# 臺灣二ＯＯ六年國際科學展覽會

科　　別：電腦科學

作 品 名 稱：Vison-把台北 101 玩弄於電腦之中

得 獎 獎 項：第二名
英特爾電腦科學獎 第一名

學校 / 作者：國立臺灣師範大學附屬高級中學　俞冠廷
國立臺灣師範大學附屬高級中學　黃上嘉

# 作者介紹


我是俞冠廷，自從國中開始接觸 VB 便開始對程式設計頗有興趣。高中參加資訊能力競賽，以及科展競賽，讓我累積不少經驗，也有不少感觸。在這個科展主題上的研究讓我發現，要真正模擬自然界的物體並非想像中的那麼容易，因爲人並不是神。明確一點來說就是，世界是無限的，而電腦是有限的。即使我們達到一個境界使得模擬出來的影像和用數位相機拍的照片，每個像素都吻合，但是模擬出來的仍然無法取代真正的物體。曾經聽說過，在某部 3D 動畫電影中，砸下了重金就只是爲女主角製做一個逼真的「頭髮」，然而「她」還是騙不了你的眼睛。

「模擬」簡單來說就是「模仿」，引申出來就是「學習」。人無法達到神的境界，然而這使得每個人都有繼續進步的方向。因此在研究中雖然遇到許多瓶頸，但我們仍有自信可以繼續突破。目標很明確，便是以有限來製造無限。

我叫黃上嘉，目前是師大附中數理資優班高三生，從小喜歡研究新奇或具有推理性的事物，從高一進入了附中，由於社團相同，讓我們一起做了 Vision 的研究，雖然遇到很多瓶頸，但經過克服以後，在許多科展比賽裡都得到不錯的成績，如今高三了，一做就是三年，相信付出整個高中時段的努力，能在未來留下美好的回憶。

# 目錄

# 中文摘要


## 創意發想

在學習三角函數的三角測量應用時，由於立體感並非十分容易在平面中呈現，使得解題過程並相當困難。我們希望能透過程式，實際模擬出所看到的樣子，將有利於解決這方面的問題。學習美術者也需要了解一點透視的立體概念，皆可以透過程式來模擬。

## 作品特色

我們的精神主要在於以高中的數學及物理爲基礎，來研究其中的方法。除了研究 3D 繪圖之基本原理，並著重於如何以程式實作，以達到高繪圖效能。

## 預期效果

1. 讓電腦繪出有立體感(近大遠小)的圖形。
2. 可以由不同位置及角度觀察物體。
3. 讓立體影像具有光及影的效果。

"想像您坐了一部直升機從 1 樓向上到達頂端，觀看 101 大樓有何不同的景象?!"

# Abstract


## Motive

In learning the technique of triangulation, it is hard to show 3D coordinates on 2D graphics so that this kind of math problems is difficult to solve. We hope that we can simulate the 3D surroundings by programming to provide references in dealing the problems. In addition, painting learners also need the simulation to realize the concept of one-point perspective.

## Feature

1. We do all the research based on mathematics and physics techniques learned in high school.
2. We not only figure out the method to draw 3D pictures but put some emphasis on how to use programming to run the method.

## Objective

1. Let the computer draw 3D pictures, that is, the object looks big when near and small when far.
2. Making it possible to observe the object from different positions and angles.
3. Making the 3D pictures with lighting and shading effect.

"Imagine how the sight would change while you are taking a ride on a helicopter from the ground to the top of Taipei 101."

# 壹　研究動機

電玩已在資訊產業蓬勃發展，廣泛銷售的電玩，它虛擬實境的 3D 繪圖令我們感到十分好奇。

要從電腦的高速運算產生如此逼真的畫面，需要經過相當多的圖形處理，例如：產生透視圖、光影的效果...等。以理論方面來看，需要有不少的物理的概念，例如：運用三角測量的概念，推導出如何使用電腦產生透視圖的方法。

以程式寫作的角度來看，在那麼多重的處理過程中，也需要有相當良好的程式架構。即使是單獨其中一項的繪圖效果，也許要很多小部分嚴謹地相互配合。這些是程式開發的一大挑戰。

另外，坊間所賣的 3D 繪圖軟體大多索價不斐。雖然國內 3D 藝術繪圖高手不少，但是能了解其中的繪圖理論者並不多，能真正開發繪圖軟體的又是極少數。

在一次的專題研究課程中，我們參加了 Discreet 3DSMAX 7 的發表會。在看到目前世界上相當專業的 3D 繪圖軟體後，雖然我們的研究技術或開發的軟體比不上專業的商品，但是我們的作品和一般的繪圖軟體最大的差別是：我們注重模擬實際物體，而有實驗的加入，而一般軟體純粹著重於電腦立體繪圖。

我們的精神主要在於以高中的數學及物理爲基礎，來研究其中的繪圖方法，並親自實作。在數學部份，我們引用高二上第一章的向量處理、第二章立體空間概念，以及高三數學甲的座標系旋轉技巧。物理部分我們運用三角測量的方式完成其中的繪圖原理。

