東　海　大　學

資訊工程學系

專題研究報告

3D 動畫之設計與實作：以東海資工系導覽為例

3D Animation Design and Implementation: Visiting CS Department of Tunghai as an Example

指導教授：林祝興 老師

學生：963805 林容暄

963819 張通傑

中　華　民　國　九 十 九　年　十二　月

目錄

一、緒論

1.1 摘要

由於電腦科技的發展日新月異，電腦動畫也迅速的在蓬勃發展，從早期的 2D 只有上下左右的移動方式，到 3D 的全視角的移動方式。另外在模型與人物製作上伴隨著科技的發展越來越精緻。在許多領域都能看見 3D 動畫的蹤影，大至電影、遊戲，小至廣告、文宣，電腦繪圖已經充斥在我們日常生活周圍。

結合網路的發展，例如房屋仲介透過網路來呈現房屋的樣式與格局，買家就可透過網路檢視許多房屋，便可節省買家時間，以增加閱覽房屋的數量。藉由目前 3D 繪圖軟體的強大功能，所建立出來的房屋的場景已可媲美由攝影機所拍攝出來的效果。

承襲 94 年學長的東海校園導覽使用的 MAYA 繪圖軟體，在建模方面加以學習並更加優化，有別於 94 年學長的文理大道及、圖書館以及中正堂的大規模建築，這一次我們注重在單一建築的優化與細節，進而從大規模的文理大道縮減為資工系系館(大智慧科技大樓的三樓和四樓)，除了導覽之外我們更增加了自由控制的系統，讓使用者可以身歷其境，在電腦前也可以輕鬆的了解東海資工系系館。

1.2 研究動機與目的

接下專題之後，經過多次的討論，決定承襲 94 年學長的東海校園導覽製作東海校園模組，用來介紹各個學院以及東海校園環境，然而在建模與外掛程式設定的過程中有許多的挫敗，而後聽從專題指導老師的建議，將大目標的東海校園導覽漸漸縮小到資工系系所導覽，而在建模上可以更加的精緻，外掛程式方面也可以加入更精細的裝置，最重要的事是可以讓使用者操控第三人稱單位在建築內移動，轉向這個明確的方向之後，開始著手研究在 UNITY3D 裡的外掛程式要如何編寫，與模型之間的配合度及相容性也是一個課題，而在 MAYA 建模過程中，發覺門、窗、樓梯，這些最常使用的物件在建模上是非常頻繁的，而為了真實性還是讓門跟電梯可以使用而不只是裝飾品，這方面也要下許多的功夫，門牌上的提示可以更清楚各個實驗室的研究方向以及研究目的，希望這個應用程式可以讓加入東海資工的同學或學弟可以更快速的融入東海資工這個大家庭，利用我們的專題研究的導覽可以快速的了解資工系系辦以及實驗室的位置。

二、技術背景

2.1 MAYA 2011 介紹

MAYA 背景介紹:

MAYA 原公司為 Alias Wavefront。當時所研發的軟體是專門給工業設計和動畫特效使用的。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是由 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。

在 2005 年 10 月 4 日，生產 3D Studio Max 的 Autodesk 公司宣佈正式以$1.82 億美元收購生產 Maya 的 Alias，之後也收購了 Mud box。

今年 2008 年 Autodesk 公司將旗下的四樣軟體全面翻新，推出最新版本，Autodesk Maya 2009，Autodesk 3ds Max 2009，Autodesk Motion Builder 2009，Autodesk Mud Box 2009，稱為 Autodesk 的 M4 全方位解決方案，讓設計開發人員可以將無限的想像具體實踐，並可與開發流程進行完美整合，創造出款款創意無限的精彩作品。

Maya 在現在電影特效製作中應用相當廣泛，著名的星際大戰前傳就是採用 Maya 製作特效的，此外還有蜘蛛人、魔戒三部曲、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 功能概述:

1.基本建立模型功能：

1.1User Interface：易於使用的工具，例如：Shelf 和三維操縱器，加快工作流程。

1.2Modeling：Polygons、NURBS 和細分表面建模工具。

1.3Brush-based Technology ：Maya Artisan, Maya Paint Effects 和 3D Paint 提供全套集成的筆刷工具，用於建模、創建二維和三維效果以及在物體和紋理上噴塗。

2.特效功能介紹

1.Visual Effects：利用物理定律確定的硬體和軟體的高速、動態相互作用。

2.Maya Fluid Effects：用來模擬天氣狀況、煙火場面、粘性流體甚至海洋的效果。

3.Maya Fur：容易在 NURBS 或者多邊形模型上創建逼真的毛皮、頭髮、羊毛和草。

4.Maya Cloth：可以模擬任何布料，如：帆布、皮革、帳篷、床單等。

5.Maya Live：實現二維實拍和三維元素結合的運動匹配。Maya Live 提供更好的靈活性，一個快速的集成二維跟蹤器和一個互動式的初始幀解算器。Maya Live 也允許使用者按三維物體重新構建實拍元素並輸出到外部應用程序。

6.Maya Hair：在 NURBS 或者多邊形物體上全動態長髮的創建和渲染工具。使所有的 NURBS 曲線動態化，用於高級角色操縱和效果。也可創建許多其他「非頭髮」的效果 .

3.其他特色功能

1.Rendering：統一的渲染工作可通過共同的界面輕鬆使用 Maya 的軟體、硬體、mental rayR 和向量渲染器。Toon Shader 可以輕鬆地創作出繪畫效果，並模擬傳統 2D 漫畫書、卡通和日本漫畫的外觀。

2.Maya API/SDK and MEL：通過嵌入式腳本語言和完美的應用程序，設計人員可自行設定和製作 GUI 和擴展 Maya 軟體功能。

3.Animation：全面、廣泛的關鍵格、非線性和高級動畫編輯工具，用於創建、製作、調整和重新設計動畫數據和編輯數字角色。

電腦硬體

硬體配備

	CPU	RAM	顯示卡
A 電腦	AMD Athlon Dual Core Processor 4850e	3.0GB	ATI Radeon HD 3200 Graphics
B 電腦	AMD Athlon Dual Core Processor 5200+	2.0GB	NVIDIA GeForce 6600LE

硬體概述

由於 MAYA 對於硬體設備要求非常高，若電腦配備沒有中上程度，使用 MAYA 各方面都會產生困難，A 電腦顯示卡沒有 B 電腦顯示卡好，所以在 MAYA 上跑動畫時，很不流暢；而 A 電腦的 RAM 比 B 電腦多，在物件多寡上可以比 A 電腦多一些，但仍然稍嫌不足。

2.2 UNITY 3D 介紹:

Unity 是一個用於創建諸如三維視頻遊戲、建築可視化、實時三維動畫等類型互動內容的綜合型創作工具。Unity 類似於 Director,Blender game engine, Virtools 或 Torque Game Builder 等利用交互的圖型化開發環境為首要方式的軟體

在正式發布前，Unity 經歷的多年的開發階段，Gooball 在 2005 三月發布，運用了 Unity 的預發售版本開發

- 2005 六月 Unity1.0.1 發布
- 2009 三月 Unity2.5 加入了對 Windows 的支持
- 2009 十月 Unity2.6 獨立版開始免費

其編輯器運行在 Windows 和 Mac OS X 下，可發布遊戲至 Windows,Mac,Wii, 或 iPhone 平台。也可以利用 Unity web player 外掛程式發布網頁遊戲，支持 Mac 和 Windows 的網頁瀏覽。它的網頁播放器也被 Mac widgets 所支持。

已知用 unity 開發的知名遊戲有:Bionicle Glatorian Arena - a single player game based on the Bionicle Glatorian line of 2009.、Cartoon Network Universe: FusionFall、Foreign Legion: Buckets of Blood、GooBall、Global Conflicts: Palestine、My Animal Centre、Off-Road Velociraptor Safari、Venus Redemption、WolfQuest、Fatale、Tumbledrop

主要特性

1. 層級式的綜合開發環境，可視化編輯，詳細的屬性編輯器和動態的遊戲預覽。Unity 也被用來快速的製作遊戲或者開發遊戲原型，
2. 可開發微軟 Microsoft Windows 和 Mac OS X 的可執行文件，在線內容（通過 Unity Web Player 外掛程式支持 Internet Explorer,Firefox,Safari,Mozilla,Netscape,Opera 和 Camino,Mac OS X 的 Dashboard 工具，Wii 程序和 Iphone 應用程序. 開發 Wii 和 iPhone 需要用戶購買額外的授權，在價格上不同
3. 自動資源導入-項目中的資源會被自動導入，並根據資源的改動自動更新。雖然很多主流的三維建模軟體為 Unity 所支持，不過 3ds Max,Maya,Blender,Cinema 4D,和 Cheetah3D 可以被更好支持，並支持一些其他的三維格式
4. 圖形引擎使用的是 Direct3D(Windows),OpenGL(Mac,Windows)和自有的 APIs(Wii)

5. 支持 Bump mapping,Reflection mapping, Parallax mapping, Screen Space Ambient Occlusion,動態陰影使用的是 shadow map 技術，並支持 Render-to-texutre 和全屏 post processing 效果。
6. shaders 編寫使用 ShaderLab 語言,同時支持自有工作流中的編程方式或 Cg.GLSL 語言編寫的 shader.一個 shder 可以包含眾多變數及一個參數介面，允許 Unity 去判定參數是否為當前所支持並適配最適合參數，並自己選擇相應的 shader 類型以獲得廣大的兼容性
7. 內置對 Nvidia's 的 PhysX physics engine 支持
8. 遊戲腳本為基於 Mono 的 Mono 腳本，一個基於.NET Framework 的開源語言，因此程式設計師可用 JavaScript. C#或 Boo 加以編寫
9. The Unity Asset Server -一個支持各種遊戲和腳本的版本控制方案，使用 PostgreSql 作為後端
10. 音效系統基於 OpenAL 庫，可以播放 Ogg Vorbis 的壓縮音效
11. 視頻播放採用 Theora 編碼
12. 山體和植被引擎，支持樹木貼片

2.3 FBX 介紹:

由於 3D 軟體的檔案格式眾多，如果您需要將不同的檔案格式從一套軟體轉換到另一套軟體時，您將會發現到檔案格式受到相當多的限制，這對於一個大型的後製作公司來看，將會在轉換的過程中耗費掉許多時間及預算成本。因此 Alias 提供了免費的 FBX 外掛程式給市場上所有主流的 3D 與後製作軟體，您可以藉由她來做為軟體之間的溝通橋樑，由於您可以輕易地利用 FBX 來轉換不同軟體之間的模型、貼圖、動畫、骨架、燈光、攝影機或聲音等資料，這對於沒有 R&D 部門的公司來說，這將是一項整合性的大利多。

fbx 是 filmbox 這套軟體所使用的格式，現在改稱 Motionbuilder。Motionbuilder 是一套專攻動作製作的軟體(尤其是配合 mocap 系統)，詳細的資料可以查網路上的資訊，這裡就點到為止了。

