前．大學四年的時間非常短，幾乎沒有什麼時間失敗，請閱讀此文件的人以戒慎恐懼的心情閱讀及修改，若有修改一定要討論並更新，另外每天最晚12:10一定要睡。

遊戲設計規範

序論．

經研究發現程式設計師的經驗與程式碼品質或生產力一點關係也沒有，不會因為你沒有經驗而導至品質低於平均，影響品質的原因常是心態問題。

在正式的程式碼視察可找到約60%的缺失。

在模型或原型可找到約65%的缺失。

在單元測試可找到約30%的缺失。

在整合測試可找到約35%的缺失。

程式碼個人桌上檢查可找到約35%的缺失。

人工視察找到錯誤的效率為測試的20倍，平均一小時的視察可省去平均33小時的維護工作，所以請多看他人的程式碼並近可能的雙人組程式設計。

第一個可玩版後必需每日建置與煙霧測試。

建議 去買一本CODE COMPLETE 2(中譯：軟體建構之道第二版)來讀，約30個小時可讀完一遍。

主要使用語言：C++ native

程式平台(Win32)：C++, directX9, OpenGL 2.0, Windows XP, Vista, [masm32]

Not (.net, mfc)

建議設備：GeForce 7400M RAM:1G OS: Windows XP, Vista  
建議軟體及IDE

．使用Visual Studio 2008 來作為程式開發工具，編譯器設定開始所有警告。

．使用poseidonPE-6.02來設計UML

．使用TortoiseSVN來作為程式碼版本管理的工具

．使用lua用(Notepad++來編輯)來作為描述介面與人物的腳本語言

．使用WinAsm 5.15 來作為編寫masm32的工具

．使用虛擬碼程式設計流程(PPP,Pseudocode Programming Process)來撰寫程式。

．使用Office 2001 Groove軟體作為專案部份軟體分享  
．網路上的Google協作平台用於宣傳當免空用= =

規定功能：每一個類別一定要有下列功能，

一．程式人員簽名加修改日期 二．單鍵測試 三．版本控制 四．說明注解   
五．UML描述圖 六．無警告(要提交的類別一定不能有編譯警告)

類別產生的方法最好是先在poseidonPE-6.02把UML畫好再一次產出來。

(原因：版本控制不消說，很重要，程式人員簽名代表你有改過這份文件了，有說明文件才知道這個類別的功能與如何正確使用，有UML描述圖可以大概了解這個類別的屬性與方便整合，有單鍵測試才知道這個類別能不能在可預測的情況下不出錯)  
此專案以可讀性為第一來編寫，所有參與的程式人員所寫程式為共同擁有。

建議命名慣例：  
(XXXX、YYYY、ZZZZ是名字)  
  
對基本型別使用匈牙利命名法的前置詞

int=i long=ld float=f double=lf char=c String=sz char\*=sz boolean=b   
  
ClassName  
類別名稱混合使用大小寫，且第一個字母大寫。  
．一般類別 XxxxYyyyZzzz  
  
StructName.adjustMember

結構名同類別，成員大小寫混合，第一個字小寫。  
．Struct Xxxx.yyyyZzzz  
  
TypeName  
型別定義(包含列舉型別和typedefs)混合使用大小寫，同類別。  
．Typedef XxxxYyyy

EnumeratedTypes  
除上述規則之外，列舉型別永遠以複數型式陳述。  
．列舉型別 XxxxYyyys

localVariable  
區域變數混合使用大小寫，且第一個字母小寫。該名稱應與基礎資料型別無關，並且應與變數表示的一切有關。  
．一般變數 xxxxYyyyZzzz

\_routineParameter  
常式參數的格式與區域變數相同。

．傳入參數 \_xxxxYyyyZzzz

RoutineName  
常式混合使用大小寫。

．一般函數 XxxxYyyyZzzz()

m\_ClassVariable  
可供類別內多個常式使用的成員變數(但僅限於類別內) 前置詞為：  
int=m\_i long=m\_ld float=m\_f double=m\_lf String=m\_sz

char\*= m\_sz boolean= m\_b

g\_GlobalVariable  
全域變數以g\_作為字首。

．全域變數 g\_XxxxYyyyZzzz   
．全域類別 gc\_XxxxYyyy  
  
u\_UnionName  
聯集以u\_作為字首。  
．Union u\_Xxxx  
  
CONSTANT  
具名常數為ALL\_CAPS

．常數 XXXX\_YYYY\_ZZZZ

MACRO  
巨集為ALL\_CAPS

Base\_EnumeratedType  
列舉型別以單數陳述之基底型別的助憶鍵作為字首一例如：Color\_Red,Color\_Blue  
．列舉 Xxxx.YyyyZzzz or XXXX\_YYYY  
  
generic\_function

全部小寫，以底線相隔。  
．泛型函數 xxxx\_yyyy\_zzzz()