# 貳　研究目的

研究如何讓電腦畫出有立體感圖形的方法，進而模擬生活中人眼看到的景象。

## (一)研究項目

### 1　基本 3D 繪圖方法

研究如何呈現有立體感的圖形，並與現實狀況結合

### 2　繪圖物件之繪圖方法

1. 點
2. 線
3. 三角面
4. 圓面

### 3　特殊效果之處理

1. 由不同角度觀測的效果
2. 光與影的效果

### 4　以建築實體驗證繪圖理論

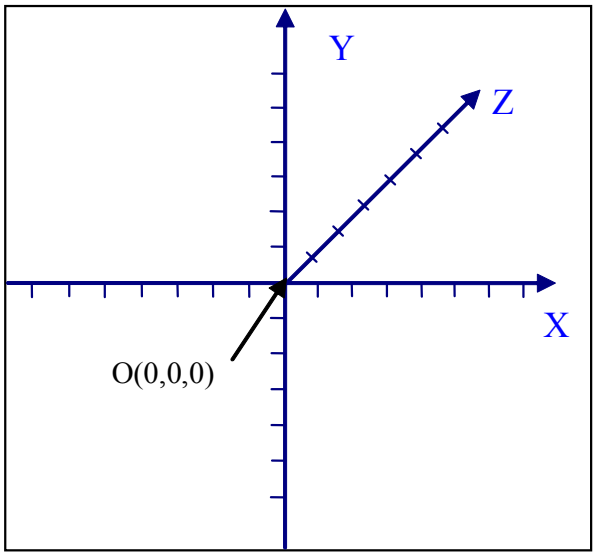
1. 建立模型
2. 以連續影格產生動態效果

# 參　研究方法

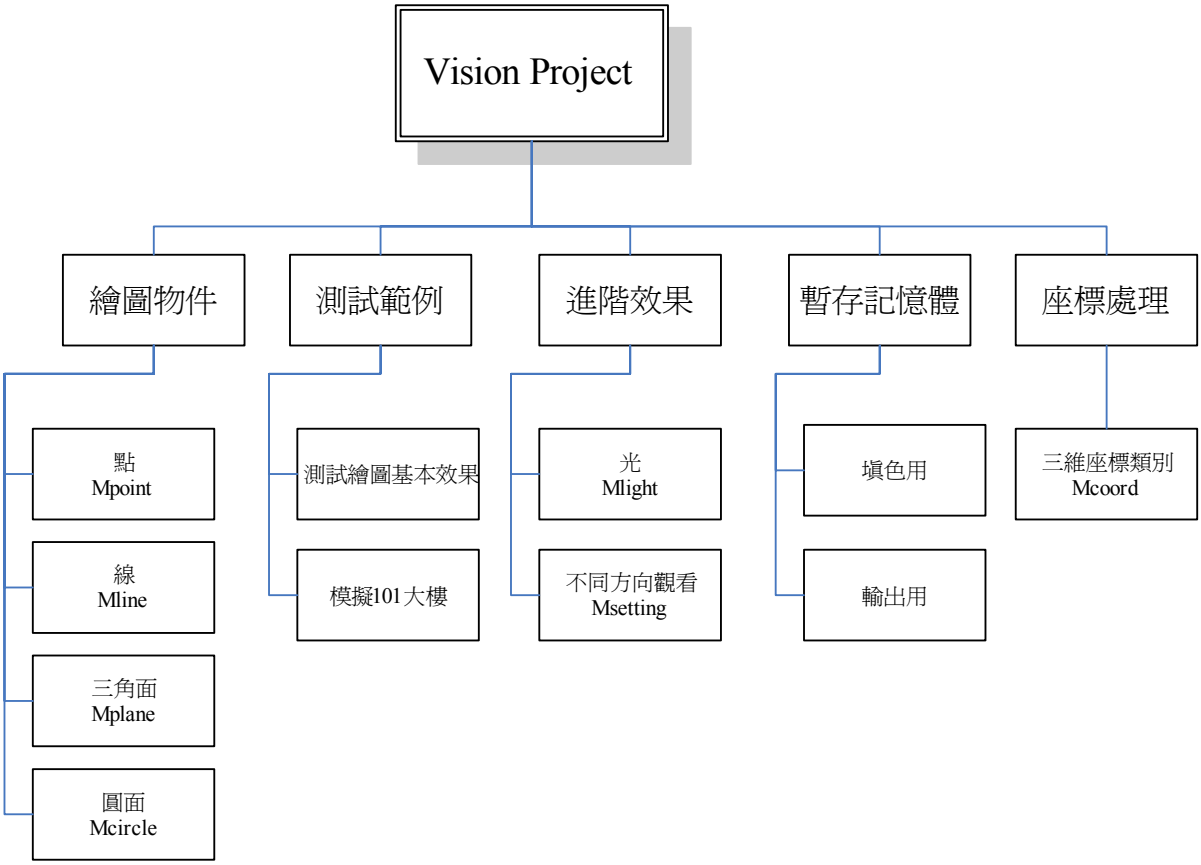
## (一)前言

使用 Borland C++ Builder 6，配合物件導向方式紀錄立體物件的節點及屬性，經過公式處理後畫出有立體感（近大遠小）的影像。

下圖是我們所訂定的座標軸，即是在熟悉的 X-Y 平面座標中加入 Z 軸代表遠近，遠離觀察者時 Z 值增大。



## (二)系統架構圖




圖片 參-1　程式規劃圖

圖中加入的英文字爲在程式中所建立的類別名稱，本程式中所有自訂類別名稱皆冠上M，以和開發系統本身作區別。

# 肆　研究設備

## (一)硬體

桌上型電腦 - AMD AthlonXP 1.7Ghz、RAM512MB、顯示卡 128MB

## (二)軟體

Microsoft Windows XP 作業系統
Borland C++ Builder 6 軟體開發工具
OpenGL 函式(僅用於呈現繪圖結果)

# 伍　研究過程

## (一)如何使圖形遠小近大？

### 1　實驗一 － 測量距離與物體縮小比例之關係

#### (1) 過程

圖片 伍-1　找一張寬 10 公分的紙。

圖片 伍-2　將紙放在卷尺的起點，並固定好。

圖片 伍-3　將直尺固定在眼前 20 公分，使眼睛、尺的刻度和紙成一直線。再調整直尺和紙的距離，直到從眼睛透過尺看到紙只有實際大小的十分之一。

#### (2) 結果

圖片 伍-4　由眼前看到的狀況：測出在 240cm 處，物體看起來縮小爲 1/10（1 公分）。

## 2　實驗二 － 測量人眼視線範圍之大小

測量人眼直視前方 20 公分上下各能看到多大的範圍。以黑板爲實驗工具，眼睛直視黑板上畫 **X** 處，然後用粉筆自此處向上移動至看不見爲止，則此時 **X** 到粉筆的距離就是人眼向上所能及的範圍。此數值再乘以 2 就是我們所要的垂直範圍。

圖片 伍-5　測量垂直之可視範圍。

圖片 伍-6　測量水平之可視範圍。

### 結果:

測出人眼在 20 公分前可看到的垂直範圍是 36 公分，水平範圍則是 60 公分。

# 3 分析

計算人眼的垂直視角 H 以及水平視角 W

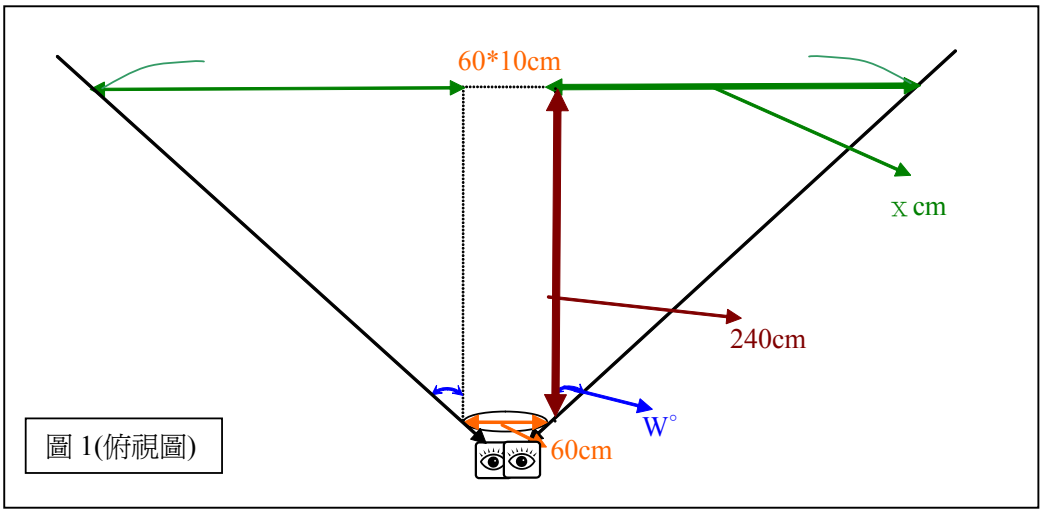



圖 1(俯視圖)

由實驗一得到物體在 240 公分外，物體看起來會縮小爲 1/10，又由實驗二測出眼睛的水平視線範圍在 20 公分前爲 60 公分。根據這些條件畫出了圖 1，現在要求的就是水平視角 W。