Motionbuilder扮演的是動作製作的平台，所以在前端的modeling和後端的rendering也都有賴於其他軟體的配合，所以 Motionbuilder 在檔案的轉換上自然下了一番功夫。

FBX 有提供 Quicktime6 的 Plugin，意思就是說只要有安裝這個插件，就可以直接在 Quicktime Player 裡面顯示 FBX 的模型，Viewport 的操作方式和 Motionbuilder 相同，滑鼠只有左鍵有功用。

三、程式概述

在 unity3D 軟體裡面，我們用的是裡面內建的編輯程式，它提供了 C#，JavaScript 等程式語言，而我們採用的是 JavaScript，這是我們之前沒使用過的語言，但一些基本的語法其實都跟我們之前學過的 C++差不多，程式寫作最困難的地方就是一開始完全不知道該怎麼使用 unity3D 裡內建的 function，參數要怎麼傳都一概不知，只能用 unity3D 官方網站裡面所提供的 Scripting Overview，來一個個慢慢查出我所需要功能相對應的 function。

最初，我們用網路上的教學影片來慢慢認識這個軟體的使用和程式碼的寫作，之後才開始做我們自己想呈現的東西。

以下是一些我們撰寫的程式碼:

3.1 人物的主要部分:　　程式碼:ThirdPersonController

人部份的程式碼是整個專題中最為複雜的，光是用在寫人物的移動、動畫的控制、各種動作，就需要大量的撰寫。大致可分為四個部份，一開始是宣告將要用到的變數或函數，一部分是動畫的設置，設定在適當的時機播放出欲讓人物動作的動畫;像是按住 SHIFT 就能跑步,平時不動時會變成 idle 狀態，一部分就是最重要的移動功能,設定輸入上下左右能夠控制人物的前進後退和轉向，function UpdateSmoothedMovementDirection () 此 function 就包含了所有移動的控制，包括了判定前進後退，是否是著地的狀態，跑步跳躍和走路速度等等。一部分就是 function Update() 這 function 表示著是隨時都在更新的，上述的移動功能會因為人物隨時在走動而改變參數，依照人物現在的狀況來做更新動作。還有一部分是比較特別的，是為了修改我們當初碰到的一個問題；我們發現把人物送上電梯後，到達某一高度或是任意上下時，人物會從平台上往下掉，這讓我們思考了很久，問題就出在當人物上電梯後，電梯平台雖然是在上升，但人物的座標確沒跟著改變，因此我們加入了一些程式碼，讓人物的移動和電梯的上下是同步的，才解決了這部份的 bug。

3.2 電梯:

3.2.1 電梯:升降台部份　　程式碼:elevator

電梯部分的程式碼比較瑣碎，但是也比較簡單，我用 function Update() 來做移動平台的停止，用的是 IF 判定句和 TAG 標籤來標示當電梯到達最高點，則 TAG 設為 Finish，當平台的 TAG 被設定成 Finish 時，就判定他停止。

```
if( targetA.gameObject.tag == "Finish" )
```

```
{
    onelevator = false ;                    // onelevator 設為 fals 則停止移動
    targetA.gameObject.tag = "Untagged";    // targetA 是指移動平台
                                            //當停止後將 TAG 設為初值
}
```

判定電梯到達最高點，是使用一個觸發器，當電梯平台觸碰到觸發器，則電梯的 TAG 被改變。function FixedUpdate () 也是隨著時間不斷的執行的 function，用來給電梯移動的速度，當 onelevator == true 時才會讓電梯有速度。

3.2.2 電梯:用來控制電梯停住

程式碼:trigger up down

解說如 3.2.1，我們使用 Trigger(觸發器)來停止電梯，function OnCollisionEnter(collision : Collision) 此 function 用來判別是否有東西撞擊到該物件，若有則將電梯 TAG 改成 Finish 讓它停止。

3.2.3 電梯:按鈕控制(下)　　程式碼:Text buttons Down

我們在電梯裡做了兩個按鈕，用來控制電梯的上和下，當滑鼠移到按鈕上面時，按鈕顏色會改變。用 function OnMouseUp() 來判定使用者是否點擊了按鈕，用 OnMouseUp() 而不是 OnMouseDown() 是為避免使用著點擊按鈕不放恐造成錯誤。當使用著點擊後，會更改電梯的 TAG 成 up or down 來促使上述 elevator 之程式碼讓電梯移動。

3.3 視窗功能　　程式碼:Window1

這是我們用來和標示牌對話的功能，我們能點擊在某些教室外面的標示牌，會跳出一個視窗來為我們解說此教室的內容。

3.4 拖曳剛體(門)　　程式碼: DragRigidbody

此功能讓我們可以做出有如真實的門，用滑鼠按住並且拖曳就可以達成把剛體任意拖拉的效果。程式碼部分是參考 UNITY3D 的 Scripting Overview。

3.5 順移程式碼　　程式碼:Domywindow

此部分的程式碼是包含在 ThirdPersonController 程式碼裡面，此工能提供了一個功能視窗，標示了電梯的移動狀態和許多按鈕，按鈕上面標示著各個 ST 教室的名稱，當我們想直接移動到某地點，可以點擊按鈕直接讓腳色移動到該位置，讓導覽功能更快速和方便。

Transform.position = Vector3(-314, 51, 140);

// Transform.position 表示著人物的三維座標。

3.6 影子程式碼 程式碼:BlobShadowController

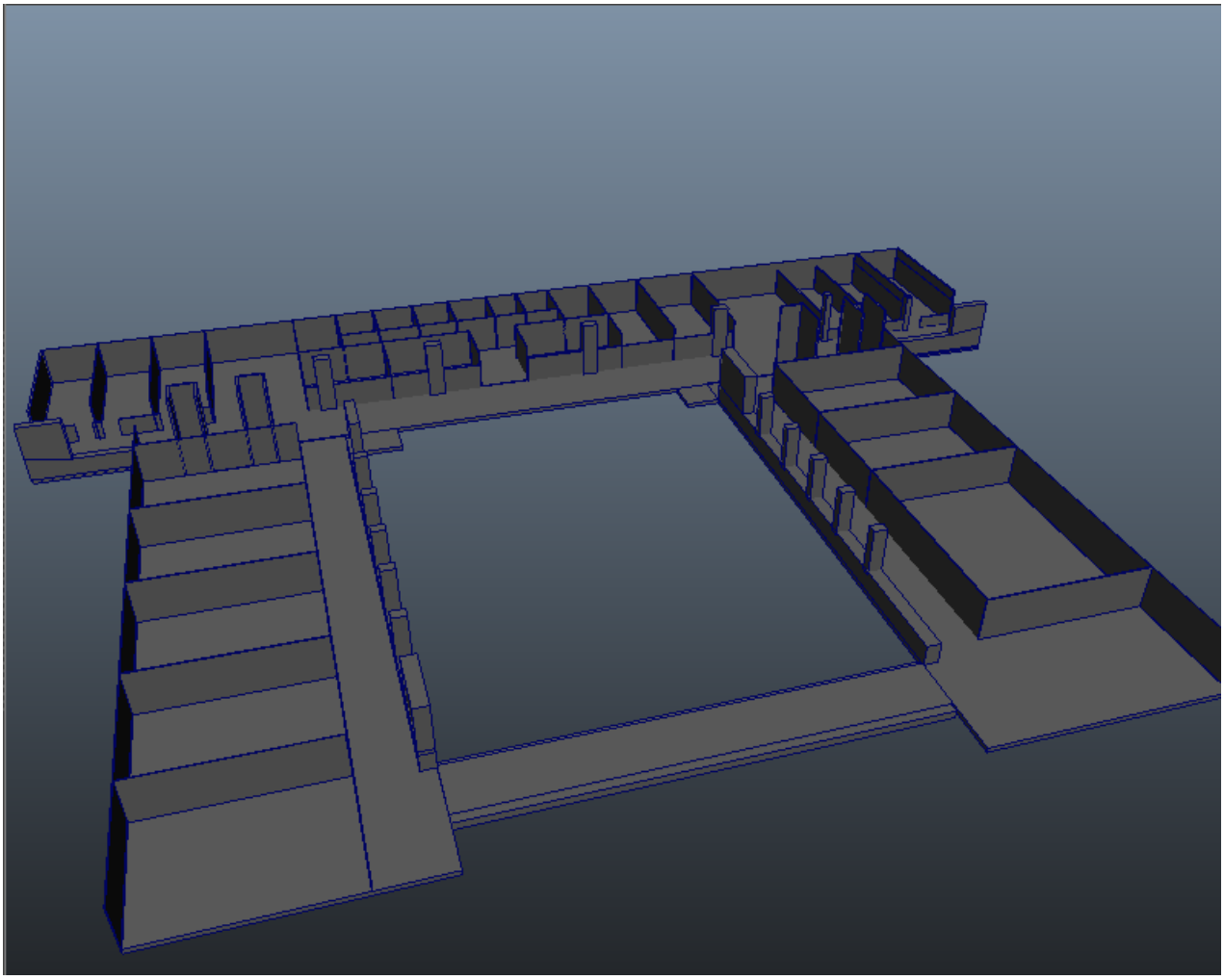
此程式碼用來控制影子的部份讓 projector 能確實的將貼圖投影出來，由於真實的影子判定需要用到 PRO 的軟體，所以我們用模擬的方式做一類似影子的效果。

四、專題實作

建模:建模方面我們主要是在 MAYA 程式下建模，因為他在建造模型的性能比 3DMAX 簡單而且容易操作，相較於 UNITY3D 也有更多的物件可以讓模型呈現更完整的樣式。

