

輔仁大學資訊工程學系  
畢業專題報告  
B03 組

---

# Rish Engine - 2D 遊戲引擎

---

## 從零開始自幹遊戲引擎

### 成員

406262515 鍾秉桓 me@roy4801.tw  
406262084 梁博全 me@icejj.tw  
406262319 黃育皓 suntalk1224@gmail.com  
406262163 黃品翰 william31212@gmail.com

報告編號: CS109-PR-B03

指導教授  
鄭進和

December 8, 2020

# Contents

|          |                                   |          |
|----------|-----------------------------------|----------|
| <b>1</b> | <b>摘要</b>                         | <b>1</b> |
| 1.1      | 背景簡介 . . . . .                    | 1        |
| 1.2      | 問題說明 . . . . .                    | 1        |
| 1.3      | 實作結果 . . . . .                    | 1        |
| <b>2</b> | <b>緒論</b>                         | <b>2</b> |
| 2.1      | 動機 . . . . .                      | 2        |
| 2.2      | 問題描述 . . . . .                    | 2        |
| 2.2.1    | 歷史介紹 . . . . .                    | 2        |
| 2.2.2    | 遊戲是什麼 . . . . .                   | 2        |
| 2.2.3    | 遊戲引擎是什麼 . . . . .                 | 2        |
| 2.2.4    | 遊戲引擎具備的功能 . . . . .               | 3        |
| <b>3</b> | <b>系統說明</b>                       | <b>5</b> |
| 3.1      | 使用技術 . . . . .                    | 5        |
| 3.1.1    | 使用語言 . . . . .                    | 5        |
| <b>4</b> | <b>系統實作結果</b>                     | <b>7</b> |
| 4.1      | 引擎介紹 . . . . .                    | 7        |
| 4.1.1    | 架構 . . . . .                      | 7        |
| 4.1.2    | Entity Component System . . . . . | 7        |
| 4.1.3    | Batch Rendering . . . . .         | 7        |
| 4.1.4    | Particle System . . . . .         | 8        |
| 4.1.5    | 2D Lighting . . . . .             | 8        |
| 4.1.6    | Scriptable Entity . . . . .       | 8        |
| 4.1.7    | Physics Engine . . . . .          | 8        |
| 4.2      | 編輯器介紹 . . . . .                   | 8        |
| <b>5</b> | <b>結語</b>                         | <b>9</b> |
| 5.1      | 鍾秉桓 Roy . . . . .                 | 9        |
| 5.2      | 梁博全 ICEJJ . . . . .               | 10       |
| 5.3      | 黃育皓 SunTalk . . . . .             | 10       |
| 5.4      | 黃品翰 Halloworld . . . . .          | 11       |

# Chapter 1

## 摘要

### 1.1 背景簡介

本專題是以 C++ 開發之 2D 遊戲引擎，開發者可以使用引擎提供之編輯器 (RishEditor) 編輯遊戲場景 (Scene)，並且可以對遊戲物件 (Entity) 附加遊戲邏輯 (使用 C++ 撰寫，並和編輯器一同編譯)，並支援 Batch Rendering <sup>1</sup>、2D Lighting (Point Light, Ambient Light), Particle System, Constrain-based Physics (支援圓形、多邊形等)

### 1.2 問題說明

TODO

### 1.3 實作結果

TODO 本專題製作解決問題方法、創新所在、與實作結果等概要陳述

---

<sup>1</sup>一種 Rendering 技巧，可支援同屏幕高達 100000 個 sprites

# Chapter 2

## 緒論

### 2.1 動機

隨著遊戲的發展，現代的遊戲越來越趨於複雜，從數人的小型獨立製作，到數百人的大型 3A 級遊戲，遊戲的規模與以前不可同日而語，現今一個獨立開發工作室製作的 2D 遊戲，在十多年前要達到同樣的規模可能要數十人的團隊才可能達到，而這些節省下來的時間成本，就是多虧了遊戲引擎的強大之處。現代遊戲引擎的複雜度，先進的圖學技術，複雜的物理模擬，是需要一個具有規模的團隊來開發的，我們嘗試以此為目標，試著實作了一個具有一定規模的 2D 遊戲引擎。