方程式：

$$\frac{60}{2x+60}=\frac{1}{10}\Rightarrow x=270$$

$$\tan W = 270 \div 240 = 1.125$$

$$W = \tan^{-1} 1.125 = 48.366460$$

同樣的也能算出垂直的視角 H

方程式：

$$\frac{36}{2x+36}=\frac{1}{10}\Rightarrow x=162$$

$$\tan H = 162 \div 240 = 0.675$$

$$H = \tan^{-1} 0.675 = 34.01935$$

## (二)畫點

### 1　作圖分析

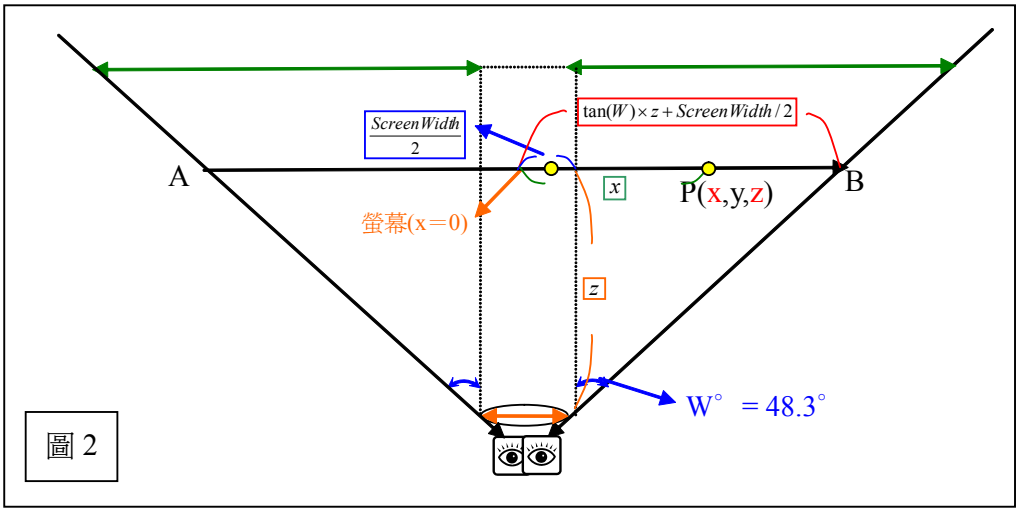


圖 2

假設一點 P 在視角內座標爲(x,y,z)，把 AB 當作螢幕的 X 座標軸

$$\text{ScreenX}=\frac{x}{\tan(W)\times z+ScreenWidth/2}\times\frac{ScreenWidth}{2}$$

同樣地

$$\text{ScreenY}=\frac{y}{\tan(H)\times z+ScreenHeight/2}\times\frac{ScreenHeight}{2}$$

對於 ScreenWidth 和 ScreenHeight 我們取寬爲 800pixel，而高依人眼可觀察的比例(60 公分:36 公分)，所以取高爲 480pixel。也就是會輸出 480*800 的矩形。

不過要注意的一點：z 不可以爲負值。z 若爲負值代表在觀察者後方，將不會看到，所以也不會被顯示出來。

### 2　小結

1. 根據推出的換算公式，本程式的像素長度與實際長度比例是 $7.5E^{-4}$ 公尺/pixel。
2. 由導出的公式就能使得遠方的物體實際座標放到適當的螢幕座標。而我們發現越遠的點會越接近螢幕的中心點(0,0)也就是一點透視的概念。

## (三)畫線

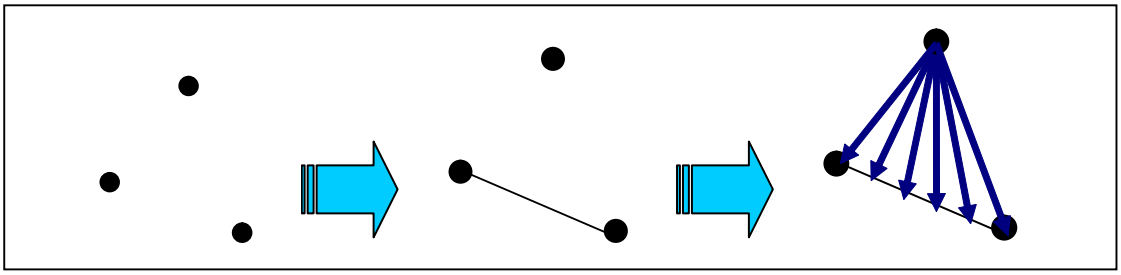
給程式任意兩個三維座標的點，先轉換成直線的參數式，再利用迴圈將 t 代入 0~1 之間的數值，如此就可把線上的所有點描繪出來，形成一線段。

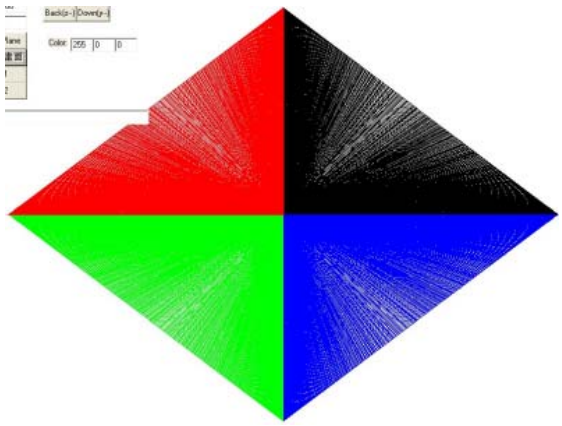
$$\begin{cases}P_0(x_0,y_0,z_0)\\P_1(x_1,y_1,z_1)\end{cases}\Rightarrow\begin{cases}x=x_0+\Delta x\times t\\y=y_0+\Delta y\times t\\z=z_0+\Delta z\times t\end{cases}\quad 0\le t\le 1$$

## (四)畫面 - 方法討論

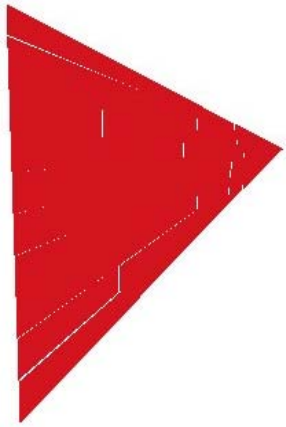
### 方法1　以自三角形頂點連線

給三個點，連接其中兩點，形成一線段，再從另一點向此線段連線，形成一個面。





圖片 伍-7

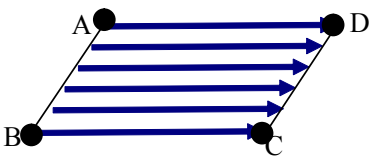


圖片 伍-8

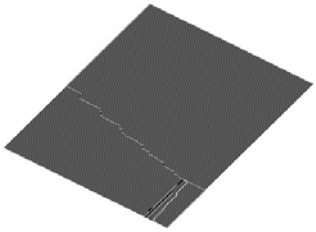
上圖爲以此方法所畫出的三角面，結果發現，因爲實際座標(浮點數)的運算結果轉換爲螢幕座標(整數)可能會出現空隙，無法保證能夠填滿。

### 方法2　平行四邊形的填色法

爲了讓連線色過程盡量是平行的，可能較不易有空隙，所以有了這個想法。
過程是：輸入三個點，先找出平行四邊行的第四個點，使用迴圈，從 AB 線段不斷連線至 DC 而畫出圖形。



