

# 致理技術學院

## 商務科技管理系 實務專題報告

### 3D Island Paradise

3  
D  
  
I  
S  
L  
A  
N  
D  
  
P  
A  
R  
A  
D  
I  
S  
E

指導老師：郭正華老師

學生：卓庭瑋(19933113)

羅貫旗(19933217)

黃政達(19933219)

一  
百  
零  
二  
年  
十  
二  
月

中華民國 102 年 12 月

# 致理技術學院

商務科技管理系  
實務專題報告

3D Island Paradise

學生：卓庭瑋(19933113)  
羅貫旗(19933217)  
黃政達(19933219)

本成果報告書經審查及口試合格特此證明。

指導老師：\_\_\_\_\_

中華民國 102 年 12 月

# 誌 謝

首先要感謝我們的專題指導老師—郭正華教授，在這一年來為我們的實務專題細心指導與諄諄教誨，在我們毫無實務專題的經驗，給予方向，並讓我們自己從專題研究過程中學習、摸索與成長，讓我們受益良多，在此誠摯的向您致謝。

特別致謝的是：感謝商務科技管理系的主任與老師以及同學，提供的非常寶貴意見，更是彌補了我們專題內容不足之處，以及看法上的見解，擴展了我們觀點及視野，使我們的研究流成、方法，更加充足，研究內容更加豐富。真心感謝你們全力的參與配合。

本實務專題能順利完成，感謝各位師長、朋友的關心、鼓勵與支持，再次深深的表達我們內心的感動及深刻的感謝。

黃政達、羅貫旗、卓庭瑋 謹摯

民國一百零二年十二月

# 摘要

資策會產業情報研究所（MIC）研究報告指出，全球行動遊戲市場年複合成長率高達 12.3%，預估 2015 年規模將超過 150 億美元。然而，對此龐大商機，許多開發商往往因無法負擔高額的技術授權費，不同平台間的研發成本，而放棄往進階、高品質 3D 遊戲領域拓展。

且近年來，遊戲業出現了前所未有的震盪期，各種平台的湧現使得行業內部的競爭愈演愈烈。前幾年，想要製作好的遊戲，肯定就需要強大硬體的支持，所以大部分的 3D 遊戲都出現在 PC 或者 PS3、XBOX 等專業遊戲主機上，其他平台則由於受硬體條件的限制無法製作出較好的遊戲。

目前市面上的跨平台遊戲引擎已經有好幾款，其中最專業、專穩定、效率最高並且支持遊戲平台最多的就是 UNITY(免費)。可跨九種主要的遊戲平台，包刮 WEB 平台、PC 平台、MAC 平台、IOS 平台……等等。無論是電腦遊戲、網路遊戲，都可以使用 UNITY 輕鬆實現跨平台遊戲開發。

在眾多的 3D 引擎中，Unity 3D 遊戲引擎在近年來廣被採用於許多的專案應用上，製作本專題所需之 3D 動作，也因此開啟了本研究計畫的構想。本組進行了應用 Unity 3D 遊戲引擎開發 3D 遊戲動在查閱了許多應用實務方法以及該產品網站討論版中各項相關的發言內容，令我們覺得該引擎很適合於用來畫的研究計畫。

關鍵詞(Key words):Unity、Dead Island、Maya、FPS 遊戲

# 目 錄

審查簽名頁	i
授權書	ii
誌謝	iii
摘要	iv-v
目 錄	v
圖目錄	vii
表目錄	viii
第一章 緒論	1
第一節 研究動機	1
第二節 研究目的	2
第三節 研究流程	3
第二章 文獻探討	4
第一節 射擊遊戲概述	4
第二節 Unity 系統	5-13
第三節 Maya 系統	14-24
第四節 SketchUp	25-28
第三章 專題設計	29
第一節 專題規劃	29-30
第二節 研究架構	31
第三節 時程計劃	31-32
第四章 遊戲畫面	33
第五章 結論與建議	34
第一節 研究結論	34
第二節 研究建議	34
參考文獻	35
附錄 Unity 程式碼	36-44

# 圖目錄

圖 1-1	研究流程.....	3
圖 2-1	Demo 中的截圖.....	7
圖 2-2	Unity 介面截圖.....	8
圖 2-3	預覽圖.....	11
圖 2-4	假想敵人程式碼.....	12
圖 2-5	操作畫面.....	13
圖 2-6	平面圖型視覺化.....	15
圖 2-7	臉部圖.....	16
圖 2-8	Maya 介面圖.....	17
圖 2-9	Alt+左鍵.....	18
圖 2-10	Alt+中鍵.....	18
圖 2-11	Alt+右鍵.....	18
圖 2-12	視角羅盤.....	19
圖 2-13	空間關係圖.....	19
圖 2-14	各視角工作視窗皆可開啟/關閉 View Compass.....	20
圖 2-15	編輯物件中心軸點之狀態.....	22
圖 2-16	快速鍵圖示.....	23
圖 2-17	SketchUp 介面圖.....	25
圖 2-18	Model Info.....	26
圖 2-19	Export Model.....	27
圖 2-20	FBX Export Options.....	28
圖 2-2	圖片匯入成功.....	28
圖 3-1	甘特圖.....	31
圖 3-2	PERT 圖.....	32

# 表 目 錄

表 2-1	菜單欄英翻中.....	9-11
表 2-2	<b>Tool Box</b> 工具列表.....	21
表 3-1	權責指派表.....	29
表 3-2	專案章程.....	30
表 3-3	專案時程.....	30
表 3-4	工作分解結構表.....	31



# 第一章 緒論

## 第一節 研究動機

近年來電玩產業快速成長，經研究指出亞洲最大的遊戲市場分別為韓國、台灣、中國大陸，而其中韓國線上遊戲市場的發展已近飽和，台灣又為韓國線上遊戲的忠實擁護者，台灣廠商多以代理韓國產品為主，且台灣的多數人也皆都忠愛韓國的遊戲產品，且韓國大力推動遊戲產業，實力日漸雄厚，而中國大陸將成為龐大華文遊戲市場使我國感到有所威脅。

最後，經由本組組員一連串的討論後，認為 Dead Island 是個不錯的方向；且在近幾年不斷有生存、荒島、等知名的故事及遊戲的方式呈現出來。而『Dead Island』是個知名的殺殭屍的生存遊戲故事，更是大朋友小朋友都愛玩的第一人稱射擊經典遊戲，此款遊戲的真實性和戰鬥系統，與每次遭遇到活屍都是毛骨悚然的體驗。玩家還在場景中，可以使用身邊任何東西當作武器，而在小島上的任何決定都將改變此小島的命運。因此，本組將打造易上手且能夠發揮反應和思考力的射擊遊戲。

## 第二節 研究目的

不論是從前，或是現在，大部分學生一直都熱愛著電玩遊戲。基於上述這原因，在選擇專題研究報告的主題時，我們毫不考慮地當然將遊戲製作列入選項。雖然在大學四年中，在學校的學術課程裡並沒有直接接觸到遊戲製作之相關課程，雖然為此感到遺憾，但也為能夠在專題研究中運用學校所學與自修而得的知識，製作出一款自身所學所愛的電玩遊戲而感到高興及興奮。

目前，選擇3D 射擊遊戲之類型，是因其較為適合我們研究之作業。主要在於該遊戲類型之程式是以數理運算為主，再經過適當的美工圖片包裝，相信，應能完成一般水平之作品。

此次之目的，希望能經由專題之過程，學習到遊戲製作之相關經驗。從立案、企劃、審合、到實作，一步步地來學習遊戲之製程。再者，也測試自我，能將腦中所構思的遊戲，實體化到何種地步。因此，藉由歷年來的玩遊戲經驗，來製作此次專題，以求完成更為出色之遊戲。

我們選擇射擊遊戲是對於作業上之考量，相較於角色扮演(RPG)、動作類型(ACT)之下，射擊遊戲在遊戲之設計與程式上之鑽研可有較多之琢磨。射擊類型的遊戲，對於玩家在介面的操作上及遊玩時的沉浸度為何。在 2D 及 3D 的遊戲畫面上，玩家在遊玩時的沉浸程度是否有所差異。

在眾多的 3D 引擎中，Unity 3D 遊戲引擎在近年來廣被採用於許多的專案應用上，且具有三大特點：跨平台支援、高質感 3D 視效、縮短開發時程，因此開啟了本研究計畫的構想。在查閱了許多應用實務方法以及該產品網站討論版中各項相關的發言內容，令我們覺得該引擎很適合於用來製

作本專題所需之 3D 射擊，，本組將應用 Unity 3D 遊戲引擎開發 3D 遊戲動畫並結合 3D 模型繪製軟體 Autodesk Maya 增加遊戲之真實性，設計出一款名為 3D Island Paradise 之遊戲，作為大學四年之成果驗收。

### 第三節 研究流程

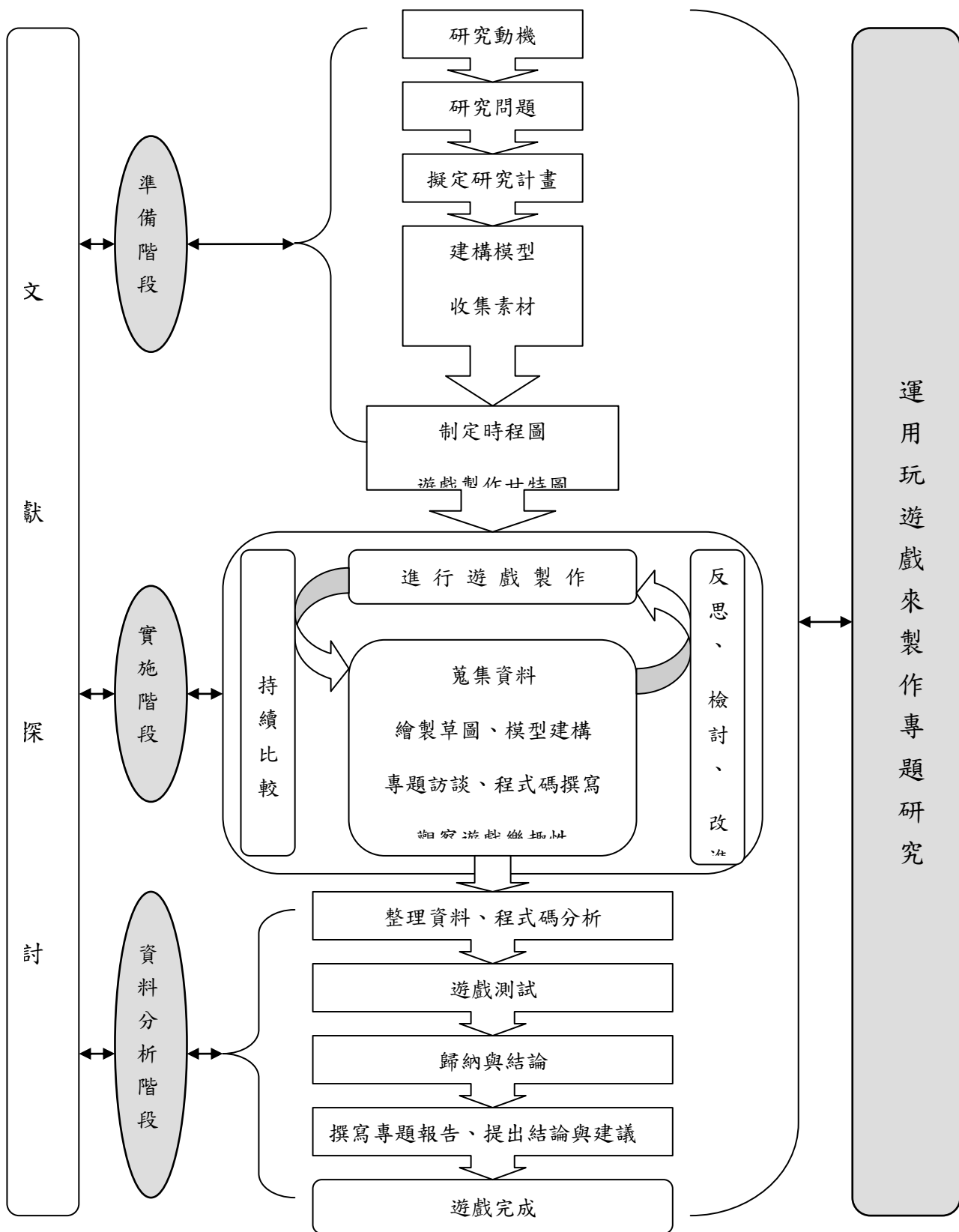


圖 1-1 研究流程

## 第二章 文獻探討

### 第一節 射擊遊戲概述

於1978年發表時即刻成為社會現象的「Space Invaders(太空侵略者)」，正是日本第一個電玩遊戲。而後以「Space Invaders」為發端，當時市面上緊接著出現一系列的侵略者遊戲，都承襲了同樣的特徵，發展至今日，成為「射擊遊戲」(Shooting Game，以下簡稱為STG)之典範。