4.1 建模

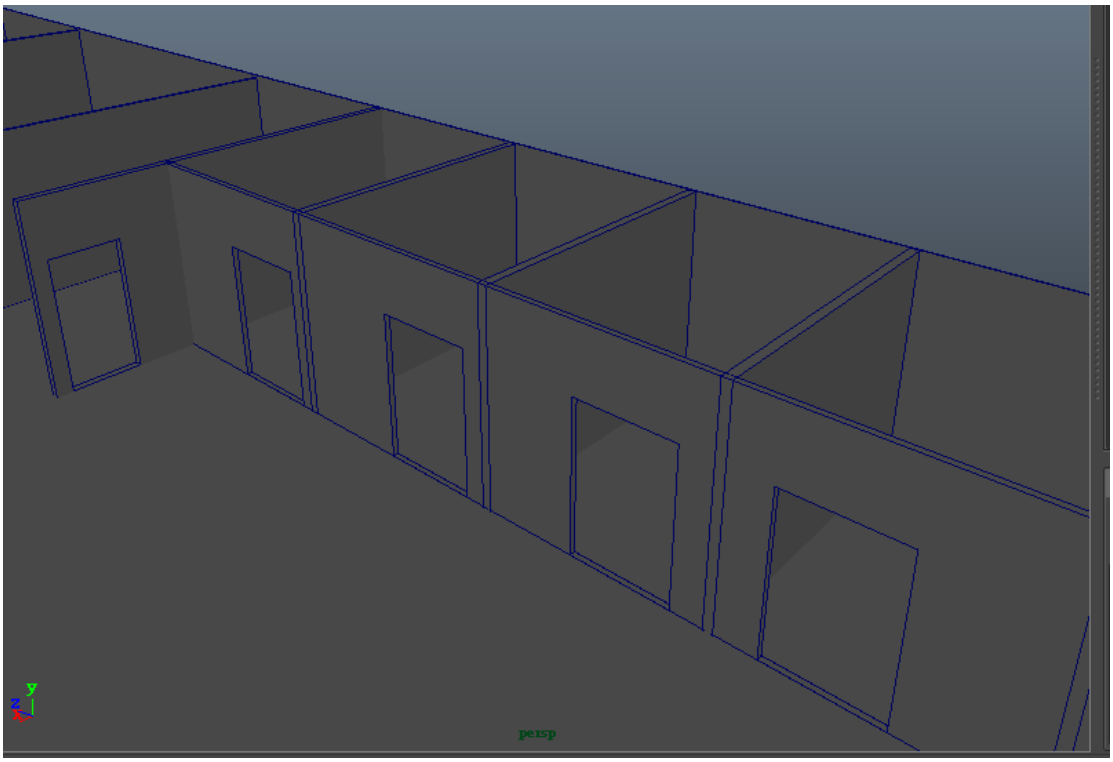
ST 大樓三樓和四樓，首先建構建築大概，參見附圖一



(附圖一)

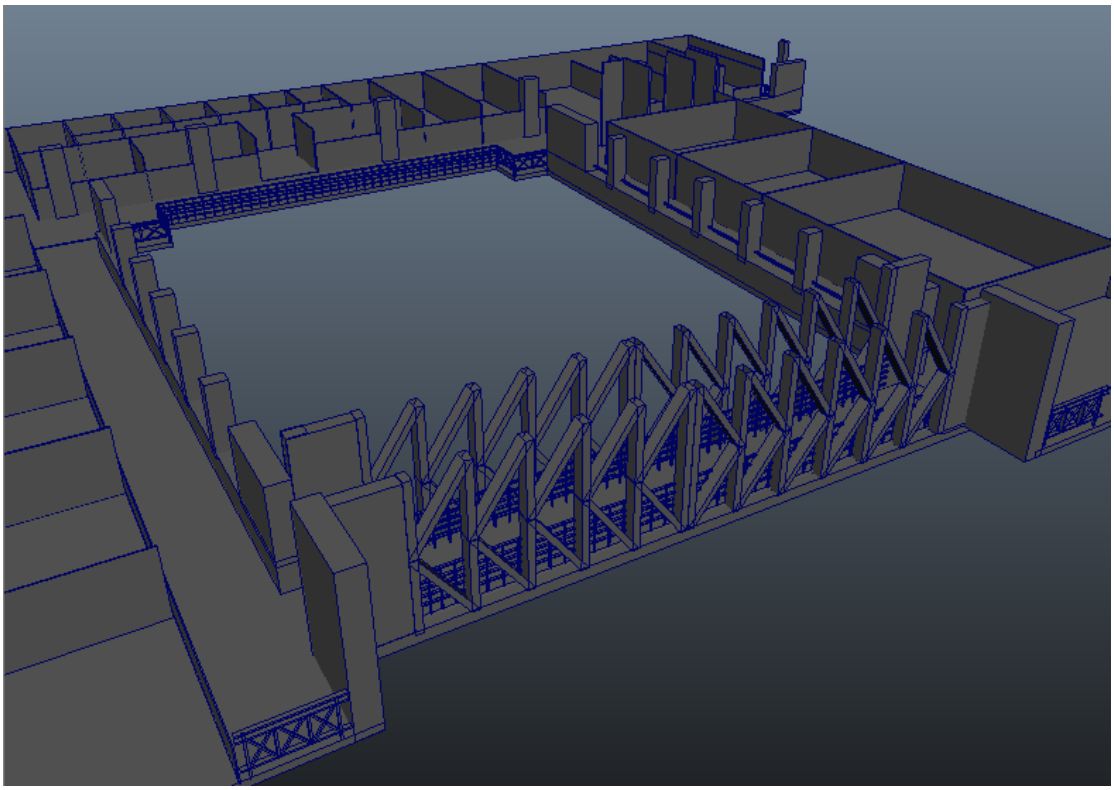
依各個房間的需求作挖門的動作， 要先做出所需門的大小 CUBE

Mesh->Boolean->Difference



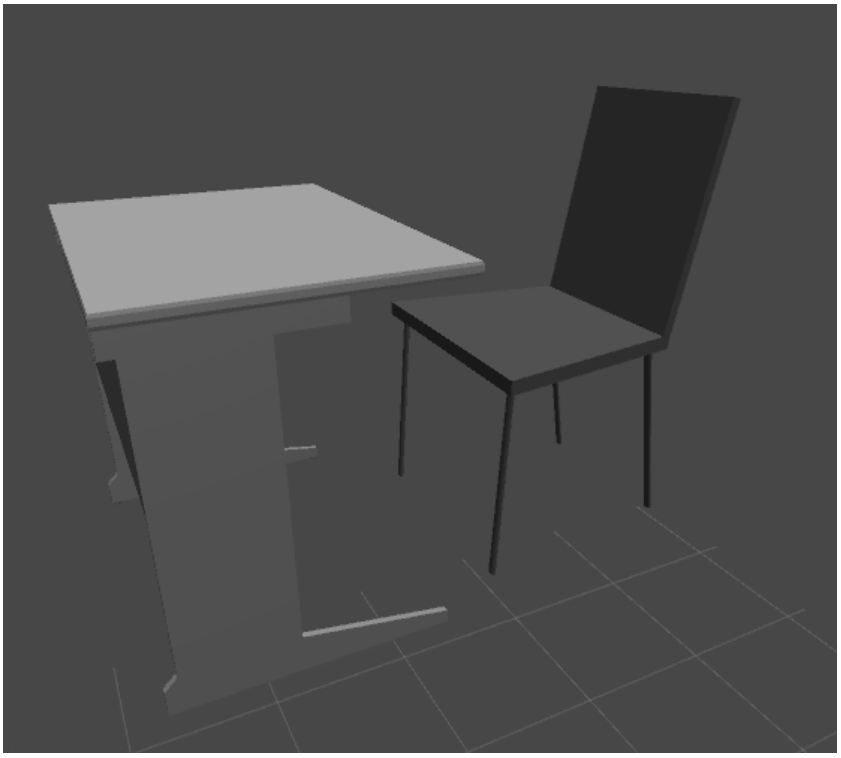
(附圖二)

最後架入欄杆

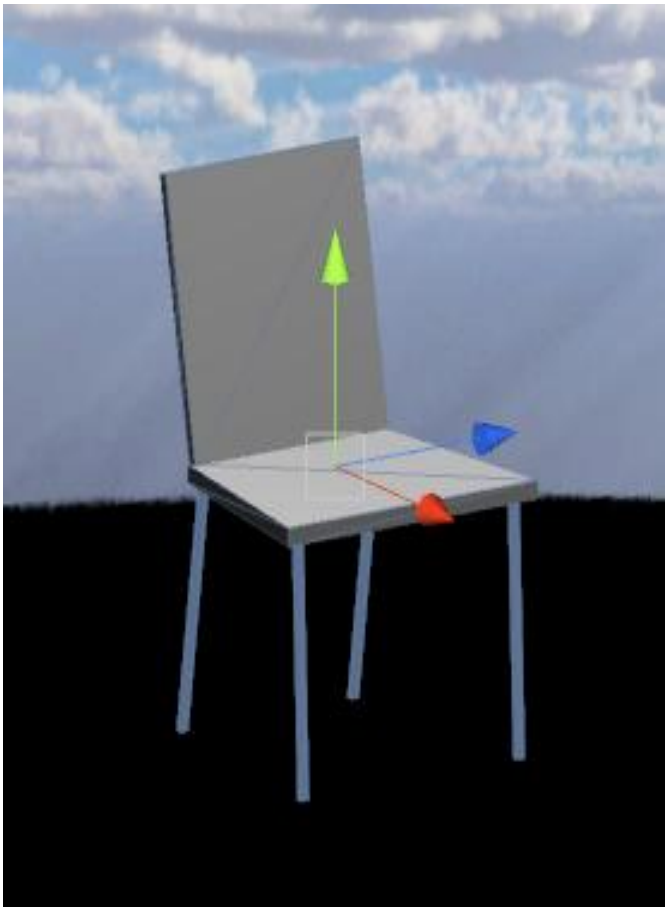


(附圖三)

課桌椅:

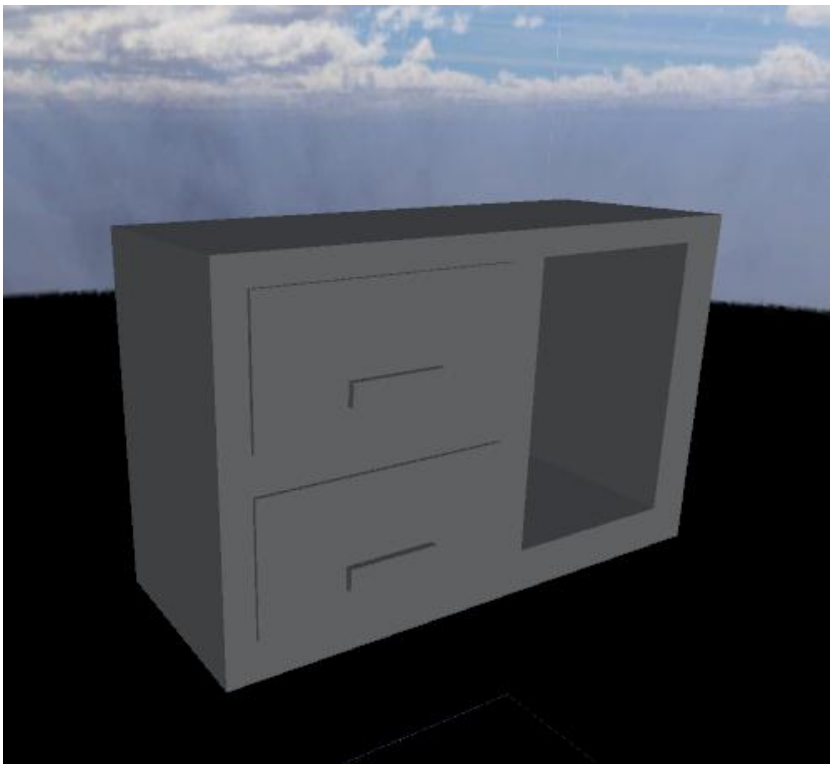


(附圖四)



