

國立臺東大學

資訊工程學系

101 學年度資訊專題成果報告書

Marble Run 自己找出路

Android 三向水平軸感應器之應用

指導教授：王忍成老師

學生：林立昇(9811230)

李東昇(9811201)

郭維倫(9811271)

中華民國 101 年 12 月 30

一、 摘要

我們製作一款 2D 闖關俯視平台遊戲，其遊戲規則是要讓一顆球進入終點。玩家透過 Android 智慧型手機裡內部的三軸感應器來進行控制紅色的球體，要想辦法引導該球體進入為終點的高爾夫球洞。當玩家控制的球體一進入終點，即可得到分數，並且可以進一步的繼續玩下一個關卡。

除了玩家在引導球體用滾動的方式移動，玩家可以拿手機以抬起的動作向上動，可以讓球體進行跳躍的動作，並且可以在遇到障礙物時使用。我們是透過手機內部的三軸感應器，我們可以利用時間與位置的差異性來判斷並且算出是否該讓球體本身進行跳躍的動作。

關卡的設計是使用「小畫家」，以邊長平均為 10 pixel 高與寬的四方形作為圖片使用。製作關卡時，我們透過遊戲物件之 ID 編號作為圖片上代表物件的顏色進行分配位置，再以遊戲程式載入關卡讓程式自行展開關卡的圖片，而呈現出遊戲關卡真正的面貌。

遊戲音樂最主要是以 IT 和 XM 等等模塊文件作為音樂格式來播放音樂使用。這些模塊文件是以比較新的 OpenMPT 音樂編輯軟體，是利用取樣的波型與樂器作組合，再加以類似於 MIDI 等等音符進行編湊成音樂。其優點是檔案大小非常小，不會佔檔案空間。

以上是我們在製作遊戲的過程中，全程利用 Android API 來進行遊戲開發，製作出符合專題的遊戲產品。

關鍵字：Android、三軸感應器、OpenMPT、載入關卡圖片

二、 研究背景與動機

壹、 研究背景

一、 歷史 ^[L1]

2007 年 10 月，『Google 手機』即將出現的傳聞吵得沸沸揚揚，終於在 2007 年 11 月 5 日，Google 公布了 Android 手機開發平台。

原來 Google 並非要做手機，而是直接釋出了一個基於 Linux 的手機平台 Android。Android 平台的核心採用了 GPL v2 的授權，應用

部分則採用了 Apache Software License 授權。開放手機聯盟是一個由 65 間企業組成的商業聯盟，為了行動裝置開發自由標準。這 65 間企業有包括谷歌、HTC 和三星等設備製造商，無線運營商如 Sprint Nextel 和 T-Mobile 公司，高通和德州儀器等芯片製造商等等。

而在那一天，是由開放手機聯盟推出了第一款 Android 產品，其產品是一個建立在 Linux 內核 2.6 版本行動平台上。第一款可執行 Android 的商用手機為 HTC Dream，是在 2008 年 10 月 22 日發布。

自 2008 年以來，Android 有經過多次的更新、改進了作業系統，也在過去的作業系統上不斷地增加新的功能與修正錯誤。每一個完整版本是以甜品或舔點按照英文字母順序命名，例如，版本 1.5 是蛋糕，而其次的 1.6 版本是甜甜圈。最新的版本是果凍豆 4.2。

2010 年，谷歌推出智能手機和平板電腦執行 Android 作業系統的 Nexus 系列產品，由合作製造商製作。HTC 與谷歌進行合作，推出「Nexus One」，是第一款 Nexus 系列的智能手機。該 Nexus 系列產品不斷的更新硬體設備，如 Nexus 4 手機和 Nexus10 平板電腦，分別由 LG 和三星製造。

谷歌以發布的 Nexus 手機和平板電腦作為他們的旗艦級的 Android 產品，展示著 Android 最新版本的軟體和硬體功能。

這意味著手機產業可能會有所改變，目前已經有 65 家廠商參與 Android 的 Open Handset Alliance 聯盟。這是所有手機軟硬體開發者都應該仔細研究的一項新興技術。



圖 1：Open Handset Alliance 標誌。

二、 Google ^[L2]

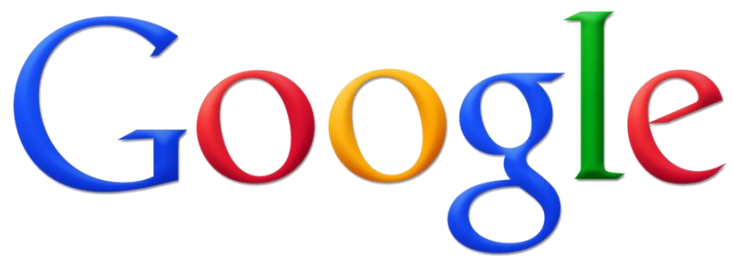


圖 2：Google 標誌。

Google 公司是一家美國的跨國科技企業，致力於網際網路搜尋、雲端運算、廣告技術等等領域。開發並提供大量基於網際網路的產品與服務，其主要利潤來自於 AdWords 等廣告服務。

Google 由當時在史丹佛大學攻讀理學博士的拉里·佩奇和謝爾蓋·布林共同建立，因此兩人也被稱為『Google Guys』。1998 年 9 月 4 日，Google 以私營公司的形式創立，設計並管理一個網際網路搜尋引擎「Google 搜尋」。Google 網站則於 1999 年下半年啟用。

2004 年 8 月 19 日，Google 公司的股票在納斯達克上市，後來被稱為「三駕馬車」的公司兩位共同創始人與出任執行長的埃里克·施密特在當時承諾：「共同在 Google 工作至少二十年，即至 2024 年止」。

創始之初，Google 官方的公司使命為「整合全球範圍的訊息，使人人皆可存取並從中受益」(to organize the world's information and make it universally accessible and useful)，而非正式的口號則為「不作惡」(Don't be evil)，由工程師阿米特·帕特爾 (Amit Patel) 所創，並得到了保羅·布赫海特 (Paul Buchheit) 的支援。

Google 公司的總部稱為『Googleplex』，位於美國加州聖塔克拉拉縣的山景城。2011 年 4 月，佩奇接替施密特擔任執行長。據估計，Google 在全世界的資料中心內運營著超過百萬台的伺服器，每天處理數以億計的搜尋請求和約二十四 PB 使用者生成的資料。



圖 3：Googleplex，Google 公司總部。

Google 自創立開始的快速成長同時也帶動了一系列的產品研發、併購事項與合作關聯，而不僅僅是公司核心的網路搜尋業務。Google 公司提供豐富的線上軟體服務，如雲端硬碟、Gmail 電子郵件，包括 Orkut、Google Buzz 以及最近的 Google+ 在內的社群網路服務。

Google 的產品同時也以應用軟體的形式進入使用者桌面，例如 Google Chrome 瀏覽器、Picasa 圖片整理與編輯軟體、Google Talk 即時通訊工具等。另外，Google 還進行了行動裝置的 Android 作業系統以及小筆電的 Google Chrome OS 作業系統的開發。

網站資訊分析網 Alexa 資料顯示，Google 的主域名 google.com 為全世界存取量最高的站點，除此之外，Google 搜尋在其他國家或地區域名下的多個站點 (google.co.in、google.de、google.com.hk 等等)，及旗下的 YouTube、Blogger、Orkut 等的存取量都在前一百名之內。

2007 年至 2010 年，Google 連續四年蟬聯 BrandZ 全球品牌價值榜首，但在 2011 年被蘋果公司超越。在市場競爭中處於領先地位的

現實也使 Google 公司在使用者隱私保護、版權、網路審查等等方面存在一些爭議。

三、Android^[L3]



圖 4：Android 標誌。

中文俗稱安卓是一個以 Linux 為基礎的半開放原始碼作業系統，主要用於行動設備，由 Google 成立的 Open Handset Alliance (OHA，開放手機聯盟) 持續領導與開發中。

Android 系統最初由安迪·魯賓 (Andy Rubin) 開發製作，最初主要支援手機，於 2005 年 8 月被美國科技企業 Google 收購。

2007 年 11 月，Google 與 84 家硬體製造商、軟體開發商及電信營運商成立開放手持設備聯盟來共同研發改良 Android 系統。隨後，Google 以 Apache 免費開源許可證的授權方式，發布了 Android 的源代碼。讓生產商推出搭載 Android 的智慧型手機，Android 作業系統後來更逐漸拓展到平板電腦及其他領域上。

Google 透過官方網上商店平台 Google Play，提供應用程式和遊戲供用戶下載，截止至 2012 年 6 月。Google Play 商店擁有超過 60 萬個官方認證應用程式，同時用戶亦可以通過第三方網站來下載。



圖 5：Google Play 標誌。

2010 年末數據顯示，僅正式推出兩年的 Android 作業系統在市場佔有率上已經超越稱霸逾十年的諾基亞 Symbian 系統，成為全球第一大智慧型手機作業系統。

四、Android 平台的架構

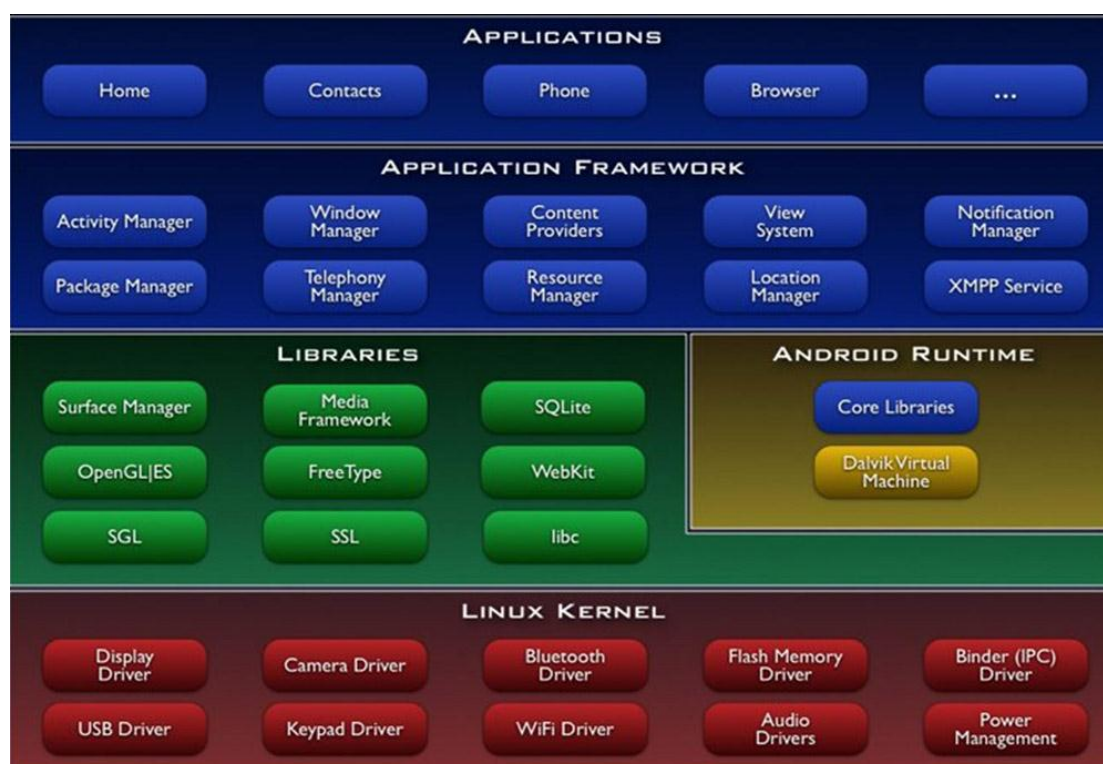


圖 6：Android 平台的整體架構。

在上圖（圖 6），裡面的藍色部分是 Android 平台的應用程式，這些應用程式以 Java 作為主要的開發語言。透過這樣的布局，Google 可以有效的引入 Java 社群長期累積的資源，以便利用 Java 程式設計社群的力量。

承上圖（圖 6）所顯示了 Android 平台的整體架構，平台的底層採用了 Linux 作為作業系統（圖 6 的紅色部分）。在 Linux 作業系統之下、內建了許多控制裝置的驅動程式，包含藍芽（Bluetooth）、無線網路（WiFi）、隨插即用的 USB 介面等等，當然還有記憶體、銀幕、鍵盤、照像、音效等裝置的驅動程式。

在 Linux 作業系統之上，內建了許多由 C/C++ 語言所開發出來

的函式庫 (Libraries) (圖 6 的綠色部份)，包含 libc、OpenGL/ES、WebKit、Sqlite 等。

雖然 Android 平台中大量的使用了開放原始碼的資源，但是為了避免軟體廠商被迫要開放原始碼，Android 在應用端所使用的是 Apache Software License (ASL)，ASL 並不要求軟體開發者要開放原始碼，這使得軟體開發廠商可以透過開發 Google 手機程式營利。

Google 在手機開發平台上的這種佈局是相當精巧而有彈性的。一方面藉助開放原始碼社群的力量，另一方面又可以吸引手機製造公司與軟體設計公司紛紛加入戰局，而不需要受到開放原始碼與平台授權金的限制。

在 Google 所提供的 Android 開發工具當中，附有模擬器環境，您可以在桌上型電腦中開發 Android 的 Java 程式。然後在上傳時 Google 工具會將 JVM 的 bytecodes 轉換成 DVM 的目的碼，然後才傳送到 Android 平台的目標裝置中執行。^[L4]

這使得 Google Phone - Android 平台具有相當大的吸引力。

五、 Android 模擬器與開發環境

下圖 (圖 7) 為 Android 模擬環境的畫面。這個模擬器可以模擬出一般智慧型手機的所有的硬體設備以及軟體功能，只差不能夠像手機一樣去接來電電話與撥出電話這些功能。

Android 模擬器有附上各個不同的瀏覽鍵與控制鍵，可以用滑鼠或者是鍵盤來產生出給予軟體使用的事件。如果要讓程式開發者更容易的進行測試應用程式，可以在 Android 虛擬平台 (AVD, Android Virtual Devices) 上調整設定。這些設定可以依據智慧型手機上特定硬體設備，也可以提供更多不同的排列組合設定在不同的 Android 平台做測試。

當程式開發者的應用程式已經在 Android 模擬器安裝成功之後，該應用程式就可以使用 Android 作業系統所提供的服務與其他應用程式、使用網路、播放媒體、儲存資料、通知使用者以及顯式圖案、背景和樣式等等。^[L5]



圖 7：Android 模擬器畫面。

宏達電 (HTC) 是第一家販售 Android 手機的公司，第一支 Android 手機由 HTC T-Mobile G1 拔得頭籌。^[L4]

目前，各家手機製造商紛紛投入製造 Android 手機，Motorola、Samsung、Acer 等等公司都已經投入 Android 手機的製造，並且發行了數款 Android 手機。甚至，資策會、宏碁等等公司更已經將 Android 放入迷你筆記型電腦當中，準備作為免費的作業系統。

Android 平台有可能是未來系統程式設計師的主要戰場之一，對於台灣的手機與筆電製造商而言應有相當大的吸引力。^[L4]

六、 Apple 蘋果公司 ^[L6]



圖 8：Apple 標誌。

Apple Inc.，原稱蘋果電腦（股份有限）公司（Apple Computer, Inc.），於 2007 年 1 月 9 日在舊金山 Macworld Expo 上宣佈改為現名。總部位於美國加利福尼亞庫比蒂諾，核心業務是電子科技產品，目前全球電腦市場佔有率為 7.96%。

蘋果的 Apple II 於 1970 年代助長了個人電腦革命，其後的 Macintosh 接力於 1980 年代持續發展。最知名的產品是其出品的 Apple II、Macintosh 電腦、iPod 音樂播放器、iTunes 商店、iPhone 手機和 iPad 平板電腦等。它在高科技企業中以創新而聞名。

在創立電腦前，創始人之一沃茲已經是一個電子學駭客，自 1975 年，他在惠普上班和幫斯帝夫·賈伯斯設計 Atari 電子遊戲。當時沃茲向由 Alex Kamradt 開設的分時電腦系統服務公司 Call Computer 租用小型電腦使用。

甲、蘋果的由來 [L6]

在史蒂夫·賈伯斯、史蒂夫·沃茲尼克和羅納德·韋恩三人決定成立公司時，賈伯斯正好從一次旅行回來，他向沃茲建議把公司命名為「蘋果電腦」。

最初的 LOGO 在 1976 年由創始人三人其中之一韋恩設計，只在生產 Apple I 時使用，為牛頓坐在蘋果樹下看書的鋼筆繪畫。在 1976 年由賈伯斯決定重新委託廣告設計，並配合 Apple II 的發行使用，本次 LOGO 確定使用了彩虹色、具有一個缺口的蘋果圖像。這個 LOGO 一直使用至 1998 年，在 iMac 發佈時作出修改，變更為雙色系列。2007 年再次變更為金屬帶有陰影的銀灰色，使用至今。

蘋果 LOGO 的來由多被誤解為「圖靈自殺時吃了一口的氰化物

溶液蘋果」。其來源為 2001 年的英國電影《Enigma》，在該部電影中虛構了前述有關圖靈自殺與蘋果公司 LOGO 關係的情節，被部分公眾以及媒體訛傳。

而蘋果 LOGO 的設計師在一次採訪中親自證實這個 LOGO 與圖靈（或者其他的猜測，比如，被夏娃咬的那個蘋果）無關。他甚至半開玩笑地說「被咬掉一口的設計只是為了讓它看起來不像櫻桃」。



圖 9：史蒂夫·賈伯斯，攝於 2010 年 6 月 8 日。

1971 年，16 歲的史蒂夫·賈伯斯和 21 歲的史蒂夫·沃茲尼克經由朋友介紹而結識。1976 年，賈伯斯成功說服沃茲組裝機器之後再拿去推銷，他們的另一位朋友羅納德·韋恩（Ronald Wayne）也加入，三人在 1976 年 4 月 1 日成立蘋果電腦公司。

蘋果電腦公司第一個產品被命名 Apple I。當時大多數的電腦沒有顯示器，Apple I 卻以電視機作為顯示器。對比起後來的顯示器，Apple I 的顯示功能只能緩慢地每秒顯示 60 字。此外，主機 ROM 包括了引導（Bootstrap）代碼，同時沃茲也設計了一個用於裝載和儲存程式的卡式磁帶介面，以 1200 位每秒的高速執行。雖然設計相當簡單，但它仍然是一件傑作，而且比其他同等級的主機需用的零件少，使沃茲贏得了設計大師的名譽。最終一共生產了 200 部。

乙、 Apple II -- 一款早期的蘋果電腦 ^[L6]

此役後，沃茲已成功設計出比 Apple I 更先進的 Apple II。1977 年 1 月，蘋果電腦公司正式註冊成為『蘋果電腦有限公司』。同年 4 月，Apple II 在首屆的西岸電腦展覽會（West Coast Computer Fair）首次面世。



圖 10：Apple II (在 Museum of the Moving Image 博物館裡拍攝)。

Apple II 與 Apple I 最大分別包括重新設計的顯示介面，把顯示處理核心整合到記憶體中，這有助於顯示簡單的文字，影像，甚至彩色顯示。而且有一個改良的外殼和鍵盤。Apple II 在電腦界被廣泛譽為締造家庭電腦市場的產品，到了 1980 年代已售出數百萬部。Apple II 家族產生了大量不同的型號，包括 Apple IIe 和 IIgs，這兩款電腦直到 1990 年代末仍能在許多學校見到。

當蘋果在 1980 年上市的時候，他們的資金比 1956 年福特上市以後任何首次公開發行股票的公司都要多，而且比任何歷史上的公司創造了更多的百萬富翁。在五年之內該公司就進入了世界公司五百強，這是當時最快的記錄。

丙、 iOS VS Android



圖 11：Apple 和 Android 的商標。

近年來 iOS 與 Android 兩大智慧型手機平台之爭，常受到許多人的注目與討論。有一派人認為，iOS 有統一的界面與互動設計，因此帶來較佳的使用者體驗，最後必將勝出無疑；相反的，也有不少人認為 Android 的開放系統，可以授權給任何硬體廠商使用，不需要經過 Google 審核，會獲得比較多的開發商支持，所以 Android 必回得到最後的勝利果實。

兩種說法當然都有道理，封閉系統有封閉系統的優點，開放系統有開放系統的優點。有人說，封閉系統就像獨裁專制，短期間之內很有效率，但最終將不敵正義的開放系統。

這樣的講法，聽起來好像很有道理，但眼前就有一個很大的反例——正義的 Linux 何時才要打敗邪惡的 Windows 呢？哪有打了這麼久，正義的一方不但沒贏還被打趴在地上的道理呢？（附帶一提，以目前而言 Linux 與 Windows 的市佔率分別為 1.10% 與 91.77% 左右，參見圖 12）