### 2.2 問題描述

#### 2.2.1 歷史介紹

最初並沒有所謂的遊戲引擎，遊戲常常是重頭開始建構 (from scratch) 的，這個概念被眾人所知最早是由 Jhon Carmack 發揚光大，他為 Doom 以及 Quake 系列遊戲開發的 3D 遊戲引擎，深深地影響了遊戲業界，為早期 (1990 末) 的遊戲業界標準，並促成引擎授權的商業模式，使得遊戲軟體的規模上升。

#### 2.2.2 遊戲是什麼

電腦遊戲的技術本質是實時 (real-time) 可交互 (interactive) 的程式，遊戲程式會模擬出不精確但足以表現的遊戲世界，並且根據玩家的輸入做出相對的輸出 (例如操作搖桿控制角色等)，通常遊戲會實作遊戲循環 (Game Loop)，更新遊戲邏輯、物理模擬、更新 AI 等。而通常遊戲要維持在每秒更新 60 幀才能保證流暢運行 (低於 60 fps 通常會感覺卡頓)，也就是要在 16 毫秒內做完所有的遊戲更新並渲染到螢幕上頭，更何況在 VR 上要維持 90 FPS 才不會感覺暈眩，這也是為甚麼遊戲會如此要求效能。

#### 2.2.3 遊戲引擎是什麼

遊戲引擎 (Game Engine) 從字面上解釋是驅動遊戲的基礎程式，因此遊戲引擎須具備了窗口管理、輸出入管理、渲染 (Rendering) 系統、物理 (Physics) 系統…等子系統，

除了基礎程式之外，要是一個遊戲引擎還必須要有讓遊戲開發者使用之工具 (可視化、非可視化)，可能是單個工具或一整個工具鏈 (Toolchain)，使得遊戲開發者可以用該工具開發遊戲 (Developing)、進行測試 (Debugging)、封裝發布 (Shipping) 等，否則就只能被稱為遊戲框架 (Game Framework)。遊戲引擎會讀入自訂的資源 (Asset) 格式<sup>1</sup>，並且有工具支援設計師將素材 (貼圖、音效、模型等) 轉換成引擎的格式，或是引擎工具可以直接產生，因此遊戲引擎可以說是專門開發遊戲的開發環境。