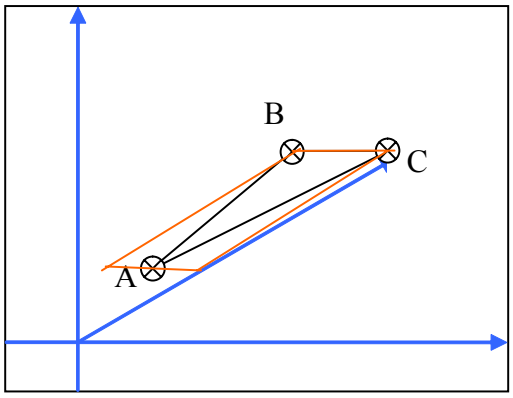

過程；輸入 A,B,C 三點先找出第四點 D，然後從 AB 上的點依序向 CD 連線。

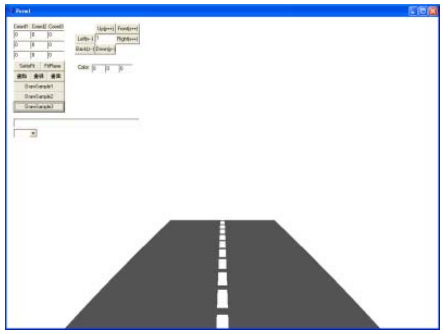


圖片 伍-9　同樣產生無法避免的空隙。

## 方法3　最大最小值

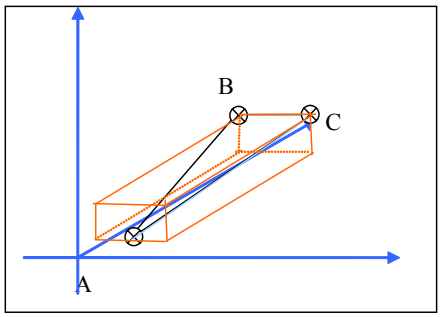
輸入三個點，從這三個點中分別取 X,Y,Z 的最大最小值，將矩形著色。



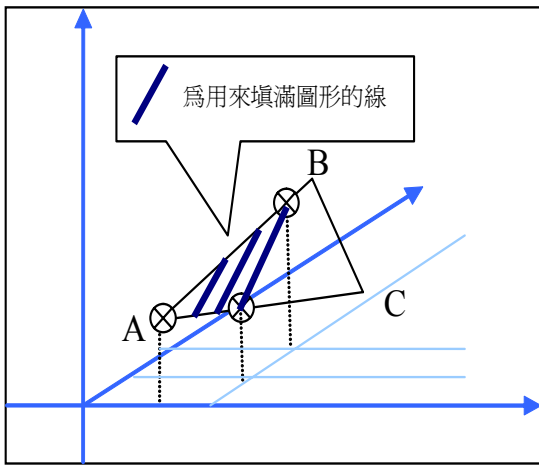
圖片 伍-10　以此方法畫的長方形，結果發現不會有空隙，但畫出來的長方形的邊是必須和 X、Y 或 Z 軸平行。

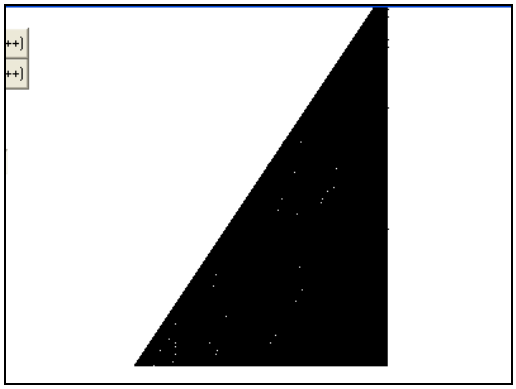
想法不完整處：




當三點的三個軸的最大最小值皆不同時，會繪出長方體內的所有點，如此一來，十分浪費時間，也不是我們預期的。

## 方法4　以平行 yz 平面或 xz 平面，將三角形切成數條線段，依序繪出。



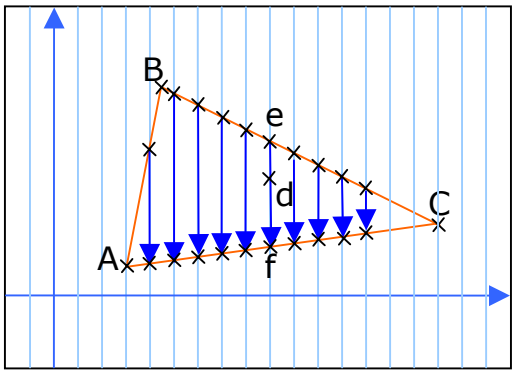
圖片 伍-11　由於每次畫的線條與螢幕還是不平行，仍然有無法避免小空隙。

## 方法5　終極填色法

概念：

先將三維空間的 3 個座標用「線」（Mline）連起來，形成一個三角形的外框。再透過函式轉成 2D 的三角形，紀錄在暫存陣列中。接著不斷地畫垂直線來填滿這個三角形。

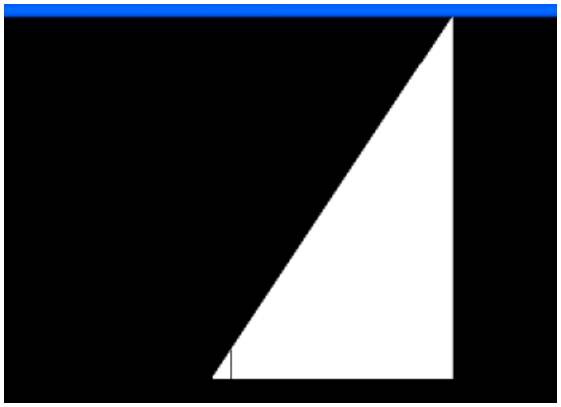
填色步驟：



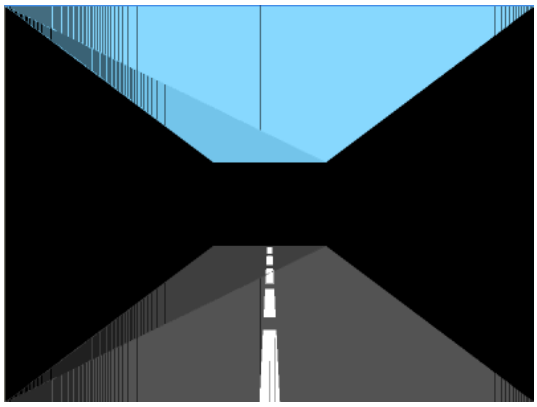

A, B, C 三點分別爲三角面之頂點。

把線段 AB, AC, BC 連接後，在二維的填色暫存記憶體[1]裡產生了一個三角形。在填滿過程中，從暫存記憶體裡的 painted[0~800][][2] 一欄一欄做處理。現在假設同欄內有 e 和 f 兩點都被線畫過（painted 屬性爲 true），就將中間的空隙填滿。並用內插法算出連起來的點的實際 3 維座標紀錄在暫存陣列中。假設 d 是線段 ef 上的一點，如此一來 d 的座標：$d=(e-f)\times\frac{\overline{ed}}{ef}+e$

測試圖樣：



圖片 伍-12



圖片 伍-13

[1] 在(七)暫存記憶體之運用有詳細說明。

[2] painted 二維陣列爲布林型態，用來記錄在填色暫存記憶體中該像素是否已被畫過。

修正：
以此方法畫的三角形，可能會因爲線段的不完全而造成有空隙。可以透過調整畫線的密度來避免線有不連續處。目前是以調整畫線密度來修正，雖然線畫得更密，但因爲是畫線之時間複雜度爲 O(n)，對於整體繪圖速度影響不大。

修正後測試圖樣：

圖片 伍-14

圖片 伍-15　修正後線段較爲緊密，不存在垂直之空隙

目前就以第 5 個方法做爲本程式的畫面方法，這是目前研究出最精確的方法。

## (五)繪製特例圓

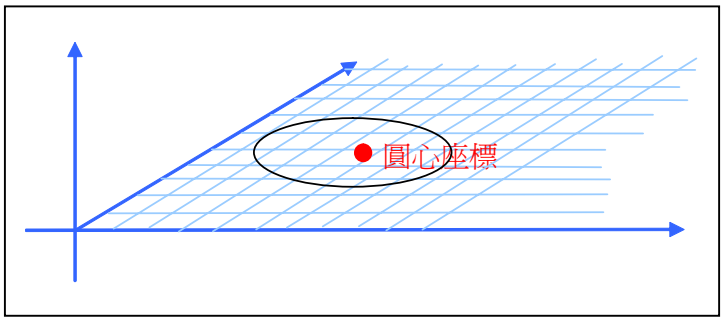
畫「圓」的方法：

輸入一個圓心座標，半徑，及選擇此圓形是對 x-y，y-z，z-x 平面其中的哪一個面平行。為了方便判定，分別定義平面編號：0:(x-y)，1:(y-z)，2:(z-x)。 有了平面編號之後便可以在 2 維座標使用「圓的參數式」做描點的動作來畫圓。

