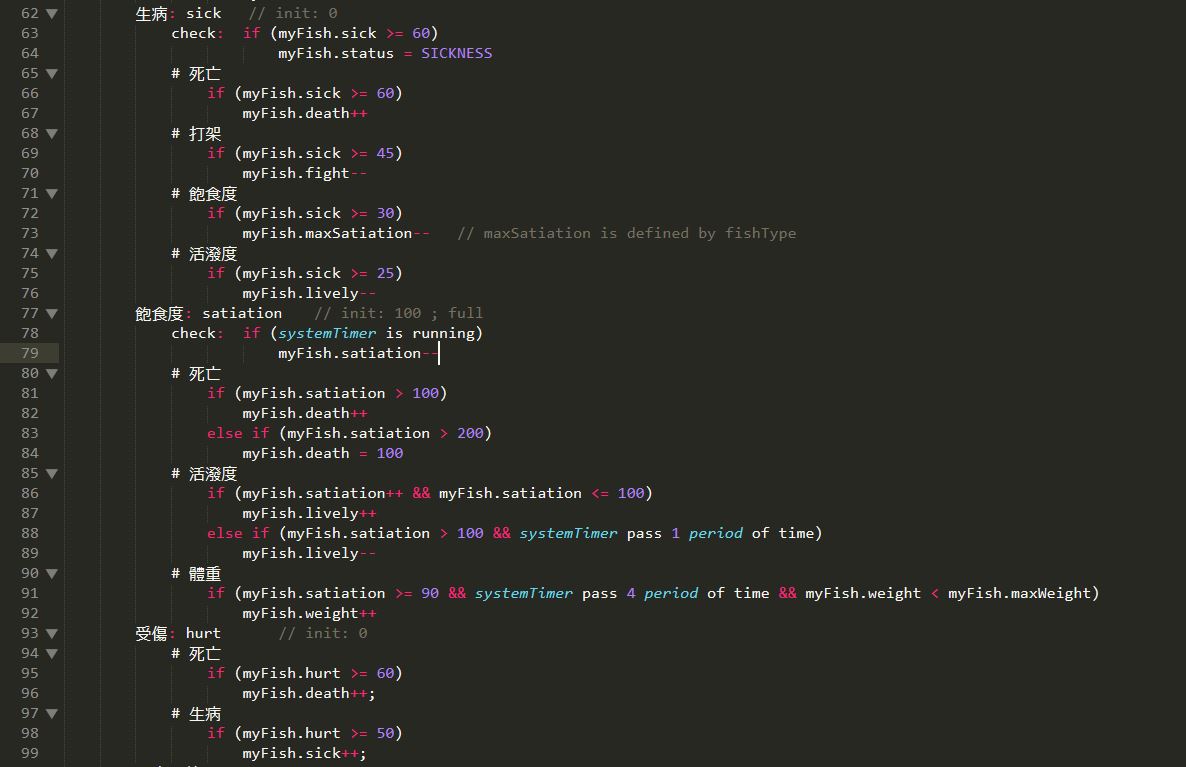
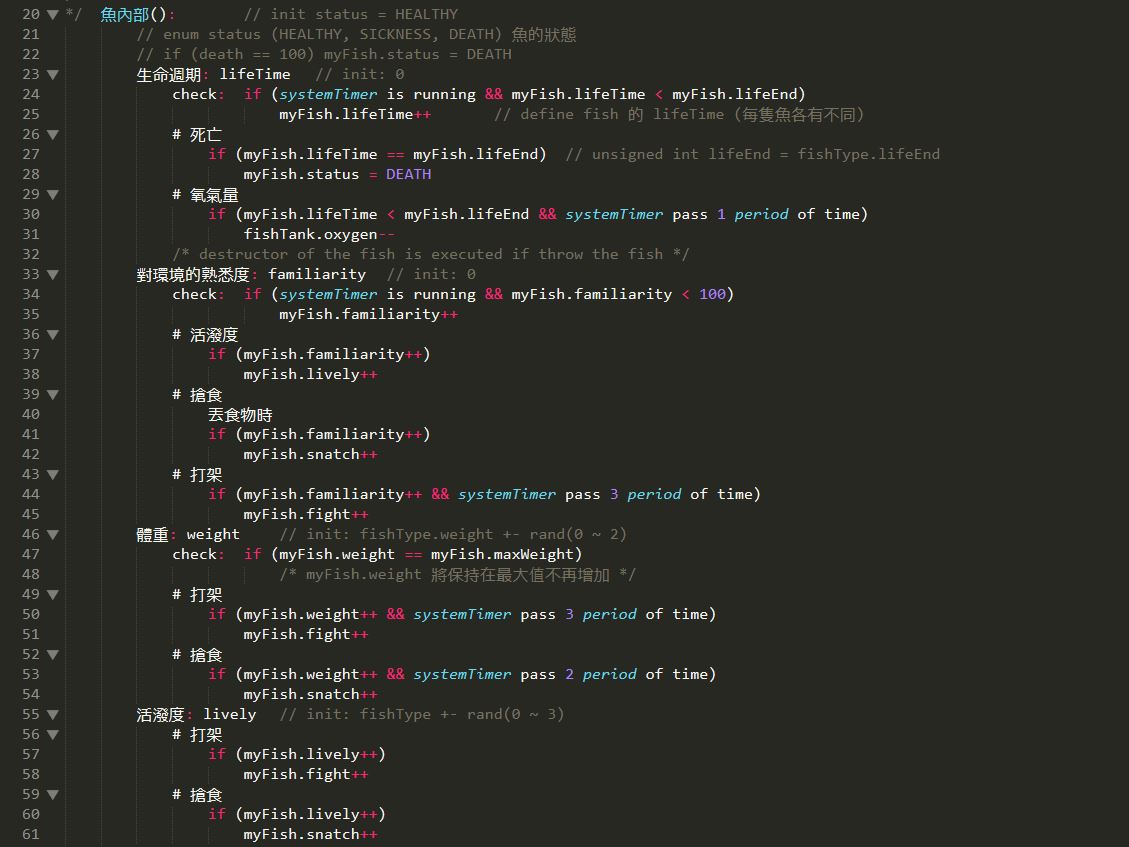
定案理由:

因為與資料庫的連線資料傳輸不大，一小時一次，預估連線時間不會到太長。使用connection pool 的需求性並不高加上程式設定使用者為一人；需用connection pool 的需求可能要到使用者二人以上才會顯示其效率。

1. **演算法設計(Algorithm Design)**

**魚類影響設計**

(一)魚類內部變因

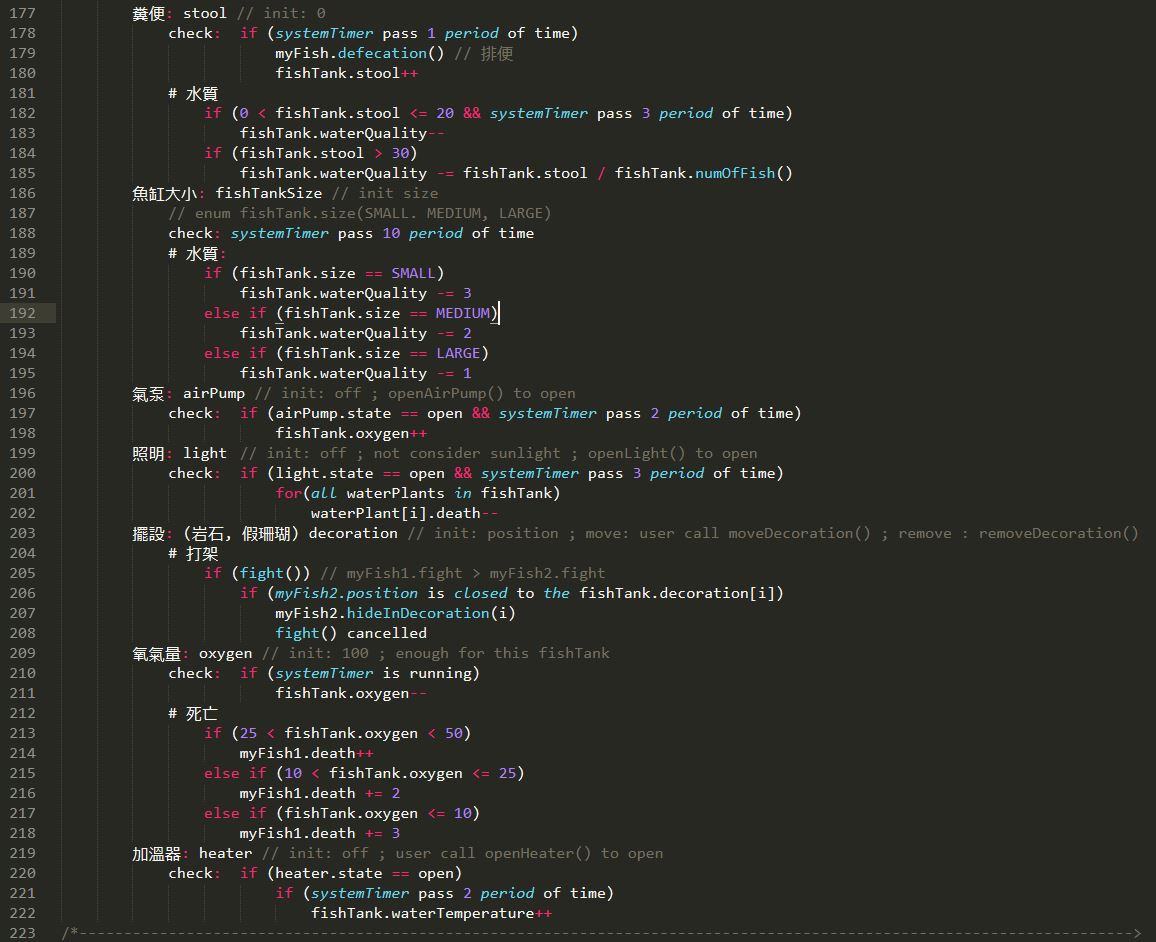
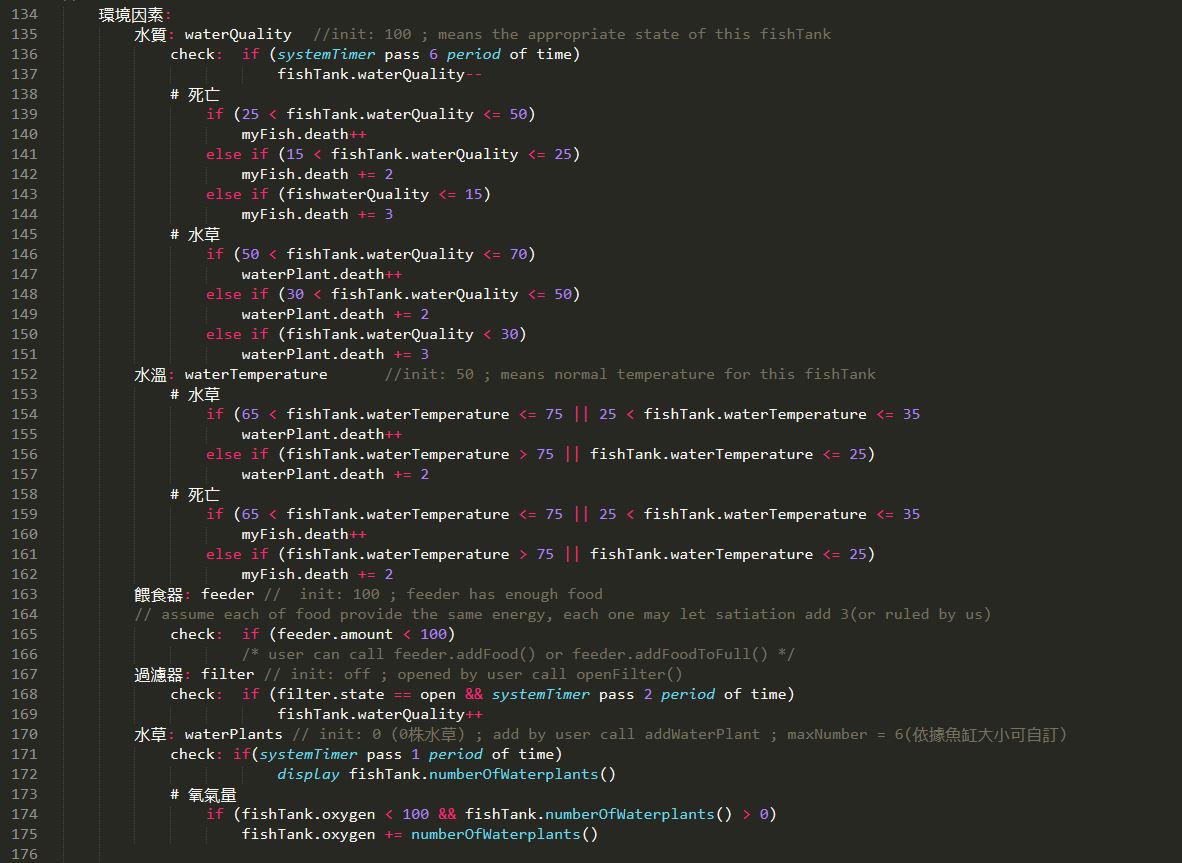