其中，又分為第三人稱射擊及第一人稱射擊。第三人稱射擊多為捲軸式(直向或是橫向)，遊戲中的佈景大都會以強迫式的手段，使玩家自機前進。而玩家的手要任務，即便是在有限的空間之中，確保自機的生存之道，並以亂槍打鳥的射擊方法來摧毀敵方的攻勢，誠如極上瘋狂大射擊、東方系列、斑鳩。

而第一人稱射擊則多為3D畫面為主。玩家的視角將座落於遊戲中主角視眼當中，換言之，畫面中只看得到怪物和大把槍，遊戲進行時迴避的成份相對於減少，而以與時間競賽為主，盡可能的在最短的時間內將眼前的敵人擊破，諸如死亡火槍、趣味射擊之笨狗守護者、火線危機、死亡之島。

然則，STG的趣味性，依舊在於閃躲攻擊過來的敵人，並伺機以子彈擊中敵人，獲得成就感。攻擊過來的敵人施加於自己的壓力，以及打中敵機後瞬間的快感等，一般認為這種緊張狀態的收放，會讓人感到興奮。

## 第二節 Unity 系統

Unity 是一套跨平台的遊戲引擎，可開發執行於 PC、Mac OS 單機遊戲，或是 iOS、Android 手機或平板電腦的遊戲。Unity 也可開發線上遊戲，只需在網頁瀏覽器安裝外掛程式後即可執行 Unity 開發的遊戲（例如近期推出的亞特蘭提斯）。Unity 也可用於開發 PS3、XBox360、Wii 遊戲主機上的遊戲。

Unity 提供了人性化的操作介面，支援 PhysX 物理引擎、粒子系統，並且提供網路多人連線的功能，不需要學習複雜的程式語言，符合遊戲製作上的各項需求。Unity 大幅降低了遊戲開發的門檻，即使是個人工作室製作遊戲也不再是夢想。對於遊戲公司而言，使用 Unity 也可以縮短遊戲的開發時間，並且降低遊戲的製作成本。Unity 4 於 2012 年 11 月 15 日推出，每隔一段時間就會發佈新的版本，修正發現的問題並且加入新的功能，目前最新版本是 Unity 4.1.5。

### Unity 遊戲引擎的主要特色

地形編輯器 - 快速製作自然場景的地形效果。

內建 NVIDIA PhysX 物理引擎 - 提供逼真的物理效果。

圖形最佳化 - 擁有支援 DirectX 與 OpenGL 的圖形最佳化技術。

多平台發佈 - 可將製作的遊戲發佈到 PC、Mac、iOS、Android 平台。

多人網路連線 - 透過 RakNet 支援多人同時上線遊戲。

## Unity Pro 版本功能

Unity Pro 專業版屬於付費購買的商業軟體，除了擁有 Unity 免費版本的所有功能，還提供燈光的即時陰影、更逼真的水面效果、影像效果 ( Image Effect )，與遮蔽刪除 ( Occlusion Culling ) 的功能，可提升遊戲的執行效能。

此外 Unity Pro 還可以選擇黑色的使用者介面，即使您的工作環境為了避免螢幕反光而降低光源，也不會感到畫面太亮而刺眼。

## Unity 系統需求

### Unity 單機遊戲開發的系統需求

- 作業系統 Windows XP SP2 / Mac OS X Leopard 10.5 以上版本
- 使用 Occlusion Culling 遮蔽刪除的功能需要支援 Occlusion Query 的圖形處理器

### Unity iOS 遊戲開發的系統需求

- 硬體 Intel-based Mac 處理器
- 作業系統 Mac OS X Snow Leopard 10.6 以上作業系統

### Unity Android 遊戲開發的系統需求

- 作業系統 Windows XP SP2 以上/Mac OS 10.5.8 以上
- Android SDK and Java Development Kit (JDK)

測試用 Android 手機或平板電腦的系統需求

- OS Android OS 2.0 以上作業系統
- CPU ARMv7 (Cortex family) 處理器
- 建議採用支援 OpenGL ES 2.0 的圖形處理器

需注意 Android 行動裝置擁有多種不同的規格，遊戲在不同的 Android 手機或平板電腦上執行時，玩家對於遊戲的體驗將會有不可避免的落差，遊戲設計者應考慮應用程式的執行效能。



## Unity 是個強大的跨平台 3D 遊戲開發工具

尤其是它在 3.0 裡面製作那款第一人稱戰爭遊戲，如圖 2-1 畫質效果絲毫不遜色於當下十分流行的《穿越火線》、《戰地之王》等等的遊戲。



圖 2-1 Demo 中的截圖

## 一、Unity3D 的基本界面介紹

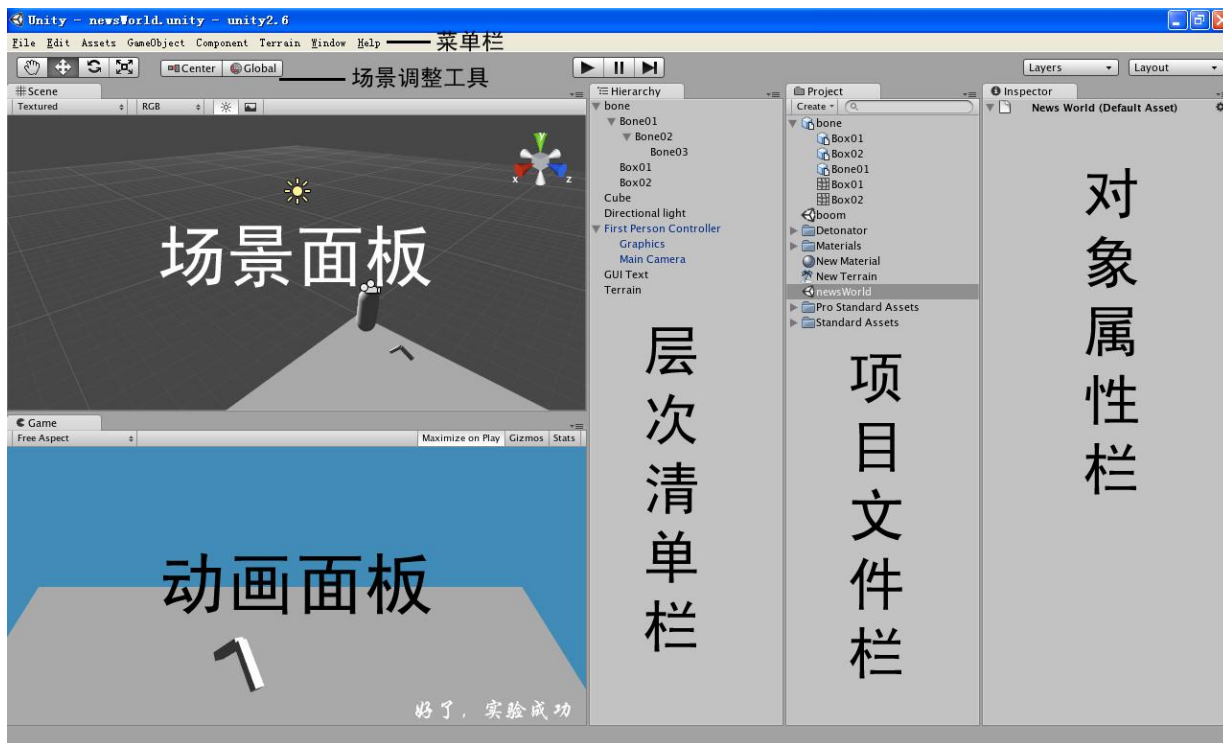


圖 2-2 Unity 介面截圖

**場景面板：**圖 2-2 該面板為 Unity3D 的編輯面板；您可以將您所有的模型、燈光、以及其他材質對象拖放到該場景中。構建遊戲中所能呈現景象。

**動畫面板：**與場景面板不同，該面板是用來渲染場景面板中景象的。該面板不能用作編輯，但卻可以呈現完整的動畫效果。

**層次清單欄：**該面板欄主要功能是顯示放在場景面板中所有的物體對象。

**項目文件欄：**該面板欄主要功能是顯示該項目文件中的所有資源列表。除了模型、材質、字體等，還包括該項目的各個場景文件。

**對象屬性欄**：該面板欄會呈現出任何對象的所固有的屬性。

下面為重點詳解一下清單欄：

菜單欄中包含有八個菜單選項如表 2-1：分別是 File【文件】 Edit【編輯】 Assets【資源】 GameObject【遊戲對象】 Component【組件】 Terrain【地形】 Window【窗口】 Help(幫助)。其各自又有自己的子菜單：


表 2-1 菜單欄英翻中

主菜單	包含的子功能表
File【文件】	New Scene【新建場景】
	Open Scene【打開場景】
	Save Scene【保存場景】
	Save Scene as...【場景另存為...】
	New Project...【新建工程檔】
	Open Project...【打開工程檔】
	Save Project...【保存工程檔】
	Build Settings...【創建設置】（這裡指創建遊戲設置）
	Build & Run【創建並運行】（這裡指創建並運行遊戲）
	Exit【退出】
Edit【編輯】	Undo【撤銷】
	Redo【重複】
	Cut【剪切】
	Copy【拷貝】
	Paste【粘貼】
	Duplicate【複製】
	Delete【刪除】
	Frame selected【當前鏡頭移動到所選的物體前】
	Select All【選擇全部】
	Preferences【首選參數設置】
	Play【播放】
	Pause【暫停】
	Step【步驟】
	Load selection【載入所選】
	Save selection【存儲所選】
	Project settings【工程檔設置】（包含：可執行檔EXE圖示設置，畫面抗鋸齒功能設置等）
	Render settings【渲染設置】（如果您覺得整體畫面的色彩品質不盡人意，可在此處進行調節）