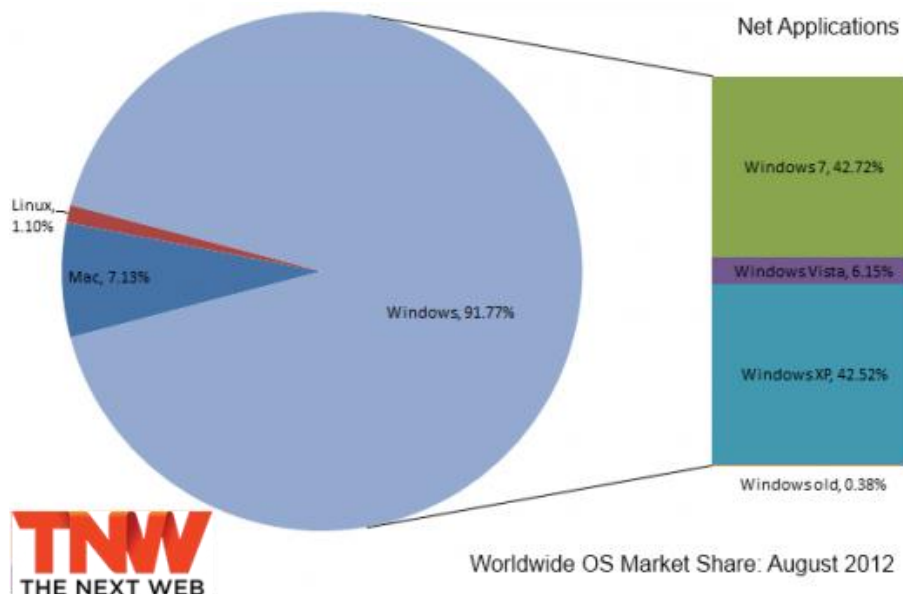


圖 12：各種作業系統的市場占有率 (2012 年 8 月)。

因此，我們不妨對 Android 4.0 和 iOS 5 兩個系統進行一些功能性的比較。兩款系統都有一些共同點，在雲端系統方面，iOS 5 提供了 iCloud 技術，而 Android 4.0 則支持 Google 的雲端技術；在社交功能上，Android 4.0 內置了 Google+，而 iOS 5 內置的則是 Twitter；語音功能上，Android 4.0 使用的是較為普通的語音輸入，而 iOS 5 則是相當強大的語音操作助手 Siri。

另外，Android 4.0 的照相功能變得更為強大，例如可添加各種濾鏡和相框，可進行超寬全景照片拍攝等等。可以說，在功能上，兩個系統可以說是各有千秋。

但是系統功能並不是評價產品好壞的唯一標準，畢竟它們都可以通過第三方軟件擴展其他功能。對用戶來說，系統的使用體驗才是最重要的，這也是 Android 系統最大的不足，由於 iOS 僅在蘋果的設備上使用，因此無論在系統優化上，iOS 設備是不用質疑的。但是 Android 系統會用在不同的產品上，而且廠商都喜歡定制自己的 Android 系統，到最後 Android 系統能否給用戶良好的體驗，最終還是要看廠商的實力，這也正是 Android 系統體驗良莠不齊的原因。



圖 13：Windows Phone 標誌 (2012)。

不過無法否定的是，目前能夠與 iOS 抗衡的也就只有 Android 系統了，微軟的 Windows Phone Mobile 系統可能會很強大，但是目前仍然未形成氣候。而且 Android 4.0 系統同時支持智能手機和平板電腦，從成熟度來說已經可以媲美 iOS，誰勝誰負，已經無法輕易斷言了。希望兩者能展開一場公平、正當的競爭，讓消費者享受不同設備帶來的美好體驗。

七、 版本控制

維護工程藍圖的標準作法，能追蹤工程藍圖從誕生一直到定案的過程。此外，版本控制也是一種軟體工程技巧，藉此能在軟體開發的過程中，確保由不同人所編輯的同一程式檔案都得到同步。

我們是使用 Subversion，簡稱 SVN，是一個開放原始碼的版本控制系統，相對於的 RCS、CVS，採用了分支管理系統，它的設計目標就是取代 CVS。網際網路上越來越多的控制服務從 CVS 轉移到 Subversion。在不同版本間，只要檔案內容一樣就只會有一個實體，如此能節省空間及提升效率。



圖 14、TortoiseSVN 的主選單畫面。

在 2000 年初，開發人員要寫一個 CVS 的自由軟體代替品，它保留 CVS 的基本思想，但沒有它的錯誤和局限。2000 年 2 月，他們聯繫了 Open Source Development with CVS (Coriolis, 1999) 的作者 Karl Fogel，問他是否願意為這個新項目工作。^[L7]



圖 15：CVS 標誌（舊）。

碰巧的是，這時 Karl 已經在和他的朋友 Jim Blandy 討論一個新的版本控制系統的設計。在 1995 年，兩人開了一家提供 CVS 技術支持的公司，叫作 Cyclic Software。雖然公司已經賣掉了，他們仍然在日常工作中使用 CVS。

在使用 CVS 時受到的束縛已經讓 Jim 開始仔細思考管理版本化數據的更好的路子。他不僅已經起好了名字「Subversion」，而且有了 Subvesion 資料庫的基本設計。

當 CollabNet 打來電話時，Karl 立刻同意為這個項目工作。Jim

徵得他的老闆 RedHat Software 的同意，讓他投入這個項目，而且沒有時間限制。CollabNet 僱用了 Karl 和 Ben Collins-Sussman，從 5 月份開始詳細設計。

由於 Greg Stein 和 CollabNet 的 Brian Behlendorf 和 Jason Robbins 作了恰當的推動，Subversion 很快吸引了一個活躍的開發人員社群。這說明了許多人有相同的受制於 CVS 的經驗，他們對終於有機會對它做點什麼表示歡迎。

最初的設計團隊設定了幾個簡單的目標。他們並不想在版本控制方法論上有新突破。他們只想修補 CVS。他們決定 Subversion 應該與 CVS 相似，保留相同的開發模型，但不複製 CVS 最明顯的缺點。雖然它不一定是 CVS 的完全的替代品，它應該和 CVS 足夠象，從而任何 CVS 用戶可以不費什麼力氣的轉換過來。

經過 14 個月的編碼，在 2001 年 8 月 31 號，Subversion 可以「自我寄生」了。就是說，Subversion 開發人員停止使用 CVS 管理 Subversion 的原始碼，開始使用 Subversion 代替。雖然 CollabNet 發起了這個項目，而且仍然支助一大部分的工作（它為一些專職的 Subversion 開發人員發薪水）。

但是 Subversion 像大部分開放源碼的項目一樣運作，由一個鬆散透明，鼓勵能者多勞的規則管理。CollabNet 的版權許可證和 Debian FSG 完全兼容。換句話說，任何人可以免費下載，修改，按自己的意願重新分發 Subversion，而不必得到來自 CollabNet 或其他任何人的許可。^[L7]

八、Java 程式語言

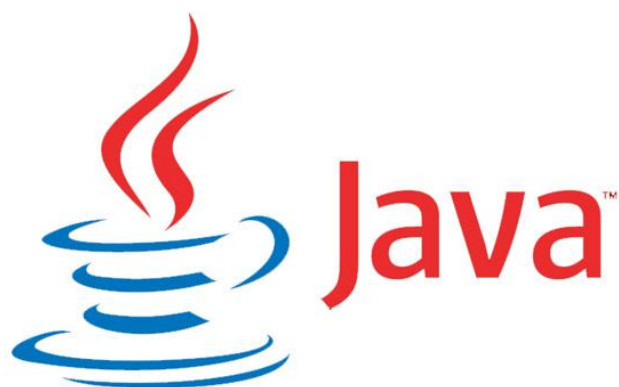


圖 16：Java 標誌。

Java 是一種電腦程式設計語言，擁有跨平台、物件導向、泛型程式設計的特性。任職於昇陽電腦的詹姆斯·高斯林 (James Gosling) 等人於 1990 年代初開發 Java 語言的雛形。

最初被命名為 Oak，目標設定在家用電器等小型系統的程式語言，應用在電視機、電話、鬧鐘、烤麵包機等家用電器的控制和通訊。

由於這些智慧型家電的市場需求沒有預期的高，昇陽公司放棄了該項計劃。隨著 1990 年代網際網路的發展，昇陽公司看見 Oak 在網際網路上應用的前景，於是改造了 Oak，於 1995 年 5 月以 Java 的名稱正式發布。

Java 伴隨著網際網路的迅猛發展而發展，逐漸成為重要的網路程式語言。^[L8]

九、 SQLite



圖 17：SQLite 標誌。

SQLite 是遵守 ACID 的關聯式資料庫管理系統，它包含在一個相對小的 C 庫中。它是 D. R. Hipp 建立的公有領域項目。不像常見的客戶端/伺服器結構範例，SQLite 引擎不是個程序與之通信的獨立進程，而是連接到程序中成為它的一個主要部分。

所以主要的通信協議是在程式語言內的直接 API 調用。這在消耗總量、延遲時間和整體簡單性上有積極的作用。整個資料庫(定義、表、索引和資料本身)都在宿主主機上存儲在一個單一的文件中。它的簡單的設計是通過在開始一個事務的時候鎖定整個資料文件而完成的。^[L9]

十、 Activity：Android 應用程式之機動程式

Android 作業系統的機動程式是基本的處理工作單元。大部分的機動程式都會有一個相對應的顯示畫面(View)，就好像 HTML 程式都會針對某一個網頁來描述。^[L10]

Android 應用程式的顯示畫面(View)就好比一個配置了按鈕、信息輸入欄位和顯示文字欄位的畫面，一個可以瀏覽器顯示的 Web 網頁或一個播放 3D 影像的畫面。

一般所指的活動是使用者介面。一支應用程式可能有一個或以上的活動存在，每個活動也都會有自己的 View。

所有的活動在系統裡由活動堆疊所管理，當一個新的活動被執行後，它將會被放置到堆疊的最頂端，並且變成啟動程式 (running activity)，而先前的活動原則上還是會存在於堆疊中，但它此時不會是在前景的情況，除非新加入的活動離開。

一個 Android 應用程式是由一到多個機動程式所組成(最簡易的應用程式可以只有一個機動程式)，當使用者切換顯示畫面時，Android 提供 Intent 的方法來完成，比如說：用 HTML 連結到另一個網址。Android 提供一些方法和指令，Activity 不需要一開始執行就要到結束狀態。Activity 在 Android 中可以是在 Running, Pause 或 Stop 狀態。^[L10]

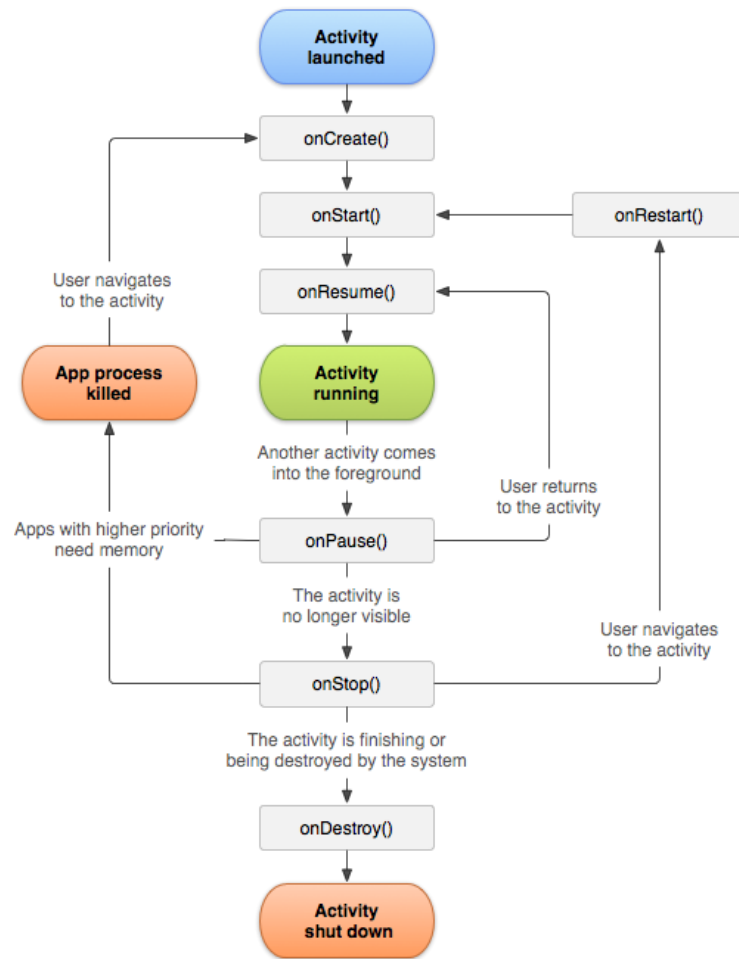


圖 18： Activity 流程圖。

可以參考 ^[L11] 以下是我們自己解釋以上的 Activity 狀態：

- 執行 (Running)：

當 Activity 是在執行狀態時，它是在前景(Foreground)處理中，所相對應的顯示畫面會出現在螢幕上。

- 暫停 (Pause)：

當 Activity 是在暫停狀態時，其他 Activity 已凌駕其上，螢幕上展現的是其他 Activity 相對應的顯示畫面。

但是該 Activity 的對應的顯示畫面仍然可以看見，只是沒有佔據整個螢幕，它只是在待命中，隨時可以回到執行狀態(`onResume`)，但是當記憶體不足時，會被強制結束。

- 停止 (Stop)：

當 Activity 是在停止狀態時，其他 Activity 已駕臨其上，螢幕上展現的是其他 Activity 相對應的顯示畫面，該 Activity 的對應的顯示

畫面已完全看不見，它是在背景 (background) 待命中，可以被呼叫再開始 (onRestart)；但是當記憶體不足時，會被強制結束。

一個 Activity 基本上有三個生命狀態，可以參考圖 18。在 Activity 裡，每一個狀態都具有不同的定義。以下是每一個狀態的解釋：

- 執行狀態：

當一個活動在螢幕的最上層時(系統堆疊中的最頂端)，此活動就是屬於 active 或 running。

- 暫停狀態：

當一個活動失去焦點(Focus)但還看得到它的畫面，那失去焦點的這個活動則處在 Paused 狀態，像這種屬於 Paused 暫停狀態的活動，當系統的記憶體不夠用時，系統會自動判斷，把優先順序較低的活動移除。

- 停止狀態：

當一個活動被其它的活動完全遮蔽時，被遮蔽的活動就是處於 Stop 的狀態，不過，仍保有全部的狀態及資料，但是因為已不再被使用者看見，所以當系統記憶體不足時，這種 stop 狀態的活動是最先考釋放記憶體的。

貳、 研究動機

一、 解釋

本研究動機是為了效仿困難的遊戲對遊戲玩家的影響而設計出簡單操作的遊戲。操作越簡單，玩家越容易上手。遊戲困難度從極簡單開始到極困難，遊戲關卡越難越少，因為不希望玩家無法破關而喪失信心。

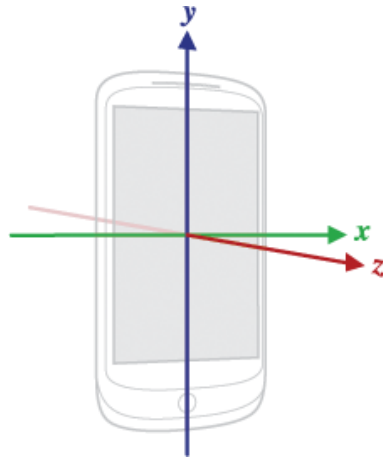


圖 19：Android 手機內部三軸感應器之軸向。

Android 手機上所擁有的三軸感應是一個可以在智慧型手機上實際操作使用的感應器，其應用方面與層次非常廣。目前來說，其 Android 手機上的三軸感應器可以當作是遊戲輸入操作，利用感應器上的 X 和 Y 軸呈現出在手機螢幕上物體滾動方向。而 Z 軸則可以用在手機的向下地心引力作為感測，使得物體在螢幕上滾向手機螢幕最低位值。

Android 是以 Linux 為底的作業系統，主要是以 Java 程式語言進行開發。為了能夠提升我們自己開方程式能力，才能夠在社會上共存，我們選擇對 Android 平台進行遊戲開發，同時也能夠學習到如何維護與寫出大型程式專案。以另一個角度來說明，我們要在 Android 平台上開發 APP 軟體，是為了能夠自我學習軟體工程所應有的技術。

遊戲可以說是應用程式的子集合，也就是我們一般所認定的遊戲，其實是應用程式的一種。我們也因此決定先從基礎開始進入學習狀況，而使用了一本介紹 Android 應用程式的開發，來增加我們對 Android 作業系統的認知。

我們的遊戲是一個可以做互動的應用程式。玩家可以使用自己的手先做平衡，為了能夠讓手機內的三軸感應器感知道手機是平衡了。

遊戲它本身是從應用程式分類出來成一個獨立的類別，也因為這樣我們才可以從網際網路上找出各種不同有關於遊戲方面的資訊。對於我們的專題來說，資訊的多寡會取決於製作過程的困難度，以及熟悉度。我們的認定是，遊戲的資訊非常龐大又非常多，製作遊戲在某些方面的考量上，會是一個很好的選擇，可以作為我們的專題之方

針。

我們所得到資訊來源其中之一的地方是從 Android Developers 中取得有關 Android API 函式庫的 Android 文檔 (Android documentation)。這些資料是提供 Android API 函式庫的使用、呼叫、等等方法，並且說明該函式庫的一些特定的函式在程式裡會產生什麼樣的事件。



圖 20：Android Developers 標誌。

第二個取得資訊的來源之地方是由 Stack Overflow 問答區，所提供的 Android 問題與解答算是很準確。該問答區類似於 Yahoo!奇摩知識，但是本身是針對在進行開發應用程式的時候會遇到的問題提出相關解答，來幫助別人。同時，Google 有推薦 Android 應用程式開發的初學者們去這個問答區提出他們有關 Android 的問題。



圖 21：Stack Overflow 問答區。

第三個資訊來源的地方是 GameDev.net 論壇區。所有程式方面的資訊可以透過人與人的互動與交流來進行知識督導。我們其中之一的組員有從這個論壇區得知我們應該要針對遊戲的方向為何。對方也有提出我們要看對方所推薦給我們的手冊，我們相信這會對我們有所幫助。



圖 22：Gamedev.net 論壇區。

當然，我們也不是說我們什麼都可以做到就能夠可以應用在 Android 平台上使用。由於我們自己對於 Android 所支援的 OpenGL ES 的概念、知識、技術、以及效能度感到模糊，目前沒有任何打算往該 OpenGL ES 方向進行開發。我們主要講究的是，從以上的資訊來源所得到的資訊，可以合併運用而做出一套簡單的遊戲。我們並沒有限定遊戲本身要多簡單，只要求說我們自己創造出來的遊戲一定要出現在玩家者的手裡。

專題組員的時間分配，因為有許多因素，有考量到我們沒有辦法做到正式的 Scrum，因此我們在整個開發過程，原則上，是接近於 Scrum 的一種，但是並不是 Scrum 本身。

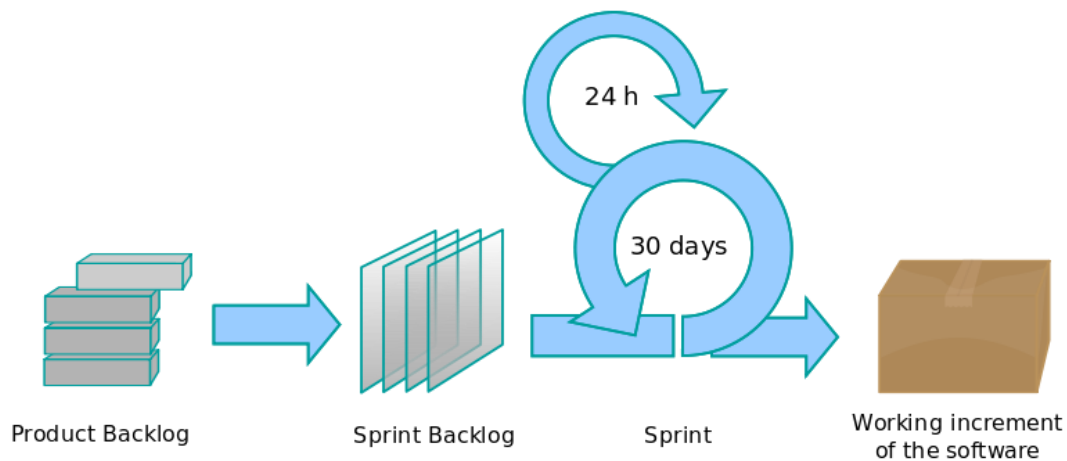


圖 23：Scrum 軟體開發流程圖。

二、計畫目的與範圍

我們的目的是希望能夠達到以下五項目的中的三項或三項以上

項目。以下是我們的專題範圍：

- 遊戲概念或遊戲理念
- 遊戲玩法的完整性
- 遊戲結構的流暢度
- 遊戲關卡困難度的平衡性
- 遊戲操作的使用方便性
- 遊戲要具備遊戲五大基本內容：
- 遊戲流程
- 人性化介面圖 GUI
- 設定
- 控制器
- 說明

我們專題的目的是設計出一套遊戲能夠使用上或應用到 Android 手機上所擁有的三軸感應器。遊戲的五大基本內容是屬於我們專題的五大中繼站，非按照順序排列。

如果是參考各種不同遊戲，不管是電腦上或是手機上，遊戲最需要的配備就是遊戲的玩法。玩法完整性的考量與方面來說，我們不希望在所有的關卡當中，因為遊戲規則的限制，會導致有一門關卡是解不出來、或是無法破解。

遊戲的玩法，依照各種遊戲概念的不同，會出現有所不同的遊戲玩法，例如：Black and White (善與惡)是由 Lionhead Studios 製作。他們的遊戲理念是玩家自己當作是神來操作人民，並且要求玩家自己去影響整個遊戲的發展。遊戲的玩法是由該遊戲的概念所呼應的限制與範圍，而形成所謂的遊戲規則。

所謂遊戲的完整性，是指所有的玩家都能夠破解遊戲所提供的關卡，並且不會影響到遊戲的好玩度，同時也能夠讓遊戲玩家知道，遊戲本身是在玩什麼給玩家聽或是給玩家看。

在這裡，我們的遊戲理念最主要的是讓玩家可以在不經由觸控螢幕的操作下，按照我們所規定的遊戲規則，進行關卡的破解。我們也不希望在遊戲當中會出現一些我們乍看之下沒有什麼問題的小錯誤。這些小錯誤很可能以致於讓玩家無法順利進行遊戲。反過來說，如果這些小錯誤並沒有影響遊戲的玩法或是違背遊戲規則，我們反而