#### 2.2.4 遊戲引擎具備的功能

TODO

---

<sup>1</sup>常常是為了效能才會這樣做

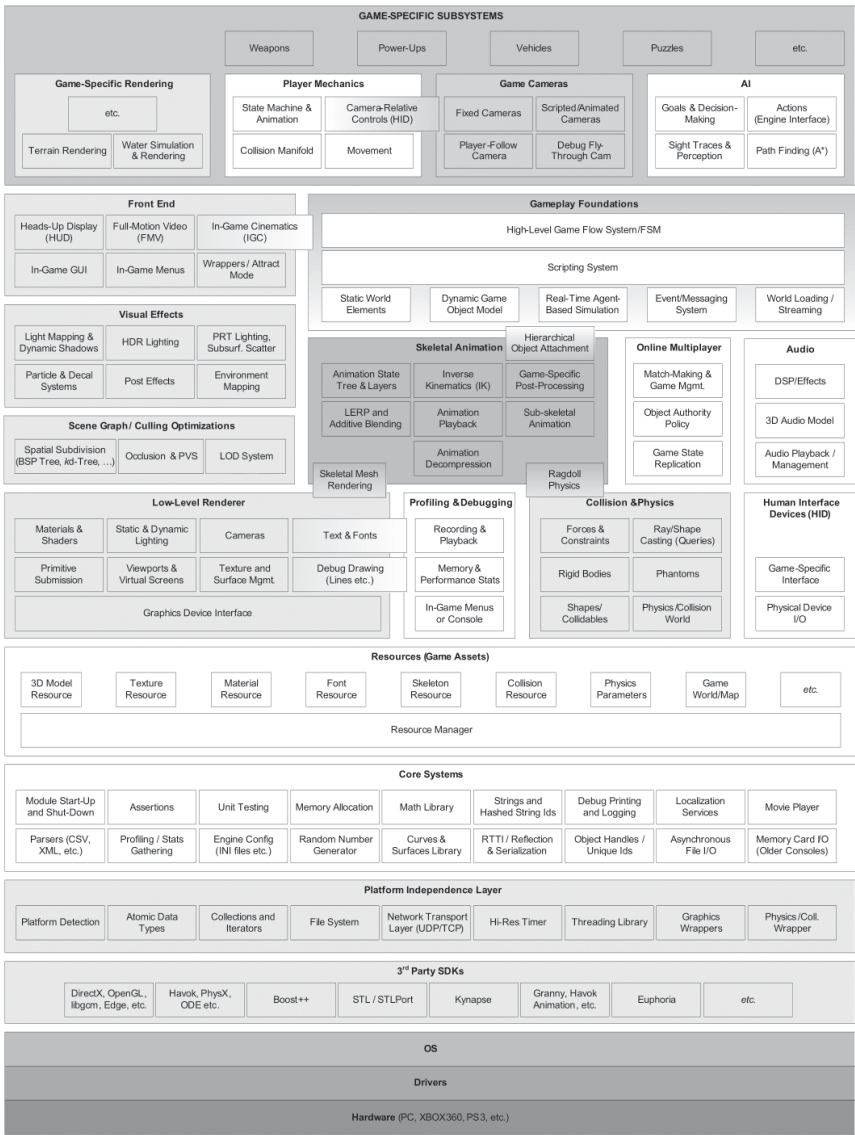


Figure 1.II. Runtime game engine architecture.

# Chapter 3

## 系統說明

### 3.1 使用技術

#### 3.1.1 使用語言

使用 C++ 11 作為主要開發語言，使用 `msys2` 作為套件管理器<sup>1</sup>，`CMake` 建置專案，使用 `doxygen` 工具將註解轉換 documentation, `cppcheck` C++ 的靜態程式碼分析器

- Slack

團隊協作軟體，用於溝通紀錄

- Trello

看板軟體，用於工作進度管理

- git, github

版本控制

TODO 寫多一點

#### 為何選 C++ 做為開發引擎之語言？

因為 C++ 讓開發者可以自由掌控記憶體，讓開發者可以精確的控制變數的生命週期，沒有垃圾回收 (Garbage Collection)，這對於效能注重的遊戲引擎來說相當重要；比 C 來得高階但效能卻沒損失很多；C++ 歷史悠久且有許多精良的函式庫可以使用。

[1]

#### 使用函式庫

- SFML

一個 C++ 的跨平台用於遊戲、多媒體程式開發的函式庫，於此專案中用於 Input 及窗口操作等

---

<sup>1</sup>在 Windows 上，沒有套件管理器會非常痛苦

- [OpenGL + glad](#)  
一個跨平台的 API，使用 glad 載入器
- [imgui](#)  
一個輕量、快速的 Immediate Mode GUI 函式庫，常在遊戲開發、工具開發等使用，於此專案中用於編輯器的 UI 開發。
- [spdlog](#)  
一個輕量、快速的 Logging 函式庫，於此專案中用於處理除錯訊息。
- [nativefiledialog](#)  
小型的 Open File Dialog 函式庫，於此專案中用於處理開啟檔案視窗。
- [IconFontCppHeaders](#)  
字體的 Helper Headers，於此專案中用於引入自訂字體。
- [fmt](#)  
補足了 C++ 標準庫缺少的格式化輸出輸入。
- [glm](#)  
OpenGL 數學函式庫，於此專案中用於處理向量、投影等相關數學函式。
- [cereal](#)  
C++ 缺少的序列化函式庫，於此專案中用於序列化儲存資源。
- [re2](#)  
來自 Google 的 Regular Expression 函式庫，標準庫的有夠慢 (確信)。
-