	Graphics emulation 【圖形模擬】（主要是配合一些圖形加速器的處理）
	Network emulation 【網路模擬】（可選擇相應的網路類型進行模擬）
	Snap settings 【臨時環境】
Assets (資源)	Reimport 【重新導入】
	Create 【創建】（包含：資料夾、材質、腳本等等）
	Show in Explor 【顯示專案資源所在的資料夾】
	Open 【打開】
	Import New Asset... 【導入新的資源】
	Refresh 【刷新】
	Import Package... 【導入資源包】
	Export Package... 【匯出資源包】
	Select Dependencies 【選擇相關】
	Export ogg file 【匯出OGG文件】
	Reimport All 【重新導入所有】
	Sync VisualStudio Project 【與VS項目同步】
GameObject 【遊戲項目】	Create Other 【創建其他組件】
	Center On Children 【子物體歸位元到父物體中心點】
	Make Parent 【創建父集】
	Clear Parent 【取消父集】
	Apply Changes To Prefab 【應用變更為預置】
	Move To View 【移動物體到視窗的中心點】
	Align With View 【移動物體與視窗對齊】
	Align View to Selected 【移動視窗與物體對齊】
Component(組件)	Mesh 【網路】
	Particles 【粒子系統】（能打造出非常棒的流體效果）
	Physics 【物理系統】（可使物體帶有對應的物理屬性）
	Audio 【音訊】（可創建聲音源和聲音的聽者）
	Rendering 【渲染】
	Miscellaneous 【雜項】
	Scripts 【腳本】（Unity內置的一些功能很強大的腳本）
	Camera-Control 【攝像機控制】
Terrain(地形)	Create Terrain 【創建地形】
	Import Heightmap - Raw... 【導入高度圖】
	Export Heightmap - Raw... 【匯出高度圖】
	Set Resolution... 【設置解析度】
	Create Lightmap... 【創建光影圖】
	Mass Place Trees... 【批量種植樹】
	Flatten Heightmap... 【展平高度圖】
	Refresh Tree and Detail Prototypes 【刷新樹及預置細節】
Window(窗口)	Next Window 【下個窗口】

	Previous Window 【前一個視窗】
	Layouts 【佈局】
	Scene 【場景窗口】
	Game 【遊戲窗口】
	Inspector 【檢視視窗】（這裡主要指各個物件的屬性）
	Hierarchy 【層次視窗】
	Project 【工程視窗】
	Animation 【動畫窗口】（用於創建時間動畫的面板）
	Profiler 【探查窗口】
	Asset Server 【原始伺服器】
	Console 【控制台】
Help(幫助)	About Unity 【關於Unity】
	Enter serial number 【輸入序號】
	Unity Manual 【Unity手冊】
	Reference Manual 【參考手冊】
	Scripting Manual 【腳本手冊】
	Unity Forum 【Unity論壇】
	Welcome Screen 【歡迎窗口】
	Release Notes 【發行說明】
	Report a Problem 【問題回饋】

## 二、Unity3D的一個簡單預覽

每個 Unity3D 版本都會自帶一個 demo 原始檔案。如圖 2-3 在 3.0 正式版中，自帶的 Demo 就是網上展示的那款強大的射擊遊戲。在一般情況下，您只要第一次打開 Unity3D v3.0 就會看見自帶的那個 demo 專案檔案。但如果 Unity3D 並沒有打開這個專案檔案，您可以在 Unity3D 裡的“File”功能表下點擊“Open Project...”，在隨後彈出的“Unity – Project Wizard”對話方塊中點擊“Open Other...”按鈕，在“C:\Documents and Settings\All Users\Documents\Unity Projects”這個路徑下找到專案檔案夾“Bootcamp Demo”，選擇並打開它。打開項目之後，在舞臺場景面板中依然什麼都沒有顯示的的話，請在 Project 【專案檔案欄】按兩下場景檔 。稍等片刻之後，該舞臺場景的所有物件就可導入舞臺場景面板中。

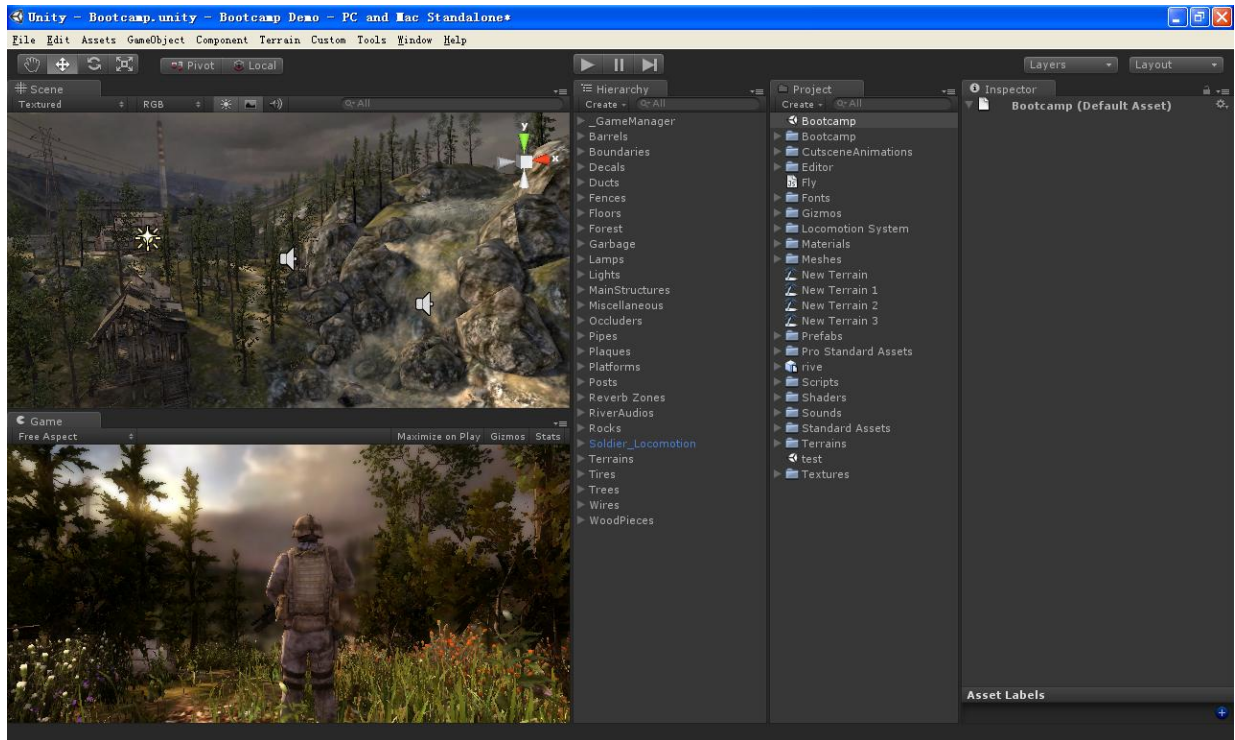


圖 2-3 預覽圖

如圖 2-3 點擊一下中間的播放按鈕做一下測試（如果您的機器配置不是太高，可能等待的時間會稍長）。過了不一會，您就可以在 Game 【動畫面板】中看到一個正在運行的射擊遊戲了。

### 三、製作假想敵人

第一步：在功能表列中選中“GameObject”——> “Create Other” ——> “Cube” ，在炮塔旁邊創建一個正方體並將其更名為“tankBody”，並用場景調整工具調整它的大小和位置。

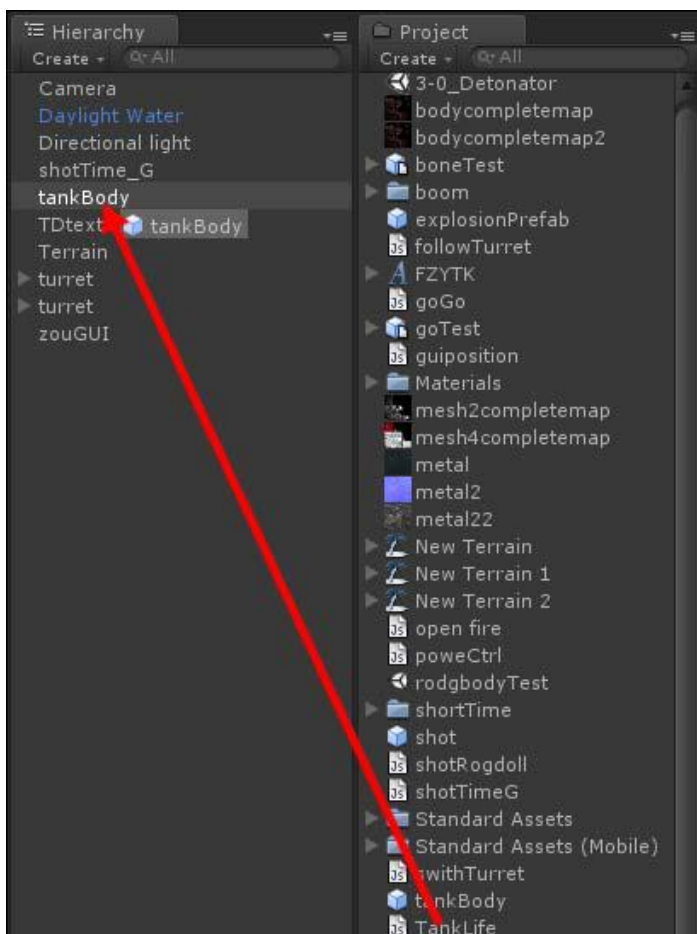
第二步：創建一個 JS 腳本代碼，並命名為“TankLife”，然後輸入如下代碼，來作為“tankBody”的生命減少函數。

```

var tankLife = 4;
function OnCollisionStay(collision : Collision)
{
//查找所有的碰撞体，如果发现有碰撞体的名字是"shot(Clone)",那么生命值减 1
    for (var contact : ContactPoint in collision.contacts)
    {
// 这里要特别注意一下，判断炮弹的名字不是"shot"而是"shot(Clone)""
        if (contact.otherCollider.name == "shot(Clone)")
        {
            if(tankLife>0)
            {
                tankLife --;
            }
//当生命值小于 1，物体就消失
            if(tankLife<1)
            {
                Destroy(gameObject);
            }
        }
    }
}
}

```

圖 2-4 假想敵人程式碼



輸入完畢之後，將這段代碼直接拖放給立方體物件“tankBody”。

第三步：點擊測試按鈕 ，對準立方體射擊。

圖 2-5 操作畫面

### 第三節 Maya 系統

Maya 是世界上最為優秀的三維動畫的製作軟體之一，是相當高階而且複雜的三維電腦動畫軟體，它是 Alias|Wavefront 公司在 1998 年才推出的三維製作軟體。被廣泛用於電影、電視、廣告、電腦遊戲和電視遊戲等的數位特效創作。曾獲奧斯卡科學技術貢獻獎等殊榮。2005 年 10 月 4 日，生產 3D Studio Max 的 Autodesk(歐特克)軟體公司宣佈正式以.82 億美元收購生產 Maya 的 Alias。所以 Maya 現在是 Autodesk 的軟體產品。

很多大片中的電腦特技鏡頭都是應用 Maya 完成的。逼真的角色動畫、豐富的畫筆，接近完美的毛髮、衣服效果，不僅是影視廣告公司對 Maya 情有獨鍾，許多喜愛三維動畫製作，並有志向影視電腦特技方向發展的朋友也為 Maya 的強大功能所吸引。

可滿足尋求大幅度性能提升而同時又處理海量資料集的遊戲、電影、廣播和數字出版專業人士的需要。Alias 對軟體進行了重新架構設計，大幅提升了軟體性能，因此，Maya 擁有處理最複雜模型、場景和動畫資料的能力和可靠性。