會去鼓勵玩家去多善利用。

流暢度是遊戲規則所擁有的一致性、簡單的解釋、以及明確的關鍵字等等這些性質，加以整合讓玩家懂得要怎麼玩、怎麼做，才能進行一個順利的遊戲。遊戲規則如果不明確，玩家無法保證可以繼續玩下去。太過複雜的規則，則會影響到遊戲的完整性，以及會讓玩家對玩法感到模糊。規則沒有在遊戲中保持一致性，則遊戲會失去整體上的流暢度，玩家會因此不懂正確玩法。



圖 24：善與惡（遊戲畫面）。

遊戲困難度可以解釋成玩家自己對這套遊戲規則的熟練度。我們不希望在遊戲關卡的建設當中出現有太多重複的要素。在這邊，我們是設計出讓玩家慢慢適應遊戲的規則和提升熟練度，並且希望遊戲能夠給予玩家一定困難度的挑戰，同時也不會造成玩家喪失個人的自信。

在遊戲操作方面，我們是希望能夠增加三軸感應功用的使用性，並且做出比較具有適度的彈性。例如：在一個關卡裡，會用到三軸感應的 Z 軸來進行籃球灌籃，灌進籃球才能破解該關卡，並且能夠進行下一關關卡。如果是用在不同的關卡，可能還會用上 X 軸來加以球道的瞄準度。（可以參考圖 19）

運用這些技巧可以帶給玩家各種不同玩法，這麼一來不會讓玩

家覺得遊戲本身是單調的、沒有多樣性等等各種感受。

遊戲五大基本要素有遊戲流程、人性化介面圖、設定、控制器和控制鍵的排列、以及說明。遊戲流程是一個遊戲進行的過程。我們是利用關卡的安排制度和獎賞來提升玩家想繼續玩這個遊戲的慾望。



圖 25：Android SDK 示範程式包：Lunar Lander。

一般來講都會有遊戲音樂或遊戲音效，例如：背景音樂、爆炸聲、等等。一旦遊戲結構上的各個方面都完成，則可以加入遊戲音樂(配樂)等等。音樂可以給予遊戲帶上一點點趣味。一些遊戲文章有指出，遊戲音樂不見得是遊戲必要的要素之一。我們有認為遊戲音樂可能是不需要的，但是遊戲音效是需要的。

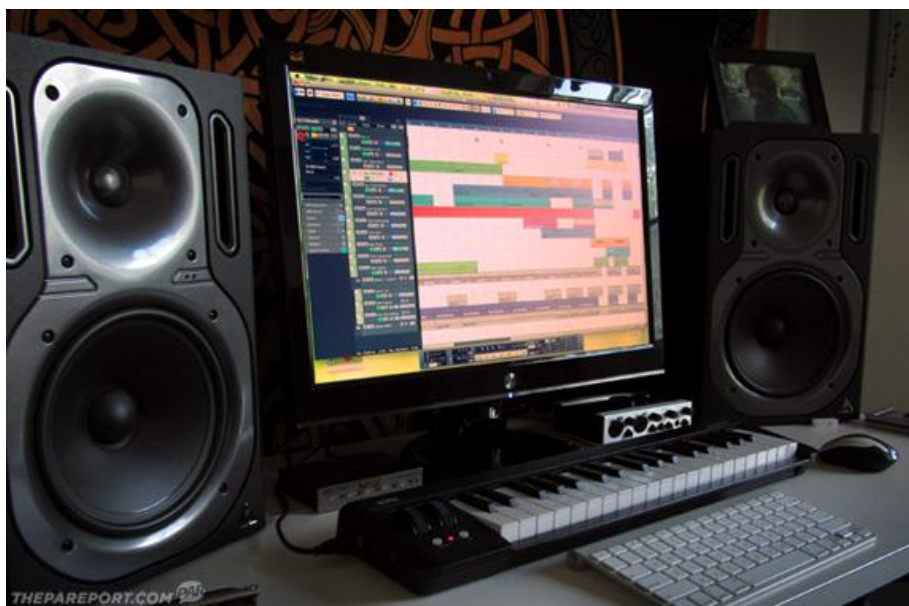


圖 26：遊戲音樂製作的過程有很多種型式可以使用。

有了簡單的方式可以告知玩家遇到什麼的事件，是能夠可以達到吸引玩家的專注力。也許，可能是因為環境因素或是我們無法預料到的原因，玩家還是會把聲音關掉，也就是會把音樂和音效同時關掉。以上所考量到的可能性，我們在這裡可以解釋設定是對遊戲來說的基本要素其中之一。

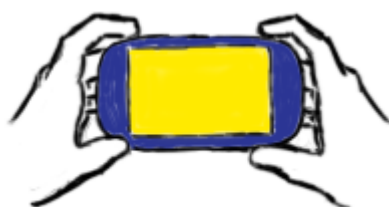


圖 27：使用三軸感應器來做為控制器使用。

控制器是遊戲給予玩家操控遊戲內的物件。我們主要的控制器是 Android 手機內部的三軸感應器。我們利用此當作我們主要的控制器，最主要的原因是它可以達到在遊戲裡可以用直覺去控制物件。在這邊，我們的主要遊戲物件是一顆球。按照人的直覺反應可以知道說一顆球會隨著斜坡的斜率和斜坡方向而前進。

如果把 Android 手機本身當作一個平板，用手把平板傾斜，就可以形成一個簡單的斜坡。再把我們的球這個物件放進去，就可以簡

單的把球當作是用直覺來操控一個物件。

控制鍵是可以用來輸入姓名以記錄最高分，或者是呼叫出來主選單的畫面。控制鍵雖然不需要很多，還是需要一些基本的控制。

最後的基本要素為說明，是用來跟玩家解釋有關遊戲的資訊和給予遊戲教學。說明的重要性，可以牽扯到智慧財產權的問題、告知玩家遊戲作者是誰，免得會玩到盜版的版本、以及玩家若想要支持遊戲的作者而提供一些資訊。說明可以扮演著作者想要表達事情給玩家的一種角色。也因此，它扮演著遊戲當中的基本要素其中之一。

三、 研究方法及步驟

一開始，我們對於 Java 程式語言的知識並不是很豐富，也有要求過我們自己去上多一點 Java 課程，促進我們學習新鮮 Java 技術。那個時候，我們只擁有一般標準的程度，已經足夠懂得對 Android 作業系統所使用的函式呼叫程序有了概念。也因此，我們能夠進行我們原先計畫好的下一個步驟。

從那個時候到現在為止，我們累積了不少和 Java 以及 Android 息息相關的知識和技術，日月累積起來而成為能夠會應用 Android 內部介面操作與功能，以及使用簡單的 XML 來做出簡單易懂的介面。

我們是採用了邊做邊調查出如何製作我們會應用上的功能，以至於能夠縮短我們完成進度的時間。

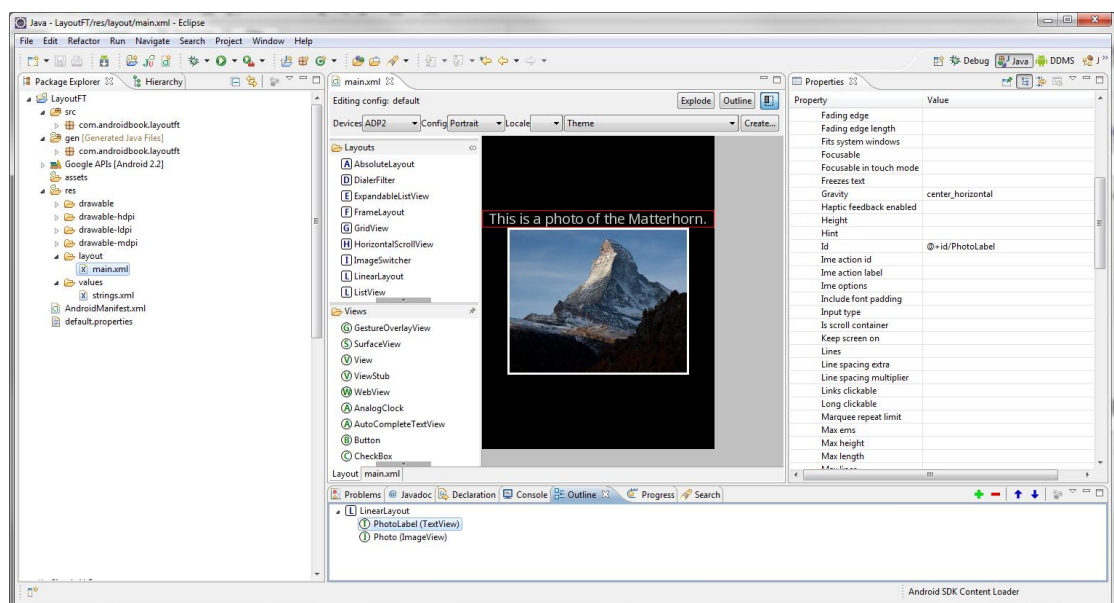


圖 28：Android XML 編輯器。

一、研究完成 Android API

原本我們是有提到過，Android API 是屬於一種自由軟體。我們是後來知道 Android API 是 Android 作業系統所提供程式設計師進行使用以及寫用的統一性標準函式庫，能夠讓開發者們寫出自己的軟體。

我們認為自由軟體這個稱呼簡單的解釋是「能夠自己決定軟體的使用權的軟體，由使用者決定該不該使用這個軟體」，而並不是說成該 Android API 是一種免費軟體讓人去使用。它還具備著龐大的 JavaDoc 程式文檔及很有特色又非常活躍的社群，而該社群可以幫助其他開發程式者們回答以及解決他們遇到的問題。

簡單來說，研究 Android 的 API 其實是比我們一開始認為的困難度較沒有那麼的困難，可是還是會在邏輯上以及隱藏之處會不斷的出現錯誤，是除了問問這個社群之外就會不太容易的能夠得到解答。

我們在研究 Android 的 API 的時候，當中有透過網際網路來進行 Android 程式問題與答案的資訊交流，兒我們之所以這麼做就是為了能夠提升學習速度以及輔助我們完成專題的進度。在加上，我們所進行開發的專題是跟在網際網路上的人所做的遊戲開發的專題所具背著有相似的共同點，從這裡就可以知道我們是需要做什麼與該捨棄我們不需要的東西。

我們也是因為資訊的交流的關係，讓我們能夠去懂得從經驗豐富程式設計師得到靈感、一些重要的領悟，技巧等等、來避免錯誤概念和迷失以及提升開發程式的效益。

甲、第一步驟

我們的第一步驟就是去上 Google 的 Android Developers 網站，找出該網站所連結到的 References 這篇文章。它是具備著 Android API 的詳細文檔 (documentations)，是可以讓我們熟知一些我們可以使用的函式以及功能。

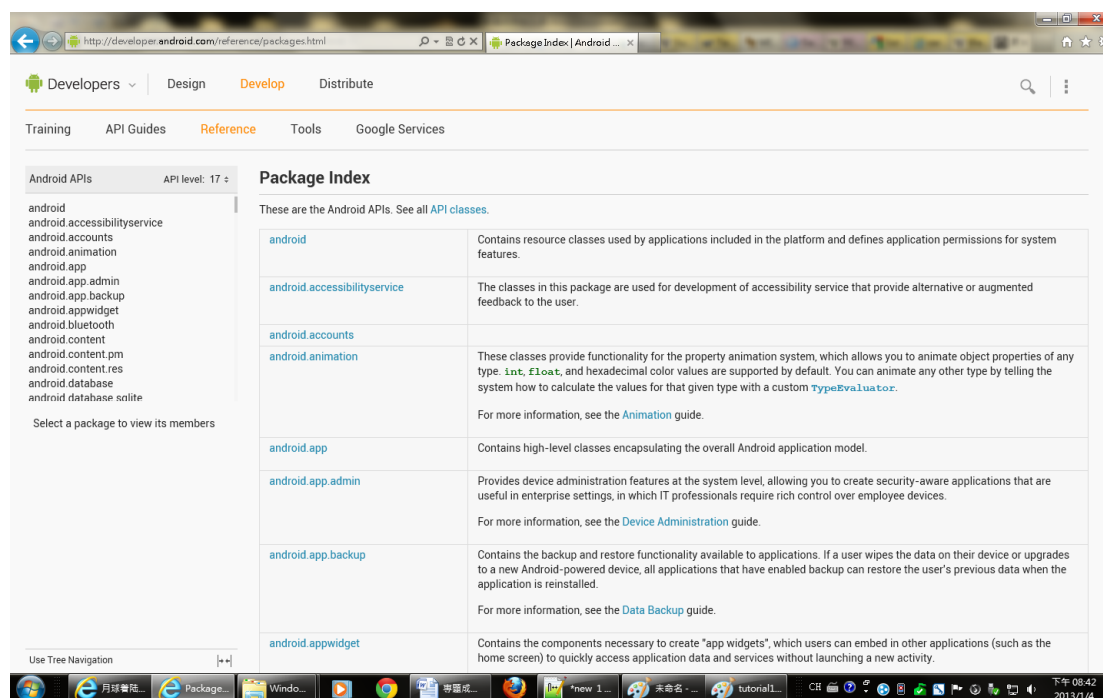


圖 29：Android Reference 頁面。

我們查到的是，Google 所附上的文檔裡當中，有一些常用的類別 (classes) 和函式 (methods) 都是會有比較詳細的介紹與說明，而有一些不常用的，則是列出它的用途是做什麼，而不會去做介紹等等。

在網際網路上的社群當中，我們有發現，有一些不知道使用基本的 Android API 函式的開發者們所問的問題，得到的解答是只接去看該 Android API 所提到的函式與介紹。

但是，有特別情況的問題，有詳細做過研究與調查的問題，還有大家常常遇到的問題等等，基本上都是會受到社群的注目，回答的答案中基本上也會有詳細的解釋和步驟說明等等。

簡單的來敘述，絕大部分的問題是透過社群的幫助才能夠解決問題，而不是只去看 Android Developers 的 Reference 文章就能夠解決的。這一點是我們在上網調查的時候所發現到的現象，也希望大家能夠注意到這一點。

根據 Android SDK 文檔和我們有購買過的 *Google SDK Android 應用開發實戰* 這一本書，我們是必須要去呼叫 `Activity.onCreate()` 才能夠有建立一個進入點 (entry point)。這個 `Activity.onCreate()` 函式是我們自訂的 Activity 程式所要去執行的進入點，就如 C++ 的 `main()`

函式類似，使得我們的程式可以去呼叫其他 Activity 出來。

除了這個要點以外，我們在網際網路上的社群還得知要透過 Eclipse 開發環境裡的 Android 專案資料夾中的 Android Manifest 來決定出哪一個 Activity 類別是成為主要的進入點和主要啟動者 (Activity Launcher) 的資訊。這也讓我們發現到，不見得在書裡面可以直接找到我們要的解答，就如上述的 Android Manifest 這一類的情況。

我們在那一開始的時候是去學習 Activity 的生命週期和呼叫方式，才能順利進行下去。從 Activity 的生命週期得知，該 Activity 有分成不同階段 (state)，而每一個階段都有屬於自己專屬的功能。呼叫的方式是透過 Activity 的 Intent 作為媒介去進行訊息 (message) 的傳送。



圖 30：Android Manifest 畫面。

我們是得知 Intent 的用途是在當我們需要傳送 Activity 與 Activity 之間的訊息的時候才會在 Intent 中加入必要的資訊。如果只是去呼叫其他的 Activity，則使用簡單的 Intent 來取而代之。

乙、 第二步驟

之後，第二步驟是我們要去學會如何使用顯示畫面 (Views)。

顯示畫面本身是涵蓋著文字顯示 (Text Views)、按鈕顯示 (Buttons)、月曆顯示 (Calendar Views) 等等。

我們在開發的過程當中，在網際網路上提到，除了這些顯示畫面是用 XML 去做設計之外，我們可以透過 Java 程式去呼叫屬於我們自己自訂的顯示畫面 (Custom Views)。

當我們學會了如何使用顯示畫面，我們做了調查，並且學會使用自訂顯示畫面，還懂得如何在其它顯示畫面中使用 setContentView()，把我們自訂的顯示畫面直入進去其他顯示畫面，呈現出具有上下結構的嫌是畫面。

在自訂顯示畫面，我們發現到可以針對畫面的排列和大小作變更和排列，特別是大小、排列對齊等等部分。

之前，我們有報告出，用手指去觸碰顯示畫面，如果該畫面太小會造成意料之外的行為發生，其實只要是用手指去按別的地方也是會造成意料之外的錯誤行為。

為了能夠修復上述情況，我們從網際網路上學到使用 Dialog.dismiss() 和 View.setOnClickListener() 等等這些功能，並且應用進來，使得我們之前的錯誤行為修復成功。

在自訂顯示畫面中，我們採用了 SurfaceView，做為我們主要顯示遊戲畫面之前端，以及使用到 Android Canvas 做為後端。控制該 Android Canvas 就可以控制住我們要顯示在螢幕上的遊戲物件等等。

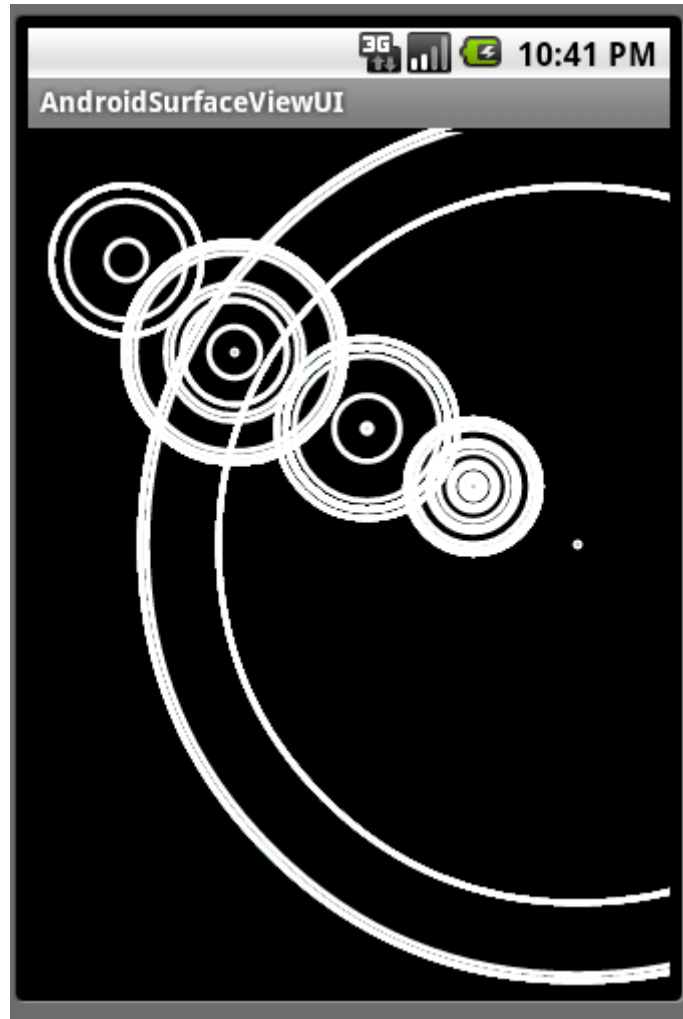


圖 31：SurfaceView 範例畫面。

完成 SurfaceView 的 Canvas 控制機能之後，我們開始處理遊戲畫面的更新率 (Frames Per Second) 和遊戲幀率 (Frame Rate)。利用 Android API 所提供的 Looper 和 Handler 類別，我們成功的實施了多執行緒的功能，使得遊戲更新率和遊戲幀率呈現出穩定的狀態。

之前是在 HTC Wildfire S 手機上進行程式偵錯，而發現在顯示出螢幕的畫面更新率為 15 FPS 以及遊戲幀率會隨著 Android 手機不同而有所不同。我們舊的假設是，擁有許多背景程式（或稱為服務 (Services)）在跑，也會影響 Android 手機幀率之頻率，但是這並不是這樣的。用同樣的程式在 HTC Evo 3D 上進行偵錯的時候，發現遊戲更新率和遊戲幀率都是很理想的，甚至是不必去理會手機裡背後有執行多少服務等等。

我們提出的心假設就是擁有好的硬體設備會大幅度的去影響遊

戲更新率和遊戲幀率等等，遊戲玩家的初步體驗也會有很大的差異性。

丙、 第三步驟

我們原先是計畫好去設計出主選單。事實上，我們後來決定先放入遊戲物件，並且開發遊戲核心，慢慢做調整，再進行外部設計，如主選單等等。

遊戲的核心是一開始決定由玩家使用手機內部的三軸感應器去控制一顆球，用控制的球去撞擊其他的球，如撞球之類的遊戲一樣。我們採用了 Paint.NET 劃出簡單的球，放上 SurfaceView 的 Canvas，再開始調查碰撞偵測 (Collision Detection)。



圖 32：Paint.NET 標誌。

一開始是先確認主要被控制的球是不是能夠用三軸感應器去控制，而且是要確認球被控制的程度是如何。這時候，我們是可以慢慢的去調整球體的移動速度。當速度調整好，我們就遇到一個小問題。

當手機在利用三軸感應器去控制球的時候，球本身經常移出手機螢幕的外面。不但如此，球的圖案也隨著浮點數的運算，造成圖案顯示的很不穩定的現象。

我們為了能夠去修正圖案不穩定的現象這個問題，必須要先固定住好球體的直徑以及圖案的邊長。

我們為了能夠避免球自己因為被三軸感應器控制到球的座標移出幕顯示的範圍之外，必須要先設定座標的最大與最小範圍。要取得這個範圍，是必須要先從螢幕管理 (WindowManager) 這個 Android API 內得取螢幕的寬度與高度。最後再加以拼湊而得到我們的最大與最小範圍。