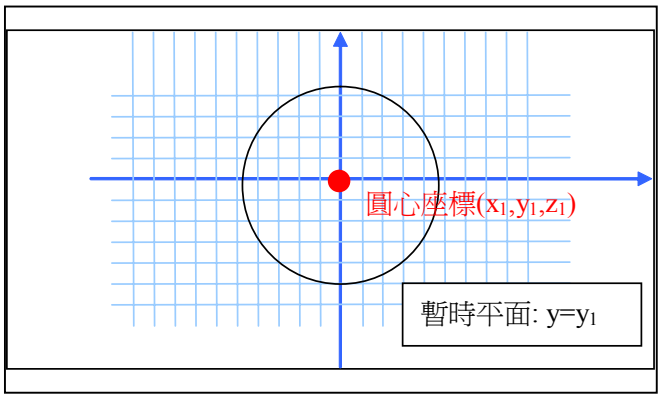
x-y 座標上圓的參數式： 令 h，k 為圓心座標，r 為半徑，$\theta$ 為參數

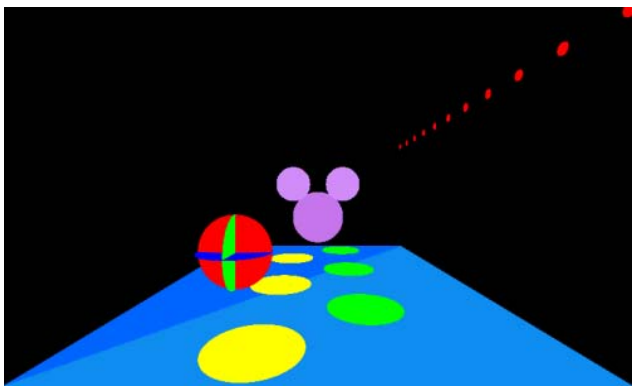
$$\begin{cases} x = h + r\cos\theta \\ y = k + r\sin\theta \end{cases}$$

如下圖是一個位在 $y=y_1$ 平面上的一個圓，也就是編號為狀況 1(平行於 y=0 平面)的圓。




放在二維座標的圓可以使用參數式描出所有圓上的點產生邊框，再以終極填色法完成填色工作。



圖片 伍-16 圓面可以達到遠近的效果，並且可以呈現三種特定方向的圓。左側一紅綠藍圓面交叉圖形中；紅色屬於 x-y 類，藍色屬於 y-z 類，綠色屬於 z-x 類。

# (六)加速 Render

經過 Vision 程式畫出的圖形，需要有快速的方式顯現於螢幕上。

## 方法1　使用記憶體處理

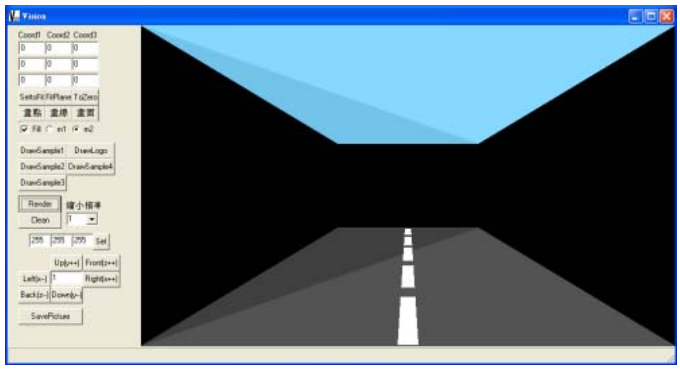
Render 最原始所採用的方式是直接將每個點輸出至 Image，但經過幾次之後就發現執行時間會隨著點的數目增加，當要一次畫很多點可是又有大量重複時，會耗費極多時間。因此想到將這些點都先使用顯示暫存記憶體處理過，去除不合規則的點，此時畫到 Image 時只需要固定的時間。

所謂不合乎規則的點有兩種：

1. 該點實際座標 z 值比已存在的點還大，也就是距離觀察者較遠，將會被已存在的點擋住。
2. 該點實際座標 z 值小於零，也就是在觀察者的後方，將不會被看到。

## 方法2　使用 GDI 函式

但是我們仍然不滿意這個固定的時間-約 3 秒鐘。我們要做的只是把固定座標的點塗上我們要的顏色，因此是非常適合使用 GDI 函數直接作繪圖動作。GDI（Graphics Device Interface），它是 Windows 的子系統，負責在視訊顯示。因此 GDI 對所有 Windows 的圖素操作是最直接的，所以我們考慮使用它。而我們做出來的結果速度確實增快許多。Render 時，只需花費 1 秒鐘。



輸出效果不錯，效率較高。

但是不同問題出現：


圖片 伍-17

1. 只能畫在 Form 上，所以儲存圖片時不能使用 Image 內建的方法，必須要經過別的方式轉換。
2. 當 Form 移動到螢幕外，圖片會消失，無法有 AutoRedraw 的功能。
3. 無法由 Image 的 MouseMove 事件來追蹤滑鼠的位置，來提供使用者介面左下角的座標資訊。

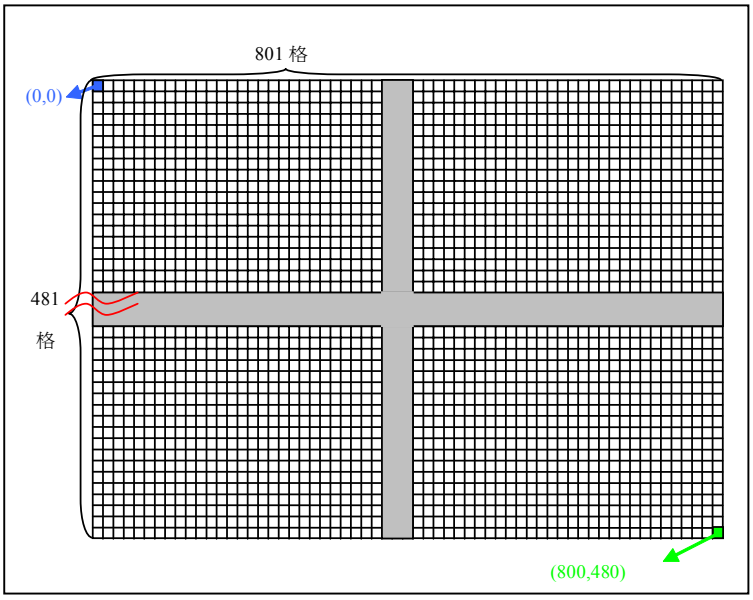
## 方法3　使用 OpenGL

OpenGL 具有直接操作硬體的特性，使得繪圖速度相當快速，將影像存於陣列後，透過 glDrawPixel()函式將可以迅速地顯示影像，由於它的繪圖速度極快，我們可以不斷的更新畫面，而不必擔心影像被抹除。所以目前採用 OpenGL 來擔任 Render 工作，而取代 GDI 的使用。