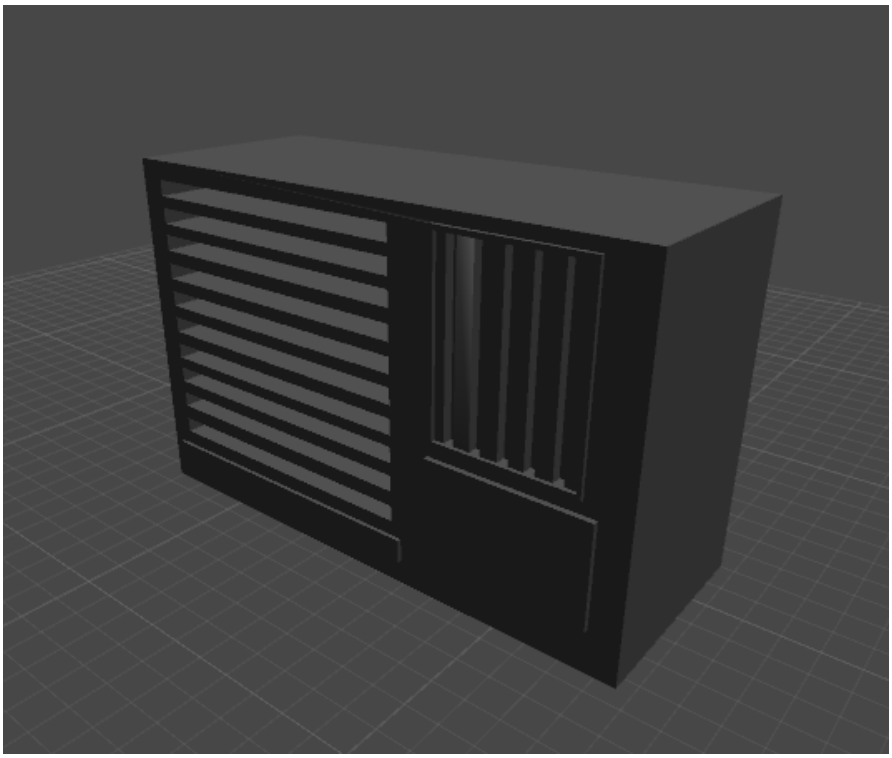
(附圖五)

器材:櫃子



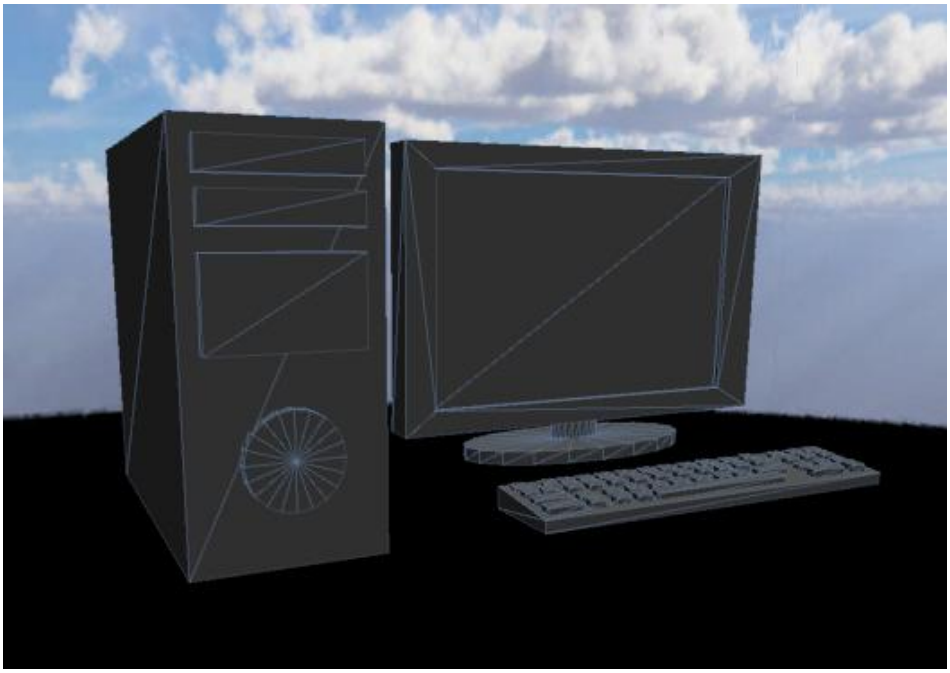
(附圖六)

器材:冷氣機



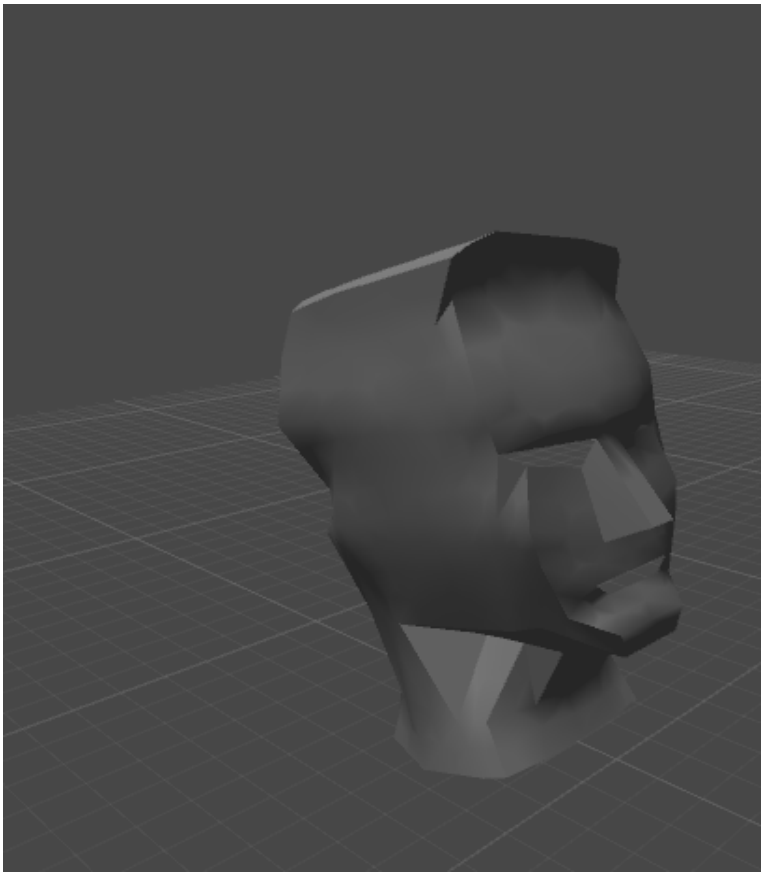
(附圖七)

器材:PC



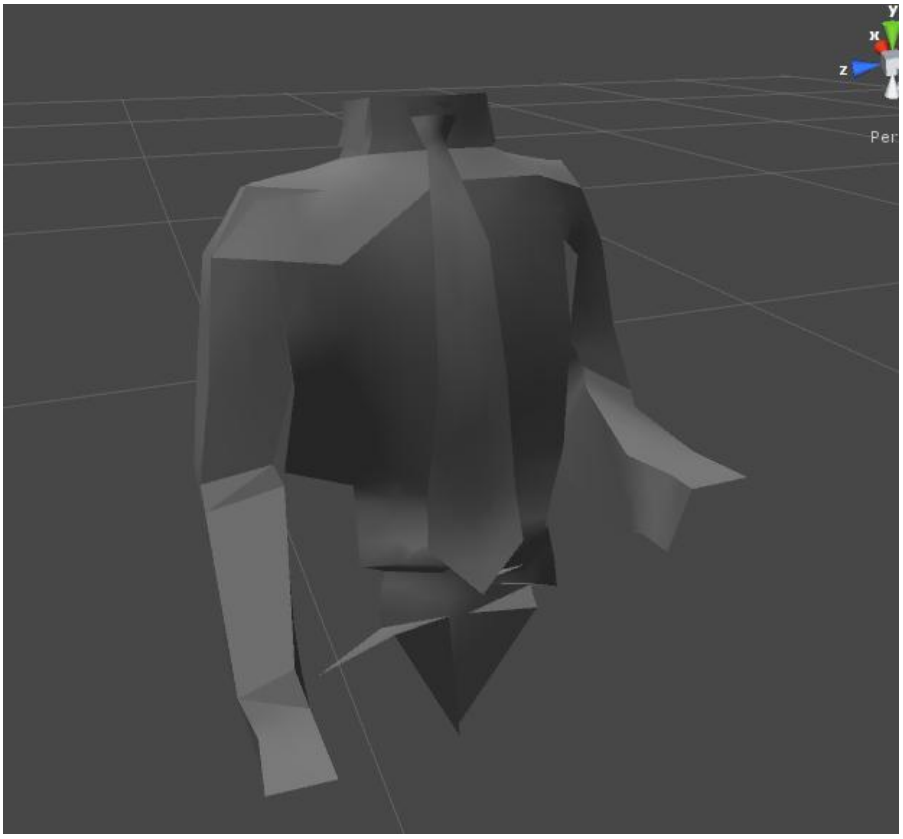
(附圖八)