# Chapter 4

## 系統實作結果

### 4.1 引擎介紹

#### 4.1.1 架構

流程圖

架構圖

TODO

#### 4.1.2 Entity Component System

TODO

#### 4.1.3 Batch Rendering

傳統上 OpenGL 要畫圖形到螢幕上時，會建立 VAO(Vertex Array Object) 用來儲存 VBO (Vertex Buffer Object) 和 IBO (Index Buffer Object)，其中 VBO 儲存圖形的點座標、IBO 儲存 index 座標，最後用 `glDrawElements()` 讓 OpenGL 將圖形經過 Rendering Pipeline 畫在 Target 上 (通常是螢幕或是 Framebuffer)。

---

```
ializeRenderer();  
  
/* 場景的所有物體 */  
  
Render();
```

---

但是遊戲引擎在畫場景中的每個物體時，如果每個物體都要重新建構 VBO 和 IBO 時，則會耗費大量時間在從 CPU 傳輸資料到 GPU 中，同時也必須減少 OpenGL 的 State 變換，因此將渲染前先將多種圖形的頂點收集起來，再一次進行繪製，減少 Draw Call 進而增加可以繪製的圖形數量。

#### **4.1.4 Particle System**

TODO

#### **4.1.5 2D Lighting**

TODO

#### **4.1.6 Scriptable Entity**

TODO

#### **4.1.7 Physics Engine**

TODO

### **4.2 編輯器介紹**

TODO

# Chapter 5

## 結語

### 5.1 鍾秉桓 Roy

其實從高中初次接觸程式設計以來，我一直想做的事情便是寫一個自己的遊戲引擎，從高中時初次嘗試撰寫 ADV 類型的遊戲引擎 (但受限當時的程式能力，並沒有完成)，而到了大學時，我從很早 (大概大二) 便找好組員，並且也在系上課程的多個專案中不斷地練習和磨合我們之間的合作，我想做的事便是寫一個屬於自己的遊戲引擎。

從三年級下學期開始，經過了九個月的開發，坐在電腦前數百小時的奮鬥，我們 RISH 終於將 RishEngine 的功能告一個段落，儘管有些功能是因為專案時程以及目前能力而有缺憾，但即便如此我們還是完成了這個專案，我們開發了一款 2D 的遊戲引擎，具有 ECS、有引擎編輯器、有 Batch Rendering、有 2D Lighting、有物理引擎、有 Particle System 等。