當我們修正這些問題的時候，就可以開始設計遊戲核心的遊戲玩法。第一個會想到的就是去創出新的遊戲物件。這些物件主要的控制方式就是由玩家用自己本身所控制的求去進行互動。該互動可以解釋成該物件的主要目的。

二、 進行遊戲開發

在還沒有開始做任何事情，我們有主動先去找老師問問看能不能讓我們使用一台伺服器。這一台伺服器的用途是讓我們去架設一台擁有版本控制系統的虛擬伺服器 (Virtual Machine，簡稱為 VM)。

架設一台擁有版本控制的伺服器的目的是讓我們擁有能力可以上傳我們的進度到雲端上面。這麼一來，當我們要去使用其他的電腦的時候，我們利用版本控制系統，抓取我們要編輯的版本，在我們使用的電腦上繼續進程式開發。這個步驟一旦成功，我們就可以進行下一步驟。

我們成功的去申請到一台虛擬伺服器的時候，我使用了 VirtualSVN 這一套免費軟體。這一套軟體可以輕鬆的架設一台可以使用 TortoiseSVN 來進行版本控制，同時也可以負責管理使用者的權限、管控 Repository (儲存庫) 等等目錄管理。



圖 33：Virtual SVN Server 標誌。

一開始，我們必須要熟悉如何操作 TortoiseSVN 以及如何連結到 VirtualSVN 伺服器去。先是在 VirtualSVN 伺服器裡建立新的儲存庫在該虛擬伺服器上的硬體。這個儲存庫類似於專題根目錄與資料庫系統並再一起使用的資料庫。

開啟 VirtualSVN 的時候，點選 Repositories，然後在 Repositories 目錄裡面的空白地方右點選取 Create New Repository。

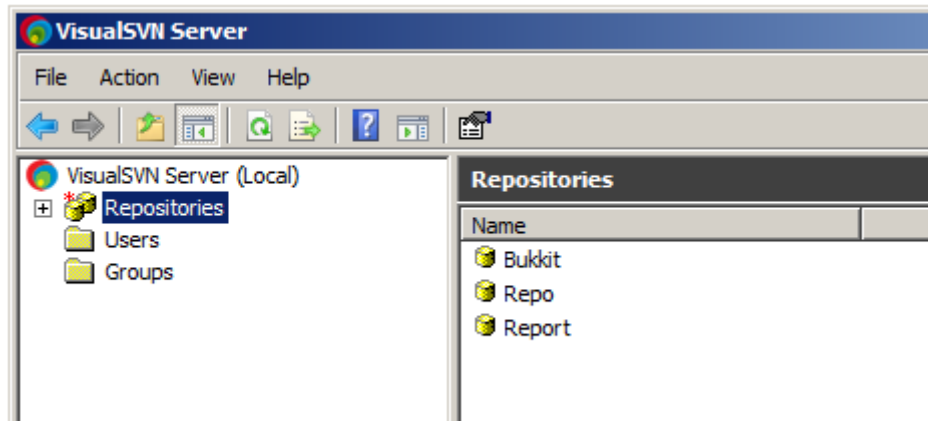


圖 34： VirtualSVN 面貌與被選取的 Repositories 目錄。

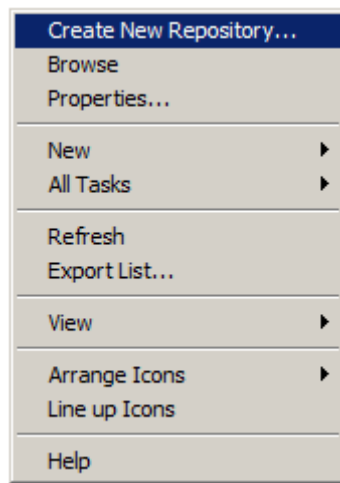


圖 35：在 Repositories 裡右點選取 Create New Repository。

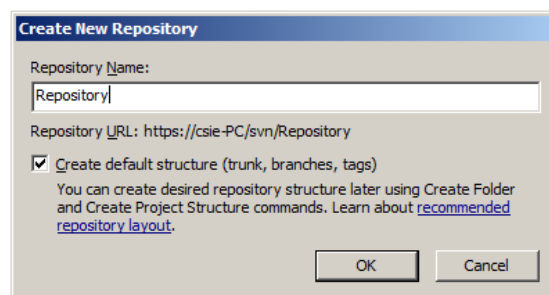


圖 36：在命名 Repository 時，會出現儲存庫 URL 位址。可打可不打勾 Create default structure。(為標準格式)

使用 VirtualSVN，必須要先建立新的 Repository，並且命名該 Repository。這個 Repository 名稱就會用在 Repository 的外部 IP 的 URL 位址使用。

建立新的 Repository 之後，我們在我們要使用的電腦上建立新的資料夾（通常我們都會在桌面上建立新的資料夾）。在新的資料夾上用滑鼠右點它，在右鍵選單裡選取 SVN Checkout。這是讓新的資料夾進行對新建立的 Repository 做連接橋梁的方式。

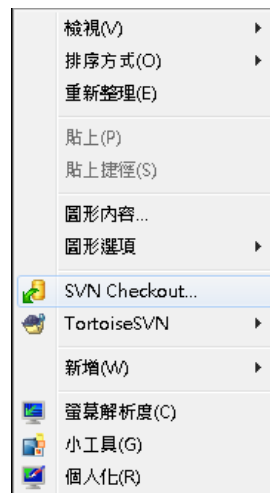


圖 37：右鍵選單上會出現使用 TortoiseSVN 的選項。

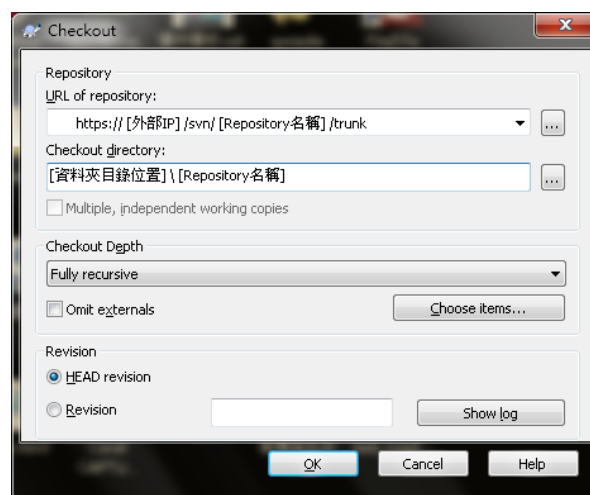


圖 38：SVN Checkout 所出現的視窗。以上為使用範例。

圖 38 為使用範例，我們是用這個命名格式，並且完成之後按 OK。

進行橋梁連接完成之後，我們放入我們的專案檔案 (Eclipse 所建立的資料與檔案等等一併放入)，在放入的資料夾上右點，點選 SVN Commit，就會進行上傳資料的動作。這個時候，VirtualSVN 會收到訊息，並且在 Repository 裡面更新，成為第一個版本。

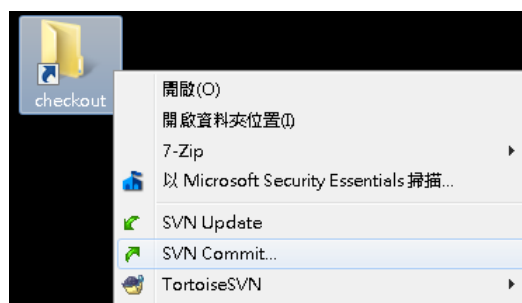


圖 39：當完成 SVN Checkout 時，右鍵選單會出現新的選項。

做完以上的步驟，我們就可以開始進行遊戲開發的部分。

遊戲物件的開發可以分成由系主題、主要的部分與次要的部分。

我們在開發遊戲之前，是必須要限設定一個遊戲主題。我們原先的遊戲主題是製作出類似於彈彈珠遊戲，是用小小顆的彈珠彈出去撞擊其他小小顆的彈珠。我們後來決定出新的遊戲主題，是控制一顆球的同時，要把該控制的球體引誘到終點。

主要的部分是先想出新的遊戲物件要在遊戲裡是扮演什麼樣子的角色、這個角色有什麼事情可以讓遊戲玩家去做互動等等。

次要的部分就是程式的部分，可以去做比較細節的調整。整體來，就是要實行把其他的遊戲物件並在起來，以至於玩家可以用不同的互動方式跟其他遊戲物件去進行遊戲。

包含在次要的部分裡，我們是想要透過闖關卡模式去進行遊戲，而每一個關卡要使用遊戲規則去闖關。我們是不能夠故意讓遊戲無法順利進行下去，否則就很沒有意義。關卡的設計可以依照我們所進行的測試作為簡單又直接的參考，並且能夠讓我們知道遊戲的困難度要如何做平衡。

有計劃好並且寫入程式的遊戲物件有包括：路線、管線、其他的球體、硬幣、分數、高爾夫球洞、無底洞、彈球緩衝器、以及起始點。

甲、路線

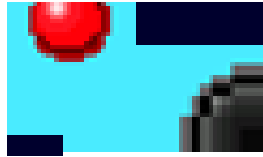


圖 40：亮藍色的區塊是可以滾動的「路線」。

路線是從起始點到高爾夫球洞連結而成的一條可以讓玩家所控制的球過去的道路。路線是我們的遊戲裡最重要的遊戲物件，也是最基本的遊戲物件其中之一。路線其實可以讓玩家去進行最直接的遊戲物件的互動，成為在遊戲裡面扮演著互動的場所這個角色。

在遊戲裡，我們可以利用路線來做為遊戲困難度的指示。如果路線的結構很簡單，就可以視為簡單的路線；反之路線本身則就可以視為困難的路線。

乙、管線

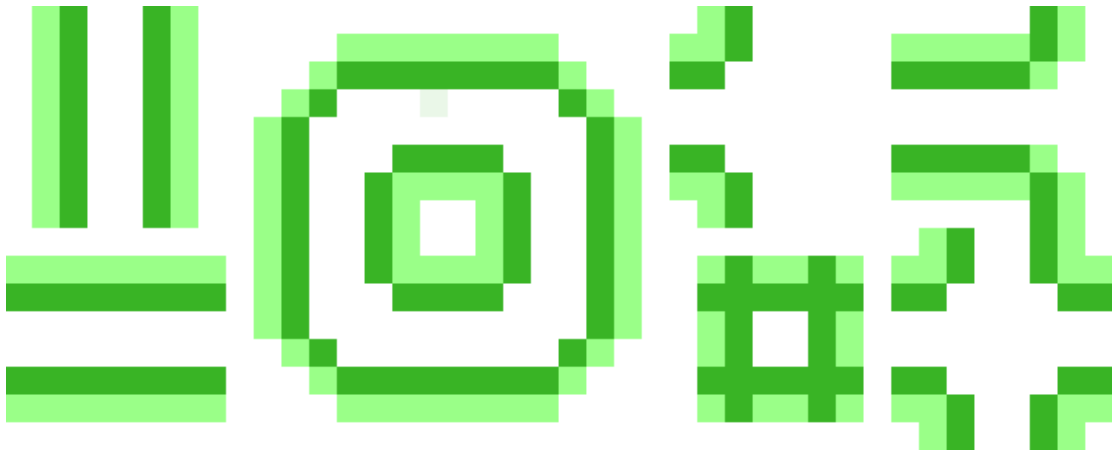










圖 41：管線（放大 16 倍後）

管線一開始是為玩家所控制的球體而用來作為簡單的引導路線。它其實是算是一種方向有所限制的路線，是限制玩家控制的球體以及其他的球體行進的路線。

管線可以細分為筆直管線以及彎曲管線。筆直管線有分成上下與左右的管線。彎曲管線有分成四種：「、、以及。上圖（參考圖 40），彎曲管線、、與是圍繞成一個圓圈。

可惜的是，我們到最後並沒有確切使用到這個程式的部分。它算是未使用到的次要部分，但是有含蓋在主程式原始碼裡面。

一開始的時候，我們只是設定玩家的球體與其他的球體如果通過一道管線，則必須要不能夠從管線的裡面穿過管線的管壁而離開。它是必須要從一個開口的一端前往另一個開口的一端，而且也要保持著速度等等。

我們有發現，筆直的管線要實行於遊戲開發次要的部分的時候，是非常容易的事情。可是，彎曲的管線造就了很多問題，例如：球體本身是要如何不離開管壁但是還可以改變球體的方向。

筆直管線的最佳解決方法就是在球體本身在固定的管線孔徑範圍內使用 `reflect()` 函式。如果球體本身超出孔徑範圍，先使用 `tweak()`^[1] 函式，使得球體向管線的中心軸有一個加速度。然後當球體非常接近管線的中心軸的時候，再切為到使用 `reflect()` 繼續進行球體的移動的運算。

`tweak()` 其實是代表在 `Cue.reflect()` 函式與 `Ball.reflect()` 函式裡面專門是運算在管線的中心球體進行的加速度，而有使用的運算部分。該運算的目的是要讓球體回到管線的中心線。筆直管線的中心線是從開口的中點到另一個開口的中點，而彎曲管線的中心線是從圓心到管線的正中間為半徑的圓弧線。



圖 42：在彎曲管線內，黑色的弧線皆是球體被吸引而去的中線。

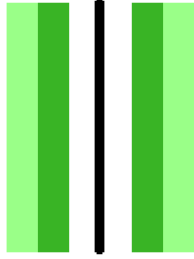


圖 43：在筆直管線內，黑色的直線是球體被吸引而去的中線。

靠這個方式就可以讓球體順利從管線的開口的一端往另一個開口的一端，而不至於因為其他的碰撞的運算造成該球體飛出去的情況。

彎曲管線並沒有一定的最佳解決方式。由於運算上出現無法確定的因素，要精確的去運算出路徑其實是很難的。我們就是基於這個理由而決定我們無法放容易出問題的管線入到遊戲裡面。

丙、球體

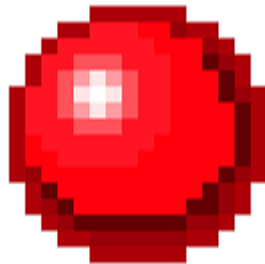


圖 44：玩家的球，顏色為鮮紅色。

每一顆球都具有不同的顏色來做為區分。除了玩家所控制的球體本身是紅色之外，其他有在遊戲關卡裡出現的球體會有不同的顏色，甚至有的時候還會出現近似於紅色的球。

我們一開始是打算要依照我們原先定出來的遊戲主題去設定成能夠用上球體與球體的碰撞當作球體的遊戲互動。同時，我們還可以去從這些碰撞的應用學到新的觀念來增加新的遊戲玩法規則。

我們是後來覺得如果增加了碰撞，就會造成更多要考慮到的額外因素必須要用其他規則來做說明，例如：當球撞上其他的球體，球體對其他遊戲物件的互動反應必須是什麼等等。

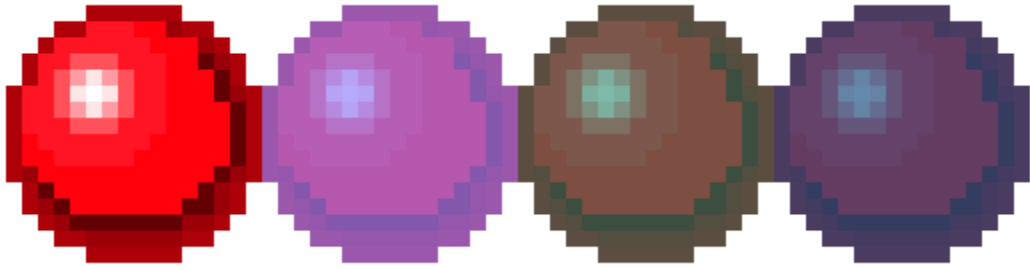


圖 45：玩家的球體與其他球的顏色。

我們並不想要增加新又無意義的規則，不然可能會讓玩家覺得遊戲要去玩它會有一點複雜。當我們覺得用球去對其他的球做碰撞的時候，我們決定不要增加新的碰撞因素而去開始朝向其他方面增加遊戲物件與互動。

玩家自己所控制的球，由於因為三軸感應器不斷的去運算手機上所感應到的傾斜程度之加速度，而無法很順利的去做調整。該三軸感應器所運算出來的數值會直接影響到玩家的球體之速度與加速度。

玩家還可以利用三軸感應器的 Z 軸進行跳躍的動作。我們是先運算出手機向上的位移差有多大。如果位移差的值夠大，我們再考慮看看位移差達到我們定的門檻的時間是不是在短時間內達成。一旦這些條件都滿足，遊戲就會告知球可以用跳的函式 `jump()`。

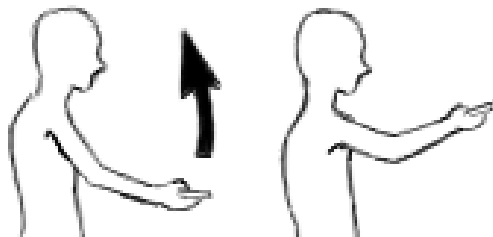


圖 46：使用向上拋的動作即可讓球跳起來。

其他的球體則是由遊戲本身去做運算，是算出該球體的移動方向、速度、以及加速度，再把數值求出來，最後再運算摩擦力來遞減球體本身的速度。

只有玩家的球可以因為碰到無底洞而死亡。

丁、硬幣

硬幣在遊戲裡是扮演著可選擇的目標以及累加玩家的遊戲分數。玩家在觀摩關卡的時候，可以去注意硬幣的位置來告知玩家要前往的方向可能會有陷阱等等會陷害玩家的遊戲物件。

硬幣有四種顏色：黃色、紅色、藍色與綠色。各個硬幣的分數從左到右分別為 200 分，500 分，1000 分，以及最高 2000 分。



圖 47：在遊戲裡會出現的圖片，說明著硬幣的分數值。

當玩家用控制的球體去碰觸到硬幣，硬幣的分數就會自行加入玩家的總分數裡面。不管玩家的球是在地面上或者是在空中，可以取得硬幣而得到該硬幣所持有的分數。

通常分數比較大的硬幣都會在關卡裡面放在比較難取得的地方。這麼做是為了鼓勵玩家運用自己本身的反應能力去嘗試看看能不能挑戰那個關卡的一部分。得到的就是硬幣所帶來的分數作為獎賞。

戊、分數

我們為了要記下玩家闖關卡的時候所得到的分數，是做為玩家與其他玩家做競爭。同時，也可以在遊戲裡提供玩家一個目標，設定了一個目標可以產生出玩家闖關卡的動機以及提升對闖關卡的意願性。

每一個關卡都有時間限制。時間限制會隨著時間而倒數計時。如果要從時間方面得到分數的話，必須在時間還沒有歸零的時候，想

辦法讓玩家的球體引入高爾夫球洞裡面。(高爾夫球洞等一下就會做介紹)

當球一進洞的時候，時間上所剩的剩餘時間會轉成分數，每一秒為 10 分，例如：一分鐘為 600 分。除了時間的分數做計算以外，如果玩家再闖關卡的過程中有取得硬幣，則時間的分數會加上玩家賺到硬幣的分數做為總分累計。

累計完了分數之後，因為玩家有讓球進洞，代表關卡過關，可以進行下一個關卡。原先的分數可以持有到下一個關卡，玩家也可以在下一個關卡繼續累計分數，一直到玩家因為球掉入無底洞而輸了關卡。(無底洞等一下會做介紹)

當玩家輸了，一個跳窗會顯示出來，並且問玩家要不要儲存晚玩家的分數。這個時候，玩家可以選擇要不要數入玩家名稱或者是留著空白，讓名稱設定為預設值“Player”、選擇重玩，再一次取得好一點的成績、或者是選擇離開遊戲，回到遊戲關卡畫面。

己、起始點

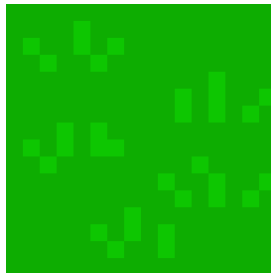


圖 48：起始點 (放大 4 倍後)

起始點是每一個關卡都一定會有的最基本遊戲物件其中之一。起始點的功能主要是標明遊戲關卡的進入點 (entry point)，也是負責告訴玩家一開始的地點是在什麼地方出現。

我們使用的起始點是取用高爾夫球的開球區作為參考，用綠色的草皮顯示出開始的地點。然後，我們先設定玩家球體的座標在起始點的正中間做為預設位置。這麼一來，如果玩家想要重玩關卡，球體可以回到預設位置上。

庚、 高爾夫球洞

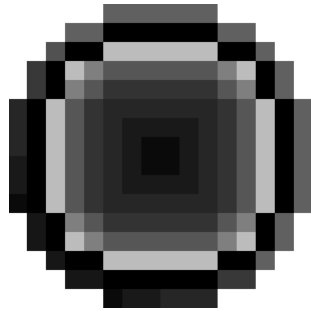


圖 49：高爾夫球洞

高爾夫球洞，或者叫做「終點」，是告知玩家控制自己的球體要讓球滾進的地方。它具有閃爍的光效果，可以凸顯出它的位置是在什麼地方。

在設計上，我們是用高爾夫球的球洞作為參考，並且盡量劃出有隱影效果，再加上白色的閃爍效果。

當玩家的球進入高爾夫球洞裡，會先鎖住玩家的球並且讓手機上的螢幕固定住遊戲畫面，已呈現出遊戲焦點 (focus) 有改變。這麼做是為了不讓玩家在處理使用者介面圖的同時而不影響到遊戲畫面。