# (七)暫存記憶體之運用

暫存記憶體在程式中擔任相當重要之角色，運用在兩部份一是在輸出前紀錄影像，以下稱作輸出暫存記憶體，一是在畫面時做填色處理用，稱作填色暫存記憶體。
它是用來暫時存放由 3 維座標轉爲 2 維座標（螢幕顯示的狀況）。大小爲 481*801 的圖形，以便做後續處理。其結構使用二維的陣列儲存每一元素的顏色（Mcolor color[][]）及實際的三維座標位置（Mcoord coord[][]）、以及該點是否有畫過(bool painted[][])。

## 1 輸出暫存記憶體

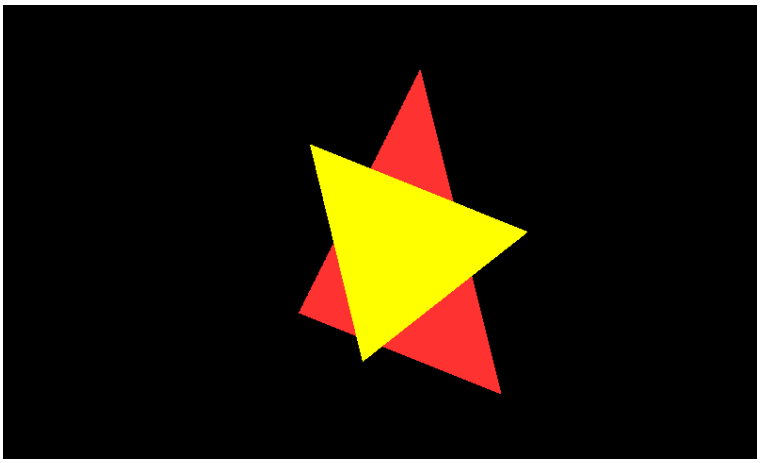



**優點**

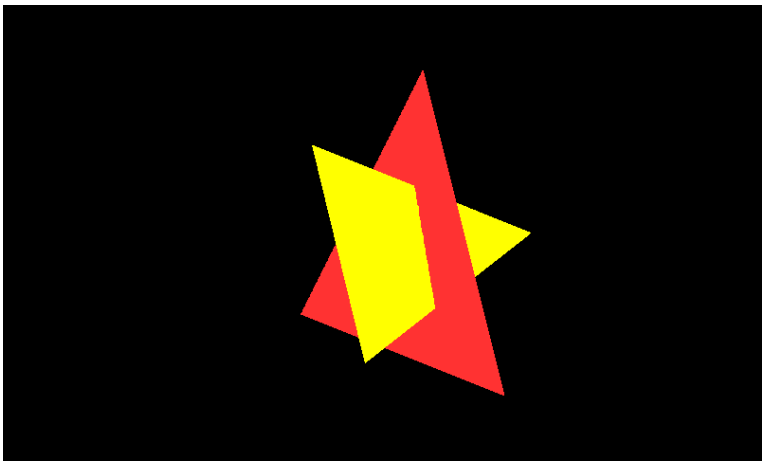
1. 速度：
   我們發現，若是把圖形直接畫在 Image 物件上，每畫一點的速度比在記憶體中的陣列中改變一點慢很多很多。其演算法時間複雜度 O(n_point)，當繪製的面越多，時間將成長十分快速。而先把圖形存入暫存記憶體，當要輸出到 Image 物件或以其他方式輸出時，時間複雜度是常數時間 O(1)。
2. 點重複時的處理
   之前每次畫一個圖形，最後畫的會把之前畫的蓋過，而我們要的不是依畫的順序取決看或看不見，而是以實際座標做判斷。當兩個 3 維座標的點轉成 2 維時是在同一點，此時比較原本已畫到暫存陣列的點的 Z 軸座標值，以及可能要畫上去的點的 Z 軸座標值。若後者較小，代表離觀察者較近而不會被前者擋住，將更改在暫存陣列的點爲新畫上的點。反之，新加入的點會被舊有的點擋住，就不需更新。

**缺點**

1. 消耗記憶體：
   一次宣告 801*481 的陣列，單位元素約 16byte，總共須佔用 481*801*16 ≒ 6.1(MB)



先畫紅色面再畫黃色面，黃色面會將紅色面覆蓋。
而實際上以三維座標來看黃色面應穿過紅色面。
紅色面頂點座標(50,200,60) (150,-200,60) (-100,-100,60)
黃色面頂點座標(-20,-150,35) (-80,100,30) (200,0,100)



以輸出暫存記憶體改善後。

## 2 填色暫存記憶體

結構與上述輸出暫存記憶體類似，但增加內插法函數，用於對面做填色時，計算實際三維座標。
其詳細過程請參考伍-(四)-方法 5 終極填色法

# (八)如何產生不同觀測位置及方向的效果

## 1 構想

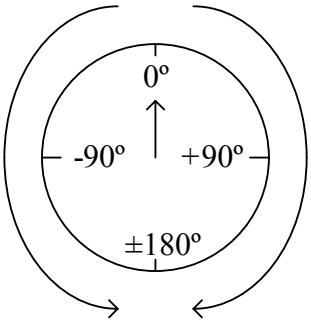
爲了想要設計出能夠讓使用者從不同角度、不同位置觀測的方法。由一個簡單的想法：被觀察者的位移和觀察者觀察方向的位移是相對的，所以藉由相對的移動物體來達到所要的效果。以下面的例子來說明。假設現在坐在電腦桌前，正前方面對著電腦，當人站起來，代表觀察位置發生改變，會發覺到電腦對觀察者而言是向下移動。而當頭順時針轉動時，代表觀察角度發生改變，此時會覺得電腦好像以逆時針方式繞著頭轉動(道理如同古代人會認爲太陽繞地球轉)。

由於位置的改變只需透過平移做調整，方法在此不多做解釋。

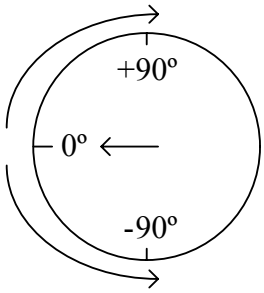
## 2 旋轉的方法

### (1) 定義觀測方向的角度

類似於地球的經緯度，我們的角度包括兩個數值(hAngle, vAngle)，hAngle 類似經度，代表水平方向的旋轉(如同頭逆時針或順時針轉動)，範圍在-180 到 180 之間。而 vAngle 類似緯度，代表垂直方向的旋轉(如同俯視或仰視)，範圍在-90 到 90 之間。




水平轉動視角




垂直轉動視角

### (2) 公式處理

先做水平旋轉，也就是將 X-Z 座標軸旋轉一個角度(Θ=-hAngle)

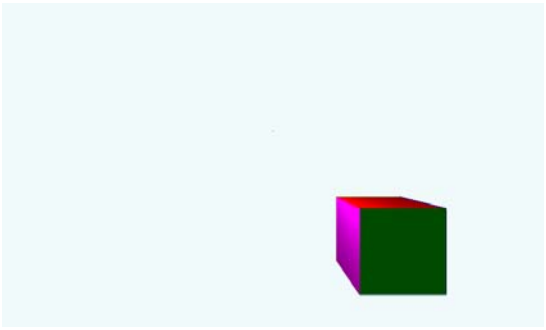
$$\begin{bmatrix} x' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\Theta & \sin\Theta \\ -\sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix}$$

再做垂直旋轉，也就是將 Z-Y 座標軸旋轉一個角度(φ= vAngle)