以下「一．」為第一可玩版的要求。

先製作含遊戲主要賣點的簡單第一版，主要用來招人與吸金= =，及說明遊戲策劃。

軟體與遊戲品質特性討論

正確性：系統規格、設計及執行未發生任何錯誤的程度。

遊戲程式不會出現錯誤的行為。

一．允許少量非關鍵錯誤。

實用性：使用者能否輕鬆學習與使用系統。

遊戲是否容易上手。

一．介面簡單美觀即可。

效率：使用最少的系統資源，包括記憶體和執行時間。

遊戲是否跑的順，記憶體又用的少。

一．不要太lag即可。

可靠性：系統能在需要時依陳述條件執行期所需的功能-意指發生錯誤的間隔。

遊戲是否能連玩8小不會有bug。

一．一小時內不要當機即可。

完整性：系統防止未授權或不正當存取程式及資料的程度。完整性的概念亦包括限制未授權的使用者存取，並確保資料正確存取，亦即含平行資料的表格以平行的方式修改，日期欄位內僅包含有效日期等。

遊戲是否防外掛。

一．不實作。

可適性：在未修改的情況下，系統在非專屬應用程式或環境下的執行能力。

遊戲是否跨平台。

一．不考慮。

準確性：系統未發生錯誤的機率，特別是關於量化輸出的作業。準確性與正確性不同，係指系統依建立目的執行工作的效率，而非是否正確執行工作。

載入遊戲是否不會太慢，遊戲是否不會lag。

一．不要太lag即可。

強固性：系統出現無效輸入或嚴苛環境條件時持續運作的能力。

遊戲是否能連玩8小不會當機。

一．一小時內不要當機即可。

維護性：您是曾能輕鬆修改軟體系統變更或新增功能、增進效能或條正缺失。

遊戲可否輕鬆更新。

一．人物腳本素質可執行期調整即可。

彈性：修改系統，使系統能於非專屬環境下運作的程度。

在linux下跑的是不是跟windows一樣順。

一．不考慮。

可攜性：是曾能輕鬆修改系統，使系統能於非專屬環境下運作。

能否在xbox360上跑的很順。

一．不考慮。

重新使用性：在其它系統中運用此系統元件的程度，以及是否能輕鬆運用。

可否將此開發的部份類別用在別的遊戲。

一．以功能獨立模組為方向。

可讀性：是否能輕鬆讀取及瞭解系統的來源碼，尤其是關於細的陳述式層。

測試性：進行單元測試及系統測試的能力，驗證此系統是否符合需求的能力。

每個類別與子系統可否單鍵測試，是否有replay機制重現錯誤。

一．可單鍵測試，有replay機制。

可了解性：是否能輕鬆瞭解系統的組織與陳述式。可了解性與系統的一般關聯比可讀性更大。

一．盡量。

遊戲α版要求

正確性：系統規格、設計及執行未發生任何錯誤的程度。

遊戲程式不會出現錯誤的行為。

α．不允許錯誤。

實用性：使用者能否輕鬆學習與使用系統。

遊戲是否容易上手。

α．遊戲玩的最少的人決定。

效率：使用最少的系統資源，包括記憶體和執行時間。

遊戲是否跑的順，記憶體又用的少。

α．記憶體不要用超過512MB、遊戲不可以LAG。

可靠性：系統能在需要時依陳述條件執行期所需的功能-意指發生錯誤的間隔。

遊戲是否能連玩8小不會有bug。

α．同上。

完整性：系統防止未授權或不正當存取程式及資料的程度。完整性的概念亦包括限制未授權的使用者存取，並確保資料正確存取，亦即含平行資料的表格以平行的方式修改，日期欄位內僅包含有效日期等。

遊戲是否防外掛。

α．請學長技術指導。

可適性：在未修改的情況下，系統在非專屬應用程式或環境下的執行能力。

遊戲是否跨平台。

α．不考慮。

準確性：系統未發生錯誤的機率，特別是關於量化輸出的作業。準確性與正確性不同，係指系統依建立目的執行工作的效率，而非是否正確執行工作。

載入遊戲是否不會太慢，遊戲是否不會lag。

α．RP不會跟戰鬥不一樣。

強固性：系統出現無效輸入或嚴苛環境條件時持續運作的能力。

遊戲是否能連玩8小不會當機。

α．同上。

維護性：您是曾能輕鬆修改軟體系統變更或新增功能、增進效能或條正缺失。

遊戲可否輕鬆更新。

α．同上。

彈性：修改系統，使系統能於非專屬環境下運作的程度。

在linux下跑的是不是跟windows一樣順。

α．不考慮。

可攜性：是曾能輕鬆修改系統，使系統能於非專屬環境下運作。

能否在xbox360上跑的很順。

α．不考慮。

重新使用性：在其它系統中運用此系統元件的程度，以及是否能輕鬆運用。

可否將此開發的部份類別用在別的遊戲。

α．以功能獨立模組為方向。

可讀性：是否能輕鬆讀取及瞭解系統的來源碼，尤其是關於細的陳述式層。

測試性：進行單元測試及系統測試的能力，驗證此系統是否符合需求的能力。

每個類別與子系統可否單鍵測試，是否有replay機制重現錯誤。

α．可單鍵測試，有replay機制。

可了解性：是否能輕鬆瞭解系統的組織與陳述式。可了解性與系統的一般關聯比可讀性更大。

α．同上。

遊戲β版要求，只允許素質效調，腳本修改，AI強化等非關遊戲架構的調整。

遊戲第一發行版，依BUG修正，視歡迎度更新。