為了能夠呈現出逼真的效果，我們加入一些碰撞互動，以至於當球體滾進入洞口裡，球不會因為慣性運動而離開洞口。

辛、 無底洞

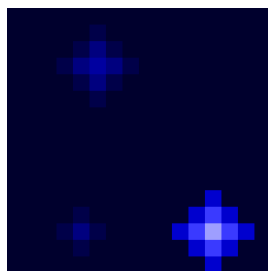


圖 50：四種不同型式的無底洞。(以左上順時針，分別為：行星、空、恆星、月亮)

無底洞是深藍色的背景圖，覆蓋著關卡。它是代表著球的死亡地帶。如果玩家的球體滾到無底洞的時候，就會立刻死亡，必須要重

完關卡或者是離開遊戲。

在無底洞上出現的星星的圖片，是透過亂數抽取 1 至 4，然後去設定出星星的位置以及星星出現的角度。這是讓玩家覺得每一個關卡所產生出來的背景凸顯出很隨興又自然的現象。

我們是設定當玩家死亡的時候，我們是不去計分數。反而，我們是鼓勵玩家盡量達到最高分數再繼續闖關，也因此分數只會重設成闖關前的分數。

壬、彈球緩衝器

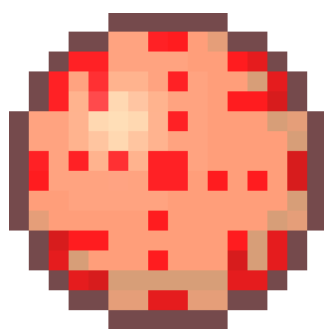


圖 51：未完成圖案 (Work In Progress) 的彈球緩衝器。

彈球緩衝器和管線一樣沒有在遊戲裡使用到。它是扮演著障礙物的角色，是能夠彈走玩家的球，前往沒有一定的方向飛出。

由於遇上三軸感應器所運算出來的數值以及必須要克服碰撞中的反彈運算這兩個問題，使得螢幕在滑動的過程當中滑的不順。我們後來也覺得加入彈球緩衝器可能會造成遊戲複雜化，決定在遊戲裡不採用。

當我們完成了以上這些遊戲物件，遊戲開發主要與次要的部分都分別完成，只剩下不定期作細微的修改、調整等等。

三、進入 Alpha 階段

我們把以上的遊戲元件設計出來之後，我們可以開始將進入下一個階段，並且稱之為 Alpha 階段。在這個階段裡，最主要的就是要完成程式碼與組建軟體這兩個部分。

所謂的程式碼要完成，是要一邊達到固定的週期之內訂出來的進度，一邊要討論出哪些東西可以達到下一個進度，哪些是必須要捨棄掉。我們一開始是決定聚會與討論的時間是固定為一個禮拜。根據我們原先計畫，在這段時間內除了要繼續我們的程式開發之外，同時要掌握好老師所提出的建議和評論，才能夠讓我們去調整計畫。

我們當初討論好的是要使用三種軟體開發模式其中之一：敏捷式軟體開發、Scrum、以及 Test-Driven Development。原本是計畫要使用 Scrum 來進行，可是到後來我們覺得這並不理想。在專題展以前，我們有改成使用 Test-Driven Development，而持續使用到現在。

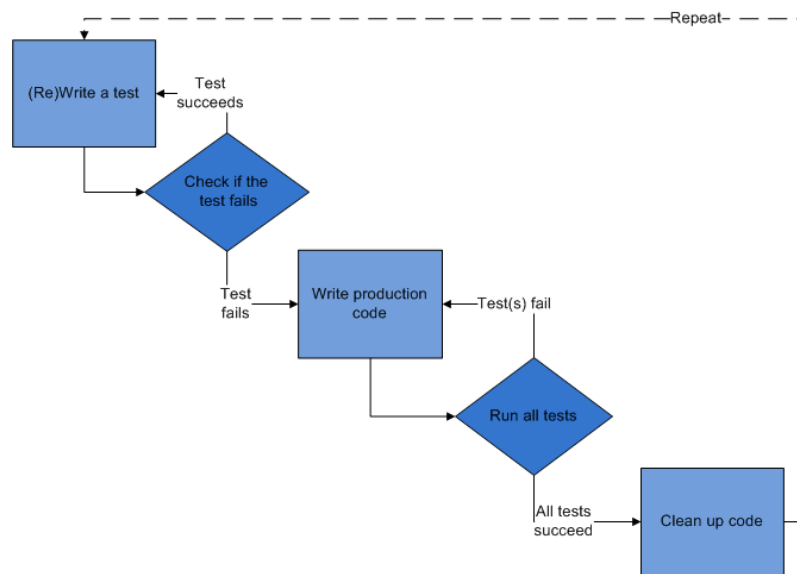


圖 52：Test Driven Development 流程圖。

依照之前提出的計畫當中，我們確實有說要使用 SQLite 當成是分數排行榜的一種，其目的是負責能夠把各個不同的玩家的最高分數紀錄其來，並且透過 Activity 呼叫 SQLite 取得資料。同時，我們可以利用 SQLite 來整理我們的分數排行榜，用分數高低來做為排列順序。

我們之所以要一個分數排行榜的目的是作為比較的工具。當玩家玩了我們的遊戲的時候，我們為了能夠促進玩家要繼續玩下去，就可以用自己所拿取的分數去跟其他玩家的分數作為競爭的媒介，看看是誰的分數最高。

結果，我們不但沒有使用 SQLite 來整理分數與玩家的資料，而

且我們還自己寫出自訂 Activity 去管理資料。我們做這個抉擇是正確的選擇，而且我們沒有後悔過。

我們管理分數的方式是經由我們自己寫出一個 class 類別，該 class 類別是擁有我們會需要在分數排行榜上會使用到的資料。這些資料之後是透過 Android API 裡的 FileOutputStream 存取成一個二位元檔案，非文字檔。我們可以利用 FileInputStream 去閱讀出上述的二位元檔案，然後經由資料的處理，我們就可以把資料放回到分數排行榜，最後再做能夠顯示這些分數的動作。

我們有提及過要在這個階段裡進行關卡地圖的設計。雖然在設計上關卡本身很簡單，要製作出夠載入原先設計好的關卡，再遊戲裡加以整合就會變得很複雜。

由於以上的原因，我們有決定出要把這個部分挪到下一個階段再進行，然後先繼續程式開發。

四、 進入 Beta 階段 (最終階段)

在當時，我們是剛剛進入 Alpha 階段持續著程式的開發。那個時候我們是有提過，當我們達到一定的水準或者是進度方面上我們覺得很滿意的時候，我們就可以成功的邁向 Beta 階段。我們並沒有做 Beta 階段的詳細規劃，只知道我們會盡量把遊戲修正到可以順利的進行遊戲，而不至於在玩的時候出現問題。

在 Beta 階段裡，我們主要是去修正我們的遊戲玩法和遊戲物件上的小細節的改良，也可能要繼續和測試者們做討論。也有提到無論是對評論或獎勵的事情，我們都會很樂意的接受這些回報，並且從這些回報取得可以用來做調整的線索。

測試者們是指無關緊要的、不是專題小組的成員的人、都去玩我們開發的遊戲，並且給我們遊戲方面的回報。至於測試者的個人資料，我們是沒有必要取得這些資料。

我們在 Beta 階段裡有做關卡的設計，全程由一個組員負責關卡設計上的路線以及硬幣擺設的位置。在擺出路線或者是硬幣之前，我們是設定了關卡設計的規則，是要求該組員必須要有起始點、簡單又不至於路線讓關卡造成會卡關的現象、以及高爾夫球洞做為關卡的終點。

我們一開始是想要利用目前遊戲內可能會用到的遊戲物件來設

計出至少 30 個關卡。在設計的時候，我們有發現遊戲物件在關卡裡可能造成遊戲平衡上的問題，也會造成某些特定的關卡會變得更簡單或者是變得更困難。也因為這樣，我們決定縮小預計會設計出來的關卡數目成為 15 個關卡，也決定不盡量使用可能會造成困擾的遊戲物件。

一開始的是在程式裡，我們把遊戲物件一個一個標示著它們的遊戲物件代碼，或者簡稱為「遊戲物件 ID」。這些遊戲物件 ID 是以 ARGB 碼為主。ARGB 碼是一個整數變數，從左到右分別為：透明度數值、紅色數值、綠色數值與藍色數值。每一個數值的範圍是從 0 到 255。

8								8								8								8							
Alpha								Red								Green								Blue							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

圖 53：ARGB 格式

關卡都設計好的時候，我們採用「小畫家」做為地圖編輯器，並且在最後以 PNG 的檔案格式做儲存。畫布是用平均為 10 px 邊長的點陣圖做為關卡的底層。在畫布上，我們主要使用的工具來上色是「筆」，並且用遊戲物件 ID 做為顏色。我們只是畫出一個大小為 1 px 的點 (pixel)。該點所指示的位置就是在關卡裡的座標位置。

關卡的點陣圖儲存成 PNG 的檔案格式的主要目的是為了節省檔案大小。如果儲存成 BMP 檔，檔案大小會很龐大。如果儲存成 JPEG 檔，可能會造成遊戲物件的圖案失真，顏色會變得不是那麼的鮮豔。

要算出座標，是必須要先懂得如何取出一張圖點陣列的資料 (pixel data)。這個資料一旦取到的時候，可以用以下的公式取得座標位置：

$$\text{pixel} = X + (Y \times \text{寬})$$

取得座標位置之後，可以在遊戲裡顯示出遊戲物件，並且順利進行遊戲物理 (game physics) 的運算。

要載入遊戲圖案的時候，必須要使用 Android API 裡的 Bitmap 類別以及 BitmapFactory 類別。其餘的載入圖案的設定都可以經由 Bitmap 做調整。在這裡，我們只是用 Bitmap.createScaledBitmap() 放

大與縮小圖案大小而已。

我們是在進入 Beta 階段的期間內，才決定要製作遊戲音樂，並且載入到遊戲裡進行播放、停止與重播這三個基本動作。前提是，我們所製作的遊戲音樂必須音樂檔案大小要盡可能不要超過 50 kB。

我們使用了可以製作「模塊文件」音樂檔的軟體，叫做「OpenMPT」。不但是可以製作出 8 位元遊戲音樂的軟體，其製作出來的 8 位元音樂檔由於檔案大小非常的小，很適合在檔案大小為前題考量之下進行音樂製作。

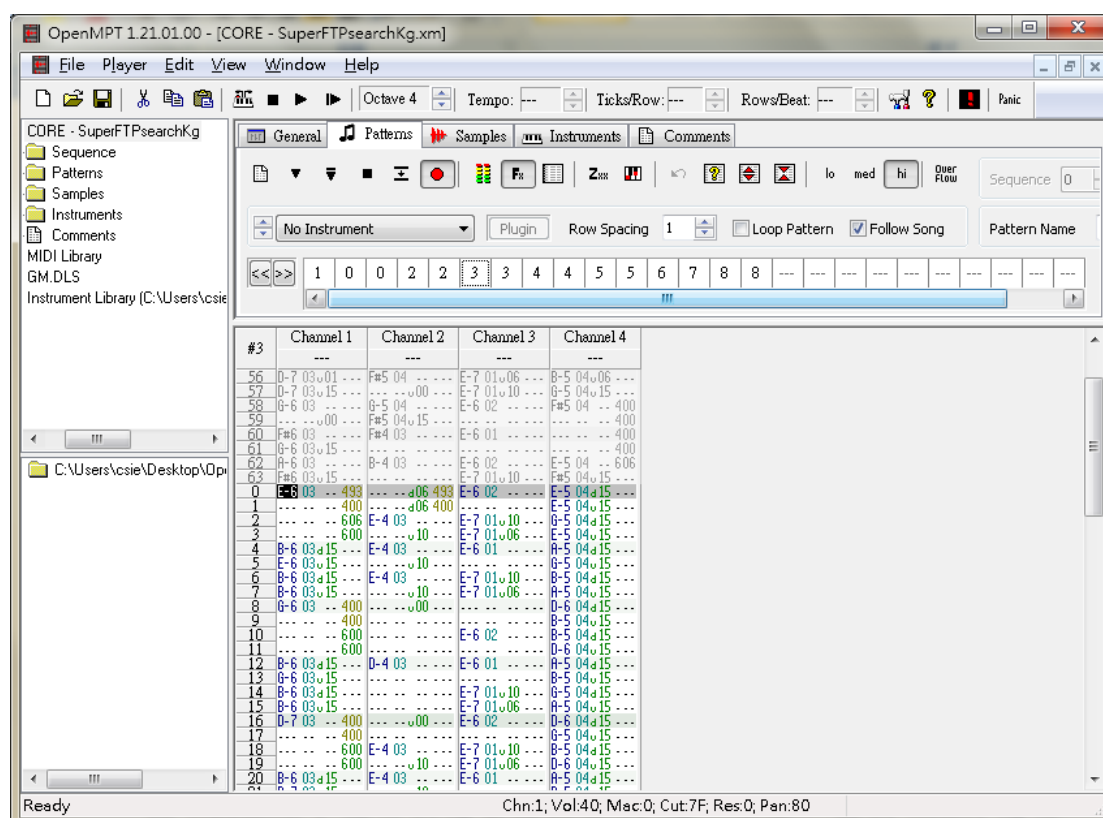


圖 54：OpenMPT 音樂編輯器。

該 OpenMPT 儲存的檔案格式是 IT 檔，不是一般的 MIDI 檔或者是常見的音樂格式檔，所以我們必須要使用不同的方式來播放 IT 格式檔的音樂。

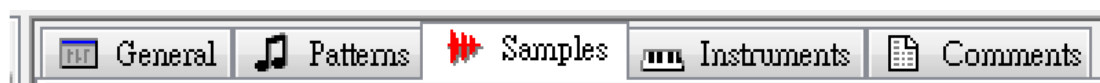


圖 55：OpenMPT 編輯器標籤，可以變更編輯器類型。

要使用 OpenMPT 的 Pattern，最簡單的方式就是在空白處用滑鼠選取一行，然後一個一個去按按看鍵盤上的按鍵。鍵盤就像是鋼琴鍵一樣，排列順序是從 QW...OP[]為低音區，ASD...KL;' 鍵為中音區，ZXC...M,./鍵為高音區。(請參考圖 53) 我們是先練習使用 Pattern 編輯器的模式，好讓我們都可以習慣它的操作方式，之後我們會再回來使用它。



圖 56：音階上的按鍵可以分為三個區域。包含著符號，紅色區域為低音區，藍色區域為中音區，綠色區域為高音區。黑色箭頭指向由低音往高音發出聲音。A 鍵為中音 C (Middle C)。

接下來就是要使用 Samples (取樣編輯器/波形編輯器)。在黑色畫布的上方有一些選項，我們主要是使用「筆」與「建立/增加空間」。(參見圖 57、圖 58 與圖 59) 每一個波需要的是空間，有了空間就可以使用「筆」去做編輯。



圖 57：左邊是用來畫出波形，右邊是建立/增加新空間。

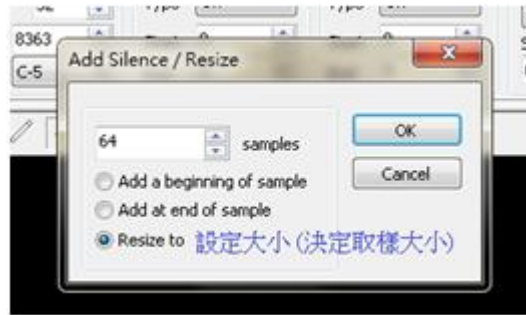


圖 58：選用「建立/增加新空間」來建立。一開始什麼都沒有，所以大小為 0。預設值為 64，單位為 byte。

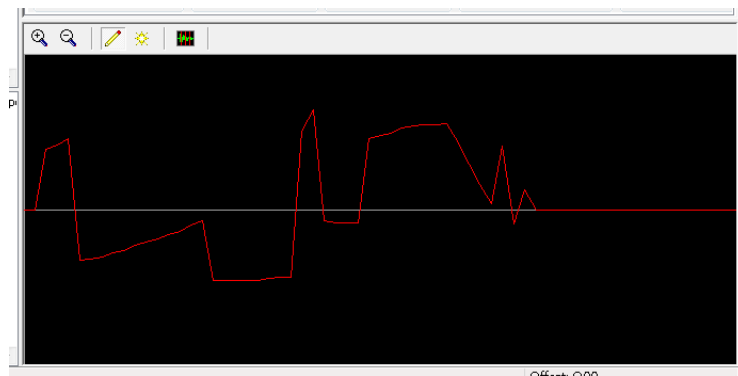




圖 59：使用「筆」的畫面。紅色的線條即為取樣的波形。

在編輯波的形狀，可以用鍵盤來測試看看波的音質是聽起來像什麼。可以用  來停止播放聲音，或者是用中調 C (Middle C) 的音調播出聲音。可以使用  來建立出新的取樣 (samples)，在音樂裡就可以使用更多取樣做為樂器使用。

接下來，我們開始使用 Instruments (樂器編輯器)。使用這個編輯器主要是讓我們可以去控制我們原先畫好的取樣，並且用那些取樣做出特殊的音樂效果。

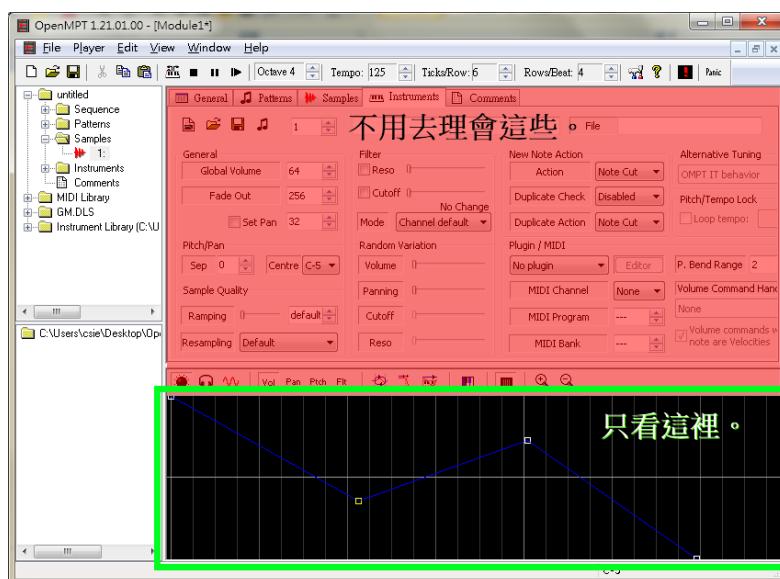


圖 60：Instruments (樂器編輯器)。我們只使用到綠色框以內的部分。

我們用滑鼠在有格子的畫布裡右點，在選單裡選取 Insert Point。我們設定的時間點 (point) 就是決定著取樣的音響 (volume)。我們可以設定許多個時間點，並且用來做出取樣聲音效果。

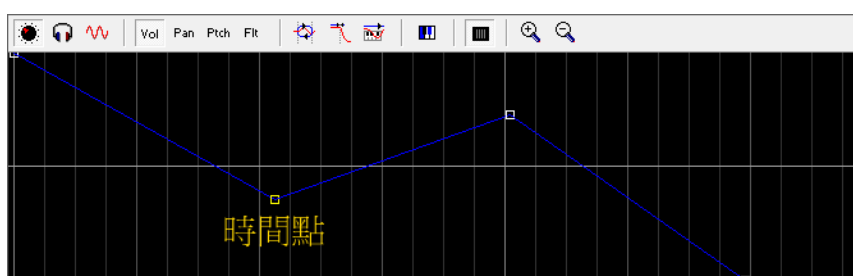


圖 61：插入許多時間點之後的畫面。

進階的控制可以在其中一點上使用 Loop 或者是 Sustain。它們會在時間點上出現灰色的線條，而且是可以滑鼠把效果拖移到我們想要的時間點上。用鍵盤播放出取樣就可以聽到聲音的效果。

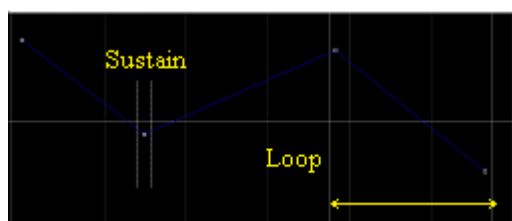


圖 62：Sustain 與 Loop 的差異性

Sustain 是當使用者壓下鍵盤上的按鍵的時候，音符會一直持續播放出聲音。Loop 是當音符播放完聲音的時候，會回到 Loop 所設的起點重複播放到 Loop 所設的終點

當我們設定好了我們要的樂器形式的時候，就可以開始設定哪一種取樣是我們會用我們的樂器播放出來。在樂器編輯器裡，有所謂的 Sample Map (取樣關聯組)，是把我們製作的樂器套用在我們指定的取樣上。這麼一來，當我們要用該樂器，就會用我們所指定的取樣播放出來。

Sample Map		
F-4	F-4	1
F#4	F#4	1
G-4	G-4	1
G#4	G#4	1
A-4	A-4	1
A#4	A#4	1
B-4	B-4	1
C-5	C-5	1
C#5	C#5	1
D-5	D-5	1
D#5	D#5	1
E-5	E-5	1
F-5	F-5	1
F#5	F#5	1
G-5	G-5	1

圖 63：Sample Map (取樣關聯組)

使用 Sample Map 很適合在取樣上添加特殊效果，又同時能夠可以保存原先的取樣的波形。

我們之後用滑鼠右點在 Sample Map 上，選取「Map all notes to sample X」。X 在這裡指目前在 Samples 取樣編輯器裡開啟的取樣。

Edit Sample Map	Shift+Ctrl+E	設定成全部 音符使用指 定取樣。
Edit Sample	Ctrl+E ▶	
Map all notes to sample 1	Ctrl+M	
Map all notes to C-5	Shift+Ctrl+M	
Transpose map up	Ctrl+Q	
Transpose map down	Ctrl+A	
Reset note mapping	Ctrl+R	
Duplicate Instrument	Ctrl+D	

圖 64：我們唯一會使用到的功能。X 這裡為 1，也就是第一個取樣。

最後，我們回到 Pattern 編輯器。我們開始寫下音樂的符號之前，我們必須要先選取一個樂器來使用。在 Pattern 編輯器的上面有一個從這裡可以選取樂器的拉下式選單，稱為樂器選單。如果沒有選取樂器的話，會出現 No Instrument 這個選項，是無法播放出音符 (notes)，只能輸入並且顯示出音符而已。

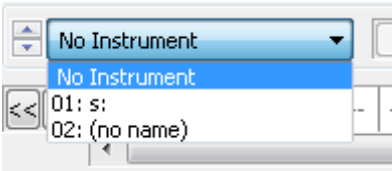


圖 65：樂器選單。

樂器都會有編號，每一個編號代表著取樣的編號。如果有用到樂器編輯器，則這個編號也會對應到那個樂器和取樣的編號。這些編號的用意是要用音符去對應我們使用的樂器。

#0	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6
	---	---	---	---	---	---
	無對應的樂器			對應到樂器1		
0	C-5	==	^^	C-5 01	== 01	^^ 01
1	D-5			D-5 01		
2	E-5			E-5 01		
3	F#5	==	^^	F#5 01	== 01	^^ 01
4	F#5			F#5 01		
5	F#5			F#5 01		
6	D#5	==	^^	D#5 01	== 01	^^ 01
7	D#5			D#5 01		
8	D#5			D#5 01		
9	C-5	==	^^	C-5 01	== 01	^^ 01
10	D#5			D#5 01		
11						
12	D#5	==	^^	D#5 01	== 01	^^ 01
13						
14	D#5			D#5 01		
15	C-5	==	^^	C-5 01	== 01	^^ 01
16	D#5			D#5 01		
17						
18	C-5	==	^^	C-5 01	== 01	^^ 01
19						
20						

圖 66：有無對應樂器的比較。

如果沒有對應到任何樂器 (如圖 66 左半部)，該音符都不會發出聲音。相反之，如果有對應的樂器，會在符號的右手邊出現該樂器的編號。有了這個編號，音符就可以使用樂器播放出我們要的聲音。

除了讓音符都對應到樂器之外，可以用音符去對應到不同的樂器。在這裡本來就沒有去限制樂器只能在一個頻道 (channel) 上使用，這只是為了方便整理整個音樂的製作過程。(參考圖 66)

C-5	01	...
D-5	01	...
E-5	01	...
F#5	01	...
F#5	02	...
F#5	02	...
G#5	02	...
G#5	02	...
G#5	01	...
C-5	01	...
D#5	01	...
D#5	02	...
D#5	01	...
C-5	01	...
D#5	01	...
C-5	01	...
D#5	01	...
C-5	01	...