## Maya - 應用領域

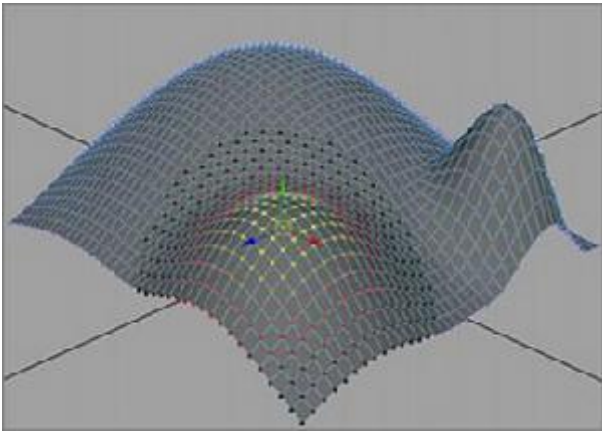
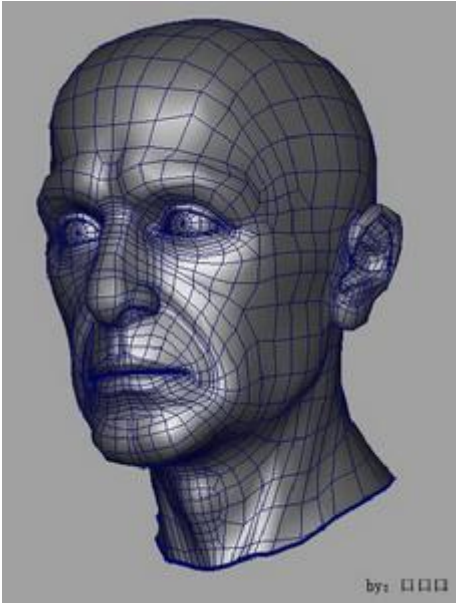


圖 2-6 平面圖型視覺化

- 
- 一、增進平面設計產品的視覺效果，強大功能開闊平面設計師的應用視野
  - 二、網站資源開發
  - 三、電影特技（蜘蛛人）
  - 四、遊戲設計及開發
-

## Maya - 學習內容



Maya 軟體在影視動畫行業有廣泛的運用，學習 Maya 基礎操作及基本建模技術，掌握 Maya 基本角色、貼圖、動畫製作方法，瞭解並掌握 Maya 動力學系統，運算式的應用及 Maya 塗刷效果、Maya 毛髮、部分 Maya 外掛程式。

榮獲美國電影藝術與科學學院獎項的 Maya® 軟體是世界上最強大的整合 3D 建模、動畫、效果和渲染解決方案。如圖 2-7 Maya 還增強了二維圖像的畫質和表現力。正因為此，電影和視頻藝術家、遊戲開發人員、視覺化專業人員、Web 和印刷設計人員在其工作

中採用 Maya，以滿足其下一代製作需要。

圖 2-7 臉部圖

Maya 的推出一舉降低了三維動畫製作的成本，在 Maya 推出之前的商業三維動畫製作基本上由基於 SGI 工作站的 Softimage 軟體所壟斷，Maya 採用 Windows NT 作為作業系統的 PC 工作站，降低了設備要求，促進了三維動畫的普及，隨後 Softimage 也開始向 PC 平臺轉移。

Maya 在現在電影特效製作中應用相當廣泛，著名的星球大戰前傳就是採用 Maya 製作特效的，此外還有蜘蛛人、侏羅紀公園、海底總動員、哈利波特甚至包括頭文字 D 在內的大批電影作品。

## Maya 基本介面及操作介紹

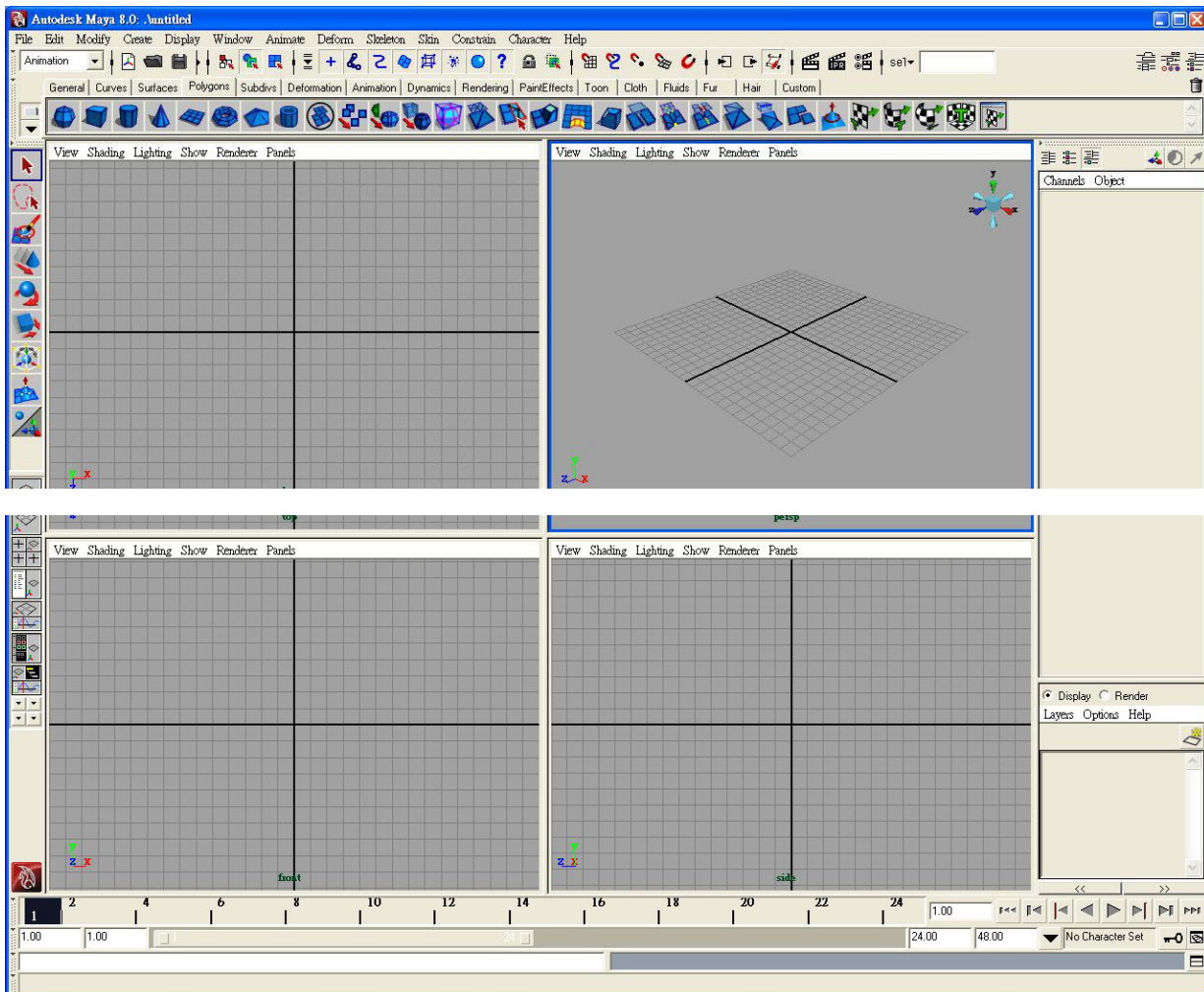


圖 2-8 Maya 介面圖

如圖2-8 在 Maya 的三個視角工作視窗及一個透視工作視窗，都是由一部預設的攝影機為視點所投射出的畫面，使用者再經由滑鼠和鍵盤操作攝影機來改變攝影機觀察的視點，以利進行工作。三個視角攝影機可以藉由滑鼠和鍵盤的操作，調整位置及遠近，透視視窗攝影機則無限制，可以調整位置、遠近及旋轉。

**Note :**

滑鼠游標移入任一視角工作視窗，不需點擊滑鼠，即可透過鍵盤空白鍵快速切換為全視窗進行編輯工作。

**基本操作方式：**

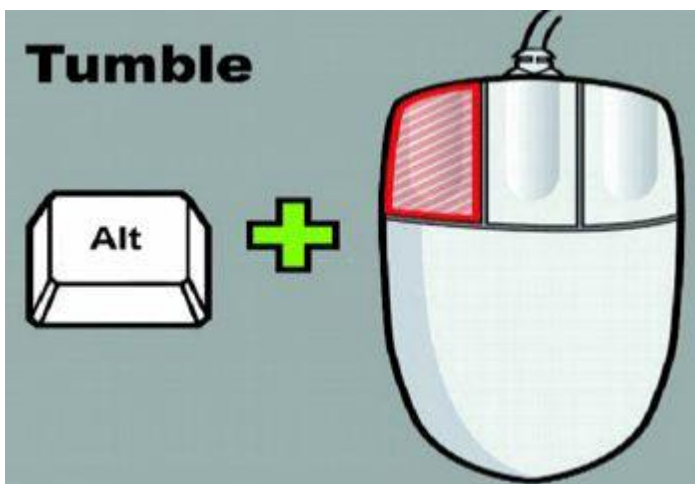


圖2-9 Alt+左鍵

如圖2-9 旋轉視點，只適用於透視工作視窗。

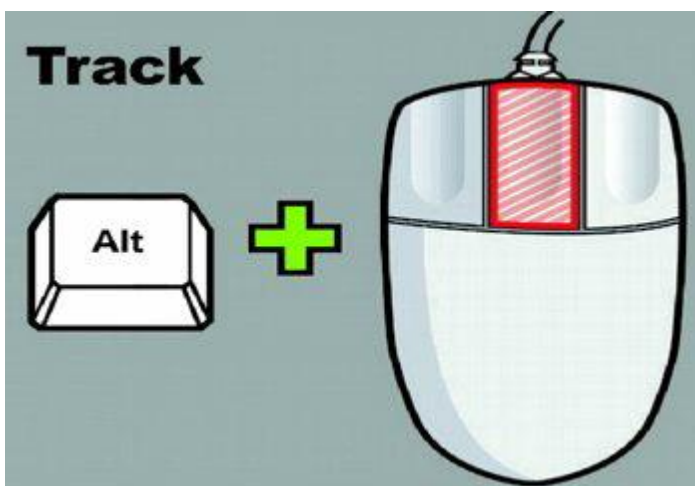


圖2-10 Alt+中鍵

如圖2-10 平移視點，可上下左右。若使用滾輪滑鼠則按下滾輪，等同於按下中鍵。

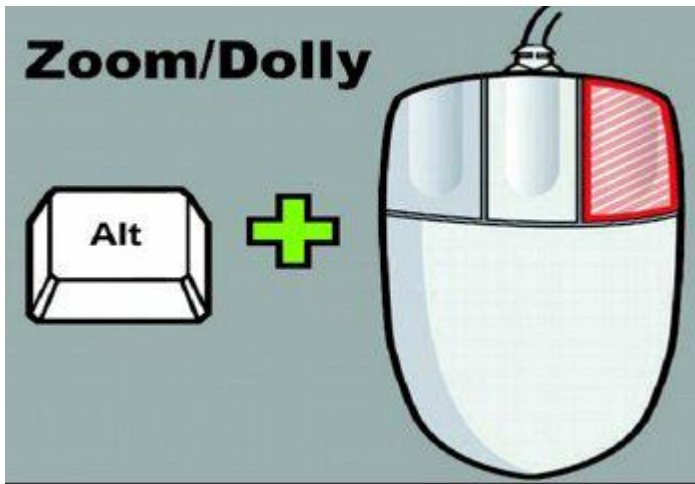


圖2-11 Alt+右鍵

如圖2-11 推移視點，攝影機前後移動。於PC的Windows作業系統亦可用滑鼠滾輪控制。

## 工作視窗切換方式：

本節開始介紹利用鍵盤空白鍵，可快速切換全視窗與四格分割視窗，並搭配滑鼠游標座落位置與空白鍵進行工作視窗的切換。除此方法外，Maya 7.0 以上版本於各視角的工作視窗提供新的視角切換工具，如圖2-12 稱之為「View Compass 視角羅盤」。



圖 2-12 視角羅盤

如圖2-13 此工具可方便使用者快速熟悉3度空間的工作環境與視角的變換。只要以滑鼠游標點擊View Compass上各軸向的圓錐形箭頭，工作畫面即切換為該軸向之平面視角，於下圖的透視視窗中可對照右上角之View Compass與左下角的三度空間座標之間的關係。

