1. **動機、想法**
2. 動機與目的

|  |
| --- |
| **（圖一）** |

本次最大動機在於該如何完美呈現物件導向的設計，能完美的展現物件導向的必要性。

這次我們選擇了製作一款遊戲當作這次專題的目標，因為遊戲本身需要靠user本身的點擊來觸發事件，所以有許多事件能以封裝的方式表現出來。許多的出發機制也有相同的特性與條件，可以完美發會出繼承與多形。

在許多遊戲中，我們選擇開發井字遊戲，雖然他的遊戲規則簡單，很好設計，但正因為他的簡單，更適合作為我們以物件導向思維的方式來訓練撰寫的對象。除了一般所認知的井字遊戲外，我們也另有設計一款進階的井字遊戲，在後面的報告會提到它的玩法。

1. 目標

* **學會物件導向的思維設計**：在撰寫前先規劃好封裝類別，類別間的關係是什麼，事件的觸發該怎麼規劃等。以UML的方式先設計出藍圖。最後將封裝、繼承、多行實現出來。
* **實現UI畫面**：為了將遊戲實現出來，我們需要UI來乘載我們的遊戲，因此在撰寫遊戲規則前，我們需要知道怎麼實現將UI實現出來。除此之外，UI畫面也需要以物件導向的形式去設計，每個button將成一個封裝，在不同場景有不同的應用。
* **完成井字遊戲**：主要希望以物件導向的形式寫出井字遊戲，並將其與UI畫面做整合，最後將其渲染。

1. **UML**
2. user model

|  |
| --- |
| **（圖二）** |
| ↑上圖為站在使用者的角度去設計的UML，稱為user model。<<Include>>代表當原本事件發生後必觸發的另一個事件；<<Extend>>代表原本事件發生後未必觸發的事件，可能會因為時間的關係不同未被觸發。  本專題主要功能是要讓玩家能遊玩一般的井字遊戲與進階版的終極遊戲，所以主要分支有兩個，而這兩種的功能性質相似，差別在於規則上的不同而已。 |

1. class model

|  |
| --- |
| **（圖三）** |
| ↑上圖是以程式設計的角度所設計的UML，主要分成兩個區塊，分別為UI設計與遊戲設計，其UML除了表達了每個封裝間繼承與多型的關係外，也表示了類別之間的關係是強還是弱。 |

1. 封裝、繼承、多型

|  |  |
| --- | --- |
| **（圖四）** | |
| **（圖五）** | **（圖六）** |
| **繼承：**圖四、五、六都包含了繼承，圖四的部分每個頁面都有相同的共通點，需要一個畫面去做負載，因此他們都有共同的父類別叫做UIcomponent，每個網頁都能用UIcomponent定義自己的功能。圖五是對基礎版與進階版的井字遊戲取共通點作父類別，稱為Game，兩者模式共有的功能。圖六的Basic4U繼承了Basic是為了include到進階的Ultimate，其功用在重點程式碼上說明。  **多型：**圖四、五的部分包含了多形，利用抽象類別來呈現多形，圖四為每個動作都需要頁面，但頁面的內容都不一樣，所以創個UIcomponent作抽象類別，每個子類別再去個別實現頁面的運作。  圖五是將Game作為抽象類別，因為基礎版與進階版的同個功能都差不多，但判斷條件有所不同，所以寫個Game做抽象類別建立個功能，其功能怎麼寫由基礎版與進階版自己決定。 | |

1. 關係

Aggregation

|  |
| --- |
| **（圖七）** |
| **（圖八）** |
| ↑Aggregation代表單向的關係，一個類別需要建立在其他類別上才有意義。如圖七每個頁面都需要透過UImanager才能顯示出完整的一套UI出來，每個UI獨立並沒意義。在圖八的部分，button需要透過頁面的實現才有意義，不然單獨的button並沒辦法實現任何事情。 |

Composition

|  |
| --- |
| **（圖九）** |
| **（圖十）** |
| ↑Compostion代表雙向的關係，彼此間為互相依賴的關係，如圖九UIcomponent與Game的關係，雖然各自能獨立運作，但UI沒有game就沒有存在UI的必要；有gameˊ有UI，game無法給user進行遊玩，因此兩者是雙向關係。 |

1. **重點程式碼說明**
2. OOP
3. 封裝

|  |  |
| --- | --- |
| **（圖十一）** | **（圖十二）** |
| ↑上圖是本專題的每個封裝，每個hpp都是一個封裝，裡面都有一個class，主要分為UI跟game兩塊。 | |

1. 繼承

|  |  |
| --- | --- |
| **（圖十三）** | **（圖十四）** |
| **（圖十五）** | |
| 上圖為每個封裝間的繼承關係，分別有UIcomponent、Game、Basic這三個父類別，照著UML的規劃分別給各自的子類別作繼承。 | |

1. 多形

|  |  |
| --- | --- |
| **（圖十六）** | **（圖十七）** |
| 上面圖十六、十七為多型，以抽象類別的方式呈現，父類別給了virtual function後，子類別在自己去實現各自的virtual function。 | |