圖 67：可以混合對應不同的樂器來使用。

在 Pattern 編輯器裡，我們並沒有使用到很複雜的巨集指令 (macro commands)。常用的巨集指令還是會用圖片解釋並且列出來。

0	...	v64	...
1	...	v63	...
2	...	v62	...
3	...	v61	...
4	...	v60	...
5	...	v48	...
6	...	v32	...
7	...	v16	...
8	...	v15	...
9	...	v14	...
10	...	v13	...
11	...	v12	...
12	...	v11	...
13	...	v10	...
14	...	v04	...
15	...	v03	...
16	...	v02	...
17	...	v01	...
18	...	v00	...
19

圖 68：音量。最大值 64，最小值 0。數值代表總音量。

...	v09	...
...	v08	...
...	v07	...
...	v06	...
...	v05	...
...	v04	...
...	v03	...
...	v02	...
...	v01	...
...	v00	...

圖 69：遞減音量。最大值 9，最小值 0。數值代表遞減量。

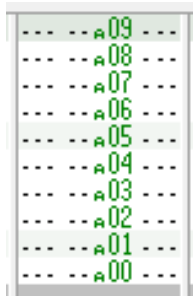


圖 70：遞增音量。最大值 9，最小值 0。數值代表遞增量。

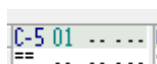


圖 71：Note Fade。用「=」鍵按出。讓音符慢慢淡出。

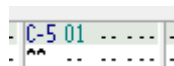


圖 72：Note Off。用「\」鍵按出。讓音符切斷。

其他巨集指令可以參考 OpenMPT Wiki 的 Basic Editor Concepts。

[L2]

要刪除不想要的音符，可以使用 Backspace 鍵去除下面的一行或者是用 Delete 鍵只刪除有選到的部分。

以上是我們使用到的 OpenMPT 就是製作我們遊戲音樂的過程。我在製作遊戲音樂的時候，發現到我們所製作的音樂缺法完整性，因此我們選用其他專業等級的音樂做為我們背景音樂。歌曲名稱為 Ode to Tracker，作者是 SaxxonPike。

最後，我們使用了 Andmodplug^[L3] 這個函式庫來播放出 8 位元音樂，因為這個函式庫是專門處理 IT 檔以及其他模塊文件檔。Andmodplug 可以直接播放出音樂、重播多少次的音樂以及停止播放音樂。















在專案裡把 Andmodplug 這個函式庫整合之後，我們的關卡設計以及遊戲音樂這兩大部份就在 Beta 階段內順利完成。

四、 研究成果

我們開發的遊戲共有 15 個關卡，以下是 15 個關卡的呈現(包含主選單、設定和積分表)。

在關卡的部分，左邊的圖片是關卡經過放大後 1400% 倍，其圖片實際大小為平均邊長為 10 pixel 大之寬與高的四邊形圖片，大小皆不超過 300 bytes。右邊的圖片皆是為在遊戲程式裡所顯示的關卡之真正的面貌。

遊戲物件的 ID 編號由以下附圖做說明，其 ID 編號是 RGB 值：

Bitmap Color Codes:		
INFO: In use with creating level designs in the assets/stage folders.		
Nothing		FFFFFF
Tee		FF0000
Marble		00FF00
Hole		000000
Void		404040
Bumper		FF9E7A
Connector		A8FF1E
Border		FF00FF
Border with Void		FFAAFF
Pipe		8563FF
		FF7580
		56FF9C
		D154FF
		FCFF60
		94FF7C
Funnel		3C706D
		34124C
		380B0D
		30510F
Ramp		4E6587
		443151
		493F30
		415B40
Yellow Coin		CCCC00
Red Coin		CC0000
Blue Coin		0000CC
Green Coin		00CC00

(Pipe，Ramp，Funnel 通通算為遊戲物件中的「管線」。)

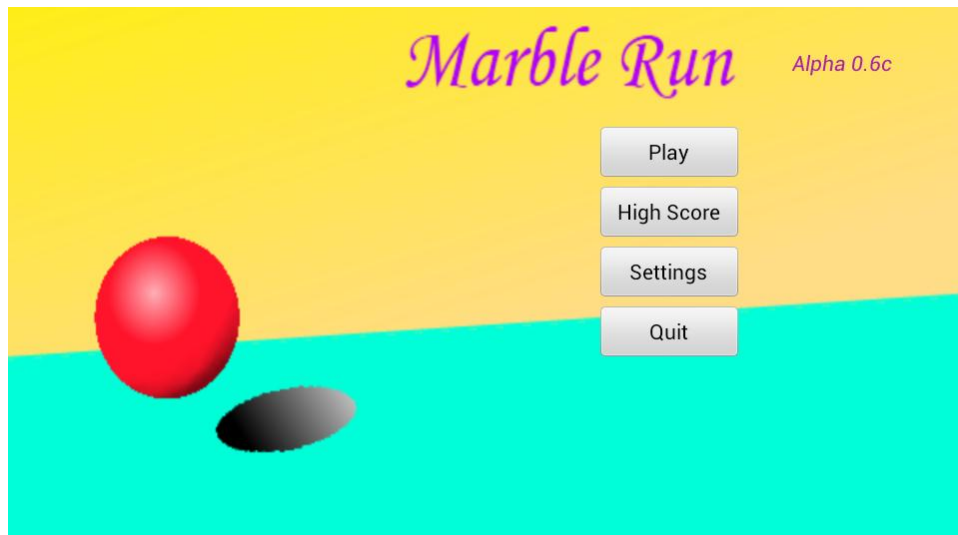


圖 73：主選單。

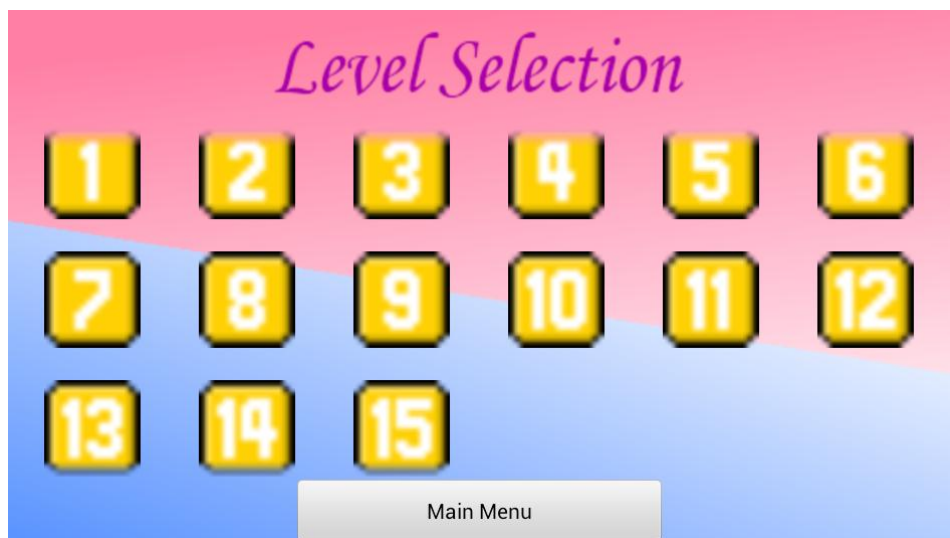


圖 74：關卡選單

Entry #	Run Ended at Stage #	Name	Accumulated Score
1	15		1480

圖 75：積分表

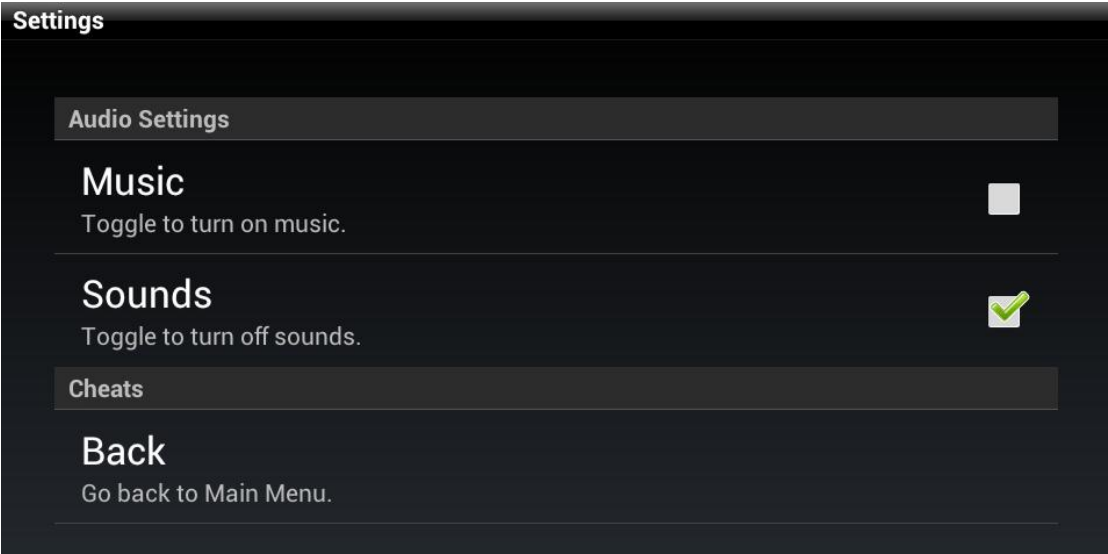


圖 76：設定

關卡的圖片實際大小如下圖：



圖 77a：第一關 (100% 放大率)