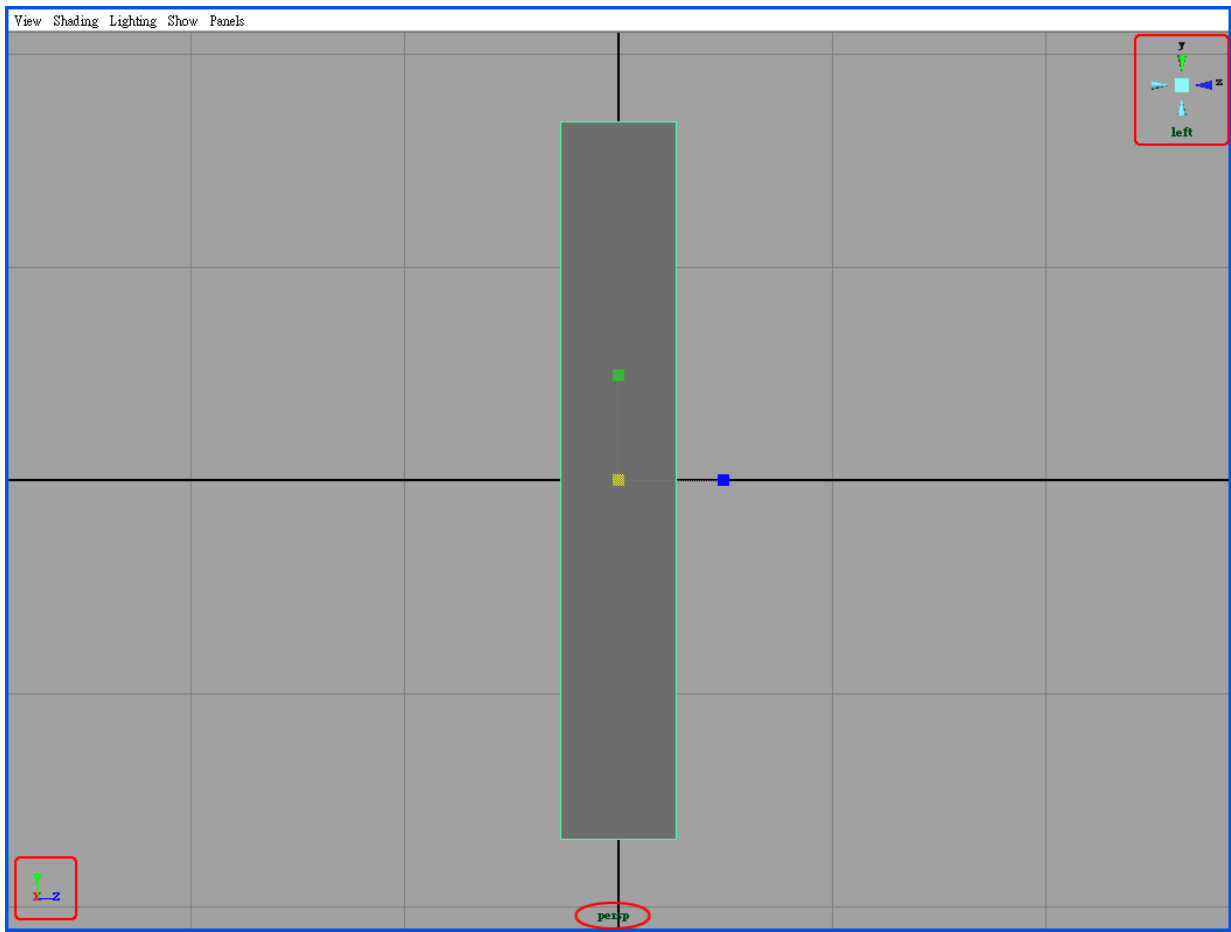


圖2-13 空間關係圖

如圖2-14 以滑鼠左鍵點擊View Compass中央的方塊，即可切換為立體透視的工作環境。Maya於四個標準工作視窗皆可啟動或關閉View Compass，預設狀態為僅透視視窗有顯示View Compass。

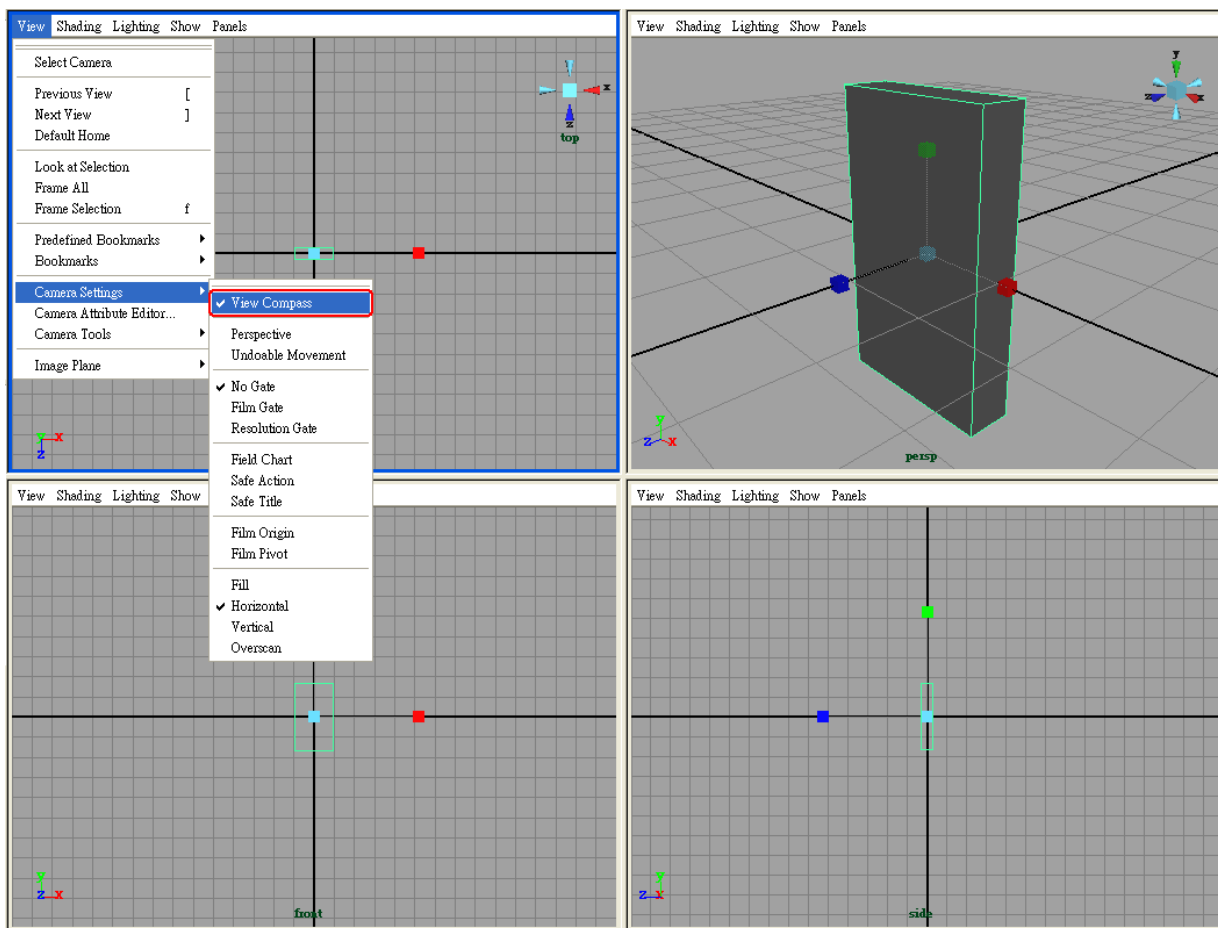


圖2-14 各視角工作視窗皆可開啟/關閉View Compass

## Tool Box工具列：

表2-2 Tool Box工具列表

	1. Select Tool：選取工具，快捷鍵 q
	2. Lasso Tool：套索選取工具
	3. Paint Selection Tool：筆刷選取工具
	4. Move Tool：移動工具，快捷鍵 w
	5. Rotate Tool：旋轉工具，快捷鍵 e
	6. Scale Tool：縮放工具，快捷鍵 r
	7. Universal Manipulator：全局操作器，快捷鍵 <b>Ctrl</b> + t
	8. Soft Modification Tool：平滑變形器
	9. Show Manipulator Tool：把手工具，快捷鍵 t
	10. Last Selected Tool：上次使用工具，快捷鍵 y



## Pivot 軸心：

如表2-2 所有物件、元件、群組被選取並啟動移動、旋轉、縮放工具時會顯現軸心位置，編輯物件時是以軸心點為作用基準。更改物件軸心位置，於選取物件後啟動移動、旋轉、或縮放工具，接著按鍵盤 **Insert** 鍵，可將軸心轉換為可移動模式，再由滑鼠進行特定軸向，或自由方向的移動。將軸心設定在物件中心，可使用 **Modify > Center Pivot** 指令。

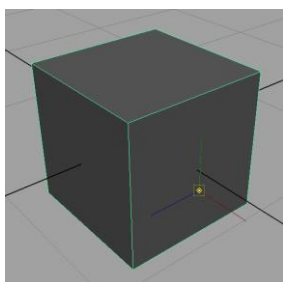


圖2-15編輯物件中心軸點之狀態

## 回復 Undo / 重複Redo 指令：

如圖 2-16 Maya 和其他軟體相同，提供Undo / Redo 指令，讓使用者可以反悔先前執行過的錯誤動作，或交替使用Undo / Redo 以進行比較執行指令前後的差別。

Undo 的快速鍵為 **Ctrl** + z，或z

Redo 的快速鍵為 **Shift** + z，亦即大寫Z


## 通用快速鍵整理：





圖2-16 快速鍵圖示

## 物件的選取：

於工作視窗中，除了基本的使用滑鼠左鍵搭配選取工具點選或框選物件外，針對複雜場景的繁多物件，另有交叉選取、加選、減選的操作方式來因應不同選取的需求：

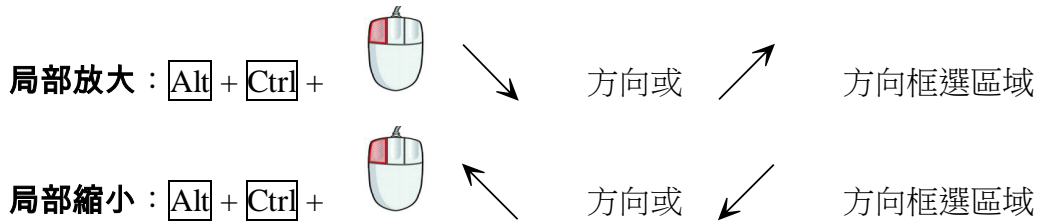
交叉選取：Shift + 

加選：Ctrl + Shift + 

減選：Ctrl + 

## 工作視窗局部放大/縮小：

除了快速鍵A, F可以對攝影機進行快速定位，於工作視窗另可搭配功能鍵進行局部放大/縮小：



## 各種 3D 動畫製作軟體的評價和選擇：

1. 如果是電影的話 Maya 最好，不過現下用 XSI 還有 LW 也很多。C4D 據說是新的一個很棒的 3D 軟體。
2. 3D 動畫還是 Maya 最出色~  
3D 構圖還是 3DS MAX 最棒~
3. Maya 的特效比較好  
Max 比較普及
4. 有插件，3D 模型在不用的軟體之間可以相互匯入，最好的一個是 bodystudio,它幾乎可以把 pposer 的所有東西（模型、貼圖、動畫、形變目標等）匯入到 maya 或 3dmax 裡，可惜一直沒有找到破解，不過只匯入模型的話不用插件，它們有一個通用的模型格式叫做.obj 格式，Maya、3dmax 和 poser 都支援該格式的模型，非常方便。