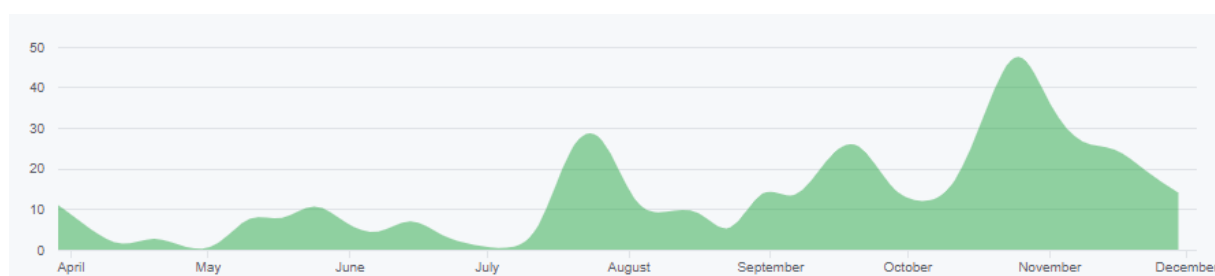


Figure 5.1: RishEngine Commit Graph

而在開發每項功能前，都花了一到兩個星期在做資料搜尋、自學，接著做出最小可行的 Demo，僅僅做出功能還不夠，因為我們做得是遊戲引擎，所以還得將 API 打磨，不斷得去想、擴充功能，因為引擎就是得提供許多功能給遊戲開發者，而在引擎開發階段時，每個功能寫出來之後，還得切換身分到遊戲開發者，試圖用自己做出的引擎來打造遊戲，透過這個過程來除錯和發想可以增加的新功能，一直不斷的迭代，而這個過程是相當漫長和痛苦的，常常在和組員討論時一個功能的規格 (spec) 時，往往最後都覺得很迷惘，總是覺得這個功能不夠好，但還是得在現實與理想中取捨。

在這次的專案中我也學到了很多東西，像是如何有效與人溝通和合作，表達自己的想法給對方清楚理解，當然過程中勢必有磨合的時期，因為我們是只有四個人的小組，

所以作為組長的我理所當然會處理許多工作，也必須擔當起管理的責任，畢竟繁雜的工作量由一個人扛起也過於不切實際，清楚了解到除了程式之外管理和與人相處溝通之道也相當重要。

這個畢業專題算是完成了我其中的一個心願，自幹一個遊戲引擎，也是我目前處理過最多行的程式，算是我的一個小小的里程碑。

## 5.2 梁博全 ICEJJ

會開始接觸程式，是因為自己很喜歡玩遊戲，而自己也想開發遊戲，所以想當個遊戲開發者，但隨著深入學習這塊領域，發現市面上遊戲引擎已經很成熟，開發遊戲門檻降低，很多人都能用熱門引擎寫出遊戲。作為開發者，我們能用現有引擎輕鬆地拉出一款遊戲，我們從中學習到的僅僅只是引擎的使用方法。若未來需要對引擎底層進行優化，但對底層的架構、邏輯都不了解，那要如何修改如何優化。開發遊戲引擎不只能讓我們更了解遊戲架構，也能更加瞭解一款遊戲的圖形渲染、物理模擬等等是如何運作，我想這會是在眾多遊戲開發者內脫穎而出的關鍵。

在開發引擎功能的，基本上都是從無開始自學，每當被分派到一個模塊時，基本上就是花大量的時間找資料，接著花大量的時間寫 Demo，寫 Demo 的期間還會遇到各種不同的問題，而做完因為是需要將模塊打入引擎裡面，因此還得將 API 想好，整個架構也得配合著引擎，最後在移植到引擎裡面之後，新的問題又會出現。

這次的專案除了深入了解整個遊戲架構、更加熟悉 C++，也知道溝通的重要性。初期常常因為溝通不清楚而導致東西做出來不合預期，因此在被分發工作時，將內容以及目標講清楚，中途遇到問題發現功能無法做出來或是需要更改 API 也需要及時提出，避免最後浪費太多時間。

雖然引擎與一開始預想的有差，但能做出這些算是不錯了，自己也很開心。

## 5.3 黃育皓 SunTalk

在剛開始接觸程式時，最直覺的就是想寫遊戲，想將腦中所想的變成一款真實的遊戲，並為此打磨自己的能力。在學習的過程中，試著寫了幾個遊戲後，發覺自己所寫出來的與市面上所存在的遊戲都有很大的差異，才發現說現如今已經不是純粹靠著自己 coding 來寫遊戲了，大多數的遊戲開發者都是使用已經打磨好的遊戲引擎來進行開發，藉由完善的遊戲引擎來解決繁瑣的底層架構，大大的提升了遊戲開發的效率與效能。也因此讓我們從想寫個遊戲轉變成想試著寫個遊戲引擎出來。

在開發的過程中，由於每個人所負責的模塊都不同，我覺得最主要的是要清楚的理解引擎中的架構，不僅僅是整體的架構，其內每個模塊的架構都要有一定程度的理解，在這的基礎上去進行編輯器的撰寫與整理，在編輯器中除了設計出讓使用者方便使用的各種功能，亦要去正確的對照到引擎中的各個功能才能使得其正常運作。