人物:頭



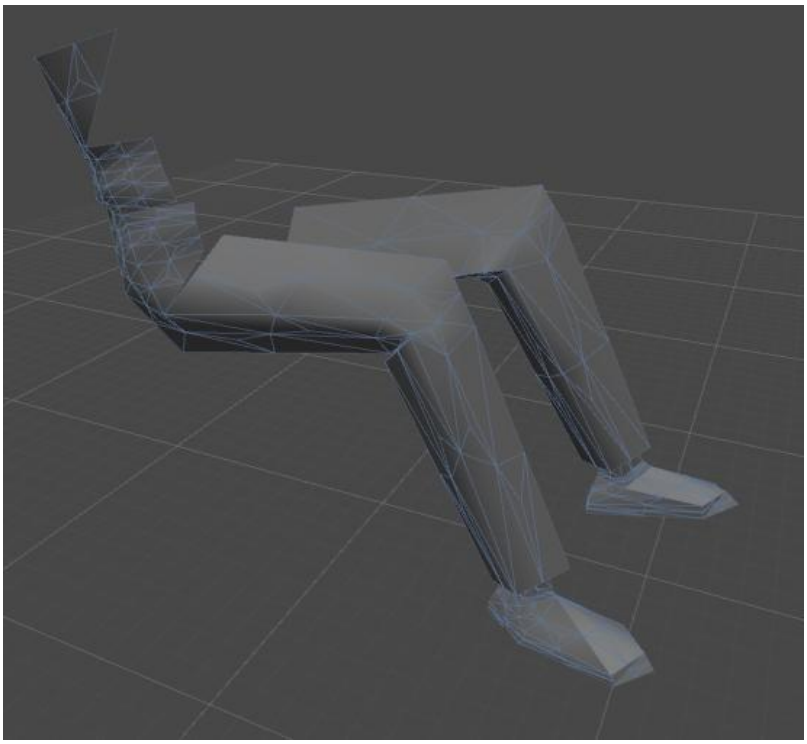
(附圖九)

人物:上半身



(附圖十)

人物:下半身



(附圖十一)

4.2. 匯入:

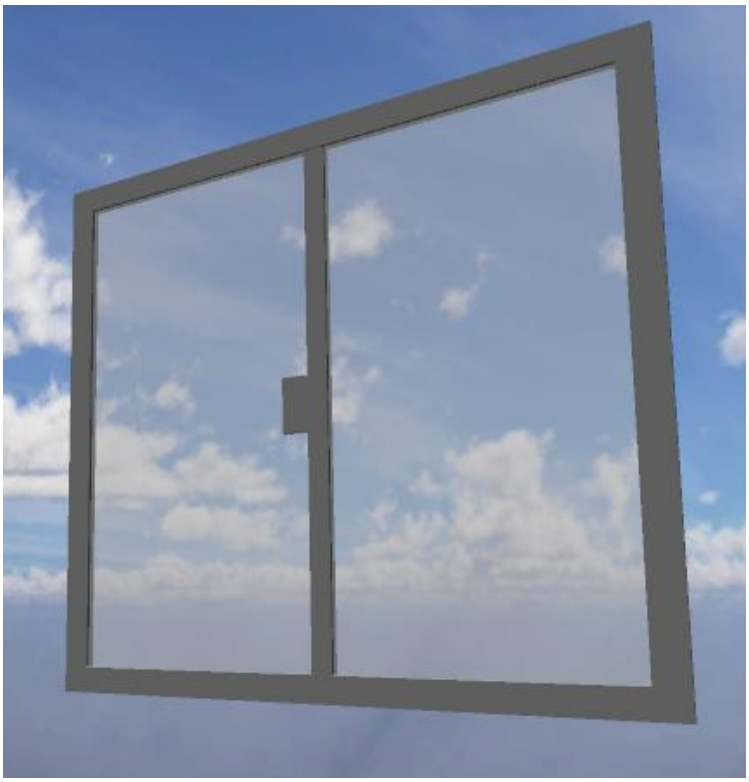
步驟 1. ,將轉檔類型調出來,預設狀態下 fbx 是被隱藏的,可選擇 load 或連 auto 一起選，auto 表示下次開啟 maya 會自動被調出,不用在手動 load 一次。
步驟 2. 可選擇 export all 表示該場景中的所有物件都導出成 fbx 檔，或是 export selection 導出選取物件，該選項必須先選中場景中的任一想導出的單一或多個物件。

FBX
貼圖:這個部份開始就把操作平台轉換到 UNITY3D，因為在 MAYA 先把圖片貼好再進行匯入的動作的話會造成圖片位移甚至錯位。

4.3 貼圖:

首先要新增一個材質球(Material)，加入所需要的材質色彩，然後選擇材質(金屬、木頭…etc)，可以調整圖案的重覆度與顏色，最後把設定完成的材質求拖曳至預賦予材質的物件。

Ex: 玻璃:

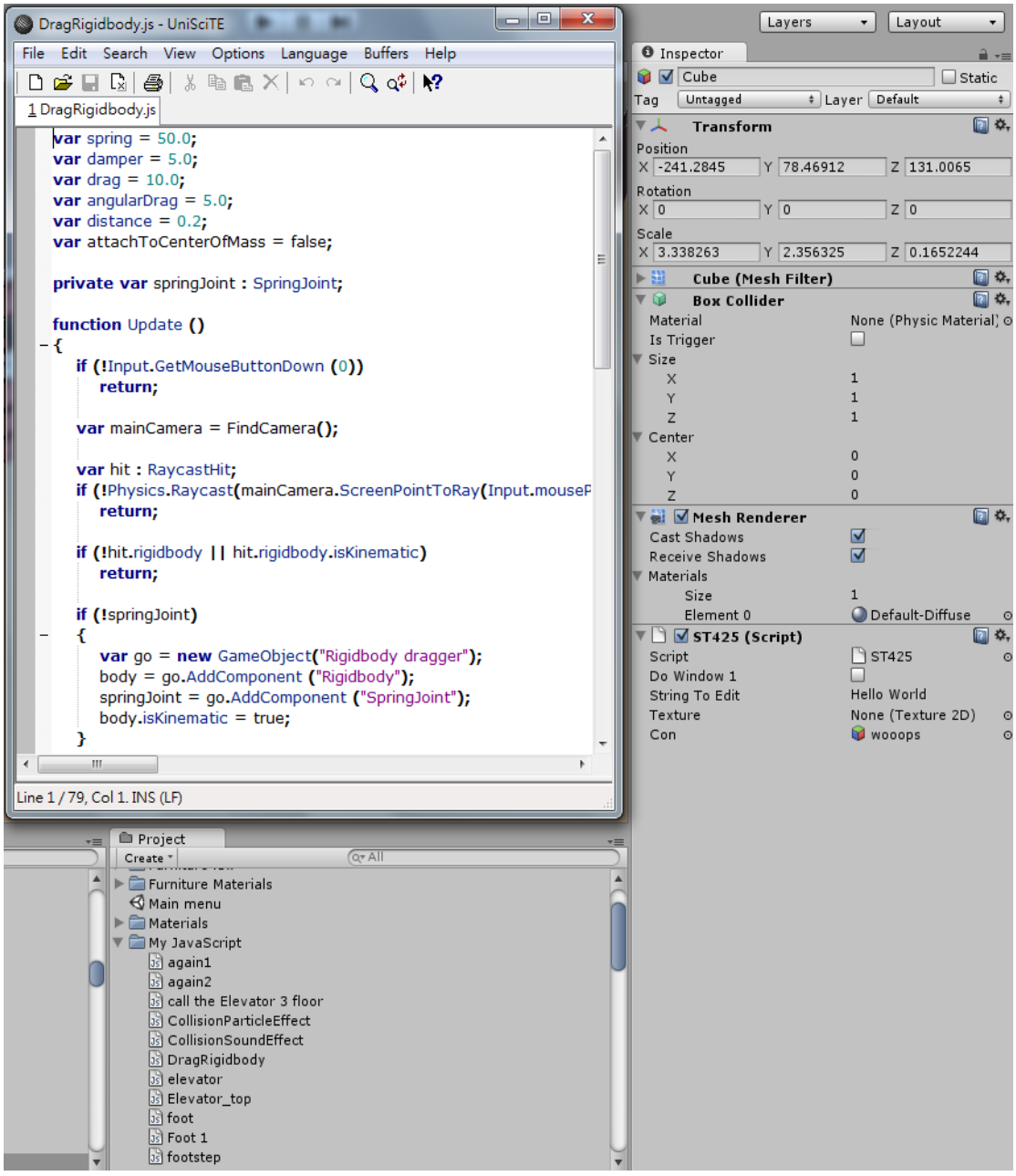


(附圖十二)


door2-trimmaterial 1
Shader
Transparent/VertexLit
Main Color
Spec Color
Emissive Color
Shininess
Base (RGB) Trans (A)
Tiling
Offset
x
1
0
y
1
0
Select
Preview

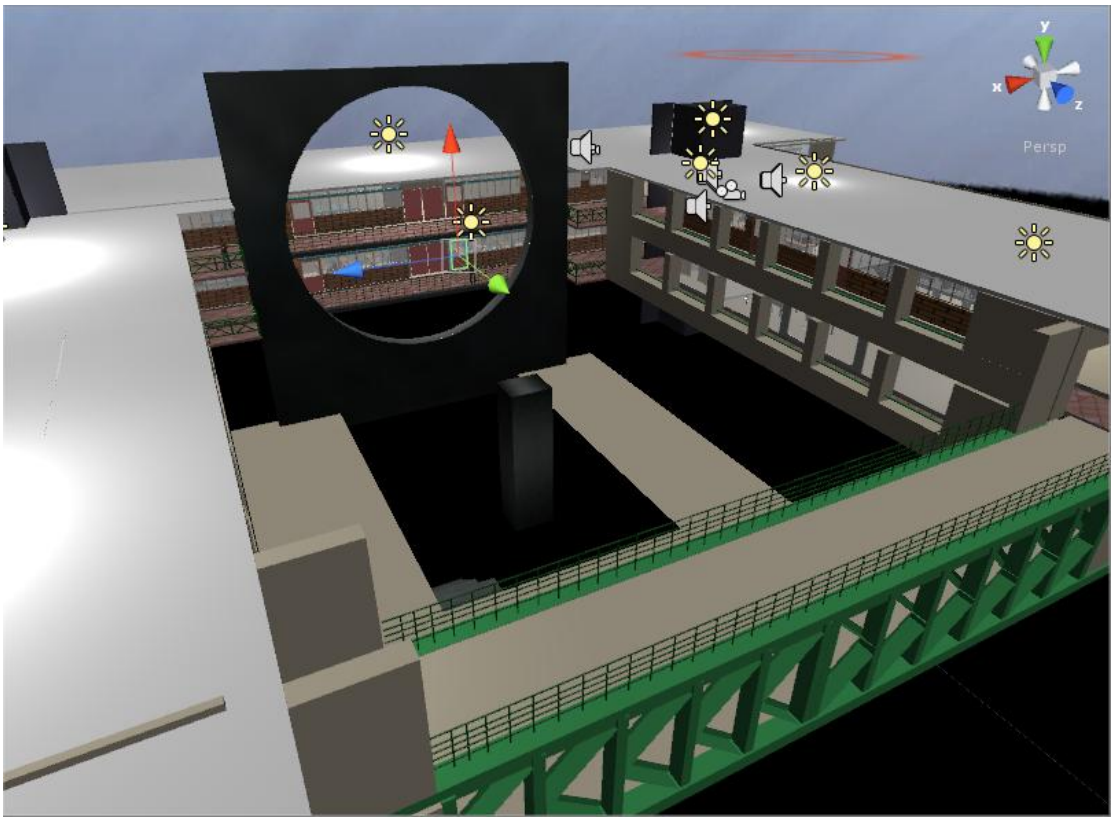
(附圖十三)