## 第四節 SketchUp

如圖 2-17 SketchUp 是一套面向建築師、都市計畫專家、製片人、遊戲開發者以及相關專業人員的 3D 建模程序。它用於 Google Earth 上的建模也十分方便。它比其他三維 CAD 程序更直觀，靈活以及易於使用。

基於便於使用的理念，它擁有一個非常簡單的界面。SketchUp 世界中一個眾所周知的特性便是 3D Warehouse。用戶可以利用他們的 Google 賬戶來上傳創建的模型，並且瀏覽其他的組件和模型。

部分關鍵特性和用處包括：

「Smart」：智能游標系統，允許用戶使用 2 維的螢幕和滑鼠來描繪 3 維的部件。

「push-pull」：通過沿預定的路徑擠壓 2 維界面從而創建 3 維物件。

「Follow Me」簡單高效的學習能力。

可以模擬攝像機和太陽的運動

與 Google Earth 的協同功能

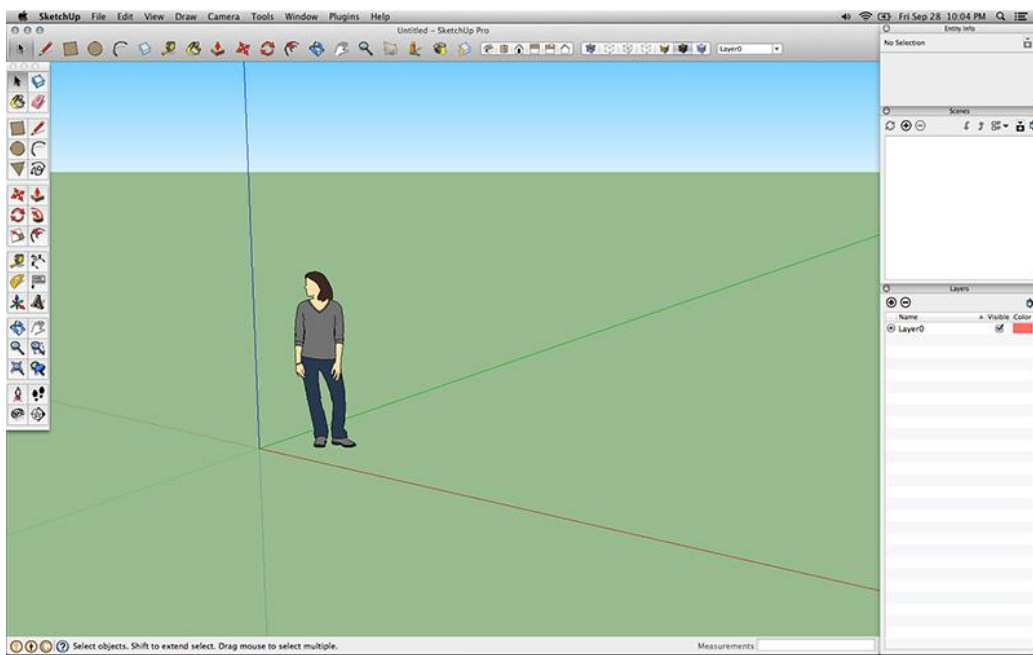


圖 2-17 SketchUp 介面圖

## SketchUp 匯出到 Unity3D 的正確方法

我們主要是利用在 3D 模型庫所找到的(.skp)檔轉為(.fbx)，才可匯入到 Unity!

但有時導入的模型很怪異，會超級大或是超級小!連根草都比你的房子大!

- 1、如圖 2-18 新建 SketchUp 工程的時候，選擇工程單位為 Meter(米)，如果不是米，則在 Tools -> Model Info -> Unit 內將 Format 修改為 Decimal : Meters 即可。

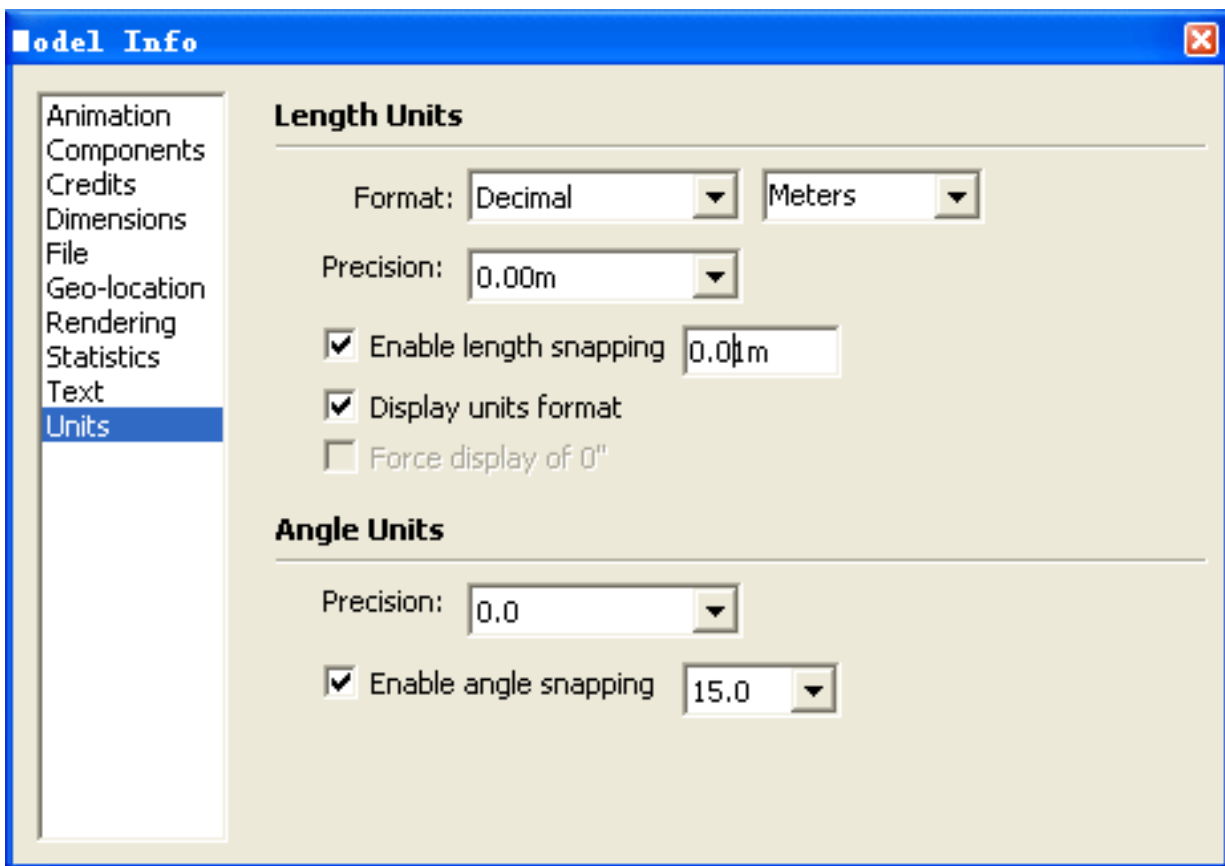


圖 2-18 Model Info

因為會影響到 SketchUp 的原圖，所以請注意如果精度不夠的話還需要把 Precision 變更為足夠的精度，把 Enable length snapping 也設置為和精度一致使其能夠正確的執行自動捕捉。

2、匯出時注意選擇參數，請選擇 File -> Export -> 3D Model 裡面有 FBX 這個格式（好像只有 Pro 版本的 SketchUp 才行哦），先不要著急 Export，選擇一下 Options 修改幾個選項

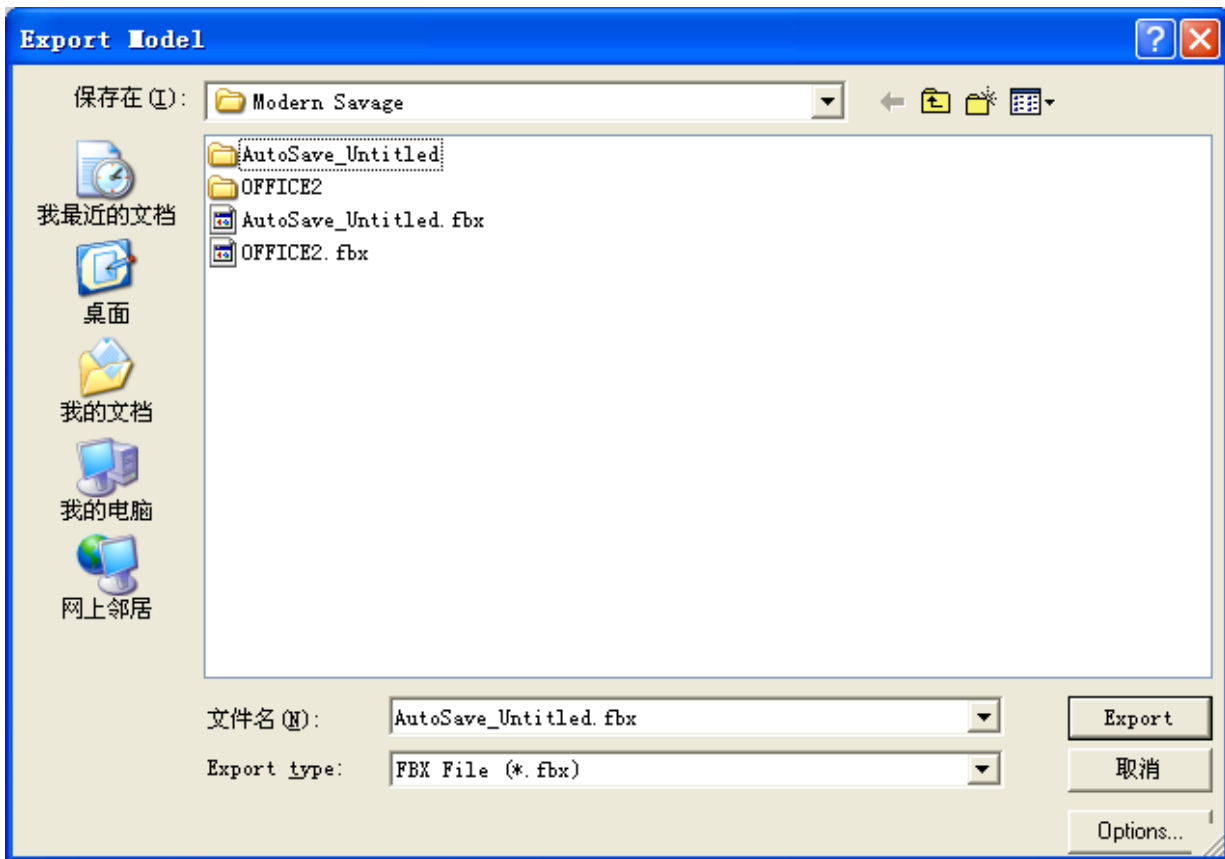


圖 2-19 Export Model

如圖 2-20 Export Options 中的

選中 Triangulate all faces 將所有面變為三角形構成的

選中 Export two-sided faces 將雙面都匯出（有的平面兩面材質不同的，這很重要）

選中 Export texture maps 匯出貼圖表面

選中 Swap YZ coordinates（Y 朝上）

單位選擇 Model Unit（如果你第一步是對的，這裡選 Meters 也是一樣的效果）

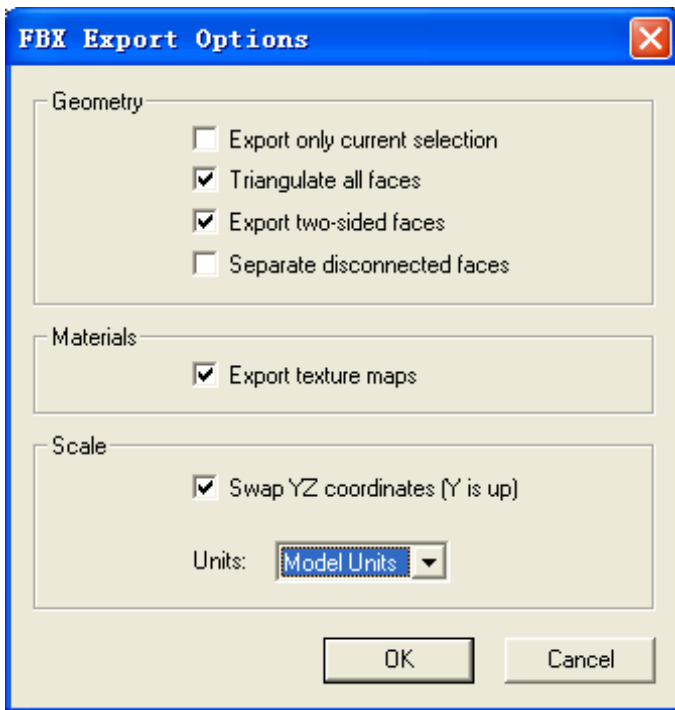


圖 2-20 FBX Export Options

OK 以後，點擊 Export 即可。

3、如圖 2-21 在 Unity3D 中選擇 Asserts -> Import New Asset，找到剛剛匯出的 fbx 檔，選中，檔導入成功後選擇 Project 視窗中的模型，然後將其 Inspector 中的(FBXImporter) -> Meshes -> Scale Factor（縮放比例）修改為 1。最後將模型拖入場景子級就大功告成啦。