1. 關係表現
2. Aggregation

|  |  |
| --- | --- |
| **（圖十八）** | ←左圖為UIManager去實現每個頁面，結合起來去做控制。除了UIManager外，每個頁面都包含了button，因此button也結合了各個頁面，每個頁面都能使用button去實作功能。 |

1. Composition

|  |  |
| --- | --- |
| **（圖十九）** | **（圖二十）** |
| 圖十九是Ultimate去引用Basic4U去實現功能，雖然沒有繼承間的關係，但沒有Basic4U，Ultimate無法被實做出來。圖二實同理，Game沒UIComponent的話Game無法給使用者遊玩。 | |

1. 重點程式碼說明

|  |  |
| --- | --- |
| **（圖二十一）** | 在UImanerger，存了很多UI介面，要怎麼通知換畫面呢? |
| **（圖二十二）** | 在UIManager currentScreen 存現在的螢幕狀態。 而進階版跟基本版都用同一個GameInterface，這樣才能凸顯動態連結的多型多好玩。他們的render要傳入currentScreen，確保要選染哪種遊戲，而且回傳整個遊戲的動態指標，讓GameEndScreen渲染遊戲最後的結果。 |
| **（圖二十三）** | 進階版以及基本版都用一樣的渲染框架，很好的表示多型的玩法。 |
| **（圖二十四）** | Basic4U唸法是 Basic for Ultimate，繼承basic增加回傳被按九宮格的哪個位子，使用tuple傳回Ultimate，以實現在大棋盤的對應位子下OX。 還有要增加現在換誰，現在換誰是basic本來就會控制的，會影響渲染的OX順序，但是主導權要在Ultimate，所以有一個setcurrentplayer |

1. **組員工作分配**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **錢昱名** | **劉沛辰** | **劉柏均** | **黃羿禎** | **孫健淳** |
| 遊戲主程式 | 70% | 10% | 10% | 5% | 5% |
| UI designer | 20% | 50% | 15% | 10% | 5% |
| UML | 10% | 20% | 5% | 35% | 30% |
| PPT | 10% | 10% | 20% | 30% | 30% |
| 專題發想 | 20% | 20% | 20% | 20% | 20% |

1. **組員心得**

**錢昱名：**

我來補充一點技術細節，我在game的多型使用了智慧指標，因為如果要維持可讀性，在哪裡new一個指標就要在哪裡delete，但是我們會把game指標傳入endscreem，直接使用game的render顯示最終結果，使用智慧指標就不用管了，哈哈哈哈。

**劉沛辰：**

這次做專題，從發想主題，討論了很久該如何實現封裝繼承多型，確定主題，畫usercase，站在使用者角度設計按鍵，設計UI、顏色、區塊，然後再畫uml，設計封裝多形繼承，再到實現uml，程式碼。

過程遇到許多困難，SFML的環境建構，git共同協作，github的branch推動，再來到SFML實作UI，多形繼承UI Component，PlayerO X實作，check的繼承，把上課學到的理論變成實作才讓我更熟悉OOP

**黃羿禎：**

在這次的專題中，我從同儕之間學到了更多關於程式的知識，像是如何使用git等，也第一次從頭與別人協同合作撰寫程式碼，除了更加熟悉OOP三大特性外，也學會了該如何與他人合作、溝通。

在過程中我覺得最為困難的點是學會如何以物件導向程式語言的思維去撰寫程式碼，以往都是撰寫程序性程式語言為主，所以一開始要開始理解、撰寫程式碼非常困難，在這部分我非常感謝並佩服我們組長--企鵝，程式碼幾乎都是由他所完成的，透過觀摩他的程式碼才讓我更加對物件導向語言有更加清晰的了解，漸漸熟悉撰寫物件導向程式語言。

**孫健淳：**

在這次專題中，除了懂得該如何應用物件導向思維的方式（利用封裝，類別關係的方式去思考如何組織程式碼）去設計程式專案外，也從其他隊友中學習到之前從未踏過的領域，以下是我這次從專題所學到的其他領域。

1. github整合程式碼

首先是為了整合大家的程式，避免大家在資訊上的不對稱，造成程式碼重複修改，不同步的情形，因此我們使用git來解決這個問題。這也是我第一次對git有了具體的概念。這將成為我日後撰寫程式的一大利器。

1. SMLF與chatgpt的應用

SMFL是專門提供圖形化介面的函示庫，這是大家第一次接觸到SMFL的函示庫，所以對裡面擁有的function並不熟悉。但在與chatgpt的配合下，算是邊用邊學，大致知道裡面的函示庫該如何使用。這也讓我意識到之後AI對於程式設計的重要性，該如何有效的對AI下指令是我必須學會的課題。

**劉柏均：**

這次的期末專題，我們選擇了井字遊戲，但因為不是只有簡單的井字遊戲，我們還加入了一些特殊的規則，我們找到了SFML來開發我們的期末專題，我主要是做UI設計，除了物件導向以外，我還學到了如何讓整個UI看起來比較順眼，還有學到到了如何使用繪圖軟體，在未來跨領域結合受益良多。