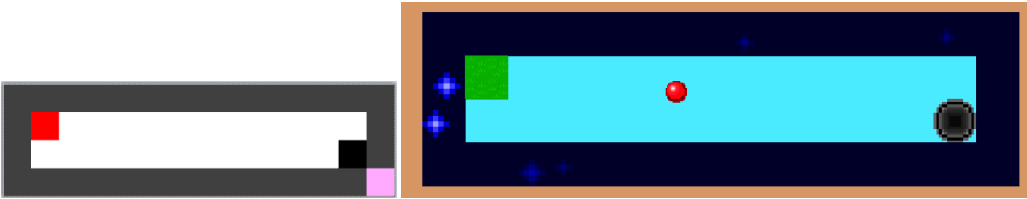


圖 77b：第一關

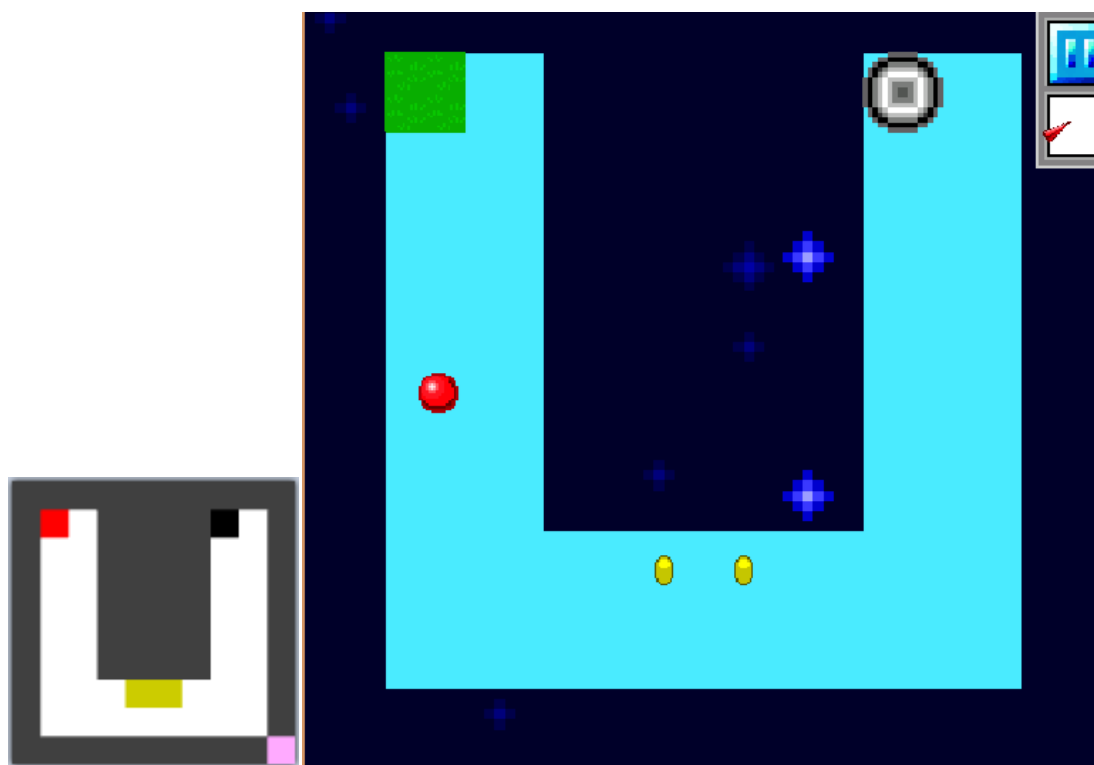


圖 78：第二關

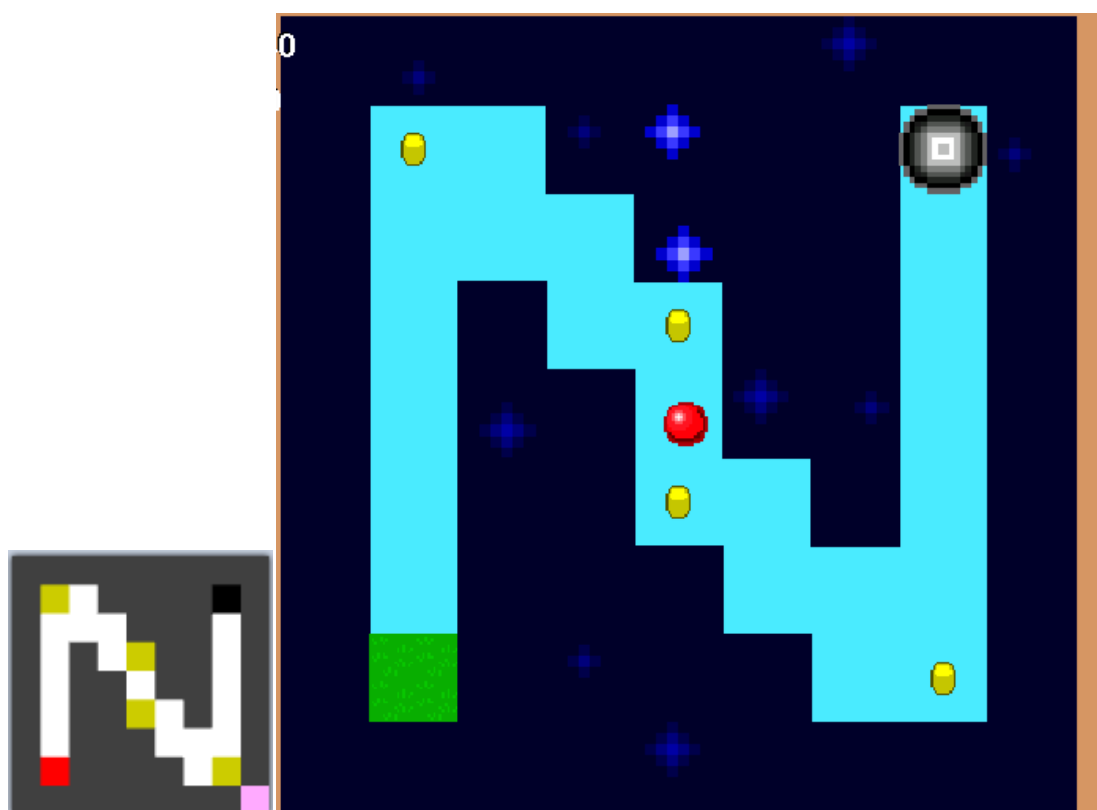


圖 79：第三關

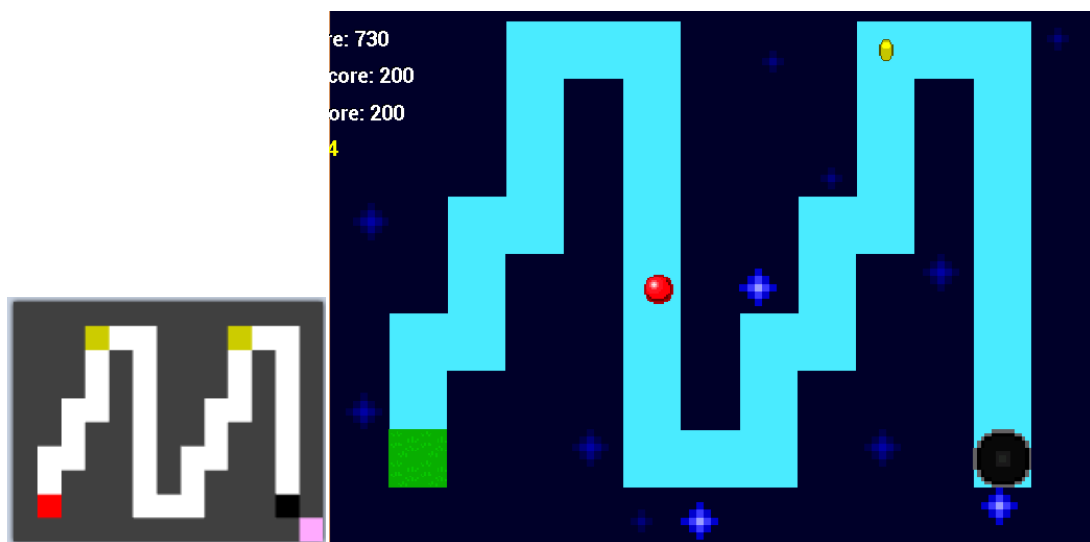


圖 80：第四關

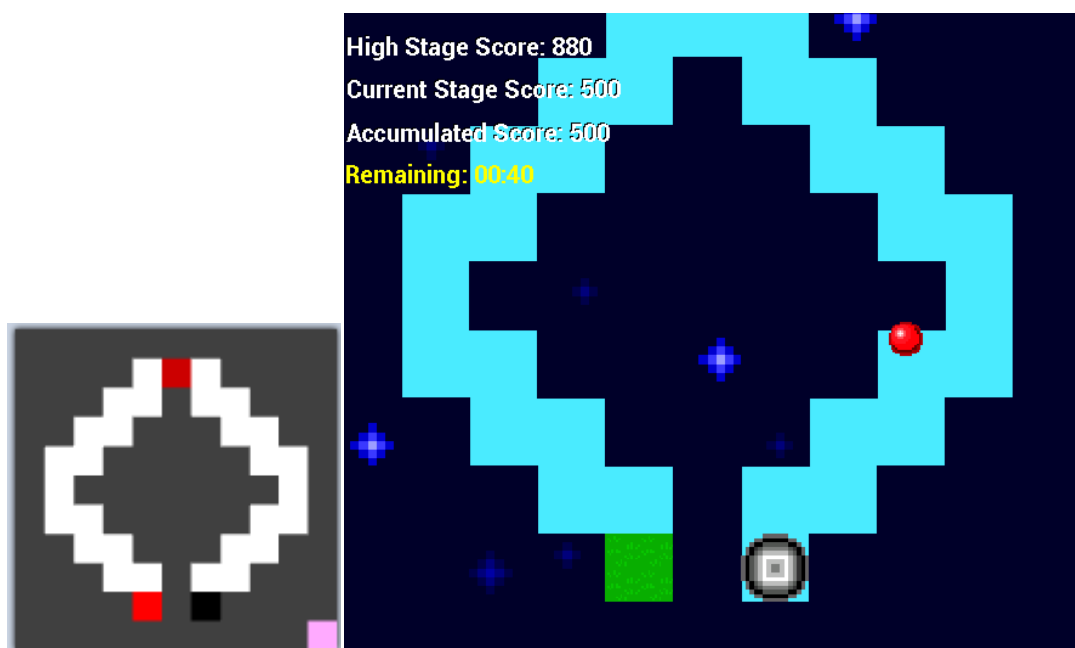


圖 81：第五關



圖 82：第六關

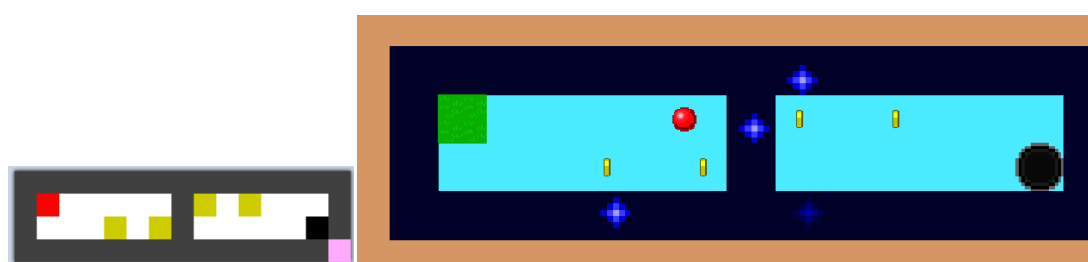


圖 83：第七關

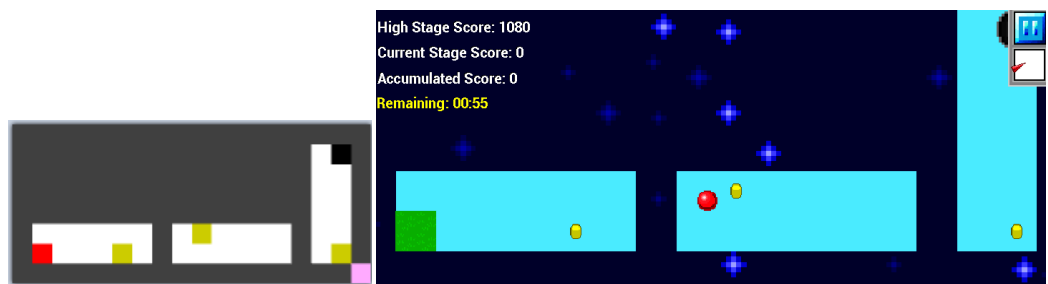


圖 84：第八關

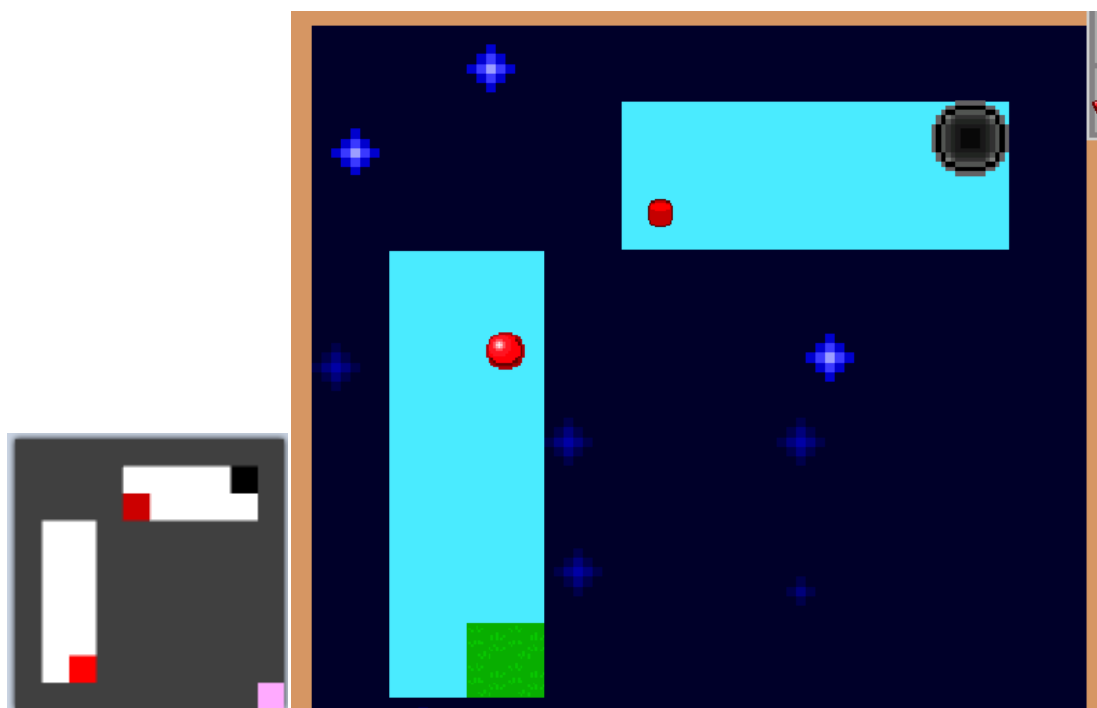


圖 85：第九關

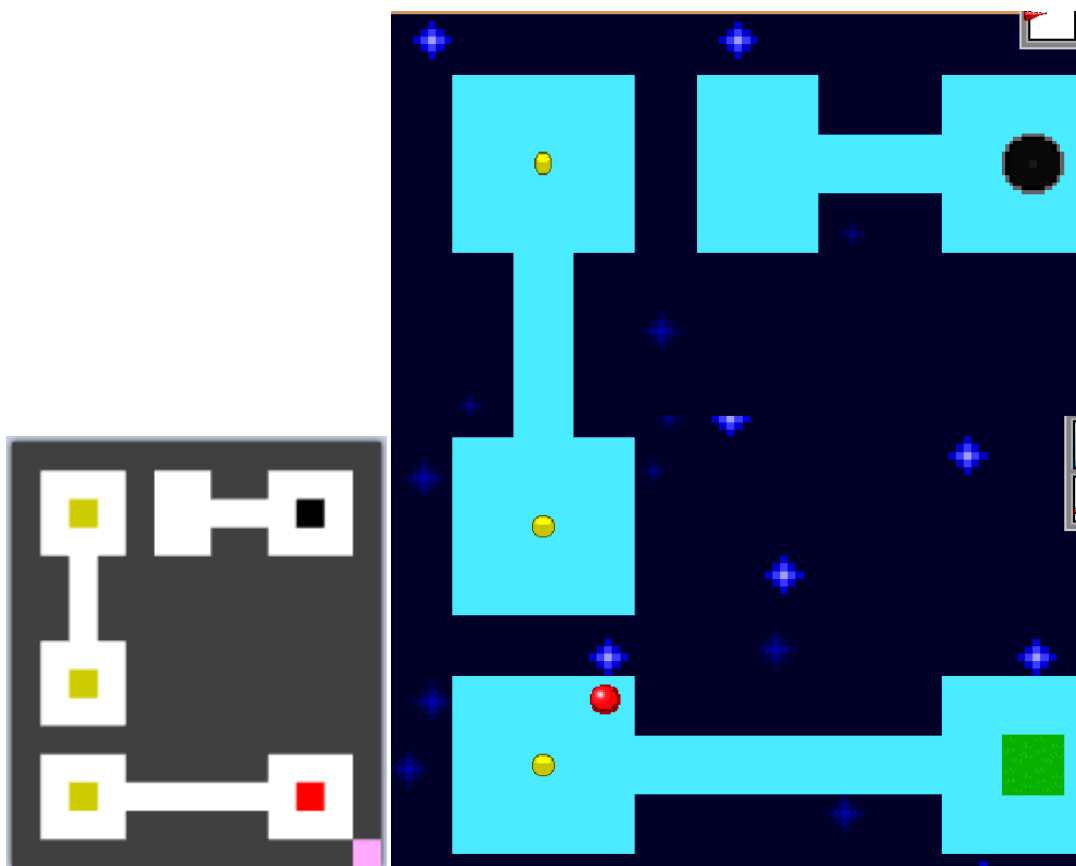


圖 86：第十關

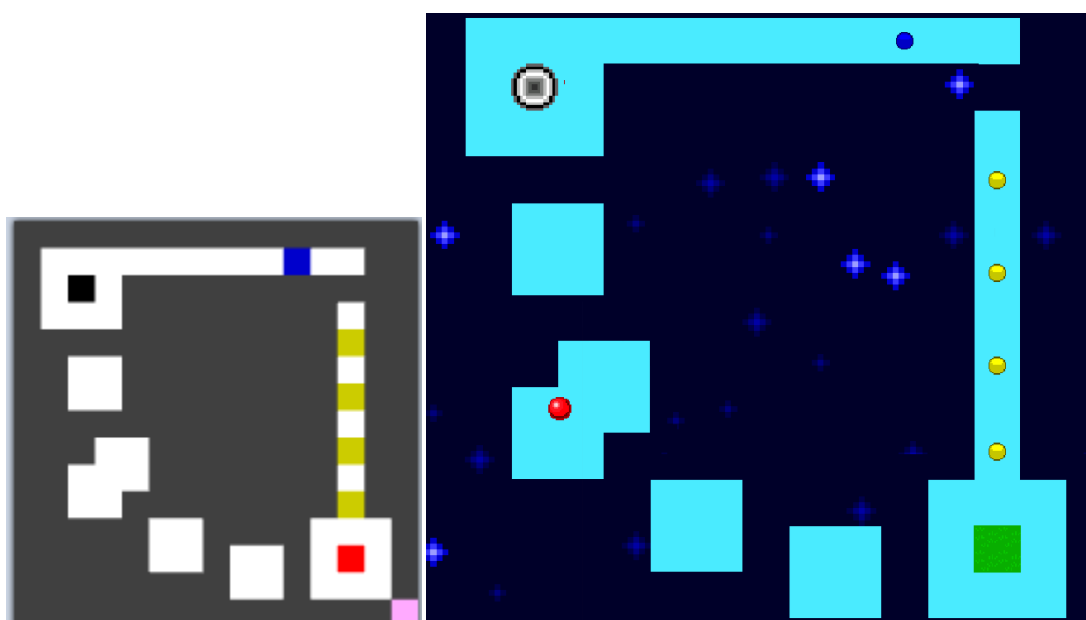


圖 87：第十一關

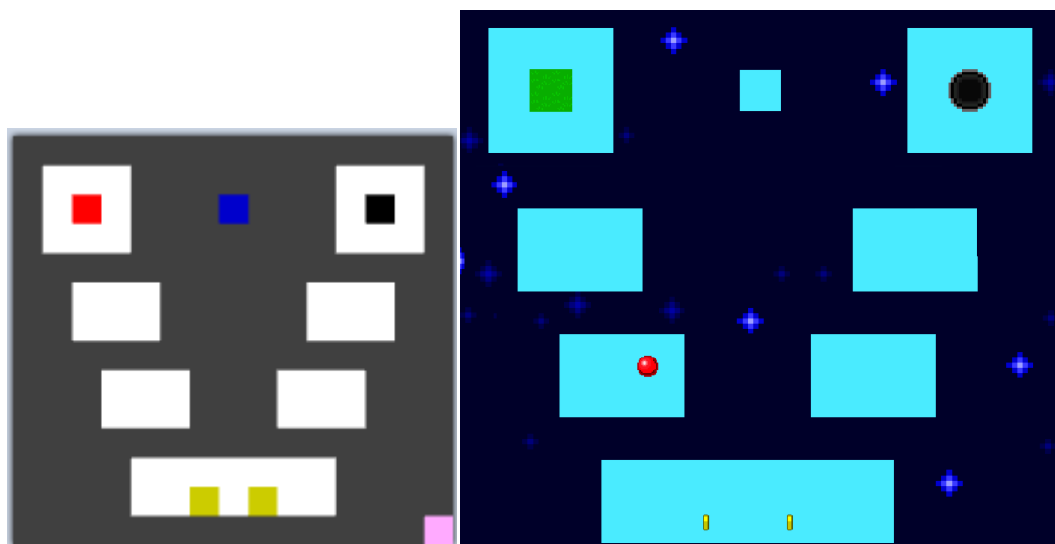


圖 88：第十二關

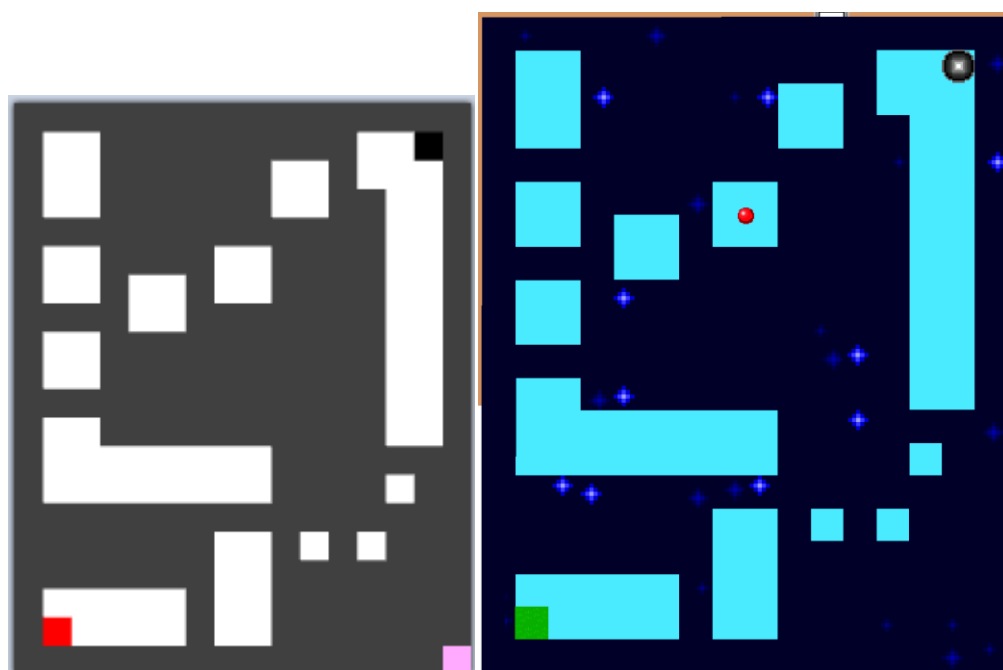


圖 89：第十三關

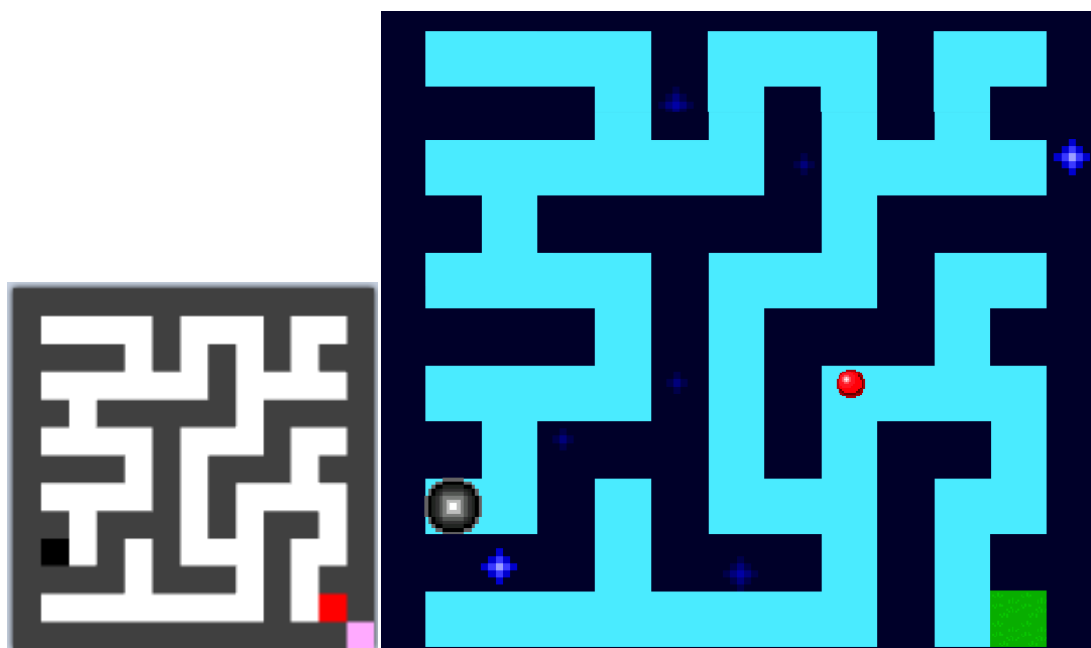


圖 90：第十四關

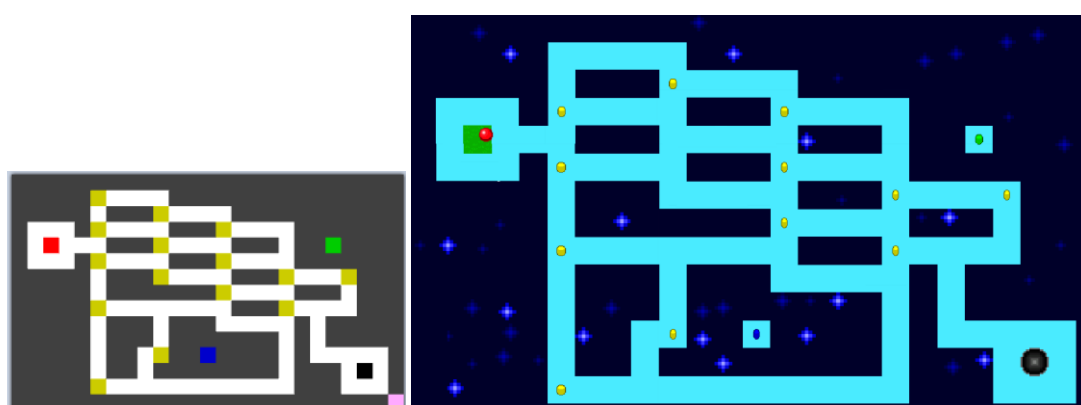


圖 91：第十五關

五、 結論

現在這個社會是個人手一機的時代，走在馬路上智慧型手機已變成當今社會的主流，傳統的手機已面臨被淘汰的命運，就連原本的手機大廠 Nokia 也不得不像智慧型手機低頭。

Nokia predicted to abandon mobile business, sell assets to Microsoft and Huawei in 2013

By Zach Epstein | BGR News – Wed, Jan 2, 2013

[Nokia \(NOK\)](#) appears to have finally turned a corner with the [Lumia 920](#), a smartphone that is seemingly selling quite well after a string of early flops from the struggling smartphone maker. According to Keeskor founder and *Forbes* contributor [Tristan Louis](#), however, [Nokia's](#) recent efforts will ultimately be for naught. 2013 will see a great deal of movement in the mobile industry according to the executive, and Nokia's departure from the handset market will be among the year's most notable events.