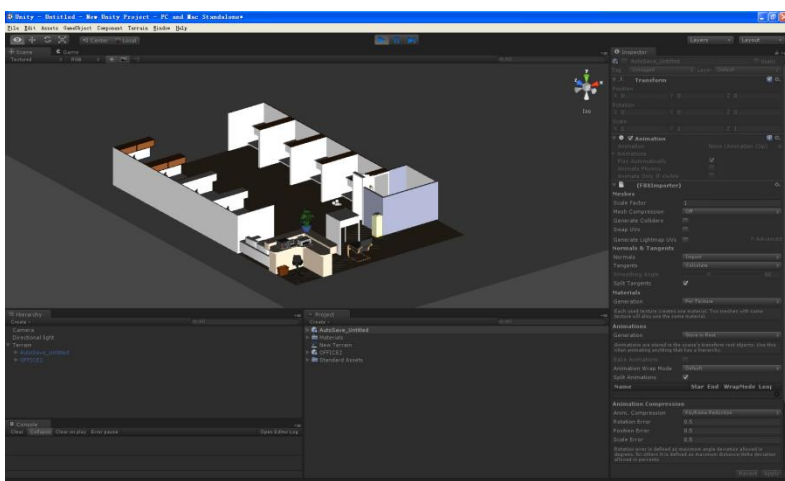


圖 2-21 圖片匯入成功

## 第三章 專題設計

### 第一節 專題規劃

#### 3D Island Paradise 遊戲特色:

本遊戲特色除了提供了景緻極優的場景島嶼外，還包括了一些動物、遺跡、等等，我們還針對遊戲內提供了即時的人物地圖位置資訊，使玩家能夠快速掌握現有的位置，除此之外，還增加了 AI 可以相互射擊!

#### 人力計畫:

主要團隊成員 3 名

專題計畫核心人員，負責專題規劃、設計及管理專題建置團隊

建置工作成員 3 名

增加趣味性及美化!未來發展研究!

#### 專案規劃:

表3-1權責指派表



P:主要責任 S:次要責任 N:應被知會 A:應予核准	專案經理	模型繪製人員	程式撰寫人員	發展研究人員
專案協調	P			A
專案發展	A			P
專案設計	S	P	N	
遊戲設計	N	A	P	S
遊戲維護		N	S	
遊戲測試		S	A	N

### 專題起始-章程:

表3-2專案章程

專題名稱: 3 D Island Paradise	
動機	近年來電玩產業快速成長，其中韓國線上遊戲的發展已近飽和，台灣又為韓國線上遊戲的忠實擁護者，台商多以代理韓國產品為主，因此認為3D FPS第一人稱射擊遊戲是個不錯的方向，將打造易上手且能夠發揮反應和思考力的射擊遊戲。
目標	於2013/12/01完成 3 D Island Paradise
預期成果	<ol style="list-style-type: none"> <li>1.了解三維圖形的視覺感官對玩家的影響以及效果。</li> <li>2.利用豐富的動畫綜合探討媒體的應用特性。</li> <li>3.可以作為日後遊戲開發之參考。</li> </ol>

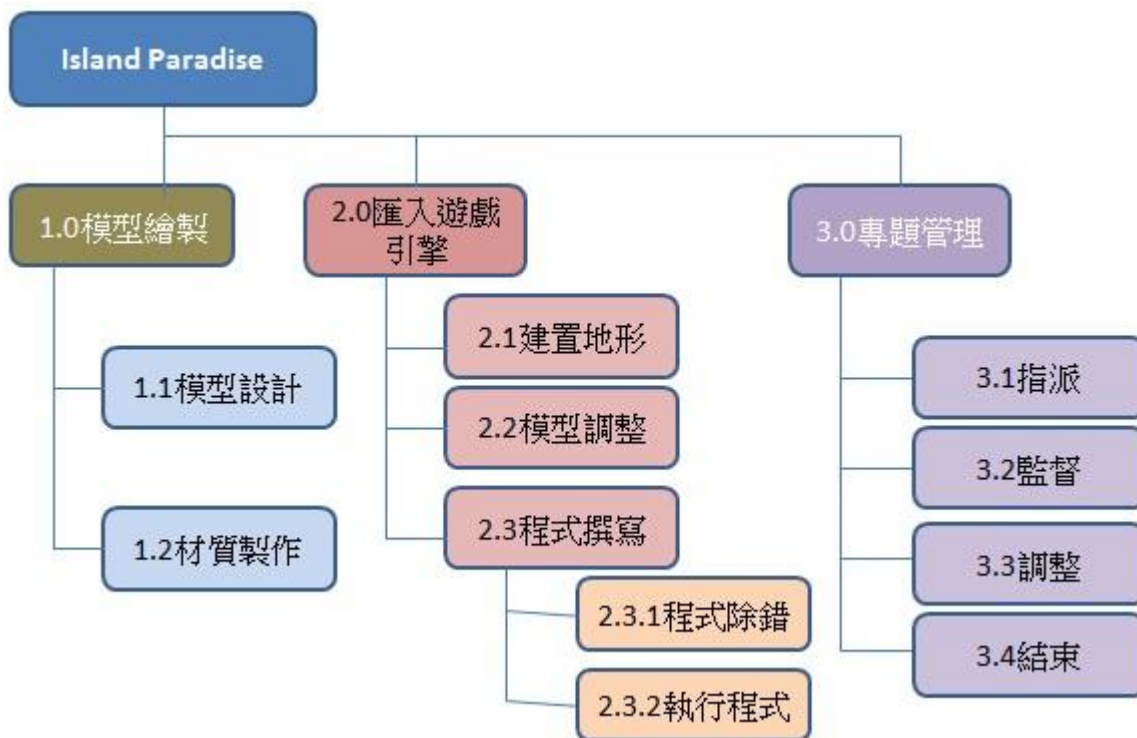
表3-3 專案時程

專題自2013年1月21日起，至2013年12月26日止，共計約11個月，其主要預期進度如下：		
專題製作時程:	起始日期	結束日期

專題 時程	選定研究主題	2013/03/20	2013/05/27
	構想初步內容	2013/04/29	2013/06/18
	遊戲製作	2013/05/20	2013/11/28
	修正畫面及內容	2013/06/28	2013/10/10
	討論及撰寫報告	2013/10/08	2013/11/10
	論文初稿	2013/10/21	2013/11/21
	細部修改	2013/10/02	2013/12/07
	遊戲完成	2013/12/09	2013/12/11
	完稿	2013/12/20	2013/12/26

## 第二節 研究架構

表3-4 工作分解結構表



### 第三節 時程計劃

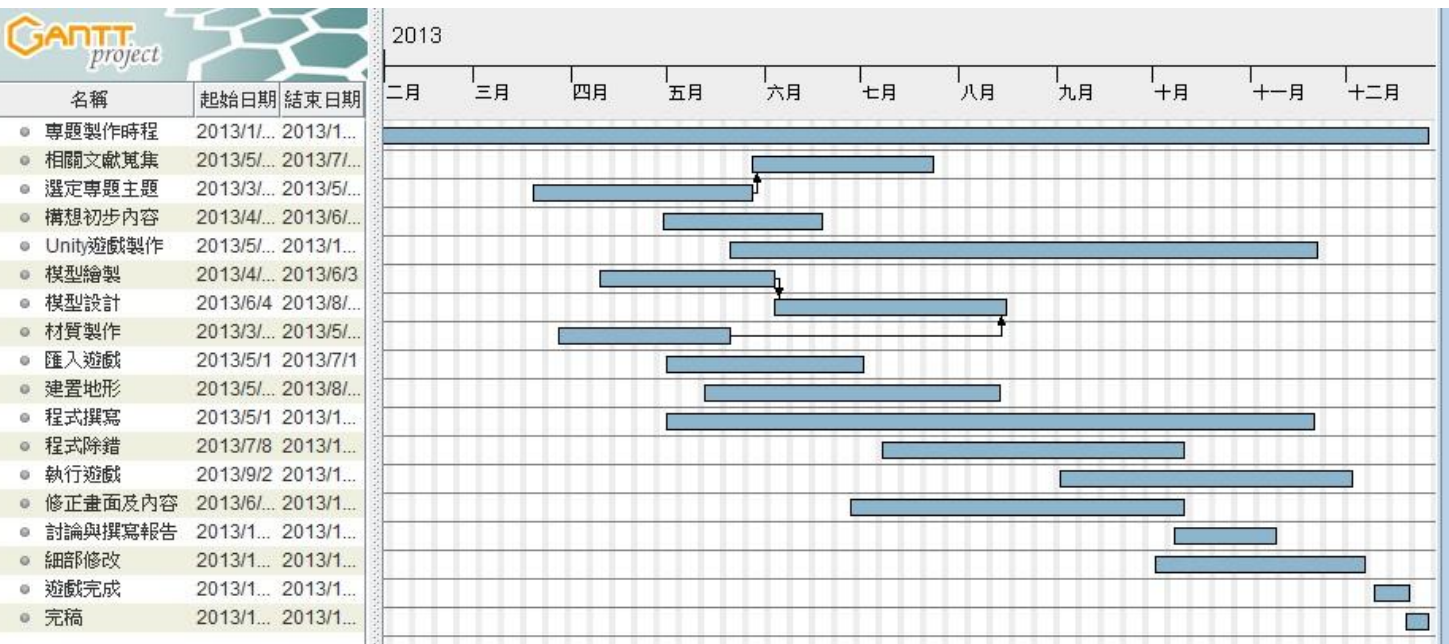


圖3-1甘特圖



圖3-2 PERT 圖

## 第四章 遊戲畫面



## 第五章 結論與建議

### 第一節 研究結論

此次專題研究，學習到遊戲製作之相關經驗。從遊戲之劇本編輯、背景、音樂 都下了不少工夫，要設計出出色之遊戲以下幾點為不可或缺之要素 1.遊戲背景 2.人物設計 3.玩家想法、設定的性向和年齡 遊戲初步企劃時本組不斷修正，直到最終確定方向再經過適當的微調以增加遊戲之趣味性，最終才完成 3D Island Paradise。