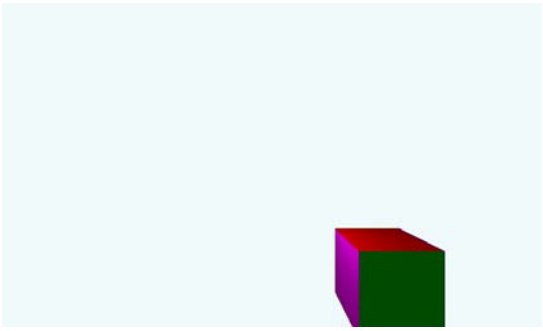
$$\begin{bmatrix} z'' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} z' \\ x \end{bmatrix}$$

## (3) 可達到的效果

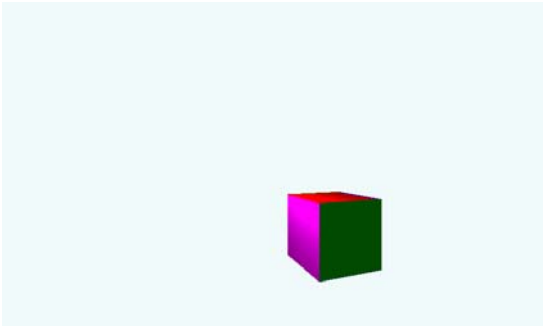
透過相對移動及相對旋轉，可以達到從不同位置以及不同角度的觀看效果。



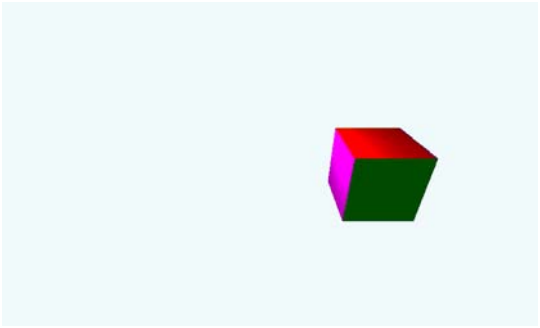
原圖:
一個正立方體在觀察者的右前方，
觀察者位於原點(0,0,0)角度(0,0)



平移:
觀察者從原點往上平移(y 軸的值增加)
觀察者位於原點(0,100,0)角度(0,0)



水平旋轉:
觀察者順時針旋轉 20º
觀察者位於原點(0,0,0)角度(20,0)



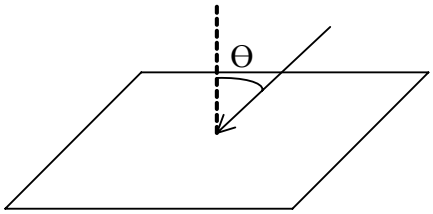
垂直旋轉:
觀察者以俯角 30º觀察
觀察者位於原點(0,0,0)角度(0,-30)

# (九)如何產生光與影的效果

由於自然情況下的光十分複雜，而我們目前對光的了解甚少，所以在此採用的爲理想的點光源。以下規則是我們從目前對物理的認知所做的推論。

## 1 理想光源之規則

1. 僅有面會產生漫射，漫射將會朝各個角度均勻發散。
2. 自然界中除了所加入的光源，尚有其他光線從四面八方出現，現實狀況亦類似，除非身處暗房中的絕對黑暗，同時也是爲了避免沒照到的面，因亮度太暗而不易觀察。
3. 光的影響力與距離平方成反比。
4. 光線照於平面時其影響力與入射角之餘弦值呈正比。



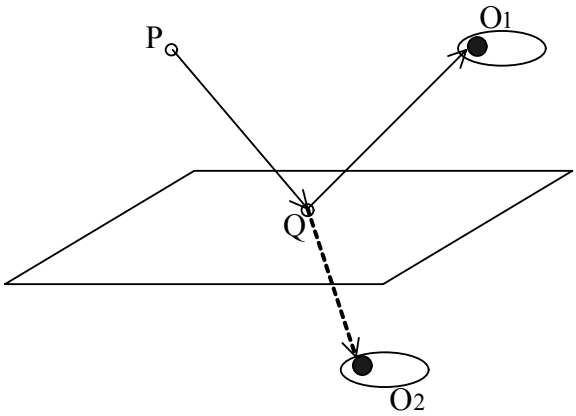

5. 光源本身無法直接被觀察到，爲一極小的點。

## 2 處理程序

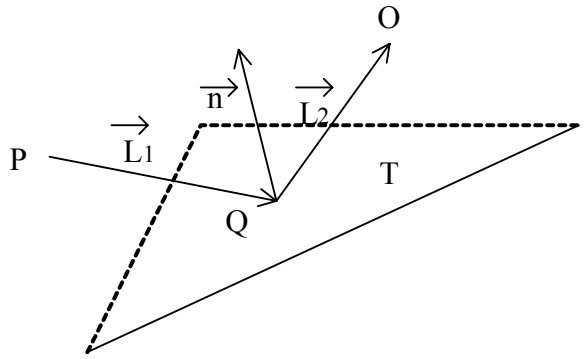
若模型中有加入光的物件，便需要將輸出暫存記憶體中的每個已被畫過的像素作處理。
假設有一光源位於座標的 P，而暫存記憶體中(x,y)有被畫過，它的實際座標爲 Q。
簡單來說，只要光源 P 到 Q 中間沒有阻礙物，則暫存記憶體中(x,y)的亮度便要提高。

### (1) 觀察者與光源需位於同側

考慮下圖，$O_1$, $O_2$ 分別在平面的兩側。在這種狀況下，很明顯地，觀察者 $O_1$ 可以感受到來自 Q 的漫射，而對於觀察者 $O_2$ 則無效果。發生這樣的情況時，簡單來說就是當 P 和 $O_1$ 在反射面的同一側，而 P 和 $O_2$ 在異側。



我們以數學向量來判斷光源 P 和觀察者 O 是否在同側




Q 爲三角面 T 上的一點

設 T 之法向量爲 $\vec{n}$ ， $\vec{L}_1 = \overrightarrow{PQ} \Rightarrow -\vec{L}_1 = \overrightarrow{QP}$

$$\vec{L}_2 = \overrightarrow{QO}$$

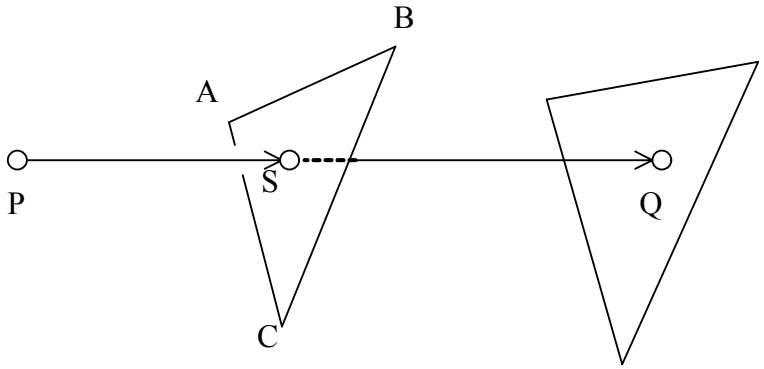
爲了讓 P 和 O 在同側，$(-\vec{L}_1 \cdot \vec{n})$ 和 $(\vec{n} \cdot \vec{L}_2)$ 必須同號，且皆不爲 0。若爲 0 代表光線平行入射或觀察者的視線平行反射面(漫射光會被鄰近組成面的點阻擋)。如此一來便能保證 P 和 O 位於同一面。

因此我們可以推得：若 $(L_1 \cdot \vec{n}) \times (\vec{n} \cdot \vec{L}_2) < 0$，則光源 P 經 Q 的漫射對觀察者 O 爲有效的。