[More from BGR: [Can Samsung survive without Android?](#)]

"The biggest shocker (and what I suspect will be my most controversial prediction), though, will be the the [sic] departure of Nokia from the phone business as the company sells its mobile operation and infrastructure divisions to [Huawei](#) in order to focus on software and services," Louis wrote. "With the company's hat on [Windows Phone 8](#) having failed in the marketplace it will see [Microsoft](#)

圖 92：最近報導指出，Nokia 很有可能即將退出手機市場。

我們可以發現坐在捷運、公車、火車等等交通工具上，大家幾乎都是埋頭看機，智慧型手機（尤其是 Android、iPhone 款式的手機）不僅僅是取代舊工具，其實也是在開發出更多新的隱藏需求，例如以前沒想過要邊通勤邊看影片，但現在智慧型手機讓我們開始對這件新的事物上癮。遊戲一直是人們喜愛的娛樂之一，自從 iPhone 的軟體商店成功擴展了手機遊戲市場，並且挑戰傳統掌上遊戲機，徹底的改變玩遊戲的平台。



圖 93：手機市場競爭非常的激烈。

本專題之研究是以開發遊戲為主，我們是針對目前未來最有前景的手機平台進行開發。融入復古的遊戲的想法，以全新的設計，使用手機的三軸感應器來進行遊戲的操作，其中包含「地形」、「物理」、「碰撞偵測」、「材質貼圖」等，讓人有如身臨其境。



圖 94：地形、物理、碰撞偵測與材質貼圖都用上了。

遊戲的內容大致上是個益智遊戲，以一個球為主，球滾進洞口即為過關，球在滾進洞口前會有許多的障礙物，待玩家們一一破關。

這次做的專題中我們一共花了將近一整年從，從一開始與專題老師討論要做甚麼專題到專題定案遞交，最後我們決定要做一款手機遊戲。重開始寫 Java 再到後面的關卡設計、遊戲規則、美術插圖和音樂編輯。

整個說起來我們的開發周期與業界相比似乎在長了。但相對的，這是我們第一次做這種完整的開發案，從會簡單的程式語言、小畫家，到專案在建構時上網查資料買書籍學會了很多 Java、Android 的知識直到專題的完成。

這中間的過程中我們從 Android 2.3.3 版本一路做到官方發布的 Android 4.2，這也不得不讓我們感嘆科技的日新月異。雖然，上面 Android 4.2 的版本向下支援 Android 2.3.3，但是該版本的差異性過於龐大，無法完全被支援到。

起先我們完全沒有想的說手機的螢幕尺寸與系統的微小差異會造成我們專題遊戲完全無法正常顯示，選單是破碎的畫面。對於我們所遇到的錯誤問題，我們還跟同學借了好幾隻不同的手機來做實體測試，是在心苦的時期中除錯階段。

我們做了這個專題讓我們學習到了非常多的東西。靠著我們在網際網路上所遇到其他程式開發者們的社群的幫助，我們應用了很多我們所討論出來的數學函式，使用一些最簡單基本的物理公式，實行

簡單但很複雜的碰撞現象，讓我們覺得這段時間過的很充實，是大學生涯中學到最多東西的時候。

在進行遊戲開發的過程中，我們常常遇上大小不同的問題。剛開始，因為我們這一組都不清楚怎麼使用 Android，不知道怎麼開始進行開發，更不知道要從哪裡開始下手。就在老師的建議下，我們那時候，便決定一起購買 *Android 2.x & Android 3.x 應用程式開發實戰* 這本書籍，期望我們都能夠從這本書裡學到一些東西可以使用。



圖 95：Android 2.x & Android 3.x 應用程式開發實戰。

剛開始的時候，那本書很好用，是讓我們有能夠成為入門等級的程式工程師，也持有一定的基礎能力。從這裡開始，我們可以寫出一些簡單的 Android 程式。

慢慢的，我們開始發現到我們設計好的計劃已經開始脫離了書本上的軌道，就在這個時候我們開始利用上網找資料，並且接觸到網際網路上各各工程師的作品與意見。我們自己使用 Google 就可以找出我們所遇到問題的解答，這也讓我們能夠可以經過調查就可以找出突破困境的好方法，也慢慢習慣不再依賴 *Android 2.x & Android 3.x 應用程式開發實戰* 這本書籍了。

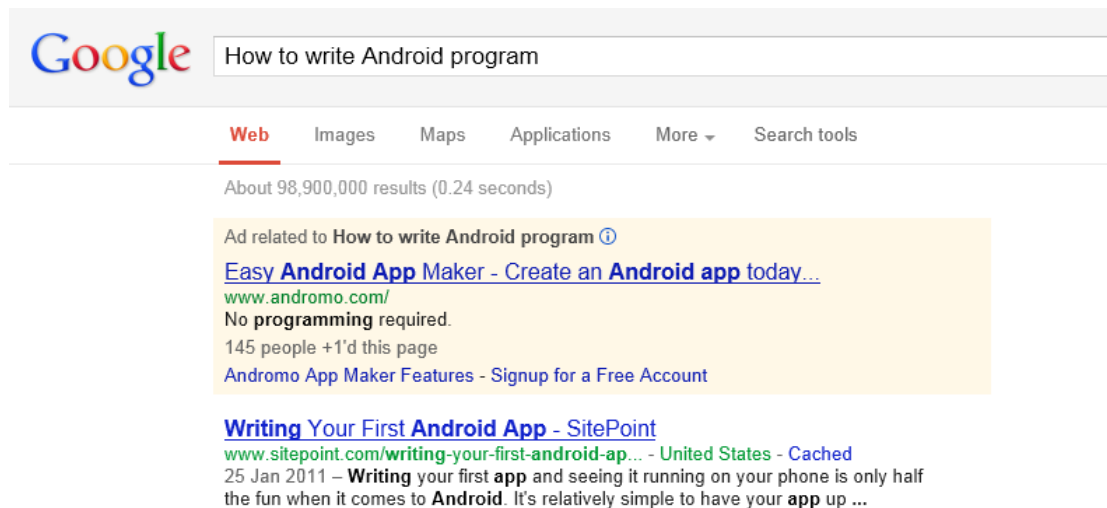


圖 96：使用 Google 搜尋就很容易找到解答。

之後，我們也慢慢發現到我們遇到的問題越來越專門化，不能單單再使用 Google 做簡單的搜尋。這個時候，我們在無意間發現到有 Android 程式問題的共同點都是在 Stack Overflow 問答區出現。我們發現 Stack Overflow 問答區是一個可以問出 Android 以及協助其他人的 Android 問題聚集在一起的專門場所。裡面很多專業的工程師會給你很多意見與幫助。



圖 96：Stack Overflow 問與答面貌。

加入了 Stack Overflow 問答區這個社群之後，我們也慢慢開始

接觸到其他經驗豐富的程式設計師們，也感慨自己沒有多多接觸這些人。我們發出的問題，很快就有人來幫忙解決了，同時還可以問到一些相關的知識與意見可以多加以應用在我們的專題上，讓我們的專題更加豐富。這一點，對我們成功的完成專題是非常有幫助的。

我們就是這樣子，一個一個問題不斷的去問出解答，偶而還會沒有辦法找出問題來。我們所有的問題都會在以下參考文獻裡出現。這些問題都是跟我們在製作的專題程式有關係。

這時我們才發現，我們不就是在做 Test-Driven Development 嗎？既然這樣，我們就想，我們去試試看用這個開發設計的方案結果會是怎麼樣。

經過了一段時間之後，我們的程式終於可以開始讓其他跟專題不相關的人試玩看看我們的遊戲，甚至還被誇獎註明我們是第一組先有一個可以試用的成果。我認為我們的開發進度算是很成功。

得知這一則消息之後，我們有發現到我們專題的進度達到快要飽和的狀態，但是並不是我們做得太快，而是我們做了之後，不知道要不要繼續開發下去。我們其實已經算是到了 Release 的階段了，也就是超出我們原先計畫書上面所寫的計畫了。

再來最重要的是我們的關卡是 BMP 圖檔 (Bitmap) 檔案所做的，所以當初設計的時候有想說可以讓玩家來自行設計關卡，這麼一來大大了提升遊戲的擴充性。未來在做更新時可將自動搜尋 Map 資料夾來增加地圖趣味性與創意性也獲得無限可能。

Release 階段是當遊戲已經達到了一定的水準之後，程式開發者變成專案管理者，是負責決定要不要繼續保持開發或者是管理下去，讓有試玩的人能夠繼續支持專題，並且能夠得知遊戲的最新消息等等。

如果專案管理者（或者是前程式開發者）不願意再繼續開發的時候，就是宣告這個專案終於結案了，不用讓其他的人繼續支持下去專案了。這也說明，程式開發者可以去做其他的專案，而不用再繼續做下去舊的專案了。

我們既然達到這個狀態，不但說明我們的時間空出許多，同時可以開始做其他事情了。我們是在這個階段開始準備考試，申請研究所等等規劃人生的事情了。

當時在做開發的時候完全沒想到說有多道屬不清的限制以及瓶頸。光是一個手機基本的 Back 鍵不能控制就要做 2 個月，而其他人

都可以為何就是我們不行？很可惜最後依然沒解決這個神秘的問題。可以說是這是最難的難題。

Android has a mechanism in place to close an application safely per its documentation. In the last Activity that is exited (usually the main Activity that first came up when the application started) just place a couple of lines in the `onDestroy()` method. The call to `System.runFinalizersOnExit(true)` ensures that all objects will be finalized and garbage collected when the application exits. You can also kill an application quickly via `android.os.Process.killProcess(android.os.Process.myPid())` if you prefer. The best way to do this is put a method like the following in a helper class and then call it whenever the app needs to be killed. For example in the destroy method of the root activity (assuming that the app never kills this activity):

Also Android will not notify an application of the **HOME** key event, so you cannot close the application when the **HOME** key is pressed. Android reserves the **HOME** key event to itself so that a developer cannot prevent users from leaving their application. However you can determine with the **HOME** key is pressed by setting a flag to true in a helper class that assumes that the **HOME** key has been pressed, then changing the flag to false when an event occurs that shows the **HOME** key was not pressed and then checking to see if the **HOME** key pressed in the `onStop()` method of the activity.

圖 97：這說明我們不能使用 Home 鍵的原因。

回想一下，其實我們做了這個專題的時候，除了學會使用我們原先不會的事情之外，還會開始發覺到我們並沒有學會很多。在外面的世界裡，還會一直出現我所不知道的事物，都出現我們值得去學習與探討的事物。這個專題也一樣，是一個很值得去探討。

其中我們覺得很需要探討的是我對於遊戲開發這個方面的經驗，我們感到很缺乏，又不見得說我們真的學會了。我們只是照著人家以前做過的東西，再加以利用與修正成我們自己想要的。我們認為我們還沒有學到要領。還有很大的進步空間。



圖 98：程式再利用。照著人家做，持續學習。

不管怎麼說，我們大家都很努力了也很拼了，要是能夠繼續開發下去會比較好。我們如果沒有「專題已經達到水準了」這個想法，或許還可以不斷的去學習其他新的又特別的知識等等。

在這個猶豫期間專題展來了，我們要負責把我們製作好的程式拿出來給人家看看。我們體會到觀賞的人們看了我們的程式，經過了

遊戲介紹與解釋遊戲規則之後就一目了然，也知道我們在這個遊戲裡做了什麼東西。我們不必講下去，人家就開始會玩我們的遊戲，這讓我們覺得我們是做到了，很成功的把比較困難的障礙一次破壞掉。

我們也是在這裡展開了我的夢想的一部分，也有達算繼續下去這一條路。

六、 參考文獻

甲、 圖片

圖 1: 開放手機聯盟 -

<http://zh.wikipedia.org/wiki/%E9%96%8B%E6%94%BE%E6%89%8B%E6%A9%9F%E8%81%AF%E7%9B%9F>

圖 2: Google - <http://zh.wikipedia.org/wiki/Google>

圖 3: Googleplex -

<http://atlantablackstar.com/2012/06/04/google-takes-number-1-on-the-top-15-list-of-mba-employers/googleplex/>

圖 4: Android - <http://zh.wikipedia.org/wiki/Android>

圖 5: Google Play - http://zh.wikipedia.org/wiki/Google_Play

圖 6: Android 平台的整體架構 -

<http://www.dmxzone.com/go/14339/google-android-sdk-released/>

圖 7: Android Emulator -

<http://developer.android.com/tools/devices/emulator.html>

圖 8: Apple -

<http://zh.wikipedia.org/wiki/%E8%98%8B%E6%9E%9C%E5%85%AC%E5%8F%B8>

圖 9: Steve Jobs -

<http://zh.wikipedia.org/wiki/%E5%8F%B2%E8%92%82%E5%A4%AB%C2%B7%E4%B9%94%E5%B8%83%E6%96%AF>

圖 10: Apple II - http://en.wikipedia.org/wiki/Apple_II

圖 11: Apple vs Android -

<http://www.oneclickroot.com/mobile-oses/apple-ios-6-versus-android-how-do-they-compare/>

- 圖 12: Market Share 2012 -
<http://thenextweb.com/microsoft/2012/09/01/windows-7-finally-over-takes-windows-xp-mac-os-x-overtakes-windows-vista/>
- 圖 13: Windows 8 Phone -
http://zh.wikipedia.org/wiki/Windows_Phone_8
- 圖 14: TortoiseSVN - <http://tortoisesvn.net/>
- 圖 15: WinCVS - <http://mindprod.com/jgloss/wincvs.html>
- 圖 16: Java - http://www.java.com/zh_TW/
- 圖 17: SQLite - <http://www.sqlite.org/>
- 圖 18: Activity workflow -
<http://developer.android.com/reference/android/app/Activity.html>
- 圖 19: Sensor -
http://developer.android.com/guide/topics/sensors/sensors_overview.html
- 圖 20: Android Developers - <http://developer.android.com/index.html>
- 圖 21: Stack Overflow - <http://stackoverflow.com/>
- 圖 22: Gamedev.net - <http://www.gamedev.net/page/index.html>
- 圖 23: Scrum - [http://en.wikipedia.org/wiki/Scrum_\(development\)](http://en.wikipedia.org/wiki/Scrum_(development))
- 圖 24: Black and White -
[http://en.wikipedia.org/wiki/Black_and_White_\(game\)](http://en.wikipedia.org/wiki/Black_and_White_(game))
- 圖 25: Lunar Lander -
http://cn.appszoom.com/android_games/casual/_osgt.html
- 圖 26: Music Production -
<http://penny-arcade.com/report/editorial-article/a-years-income-comp-osed-on-a-banjo-the-man-behind-ftls-music>
(Photo Credit: Ben Prunty - Penny Arcade Report)
- 圖 28: Using XML in Android Eclipse ADT plugin -
<http://mobile.tutsplus.com/tutorials/android/android-layout/>
- 圖 29: Screenshot of Android References -
<http://developer.android.com/reference/packages.html>
- 圖 30: Android Manifest – screenshot from HelloWorld example in Sample SDK.
- 圖 31: SurfaceView -
<http://android-er.blogspot.tw/2010/05/android-surfaceview.html>

- 圖 32: Paint.NET - <http://www.getpaint.net/>
- 圖 33: VirtualSVN Server -
http://en.wikipedia.org/wiki/VisualSVN_Server
- 圖 52: Test Driven Development -
http://en.wikipedia.org/wiki/Test-driven_development
- 圖 53: RGBA color space -
http://en.wikipedia.org/wiki/RGBA_color_space
- 圖 56: Apple keyboard - <http://support.apple.com/kb/HT1171>
- 圖 92: <http://news.yahoo.com/nokia-predicted-abandon-mobile-business-sell-assets-microsoft-145504867.html>
- 圖 93: <http://www.guardian.co.uk/business/2012/aug/20/4g-mobile-internet-everything-everywhere>
- 圖 95: <http://shopping.gotop.com.tw/showroom/view.php?C=5267700>
- 圖 97: <http://stackoverflow.com/questions/2092951/how-to-close-android-application/5036668#5036668>
- 圖 98: <http://www.alexanderinteractive.com/blog/2009/03/reuse-code-and-write-reusable-code/>

乙、連結

- [L1]: 手機程式設計 - <http://daimajishu.iteye.com/blog/1080748>
- [L2]: Google (Wikipedia) - <http://zh.wikipedia.org/wiki/Google>
- [L3]: Android (Wikipedia) - <http://zh.wikipedia.org/wiki/Android>
- [L4]: Google 的 Android 手機平台 -
<http://sp1.wikidot.com/googleandroid>
- [L5]: Using the Android Emulator -
<http://developer.android.com/tools/devices/emulator.html>
- [L6]: 蘋果公司 (Wikipedia) -
<http://zh.wikipedia.org/wiki/%E8%98%8B%E6%9E%9C%E5%85%AC%E5%8F%B8>
- [L7]: 版本控制 (Wikipedia) - <http://zh.wikipedia.org/wiki/Subversion>
- [L8]: Java (Wikipedia) - <http://zh.wikipedia.org/wiki/Java>
- [L9]: SQLite (Wikipedia) - <http://zh.wikipedia.org/wiki/SQLite>
- [L10]: *Android 2.x & Android 3.x 應用程式開發實戰*.

[L11]: Activity | Android Developer -

<http://developer.android.com/reference/android/app/Activity.html>

[L12]:

[L13]: Basic Editor Concepts -

<http://openmpt.xwiki.com/xwiki/bin/view/Manual/PatternEditor>

[L14]: Andmodplug -

<http://www.peculiar-games.com/libmodplug-in-android/andmodplug>

丙、專題使用的參考

[P1]: 中文 Android API,

<http://www.cnblogs.com/over140/category/277077.html>

[P2]: 在開始撰寫 Android Application 之前 —

<http://android.cool3c.com/article/3886>

[P3]: 昭佑.天翔: Android Emulator 變更顯示語系 —

http://tomkuo139.blogspot.com/2009/07/android-emulator_5723.html

[P4]: 昭佑.天翔: Android Emulator 透過 Proxy Server 上網 —

<http://tomkuo139.blogspot.com/2010/05/android-emulator-proxy-server.html>

[P5]: 程式天堂 - Android 應用開發・研究・與諮詢 —

<http://ysl-paradise.blogspot.com/>

[P6]: 孫傳雄談科技的 Android 教學,

<http://blog.chinatimes.com/tomsun/archive/2010/09/16/539502.html>

[P7]: Android Reference -

<http://developer.android.com/reference/packages.html>

[P8]: Android 維基

[P9]: <http://zh.wikipedia.org/wiki/Gphone>

[P10]: Android 使用者論壇網站 —

<http://ysl-paradise.blogspot.com/2008/11/android.html>

[P11]: Android beginner 的初學 Android,

<http://tyroandroid.blogspot.tw/2009/07/android-activity.html>

[P12]: Android 程式開發相關連結 —

<http://ysl-paradise.blogspot.com/2008/06/android.html>

[P13]: Android 手機安裝、解除安裝 APK 教學 —

<http://www.and-machine.com/viewthread.php?tid=74>

[P14]: Android 官方網站 <http://www.android.com/>

[P15]: Android 3D 遊戲開發——Opengl ES 遊戲實現

<http://www.hztraining.com/bbs/showtopic-26.aspx>

[P16]: Android: I have created a multi-threaded Looper game loop.

Please give feedback! -

<http://www.gamedev.net/topic/629355-android-i-have-created-a-multi-threaded-looper-game-loop-please-give-feedback/>

[P17]: AlertDialog: After showing the dialog, background Canvas rendering continues, but the dialog never close -

<http://stackoverflow.com/questions/11203662/alertdialog-after-showing-the-dialog-background-canvas-rendering-continues-but>

[P18]: Ball to Circular Wall Interior Collision Response -

<http://www.gamedev.net/topic/628261-ball-to-circular-wall-interior-collision-response/>

[P19]: Bouncing Ball on the Z axis: I can't fix a math error without help.

-

<http://www.java-gaming.org/topics/bouncing-ball-on-the-z-axis-i-can-t-fix-a-math-error-without-help-gif/27225/msg/243420/view.html>

[P20]: Canvas: How do you make the Canvas as a chase camera for moving objects? -

<http://www.java-gaming.org/topics/canvas-how-do-you-make-the-canvas-as-a-chase-camera-for-moving-objects/27210/msg/243196/view.html>

[P21]: Circle to Polygon Collision Detection and Response: Unable to successfully move the circle at inner corners of a bend -

<http://stackoverflow.com/questions/12006775/circle-to-polygon-collision-detection-and-response-unable-to-successfully-move>

[P22]: Gravity hole: Objects moving near a hole, shifts its projection towards the hole -

<http://www.java-gaming.org/topics/gravity-hole-objects-moving-near-a-hole-shifts-its-projection-towards-the-hole/26993/msg/239721/view.html>

[P23]: How do you allow objects to only move in opposite direction of a vector? Can it be applied to boundaries? -

<http://www.gamedev.net/topic/632353-how-do-you-allow-objects-to-only>

-move-in-opposite-direction-of-a-vector-can-it-be-applied-to-boundaries/
[P24]: How to reflect an object, with constant acceleration, when hitting a circle? -

<http://www.java-gaming.org/topics/how-to-reflect-an-object-with-constant-acceleration-when-hitting-a-circle/27582/msg/247471/view.html>

[P25]: Is it possible to speed up initializing AlertDialogs in an Activity subclass? -

<http://stackoverflow.com/questions/12067396/is-it-possible-to-speed-up-initializing-alertdialogs-in-an-activity-subclass>

[P26]: Java: Looking for tips on Square to Square stationary collision response -

<http://www.gamedev.net/topic/629899-java-looking-for-tips-on-square-to-square-stationary-collision-response/>

[P27]: Java-Gaming.org - <http://www.java-gaming.org/>

[P28]: Point Inside Circle Collision Response: How do you keep the Point inside of the circle? -

<http://stackoverflow.com/questions/11581738/point-inside-circle-collision-response-how-do-you-keep-the-point-inside-of-the/11739066#11739066>

[P29]: SoundPool: Error creating AudioTrack -

<http://stackoverflow.com/questions/9599059/soundpool-error-creating-audio-track/9724138#9724138>

[P30]: Stack Overflow - <http://stackoverflow.com/>

[P31]: SoundPool Broken Method: load(String path, int priority) -

<http://www.gamedev.net/topic/631244-solved-soundpool-broken-method-loadstring-path-int-priority/>

[P32]: SurfaceView, Canvas, and Threading: Would my Java-esque concept work out well? -

<http://www.gamedev.net/topic/621829-surfaceview-canvas-and-threading-would-my-java-esque-concept-work-out-well/>

[P33]: Need help on calculating the reflection of a point hitting a circle from inside. -

<http://www.java-gaming.org/topics/need-help-on-calculating-the-reflection-of-a-point-hitting-a-circle-from-inside/26923/msg/238773/view.html>

[P34]: 2D Vector Acceleration: Trying to make an object stay on the

center line. -

<http://www.gamedev.net/topic/632267-2d-vector-acceleration-trying-to-make-an-object-stay-on-the-center-line/>

[P35]: 2D Coordinate Determination: How to determine coordinates from a line intersection? -

<http://www.gamedev.net/topic/632275-2d-coordinate-determination-how-to-determine-coordinates-from-a-line-intersection/>

[P36]: 2D Bitmap Matrix manipulation: How does one flip a bitmap? -

<http://www.gamedev.net/topic/631757-2d-bitmap-matrix-manipulation-how-does-one-flip-a-bitmap/>