4.4 程式碼給定



(附圖十四)

4.5 完成圖：



(附圖十五)

(附圖十六)

五、總結:

5.1 實作問題:

在製作 MAYA 建模與 UNITY3D 遊戲控制的過程中，從學習操作軟體到完成作品，所遭遇到的困難不勝枚舉，以下就列出較為重要的幾點：

1. 國內教學資源太少，雖然國內對動畫製作的意識漸漸抬頭，畢竟和先進國家仍有一段距離，因此在國內的書籍與網頁資源為數不多，為了解決資源不足的問題，我們先借閱圖書館內的書籍，將 MAYA 的基本功能大致了解熟悉後，進而利用教學影片學習相關技巧，也在網路平台找尋與學習更多的方法，而 UNITY3D 國內目前只有一本中文教學書至逢甲大學接來研讀之後，接下來我們用網路上的教學影片來慢慢認識這個軟體的使用和程式碼的寫作，最後才能著手做我們自己想呈現的東西。

2. 精緻模型難以塑成，需要高度的美術能力與強烈的 3D 概念，精緻的人物模型就須具備以上的能力，然而我們只有基礎的技巧並且還在學習，所以精緻的人物模型是用網路上的資源，其他較簡單的物件則是學生以所學得技巧完成的。

3. 內建編輯程式語言與所學的不同，內建的程式語言有許多種，EX:C#、Javascript、XML，我們所選用的是 Javascript 做為編輯，卻因為之前沒接觸過所以造成學習上的困擾，僅能利用 UNITY 官網上的 Scripting Overview 來查詢所許要用到的指令名稱以及其意義。

4. 物件占用資源容易過於龐大，製作 3D 動畫需要高檔的硬體配備，對我們的配備而言，製作高品質的動畫並不容易達成。製作過程中，常會發生記憶體不足或錯誤的現象，CPU 的使用率也會隨著模型的細緻度上升，造成畫面停格或工作不流暢的狀況。這樣的情形，都會增加製作動畫的困難度。

由於教室課桌椅模組眾多，因而產生許多物件，造成檔案的龐大，所以在有限的資源內用較少的指令完成這些模型，便能減少資源的浪費。

隨著大智慧科技大樓 3.4 樓逐漸成型，我們發現製作過程中，MAYA 會將使用者所操作過的指令儲存在記憶體和檔案內，如此一來就便會浪費硬體資源。會因為歷史紀錄沒有刪除而造成匯入時的錯誤，所以在 MAYA 中都一指令可以清除歷史紀錄，這個的動作能稍微解決作業流暢度與匯入錯誤的問題，但這無法根除基本的問題，因為做動畫所需的硬體資源需求本身就很高。

5.2 專題心得-張通傑:

首先要謝謝林祝興老師接下我們這組專題生,在我們模糊不清的時候適時的給我們指引明確的方向,讓我們可以更快的找到目標並步上軌道,每一次的專題報告都會給我們適當的小叮嚀和方向錯誤的指正,也要感謝專題指導學姐的幫助,由於我們使用的 UNITY3D 軟體在國內的教學資源偏少,所以常麻煩他幫我們找國外的教學資料,也常常提醒我們甚麼東西應該要補充或修正。

在這次的專題中我主要是負責建造模型的部分,所以主要是專攻 MAYA2011 軟體,一開始到圖書館借書時以為學習一個新的軟體很簡單,卻發現沒有想像中的簡單,除了要了解每一個功能鍵的用處之外,對於三維空建的概念也要非常明確,而切 MAYA 對於一個新手來說並沒有很容易上手,所以剛凱史的學習是蠻坎坷的,教學資源看多了也對於 MAYA 也漸漸上手之後,才慢慢可以做出完整的物件,但是卻發生了物件佔用資源過大的問題,接著就是盡量以較少的步驟完成物件,為了要求整個遊戲系統的流暢度,要怎麼把物件佔用資源縮減卻又不失整體的完整性,在這裡又是一個複雜的課題,匯入 UNITY3D 之後就是跟林容暄一起討論各個物件該如何擺放以及其顏色,動作特效與真實度也都有參與討論,使這個專題可以更加的完整。

在這一年的的專題研究讓我學到了許多東西,不論是技術或是團隊合作的默契,技術方面或與沒有學習得非常的精通,未來還有很多的空間可以學習,希望可以把 MAYA 與 UNITY3D 兩套系統更加的研究並練習,而經過這次的專題與專題夥伴林容暄有更好的默契。

5.3 專題心得-林容暄:

起初開始知道要做專題的時候還沒有任何的想法,林祝興教授讓我們有自由選擇題目的空間,這讓我想到了之前學長做的 3D 校園導覽,我對 3D 動畫設計十分的有興趣,雖然知道這不是以往資訊系會做的東西,但還是決定走動畫設計來當我們的專題題目,一開始我們接續學長使用的 MAYA 來做摸索,由於學長們沒有設計人物在校園裡面走動,而是用攝影機並設定路線來導覽,所以我們打算讓導覽系統更具自由度,因此有了目標跟方向。起初花了很多時間在學習 MAYA,但要建出一個很真實的人物是非常有難度的,而且 MAYA 是個注重於建模的軟體,對於人物的控制和很多的導覽功能我們不知道該如何從 MAYA 做起,於是,我們找到了一個新的軟體:UNITY3D,這是個強大的軟體能匯入當前兩大 3D 軟體 MAYA 和 3DSMAX 做出來的模型,接著我們就開始著手學習使用 UNITY3D。

我負責的項目就是 UNITY3D 的程式碼寫作,用了之前完全沒接觸過的 JavaScript 程式語言來撰寫,UNITY3D 裡面提供了龐大又複雜的 function 來使

用，然而有關於此軟體的資源又相當少，我們也借了中文書來參考，我花了相當長的時間來找尋與應用到底有哪些 function 可以使用，最後完成我想要提供給使用者導覽的功能並符合真實生活的功能。程式碼的寫作讓我更了解了另一個語言的使用，而且讓我更明白程式存在的重要性，每一個小細節都須要程式碼來控制。專題的製作也讓我清楚跟夥伴合作的重要性，一個人是無法完成這麼多事情的，所以跟組員的分工合作就十分重要，也感謝學姐提供的許多幫助讓此專題能夠順利完成。

5.4 展望

在 UNITY 裡面完成的許多程式，由於知識和技術的不足有些都還只是半成品的水準而已，而後希望可以繼續研讀國外的教學資料，把半成品的部分的程式碼都可以加以改進並完成， 而在貼圖方面可以嘗試更多的材質球跟顏料，希望可以配出更真實的色彩，在 MAYA 建模的方面，由於這部分需要較多的美工技巧與 3D 成像的概念，所以需要在多方面研讀美術書籍以提升自己的美術概念及 3D 成像的技術，還需要多閱讀 MAYA 教學的進階書以增進建模計巧，而後有機會將會把這個專題繼續的研究並完成。

六、參考文獻:

UNITY3D 官方網站 :http://unity3d.com/

UNITY3D 問答 http://answers.unity3d.com/

UNITY3D Wiki http://www.unifycommunity.com/wiki/index.php?title=Main_Page

我的 MAYA 輕鬆學 臺北市/網奕資訊科技/2006[民 95] 陳偉介

Maya 動畫設計實例 洪振偉/旗標 2004

3ds max 進入 3ds max 夢想世界 臺北市/文魁資訊/2007[民 96] 黃清永

3ds Max 2008 動畫製作實戰演練 臺北縣三重市/加樺國際/2008.01

MAYA 8 創作講堂:建模.材質.燈光.動畫 臺北市/文魁資訊出版/2007/松崗總代理 蔡哲明

附件一 . 人物的主要部分程式碼:ThirdPersonController

```
@script AddComponentMenu ("ThirdPersonController")
// Require a character controller to be attached to the same game object
@script RequireComponent(CharacterController)
var runaudio : AudioClip;

var windowRect : Rect ;
var Elevator :GameObject ;
var other : GameObject;
//var ParticleEmitter.emit = false;
var target1 : GameObject;
  public var radarObject : GameObject;
public var idleAnimation : AnimationClip;
public var walkAnimation : AnimationClip;
public var runAnimation : AnimationClip;
public var jumpPoseAnimation : AnimationClip;

public var walkMaxAnimationSpeed : float = 0.75;
public var trotMaxAnimationSpeed : float = 1.0;
public var runMaxAnimationSpeed : float = 1.0;
public var jumpAnimationSpeed : float = 1.15;
public var landAnimationSpeed : float = 1.0;
private var _animation;



//Moving Platform support
private var activePlatform : Transform;
private var activeLocalPlatformPoint : Vector3;
private var activeGlobalPlatformPoint : Vector3;
private var lastPlatformVelocity : Vector3;



enum CharacterState {
      Idle = 0,
```

```
        Walking = 1,
        Trotting = 2,
        Running = 3,
        Jumping = 4,
}

private var _characterState : CharacterState;

// The speed when walking
var walkSpeed = 2.0;
// after trotAfterSeconds of walking we trot with trotSpeed
var trotSpeed = 4.0;
// when pressing "Fire3" button (cmd) we start running
var runSpeed = 6.0;

var inAirControlAcceleration = 3.0;

// How high do we jump when pressing jump and letting go immediately
var jumpHeight = 0.5;

// The gravity for the character
var gravity = 20.0;
// The gravity in controlled descent mode
var speedSmoothing = 10.0;
var rotateSpeed = 500.0;
var trotAfterSeconds = 3.0;

var canJump = true;

private var jumpRepeatTime = 0.05;
private var jumpTimeout = 0.15;
private var groundedTimeout = 0.25;

// The camera doesnt start following the target immediately but waits for a split second to avoid too much waving around.
private var lockCameraTimer = 0.0;

// The current move direction in x-z
```

```
private var moveDirection = Vector3.zero;
// The current vertical speed
private var verticalSpeed = 0.0;
// The current x-z move speed
private var moveSpeed = 0.0;

// The last collision flags returned from controller.Move
private var collisionFlags : CollisionFlags;

// Are we jumping? (Initiated with jump button and not grounded yet)
private var jumping = false;
private var jumpingReachedApex = false;

// Are we moving backwards (This locks the camera to not do a 180 degree spin)
private var movingBack = false;
// Is the user pressing any keys?
private var isMoving = false;
// When did the user start walking (Used for going into trot after a while)
private var walkTimeStart = 0.0;
// Last time the jump button was clicked down
private var lastJumpButtonTime = -10.0;
// Last time we performed a jump
private var lastJumpTime = -1.0;


// the height we jumped from (Used to determine for how long to apply extra jump
power after jumping.)
private var lastJumpStartHeight = 0.0;


private var inAirVelocity = Vector3.zero;

private var lastGroundedTime = 0.0;


var isControllable:boolean = true;

function Awake ()
```

```
{



    moveDirection = transform.TransformDirection(Vector3.forward);

    _animation = GetComponent(Animation);
    if(!_animation)
        Debug.Log("The character you would like to control doesn't have animations. Moving her might look weird.");

    /*
public var idleAnimation : AnimationClip;
public var walkAnimation : AnimationClip;
public var runAnimation : AnimationClip;
public var jumpPoseAnimation : AnimationClip;
    */
    if(!idleAnimation) {
        _animation = null;
        Debug.Log("No idle animation found. Turning off animations.");
    }
    if(!walkAnimation) {
        _animation = null;
        Debug.Log("No walk animation found. Turning off animations.");
    }
    if(!runAnimation) {
        _animation = null;
        Debug.Log("No run animation found. Turning off animations.");
    }
    if(!jumpPoseAnimation && canJump) {
        _animation = null;
        Debug.Log("No jump animation found and the character has canJump enabled. Turning off animations.");
    }

}
```

```
function UpdateSmoothedMovementDirection ()
{
	var cameraTransform = Camera.main.transform;
	var grounded = IsGrounded();

	// Forward vector relative to the camera along the x-z plane
	var forward = cameraTransform.TransformDirection(Vector3.forward);
	forward.y = 0;
	forward = forward.normalized;

	// Right vector relative to the camera
	// Always orthogonal to the forward vector
	var right = Vector3(forward.z, 0, -forward.x);

	var v =Input.GetAxisRaw("Vertical");
	var h =Input.GetAxisRaw("Horizontal");



	/*
	if (	(v!=0) | (h!=0) |	( animation.IsPlaying("run") )	)
	{

//	Debug.Log("1");
SendMessage("OnFootStrike", SendMessageOptions.DontRequireReceiver);
	}

	*/

	/*if	(	|	(Input.GetButton("Vertical"))	|
(Input.GetButton("Horizontal"))	)
	{

	Debug.Log("1");
SendMessage("OnFootStrike", SendMessageOptions.DontRequireReceiver);
	}
*/
```

```
// Are we moving backwards or looking backwards
if (v < -0.2)
	movingBack = true;
else
	movingBack = false;

var wasMoving = isMoving;
isMoving = Mathf.Abs (h) > 0.1 || Mathf.Abs (v) > 0.1;

// Target direction relative to the camera
var targetDirection = h * right + v * forward;

// Grounded controls
if (grounded)
{

	// Lock camera for short period when transitioning moving & standing still
	lockCameraTimer += Time.deltaTime;
	if (isMoving != wasMoving)
		lockCameraTimer = 0.0;

	// We store speed and direction seperately,
	// so that when the character stands still we still have a valid forward direction
	// moveDirection is always normalized, and we only update it if there is user input.
	if (targetDirection != Vector3.zero)
	{
		// If we are really slow, just snap to the target direction
		if (moveSpeed < walkSpeed * 0.9 && grounded)
		{
			moveDirection = targetDirection.normalized;
		}
		// Otherwise smoothly turn towards it
		else
		{
			moveDirection = Vector3.RotateTowards(moveDirection,
```

```
targetDirection, rotateSpeed * Mathf.Deg2Rad * Time.deltaTime, 1000);

                    moveDirection = moveDirection.normalized;
                }
            }

            // Smooth the speed based on the current target direction
            var curSmooth = speedSmoothing * Time.deltaTime;

            // Choose target speed
            //* We want to support analog input but make sure you cant walk faster diagonally than just forward or sideways
            var targetSpeed = Mathf.Min(targetDirection.magnitude, 1.0);

            _characterState = CharacterState.Idle;

            // Pick speed modifier
            if (Input.GetKey (KeyCode.LeftShift) | Input.GetKey (KeyCode.RightShift))
            {
                //transform.tag = "move" ;
                //target1.audio.Play();
                other.particleEmitter.emit = true;
                targetSpeed *= runSpeed;
                _characterState = CharacterState.Running;
            }
            else if (Time.time - trotAfterSeconds > walkTimeStart)
            {
                //target1.audio.Stop();

                other.particleEmitter.emit = false;
                targetSpeed *= trotSpeed;
                _characterState = CharacterState.Trotting;
            }
            else
            {
                targetSpeed *= walkSpeed;
                _characterState = CharacterState.Walking;
```

```
            }

            moveSpeed = Mathf.Lerp(moveSpeed, targetSpeed, curSmooth);

            // Reset walk time start when we slow down
            if (moveSpeed < walkSpeed * 0.3)
                walkTimeStart = Time.time;
        }
        // In air controls
        else
        {

            other.particleEmitter.emit = true;


            // Lock camera while in air
            if (jumping)
                lockCameraTimer = 0.0;

            if (isMoving)

                inAirVelocity += targetDirection.normalized * Time.deltaTime *
inAirControlAcceleration;


        }



}


function ApplyJumping ()
{
    // Prevent jumping too fast after each other
    if (lastJumpTime + jumpRepeatTime > Time.time)
        return;
```

```
    if (IsGrounded()) {
        // Jump
        // - Only when pressing the button down
        // - With a timeout so you can press the button slightly before landing

        if (canJump && Time.time < lastJumpButtonTime + jumpTimeout) {
            verticalSpeed = CalculateJumpVerticalSpeed (jumpHeight);
            SendMessage("DidJump",
SendMessageOptions.DontRequireReceiver);
        }
    }
}


function ApplyGravity ()
{
    if (isControllable)     // don't move player at all if not controllable.
    {
        // Apply gravity
        var jumpButton = Input.GetButton("Jump");


        // When we reach the apex of the jump we send out a message
        if (jumping && !jumpingReachedApex && verticalSpeed <= 0.0)
        {
            jumpingReachedApex = true;
            SendMessage("DidJumpReachApex",
SendMessageOptions.DontRequireReceiver);
        }

        if (IsGrounded ())
            verticalSpeed = 0.0;
        else
            verticalSpeed -= gravity * Time.deltaTime;
    }
}

function CalculateJumpVerticalSpeed (targetJumpHeight : float)
```

```
{
    // From the jump height and gravity we deduce the upwards speed
    // for the character to reach at the apex.
    return Mathf.Sqrt(2 * targetJumpHeight * gravity);
}

function DidJump ()
{
    other.particleEmitter.emit = true;
    jumping = true;
    jumpingReachedApex = false;
    lastJumpTime = Time.time;
    lastJumpStartHeight = transform.position.y;
    lastJumpButtonTime = -10;

    _characterState = CharacterState.Jumping;

}




function OnGUI() {




// Register the window. Notice the 3rd parameter
windowRect = Rect(10 , 10 , 130 , 200 );
windowRect = GUI.Window (0, windowRect, DoMyWindow, "Where To GO?");
}

// Make the contents of the window
function DoMyWindow (windowID : int) {
```

```
if (Elevator.gameObject.tag== "up" )
{
GUI.Label    (Rect (10,100,200,50), " Going Up...");

}

if (Elevator.gameObject.tag== "down" )
{
GUI.Label    (Rect (10,100,200,50), " Going Down...");

}

if (GUI.Button (Rect (20,20,90,20), "Elevator"))
{
transform.position = Vector3(-314, 51, 140);    //電梯位址

}
if (GUI.Button (Rect (20,40,90,20), "Office"))
{
transform.position = Vector3(-88, 51, 214); //系辦

}
if (GUI.Button (Rect (20,60,90,20), "Class338"))
{

transform.position = Vector3(-275, 51 ,194); //338
}

if (GUI.Button (Rect (20,80,90,20),"Labtory"))
{
transform.position = Vector3(-390, 51, 137); //實驗室
}

if (GUI.Button (Rect (20,160,90,20),"Reset"))
{
```

```
Application.LoadLevel(0);        //reset
}




//Application.LoadLevel(0);
}




    function OnEnable()
    {
        if(radarObject != null)
        {
            radarObject.SetActiveRecursively(true);
        }
}




function Update()
{
```

```
if (Input.GetButtonDown ("Jump"))
	{

//target1.audio.Pause();

		lastJumpButtonTime = Time.time;
	other.particleEmitter.emit = true;

	}

	UpdateSmoothedMovementDirection();

	// Apply gravity
	// - extra power jump modifies gravity
	// - controlledDescent mode modifies gravity
	ApplyGravity ();

	// Apply jumping logic
	ApplyJumping ();




	// Moving platform support
	if (activePlatform != null) {
		var newGlobalPlatformPoint =
activePlatform.TransformPoint(activeLocalPlatformPoint);
		var moveDistance = (newGlobalPlatformPoint -
activeGlobalPlatformPoint);
		transform.position = transform.position + moveDistance;
		lastPlatformVelocity = (newGlobalPlatformPoint -
activeGlobalPlatformPoint) / Time.deltaTime;
	} else {
		lastPlatformVelocity = Vector3.zero;
	}
```

```
    // Save lastPosition for velocity calculation.
    lastPosition = transform.position;




    // Calculate actual motion

    var movement = moveDirection * moveSpeed + Vector3 (0, verticalSpeed, 0) +
inAirVelocity;
    movement *= Time.deltaTime;

    // Move the controller
    var controller : CharacterController = GetComponent(CharacterController);
    collisionFlags = controller.Move(movement);

    // ANIMATION sector
    if(_animation) {
        if(_characterState == CharacterState.Jumping)
        {


            if(!jumpingReachedApex) {


                _animation[jumpPoseAnimation.name].speed                =
jumpAnimationSpeed;
                _animation[jumpPoseAnimation.name].wrapMode             =
WrapMode.ClampForever;
                _animation.CrossFade(jumpPoseAnimation.name);
            } else {
                _animation[jumpPoseAnimation.name].speed                =
-landAnimationSpeed;
                _animation[jumpPoseAnimation.name].wrapMode             =
WrapMode.ClampForever;
```

```
                _animation.CrossFade(jumpPoseAnimation.name);

            }
        }
        else
        {
            if(controller.velocity.sqrMagnitude < 0.1) {
                _animation.CrossFade(idleAnimation.name);

            }
            else
            {
                if(_characterState == CharacterState.Running) {
                    _animation[runAnimation.name].speed = Mathf.Clamp(controller.velocity.magnitude, 0.0, runMaxAnimationSpeed);
                    _animation.CrossFade(runAnimation.name);


                }
                else if(_characterState == CharacterState.Trotting) {

                    _animation[walkAnimation.name].speed = Mathf.Clamp(controller.velocity.magnitude, 0.0, trotMaxAnimationSpeed);
                    _animation.CrossFade(walkAnimation.name);

                    //target1.audio.Play();
                }
                else if(_characterState == CharacterState.Walking) {


                    _animation[walkAnimation.name].speed = Mathf.Clamp(controller.velocity.magnitude, 0.0, walkMaxAnimationSpeed);
                    _animation.CrossFade(walkAnimation.name);


                }

            }
```

```
            }
        }
        // ANIMATION sector

        /*
        if( Input.GetButton("Vertical"))

        {
        target1.audio.PlayOneShot(runaudio);
        }
        //else{
        //audio.Pause() ;
        //}
        */

/*    if(    (!Input.GetButton("Vertical"))    && (!Input.GetButton("Horizontal")) )
{


target1.audio.Play();

  }




*/


// Moving platforms support
        if (activePlatform != null) {
            activeGlobalPlatformPoint = transform.position;
            activeLocalPlatformPoint       =       activePlatform.InverseTransformPoint
(transform.position);
        }
```

```
        // Set rotation to the move direction
    if (IsGrounded())
        {

            transform.rotation = Quaternion.LookRotation(moveDirection);

        }
        else
        {
            var xzMove = movement;
            xzMove.y = 0;
            if (xzMove.sqrMagnitude > 0.001)
            {
                transform.rotation = Quaternion.LookRotation(xzMove);
            }
        }

        // We are in jump mode but just became grounded
        if (IsGrounded())
        {
            lastGroundedTime = Time.time;
            inAirVelocity = Vector3.zero;

            if (jumping)
            {
                jumping = false;
                SendMessage("DidLand",
    SendMessageOptions.DontRequireReceiver);

            other.particleEmitter.emit = true;
            }
        }
    }

    function OnControllerColliderHit (hit : ControllerColliderHit )
    {
        //Debug.Log("t");
```

```
//     Debug.DrawRay(hit.point, hit.normal);
       if (hit.moveDirection.y > 0.01)
              return;

      // Make sure we are really standing on a straight platform
       // Not on the underside of one and not falling down from it either!
       if (hit.moveDirection.y < -0.9 && hit.normal.y > 0.9) {
              activePlatform = hit.collider.transform;



       }


}

function GetSpeed () {
       return moveSpeed;
}

function IsJumping () {
       return jumping;
}

function IsGrounded () {
       return (collisionFlags & CollisionFlags.CollidedBelow) != 0;
}

function GetDirection () {
       return moveDirection;
}

function IsMovingBackwards () {
       return movingBack;
}

function GetLockCameraTimer ()
```

```
{
    return lockCameraTimer;
}

function IsMoving ()  : boolean
{

    return Mathf.Abs(Input.GetAxisRaw("Vertical")) +
Mathf.Abs(Input.GetAxisRaw("Horizontal")) > 0.5;
}

function HasJumpReachedApex ()
{
    return jumpingReachedApex;
}

function IsGroundedWithTimeout ()
{
    return lastGroundedTime + groundedTimeout > Time.time;
}

function Reset ()
{
    gameObject.tag = "Player";
}
```