(二)魚類外部變因

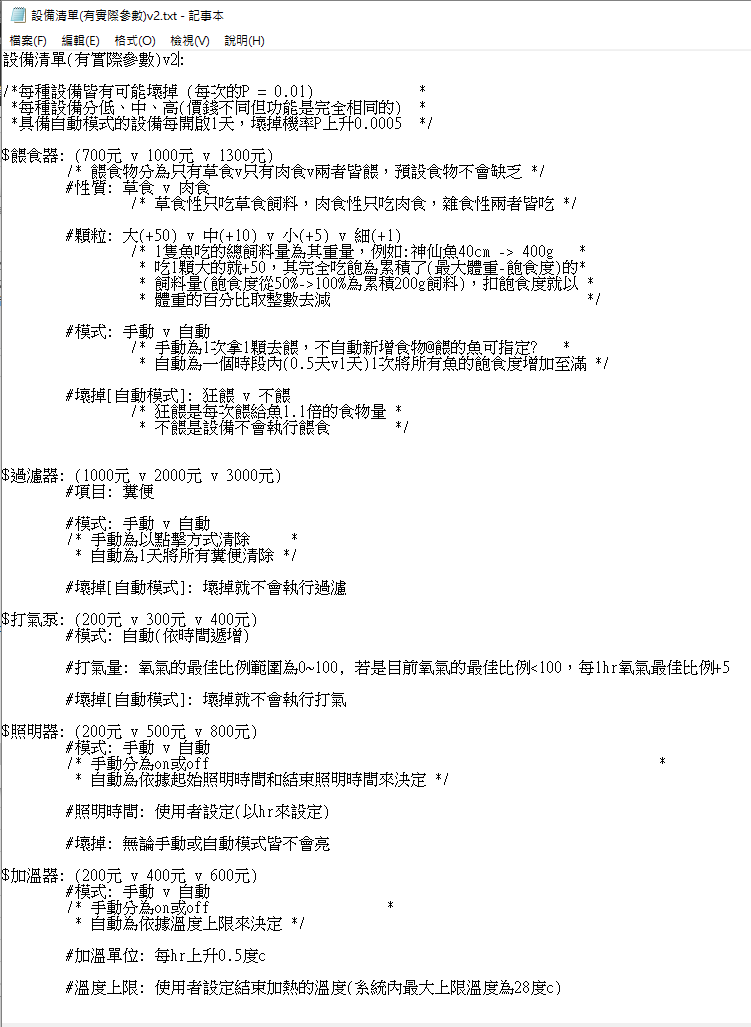


**環境影響設計**

(一)環境變因



**設備與擺設設計：**



(二)環境執行先後判斷演算法：

演算法判斷順序：環境變動 > 魚內部 > 魚外部(魚和魚之間)

**理由：由於魚外部屬於魚和魚之間的互動，需先將環境(EX:是否有灑食物)和魚內部(EX:游到哪邊)先設定好再進行判斷**

(三) 多重Thread環境下執行：

主要分成兩條thread做運行，其一是負責「控制」相關流程的thread，舉凡像是使用者的輸入操作、讀入檔案等等。另外一條thread則是負責「模擬」的部分，凡是有關模擬(包括呈現的部分)都是由這條thread來負責。來預防使用者如果控制流程的時候會影響到背後模擬的狀況。

1. **擴充插件(Expansion plug-in)**

**(一).音效、背景音樂**

**(二).成就系統**

增加使用者趣味性，並且加入冒險探索的元素讓使用者從中學習。

