Computer Organization and Assembly Language

Final Project

Keroro Invader

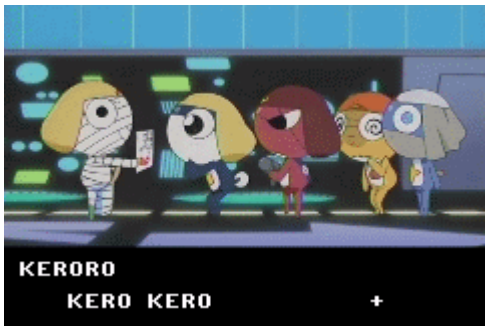
B95505013 李冠輝

B94602018 高浩軒

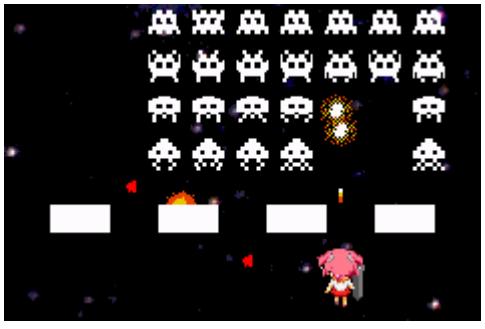
一、遊戲簡介

太空侵略者，是一款日文原名スペースインベーダー在 1978 年在日本由太東公司發行的街機遊戲，意思就是 Space invader，由西角友宏設計。常簡稱為侵略者，或翻做宇宙入侵者。西角友宏設計此遊戲時，原本的主角設計為坦克，但限於當時電腦運算速度相當慢，因此坦克移動並不順暢，才改為太空大戰的形式，太空侵略者的遊戲規格是基於 Intel 公司 8 位元 2Mhz 的 Intel 8080 處理器而設計的。此遊戲在日本與美國都廣受歡迎，在當時日本下町的廉價咖啡廳內，總可以見到大量的 Space invader 機台，無論何時總是聚集著大人小孩，足以見得當時的人氣之盛。Space invader 是影響電腦遊戲歷史的重要遊戲之一，其射擊，關卡與角色的設計在當時都相當少見，特別是在 1978 年的射擊類遊戲中，太空侵略者可說佔有相當大的領先地位。之後這個經典的遊戲也以數不清的次數被復刻（我們又作了一次）。有以 space invader 為名的暢銷歌曲，也有以 space invader 為主題的巨大壁畫，提到遊戲，space invader 可說是無人不知無人不曉。而以這個遊戲系統的完整度、人氣以及遊戲性來說，我們幾乎可以說 Space invader 是電子遊戲的最重要里程碑，因為它從此開啟了遊戲工業時代。

二、設計內容

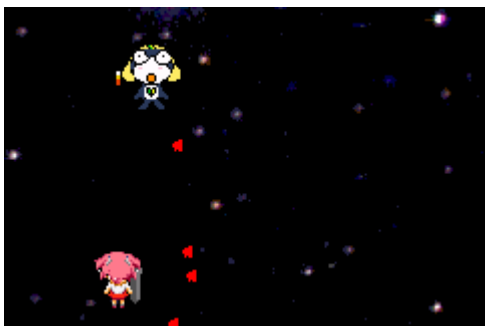


本遊戲之實作配置和傳統之入侵者相當相像：由上方數排之敵軍、中間偏下方之阻擋牆及下方單槍匹馬之我軍組成。遊戲中無計分制度，遊戲之最終目的為擊敗兩隻魔王通過訓練。本遊戲設計兩個關卡，並由開頭之劇情導入，每一關都由 4x7 之敵軍組成，越上層的敵軍越難擊敗，而擊敗所有敵軍之後，會有攻擊頻率較高、較難擊敗的魔王出來迎擊。



本遊戲以左右方向鍵控制方向、Start 鍵控制劇情流程、Z 鍵發射子彈。主角發射之子彈未消失於畫面中之前，無法發射下一發子彈，而當主角在一個關卡中被擊中太多次時則會死亡而導致遊戲結束。

在遊戲中也加入了獨特風格的圖像，結合了傳統太空入侵者的敵人和動畫 keroro 軍曹的設計，使得玩家在進行遊戲時也能有不同的趣味；而也引入了較為逼真的爆炸效果，使遊戲更具真實感。



三、實作技術

本遊戲架構部份採用 HAM Library 進行實作。HAM Library 提供了相當多方便的功能進行呼叫。但在實際進行遊戲開發時，進行過多的呼叫會使效率降低。且在遊戲進行過程中，有許多細碎的判斷和運算，如子彈的分配、碰撞判斷、勝負判斷等。

在射擊遊戲中常見的碰撞判斷問題，本實作採用分段判斷的方式。即將所有需碰撞判斷的物件分為數堆，每一個 cycle 只判斷其中的一堆，經仔細計算距離使子彈不致於在判斷的空檔再度飛離目標物之後，此方法確實可省下不少執行時間。尤其在寫判斷的初期，過多的碰撞判斷常使得遊戲在子彈滿天飛的情況下執行不順暢，使用此法可有效讓執行速度提升。

而子彈的分配也是一個大課題。由於在 GBA 的架構中，由硬體協助控制的 sprite 只能容納固定的數量，故不可能每一個敵人想要發射子彈的時候，就分配一顆子彈給他，如此可能造成記憶體不足的情況。在本實作中採用的方式是，不讓敵人自主決定要發射子彈，而是全場發配固定數量的子彈，讓子彈以亂數決定自己何時要被發射出去，再讓子彈以亂數方式決定由哪一位敵軍發射，如此便可以克服子彈數量可能突然不夠的情況。而爆炸的動畫亦是隨需求而來，但卻不能讓爆炸決定自己何時要發生，而要在一排可用的爆炸動畫中找到目前未在進行中的動畫，也會浪費許多運算時間。因此在爆炸動畫的實作中，分配了一個 stack 來存放目前非使用中的動畫。有地方發生爆炸就從這個 stack 中取用，而爆炸結束之後就放回該 stack 當中，使尋找可用爆炸動畫的過程變得快速。

另外，由於 GBA 的記憶體相當小，若在執行期間即開許多陣列將所有的圖形資料放入記憶體中，則勢必會發生記憶體不足的情況，這問題著實使我們花費了相當多的時間排除。因此在本實作中，採用動態寫入記憶體的方式進行，當記憶體中舊的圖片不用時，便利用存在 rom 中的程式碼將該記憶體區塊更新成稍後將用到的圖片，使許許多多的圖片能夠在有限的記憶體下被呈現出來。尤其利用在 C 語言中區域變數會在函式結束時消滅之特性，更可以省下許多記憶體管理的功夫。

四、參考資料

1. http://en.wikipedia.org/wiki/Space_Invaders
2. Whirlwind Tour of ARM Assembly (TONC)
   (http://www.coranac.com/tonc/text/asm.htm)
3. Knaggs and Welsh's ARM Assembly Language Programming book
   (http://www.arm.com/miscPDFs/9658.pdf)