附件二．　電梯:升降台部份程式碼:elevator

```
var targetA : GameObject;
var targetB : GameObject;
var attach : GameObject;
var position : Vector3;
var speed : Vector3;
var onelevator = false;
var elevator : AudioClip ;
var elevatorring : AudioClip ;




function Update(){


if( targetA.gameObject.tag == "Finish" )
   {

            onelevator = false ;
       targetA.gameObject.tag = "Untagged";
   }

if(targetA.gameObject.tag == "up")
{


    onelevator = true ;
    speed = Vector3(0,3,0);

  }



if( targetA.gameObject.tag == "down" )
  {
```

```
    onelevator = true ;
    speed = Vector3(0,-3,0);
  }


  }



function OnCollisionEnter(collision : Collision)
{
onelevator = false ;


targetA.gameObject.tag = "Finish";


Debug.Log("2");

}



function FixedUpdate () {

if (onelevator)
rigidbody.MovePosition(rigidbody.position + speed
*Time.deltaTime);

}
```

附件三．　電梯:用來控制電梯停住之程式碼: trigger up down

```
var Elevator : GameObject;
var Door1:GameObject;
var Door2:GameObject;

function OnCollisionEnter(collision : Collision) {


Elevator.gameObject.tag = "Finish";

}

function OnTriggerEnter (other : Collider) {

Elevator.gameObject.tag = "Finish";

}
```

附件四 .　電梯 : 按鈕控制(往下)之程式碼:Text buttons Down

```
var elevatorring : AudioClip ;
var Elevator : GameObject ;
var door1 : GameObject;
var door2 : GameObject;
var speed :Vector3;

function OnMouseEnter()
{

renderer.material.color = Color.green ;

}

function OnMouseExit ()
{

renderer.material.color = Color.yellow;

}

function OnMouseUp()
{
if( Elevator.transform.position.y > 66.2)
{

audio.PlayOneShot(elevatorring);

Elevator.gameObject.tag =  "down" ;

door1.gameObject.tag = "up";

door2.gameObject.tag = "up";
}
```

```
}

function FixedUpdate () {

if (Elevator.gameObject.tag == "up")
{
speed = Vector3(0,3,0);

rigidbody.MovePosition(rigidbody.position + speed
*Time.deltaTime);
}




if (Elevator.gameObject.tag == "down")
{
speed = Vector3(0,-3,0);

rigidbody.MovePosition(rigidbody.position + speed
*Time.deltaTime);
}


}
```

附件五．　視窗功能之程式碼:Window1

```
var doWindow1 : boolean = false;
var stringToEdit : String = "Hello World\nI've got 2 lines...";
var texture : Texture2D;
var con : GameObject ;


function OnMouseEnter()
{

con.renderer.enabled = true;

}

function OnMouseExit ()
{
con.renderer.enabled = false;

}

function OnMouseUp()
{

doWindow1  = true;

}

function DoWindow1(windowID : int) {


}
function OnGUI () {

if(doWindow1)
{
```

```
GUI.Window (1, Rect (400,200,300,150), DoWindow1, "Guide : ");


if( GUI.Button (Rect (620,350, 80,20), "Close!"))
   {
doWindow1 = false;
   }

GUI.color = Color.cyan ;
GUI.Label   (Rect (400,220,300,150), " Here is the office of Professor Chung-Chi Lin\n The expertise of Professor is\n  Image processing\n Interface technology\n Micro-computer systems\n VLSI ");


}

}
```

附件六．　拖曳剛體(門)之程式碼:DragRigidbody

```
var spring = 50.0;
var damper = 5.0;
var drag = 10.0;
var angularDrag = 5.0;
var distance = 0.2;
var attachToCenterOfMass = false;

private var springJoint : SpringJoint;

function Update ()
{
    if (!Input.GetMouseButtonDown (0))
        return;

    var mainCamera = FindCamera();

    var hit : RaycastHit;
    if
(!Physics.Raycast(mainCamera.ScreenPointToRay(Input.mousePositio
n),  hit, 100))
        return;

    if (!hit.rigidbody || hit.rigidbody.isKinematic)
        return;

    if (!springJoint)
    {
        var go = new GameObject("Rigidbody dragger");
        body = go.AddComponent ("Rigidbody");
        springJoint = go.AddComponent ("SpringJoint");
        body.isKinematic = true;
    }

    springJoint.transform.position = hit.point;
    if (attachToCenterOfMass)
    {
```

```
        var anchor =
transform.TransformDirection(hit.rigidbody.centerOfMass) +
hit.rigidbody.transform.position;
        anchor =
springJoint.transform.InverseTransformPoint(anchor);
        springJoint.anchor = anchor;
    }
    else
    {
        springJoint.anchor = Vector3.zero;
    }

    springJoint.spring = spring;
    springJoint.damper = damper;
    springJoint.maxDistance = distance;
    springJoint.connectedBody = hit.rigidbody;

    StartCoroutine ("DragObject", hit.distance);
}

function DragObject (distance : float)
{
    var oldDrag = springJoint.connectedBody.drag;
    var oldAngularDrag = springJoint.connectedBody.angularDrag;
    springJoint.connectedBody.drag = drag;
    springJoint.connectedBody.angularDrag = angularDrag;
    var mainCamera = FindCamera();
    while (Input.GetMouseButton (0))
    {
        var ray = mainCamera.ScreenPointToRay
(Input.mousePosition);
        springJoint.transform.position = ray.GetPoint(distance);
        yield;
    }
    if (springJoint.connectedBody)
    {
        springJoint.connectedBody.drag = oldDrag;
        springJoint.connectedBody.angularDrag = oldAngularDrag;
```

```
        springJoint.connectedBody = null;
    }
}

function FindCamera ()
{
    if (camera)
        return camera;
    else
        return Camera.main;
```

附件七.　瞬間移動按鈕之程式碼:Domywindow

```
function OnGUI() {


// Register the window. Notice the 3rd parameter
windowRect = Rect(10 , 10 , 130 , 200 );
windowRect = GUI.Window (0, windowRect, DoMyWindow, "Where To GO?");
}

// Make the contents of the window
function DoMyWindow (windowID : int) {




if (Elevator.gameObject.tag== "up" )
{
GUI.Label  (Rect (10,100,200,50), " Going Up...");

}

if (Elevator.gameObject.tag== "down" )
{
GUI.Label  (Rect (10,100,200,50), " Going Down...");

}

if (GUI.Button (Rect (20,20,90,20), "Elevator"))
{
transform.position = Vector3(-314, 51, 140);  //電梯位址

}
if (GUI.Button (Rect (20,40,90,20), "Office"))
{
transform.position = Vector3(-88, 51, 214); //系辦

}
```

```
if (GUI.Button (Rect (20,60,90,20), "Class338"))
{

transform.position = Vector3(-275, 51 ,194); //338
}

if (GUI.Button (Rect (20,80,90,20),"Labtory"))
{
transform.position = Vector3(-390, 51, 137); //實驗室
}

if (GUI.Button (Rect (20,160,90,20),"Reset"))
{
Application.LoadLevel(0);    //reset
}




//Application.LoadLevel(0);
```

附件八．　影子之程式碼: BlobShadowController

```
using UnityEngine;
using System.Collections;

public class BlobShadowController : MonoBehaviour
{
    void Update()
    {
        transform.position = transform.parent.position + Vector3.up
* 8.246965f;
        transform.rotation = Quaternion.LookRotation(-Vector3.up,
transform.parent.forward);
    }
}
```