在這次的專案中，我學習到了許多的事情，除了瞭解引擎與遊戲上的架構外，最重要的就是與隊友中的溝通了。像是在理解對方所寫出來的 code 時，只是看著程式碼並不能有效的理解到該區塊的意義，以及如何去使用，在經過有效的溝通後才可以快速的去諧調出 API 與協助 debug 的進行。

能夠以遊戲引擎這一個主題來作為畢業專題，並且做出這摸一個雛形，雖然還有很多可以新增與進步的空間，但也讓我感到相當的開心。

## 5.4 黃品翰 Halloworld

在整個遊戲開發的歷程，我們從一開始試著用現有的遊戲框架 (SFML)，每個人都試著實作一個簡單的遊戲。透過寫遊戲的這段過程，來構思我們的遊戲引擎該如何設計。當初想要寫遊戲引擎，一方面我自己覺得是前人沒有試著這樣做過，另一個方面是想說透過畢業專題的機會，詳細了解遊戲引擎的架構與內部的實作。我們也在網路上找了許多資源，同時也參考現有的遊戲引擎做為參考對象，想辦法設計出一套可以拿來寫遊戲的遊戲引擎。

在物理引擎的學習上，較為細節的研究歷程我都放在報告裡面，心得這邊我就純粹只分享我自己的感想。研究遊戲物理這塊，其實真的很少人在做，中文文獻也是少得可憐，大部分人以及其他遊戲相關課程，通常也只注重在使用函式庫。在最早開始，我只拿到一份真正成功將 2D 物理引擎實作的範例，但作者寫的註解也是少得可憐，於是我花了不少個禮拜，在研究他的程式碼在寫些甚麼。研究的路程中，我後來發現在研究的路上並不是只有我一個人，感謝有來我個人網頁的來自中國大陸的網友，趁機的與他交換聯絡資訊，跟他來回書信不斷地順順看自己的邏輯，互相腦力激盪出當時撰寫這份物理引擎的想法。另外，我參加了學生計算機年會 (SITCON)，很高興我花我暑假的期間，參加了用遊戲物理自幹一條繩子的議程，議程中的剛好可以跟我學的內容呼應，同時也很厚臉皮的，跟他交換了聯絡方式。我想最值得的，就是這一步步都在突破我自己的舒適圈，透過各種不同的方法找尋資源，想辦法把自己心中的那個為甚麼一一地解開。