本組專題之特色在於透過 Unity3D 遊戲引擎作出逼真的特效及人物動畫，Unity3D 是個跨平台的遊戲引擎可開發於 iOS 和 Android 手機平台上也可用來開發 PS3 Xbox360 Wii 遊戲主機上的遊戲，不需要太複雜的程式語言，大幅降低了遊戲開發時間，本次的專題成果將會作為日後遊戲開發之參考。

### 第二節 研究建議

然而 3D Island Paradise 並非一款沒有缺點的遊戲，除了目前還未開放多人連線和物件穿透的遊戲錯誤外，算是一款表現不錯的射擊遊戲，本組未來將增加多人連線模式及錯誤修正，以提升遊戲之耐玩度。

## 參考文獻

1. Unity 遊戲引擎教學 - CG 數位學習網 ·
2. 崑山科技大學資訊管理系. 作者：游峰碩 Unity 3D 基礎教學
3. Unity 台灣論壇 · 首頁
4. Unity3D 教程手冊|Unity3D 培训|中国第一免费 Unity3D 教程手冊
5. Maya- 维基百科
6. Maya教學 · CG 數位學習網
7. Maya | 3D 動畫軟體| 電腦動畫| Autodesk
8. 3D 模型庫 -SketchUp - Google
9. 崑山科技大學資訊管理系
10. 台北市立內湖高工。冷凍科三年忠班。
11. 逢甲大學 資訊工程學系



## 附錄 Unity 程式碼

AI

// 待機時間

var idleTime : float = 2.0;

// 巡邏參數

var patrolTarget : Transform; // 宣告 目前巡邏目標

var patrolTarget1 : Transform; // 宣告 巡邏目標

var patrolTarget2 : Transform; // 宣告 巡邏目標

var patrolSpeed : float = 2.5; // 巡邏速度

// 朝向目標參數

var targetPlayer : Transform; // 注視目標

var lookAtPos : Vector3; // 注視座標

// 尋追敵參數

var seachAngle :float = 20.0; // 尋敵角度

var seachDis : float = 8.0; // 尋敵距離

var followSpeed : float = 1.0; // 追敵旋轉速度

var followRotateSpeed : float = 3.5; // 追敵速度

// 攻擊參數

var attackRange : float = 4.0; // 攻擊範圍

// 警戒參數

```
var warnRange : float = 8.0; // 警戒範圍
```

```
// 宣告材質
```

```
var idleMaterial : Material;  
var patrolMaterial : Material;  
var followMaterial : Material;  
var warnMaterial : Material;  
var attackMaterial : Material;
```

```
function Start () {
```

```
// 重置巡邏目標
```

```
switchPatrolTaget();
```

```
print("迴圈前待機");
```

```
yield WaitForSeconds(idleTime);
```

```
while(true){
```

```
yield Idle(); // 呼叫執行 Idle
```

```
yield Attack(); // 呼叫執行 Attack
```

```
}
```

```
}
```

```
// 待機狀態
```

```
function Idle () {
```

```
while(true){
```

```
print("待機中");
```

```
// 如果 到達巡邏點 則 切換巡邏點 並 待機。 否則 巡邏移動
```

```
if (PatrolState()){
```

```
switchPatrolTaget();
```

```
renderer.material = idleMaterial;
```

```
yield WaitForSeconds(idleTime);
```

```
}else{
```

```
renderer.material = patrolMaterial;
```

```
Patrol();
```

```
yield;
```

```
}
```

```
// 如果敵人進入尋敵範圍及角度，跳出待機迴圈
```

```
if (SeachEnemy()){  
    print("SeachEnemy");  
    return;  
    yield;  
}  
yield;  
}
```

```
// 切換巡邏點
```

```
function switchPatrolTaget(){  
    if (patrolTarget == patrolTarget1){  
        patrolTarget = patrolTarget2;  
    }else{  
        patrolTarget = patrolTarget1;  
    }  
}
```

```
// 巡邏移動狀態
```

```
function PatrolState (){  
    if (transform.position == patrolTarget.position){  
        return true;  
    }else{  
        return false;  
    }  
}
```

```
// 巡邏移動狀態
```

```
function Patrol (){  
    // 確認巡邏點角度  
    targetAngle(patrolTarget);  
    // 面對目前目標巡邏點  
    FaceToNowTarget();  
    // 向巡邏點移動
```

```
transform.position = Vector3.MoveTowards(transform.position,patrolTarget.position,Time.deltaTime *
patrolSpeed);
}
```

// 判斷敵人是否在視野範圍內

```
function SeachEnemy (){
```

// 距離小於 視野範圍 則 回傳 true 。 否則 回傳 false

```
if (seachDis > targetDistance(targetPlayer) && Mathf.Abs(targetAngle(targetPlayer)) < seachAngle)
{
return true;
}else{
return false;
}
}
```

// 送入 Transform 參數 判斷與目標之間距離 並 回傳距離

```
function targetDistance(nowTaget:Transform){
```

```
var dis : float = Vector3.Distance(transform.position, nowTaget.transform.position);
return dis;
}
```

// 送入 Transform 參數 判斷與目標之間角度 並 回傳角度

```
function targetAngle (nowTaget:Transform){
```

// 宣告 注視目標座標 為 目標位置

```
lookAtPos = nowTaget.position;
```

// 本身與目標之間相對位置

```
var relative : Vector3 = transform.InverseTransformPoint(lookAtPos);
```

// 計算兩者之間角度

```
var angle : float = Mathf.Atan2(relative.x, relative.z) * Mathf.Rad2Deg;
```

```
lookAtPos.y = transform.position.y;
```

```
return angle;
```

```
}
```

// 跟隨敵人

```

function FollowEnemy (){
renderer.material = followMaterial;
FaceToPlayer();
transform.position = Vector3.MoveTowards(transform.position,targetPlayer.position,Time.deltaTime *
followSpeed);
}

// 面向目標
function FaceToNowTarget (){
// 注視目標
transform.LookAt (lookAtPos);
}

// 面向玩家
function FaceToPlayer (){
var rotation = Quaternion.LookRotation(lookAtPos - transform.position);
transform.rotation = Quaternion.Slerp(transform.rotation, rotation, Time.deltaTime * followRotateSpeed);
}

// 攻擊敵人狀態
function Attack (){
while(true){
if (SeachEnemy()){
if (attackRange >= targetDistance(targetPlayer)){
AttackEnemy();
yield;
}else if(warnRange >= targetDistance(targetPlayer)){
Warn();
yield;
}else{
FollowEnemy();
yield;
}
}else{
return;
yield;
}
}
}

```

```
}  
}
```

// 攻擊敵人

```
function AttackEnemy (){  
  renderer.material = attackMaterial;  
  FaceToNowTarget();  
  print("AttackPlayer");  
}
```

// 警戒敵人

```
function Warn (){  
  renderer.material = warnMaterial;  
  FaceToNowTarget();  
  print("WarnPlayer");  
}
```

發射子彈

//定義一個 FirePoint 作為發射點的接口

```
var FirePoint:Transform;
```

//定義一個 Bullet 作為炮彈對象的接口

```
var Bullet:Rigidbody;
```

```
function Update () {
```

//當滑鼠左鍵按下的時候，就發射一枚子彈

```
if(Input.GetMouseButtonDown(0)){
```

```
  var clone:Rigidbody;
```

```
  clone = Instantiate(Bullet,FirePoint.position,FirePoint.rotation);
```

//讓子彈以 100 的速度向前發射

```
  clone.velocity = transform.TransformDirection(Vector3.forward*100);
```

```
}
```

```
}
```

子彈碰撞產生的火花

```
var boomFire : Transform;
//函數判斷物體是否與其他物體產生碰撞，一旦碰撞就執行以下程式碼
function OnCollisionStay(collision : Collision) {
//創建一個火花方塊，擺放位子為子彈碰撞的位子
Instantiate(boomFire,this.transform.position,this.transform.rotation);
//刪除掉火花方塊
Destroy(gameObject);
}
```

怪物生命

```
var tankLife = 4;
function OnCollisionStay(collision : Collision)
{
//查所有的碰撞體，如有碰撞體的名字是"shot(Clone)",生命值就減 1
for (var contact : ContactPoint in collision.contacts)
{
if(contact.otherCollider.name == "shot(Clone)")
{
if(tankLife>0)
{
tankLife --;
}
}
//當生命值小於 1，物體就消失
if(tankLife<1)
{
Destroy(gameObject);
}
}
```

```
}  
}
```

人物的移動速度血量參數

```
var speed = 6.0;  
var jumpSpeed = 8.0;  
var gravity = 20.0;  
var health = 200.0;  
  
private var moveDirection = Vector3.zero;  
private var grounded : boolean = false;  
  
function FixedUpdate() {  
    if (grounded) {  
        // We are grounded, so recalculate movedirection directly from axes  
        moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical"));  
        moveDirection = transform.TransformDirection(moveDirection);  
        moveDirection *= speed;  
  
        if (Input.GetButton ("Jump")) {  
            moveDirection.y = jumpSpeed;  
        }  
    }  
  
    // Apply gravity  
    moveDirection.y -= gravity * Time.deltaTime;  
  
    // Move the controller  
    var controller : CharacterController = GetComponent(CharacterController);  
    var flags = controller.Move(moveDirection * Time.deltaTime);  
    grounded = (flags & CollisionFlags.CollidedBelow) != 0;  
}  
  
@script RequireComponent(CharacterController)
```

人物控制



```

using UnityEngine;
using System.Collections;

[AddComponentMenu("Camera-Control/Mouse Look")]
public class MouseLook : MonoBehaviour {

public enum RotationAxes { MouseXAndY = 0, MouseX = 1, MouseY = 2 }
public RotationAxes axes = RotationAxes.MouseXAndY;
public float sensitivityX = 15F;
public float sensitivityY = 15F;

public float minimumX = -360F;
public float maximumX = 360F;

public float minimumY = -60F;
public float maximumY = 60F;

float rotationX = 0F;
float rotationY = 0F;

Quaternion originalRotation;

void Update ()
{
if (axes == RotationAxes.MouseXAndY)
{
// Read the mouse input axis
rotationX += Input.GetAxis("Mouse X") * sensitivityX;
rotationY += Input.GetAxis("Mouse Y") * sensitivityY;

rotationX = ClampAngle (rotationX, minimumX, maximumX);
rotationY = ClampAngle (rotationY, minimumY, maximumY);

Quaternion xQuaternion = Quaternion.AngleAxis (rotationX, Vector3.up);
Quaternion yQuaternion = Quaternion.AngleAxis (rotationY, -Vector3.right);

transform.localRotation = originalRotation * xQuaternion * yQuaternion;
}
else if (axes == RotationAxes.MouseX)
{

```

```

rotationX += Input.GetAxis("Mouse X") * sensitivityX;
rotationX = ClampAngle (rotationX, minimumX, maximumX);

Quaternion xQuaternion = Quaternion.AngleAxis (rotationX, Vector3.up);
transform.localRotation = originalRotation * xQuaternion;
}
else
{
rotationY += Input.GetAxis("Mouse Y") * sensitivityY;
rotationY = ClampAngle (rotationY, minimumY, maximumY);

Quaternion yQuaternion = Quaternion.AngleAxis (-rotationY, Vector3.right);
transform.localRotation = originalRotation * yQuaternion;
}
}

void Start ()
{
// Make the rigid body not change rotation
if (rigidbody)
rigidbody.freezeRotation = true;
originalRotation = transform.localRotation;
}

public static float ClampAngle (float angle, float min, float max)
{
if (angle < -360F)
angle += 360F;
if (angle > 360F)
angle -= 360F;
return Mathf.Clamp (angle, min, max);
}
}

```