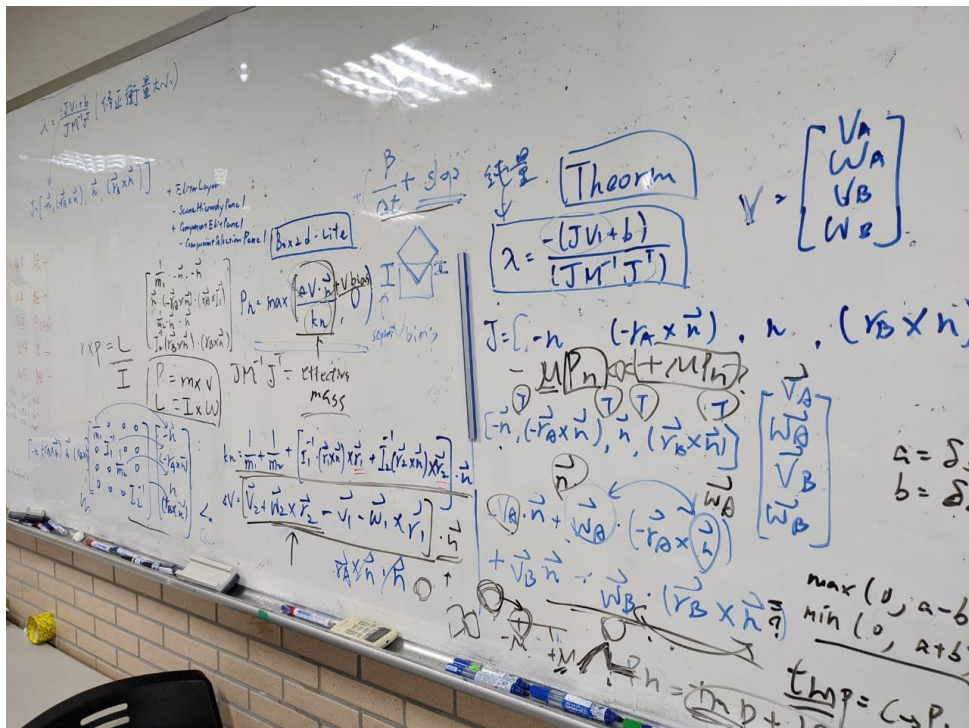


Figure 5.2: 被我寫滿物理證明的實驗室白板

除了與網路上無緣無故交到的筆友和議程上的講者，我也透過講者的關係下，拿到遊戲物理界大老的系列教學影片。那短短兩三個小時幾部的影片讓我真正了解到比較深層的理論。如果以程式語言的學習歷程來比喻，在前期自己看程式以及與網路筆友的討論，像是學習高階語言一樣，大概能知道淺層的物理原理，但是再透過那系列的教學影片之後，又知道了有些東西的係數是這樣推倒而來，就如同學習到組合語言一樣，能跟精確地知道整個程式運作的來龍去脈。除此之外，我也試著自己證明遊戲物理引擎的物理參數，看看真的是不是如程式寫得一樣，並自己在試著實作一次。此外，覺得我自己還想更精進的部分，是希望以後能夠自己實作出 3D 的遊戲引擎，以及實作空氣阻力、彈簧... 等更精細的物理模擬。

實作的過程中，很常被 C++ 的語法及語言特性坑殺。舉個最經典的例子，最常遇到的應該就是循環引入的問題，編譯器卻沒告訴你哪裡出了問題，導致我花了很多時間來找 bug。我自認為我在實作遊戲物理邏輯中的部分，跟解決 C++ 錯誤所花的時間比起來，修 C++ BUG 的時間真的佔了我大部分的時間。不過也從中學習到很多技巧。這些東西可能是課堂學不到的，大部分都得靠網路或是同學之間討論，才能夠把這些複雜又惱人的 BUG 解掉。此外，在整個專案當中，隨著時間增長的程式碼，漸漸地讓編譯速度變得相當緩慢，裡面的 Warning 也越來越多，這也是我很頭痛的部分。到專案後期，常常花的時間都不是在改程式碼，而是在等編譯。包括我現在寫心得的此時此刻，都是在等待的期間撰寫出來的。

另外，與團隊的交流也很重要，過程中有相當多的磨合以及很多的取捨，不過目的其實都是為了整個專案更好。比較愧疚的部分，是這次沒有參與到遊戲引擎整個開發的部分，引擎內部裡面的細節我沒有仔細的接觸過。主要因為光花那些時間證明物理以及實作就踩了許多坑也耗費了很多時間和冤枉路，加上我自己的資質驚鈍和我的英文能力，以及，真正研究過遊戲物理的人真的很少，資源也相對地難找，也就導致我沒有參與到整個引擎最主要的開發部分。最後，我想感謝我的組員們，雖然引擎不是最完美的，但我相信從開發過程應該都學到不少的東西以及經驗，也很感謝實驗室的學長和同學，陪我們度過一次又一次的 Bug。

# Bibliography

- [1] Valmond. *Why is C++ used for game engines? How about its future in game engines?* <https://gamedev.stackexchange.com/questions/38011/why-is-c-used-for-game-engines-how-about-its-future-in-game-engines>. Oct. 2012 (page 5).