## (2) 被其他平面阻礙

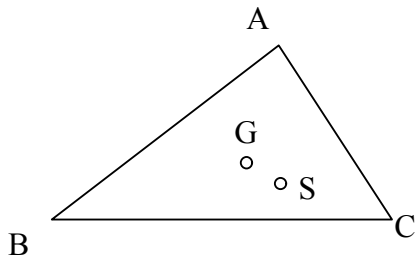
1. 判斷是否有三角面阻礙

先將三角面 ABC 之平面方程式 E 求出、$\overleftrightarrow{PQ}$ 也轉換爲參數式。如此一來便可得到 E 與 $\overleftrightarrow{PQ}$ 之焦點 S。



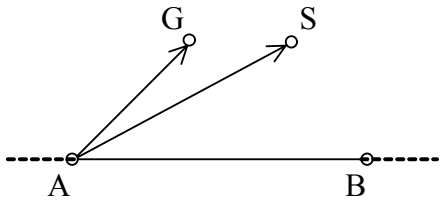

若 S 落在 $\overline{PQ}$ 之外，則三角面 ABC 就不會阻擋光線；若 S 落在 $\overline{PQ}$ 之內，此時還需判斷 S 是否在三角形 ABC 內。

此時借用三角形 ABC 之重心 G(因爲 G 一定落在三角形 ABC 內)。




判斷 G 和 S 是否皆位於 $\overleftrightarrow{AB}$ 、 $\overleftrightarrow{BC}$ 、 $\overleftrightarrow{CA}$ 之同一側即可。

就 $\overleftrightarrow{AB}$ 而言：

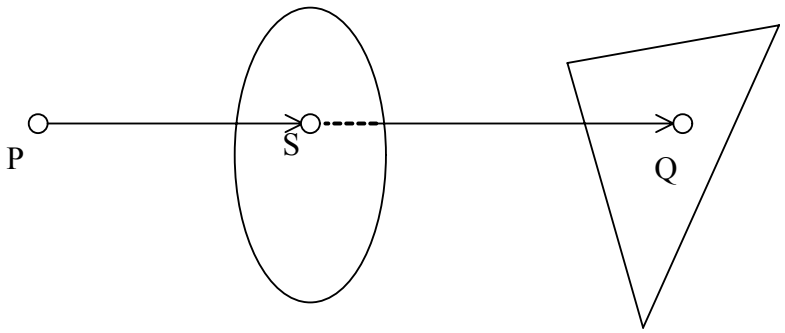



若 $\overrightarrow{AB}\times\overrightarrow{AG}$ 和 $\overrightarrow{AB}\times\overrightarrow{AS}$ 同向( 即 ( $\overrightarrow{AB}\times\overrightarrow{AG}$ ) dot ( $\overrightarrow{AB}\times\overrightarrow{AS}$ ) > 0 )，則 G 和 S 位於同側。

對於 $\overleftrightarrow{BC}$ 和 $\overleftrightarrow{CA}$ 處理方式亦相同。

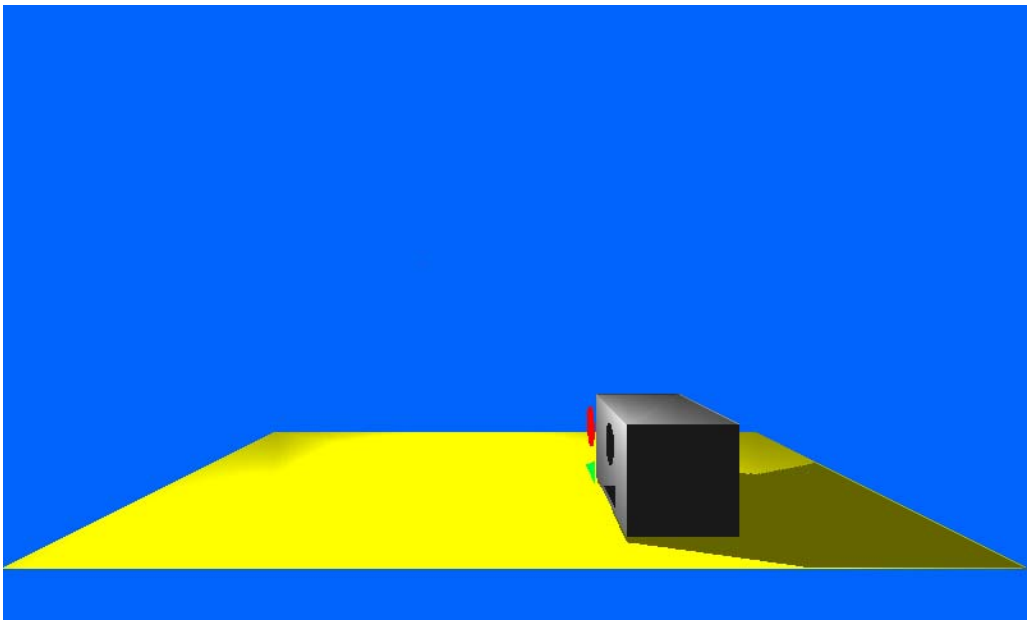
若得知 S 落在三角形 ABC 內，光線便會被阻擋，反之則否。

2. 判斷是否有圓面阻礙




方法類似於三角面，但是判斷 S 是否在圓裡較容易，只需比較 S 和圓心的距離是否在半徑以內即可。

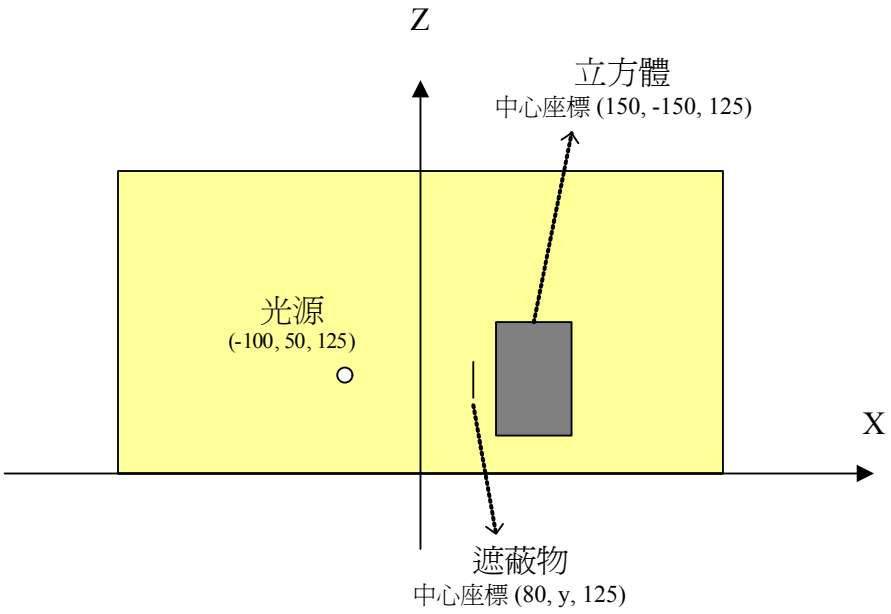
# 3 範例呈現



說明：

1. 由於距離遠近、照射角度不同，使得在立方體上表現出漸層的顏色。
2. 對於最靠近觀察者的一個面，因爲背光(觀察者和光源在面的兩側)，不受光的影響而呈現深色。
3. 顯示出當遇到障礙物時陰影的呈現，如：紅色圓面、綠色三角面皆於立方體上形成陰影。

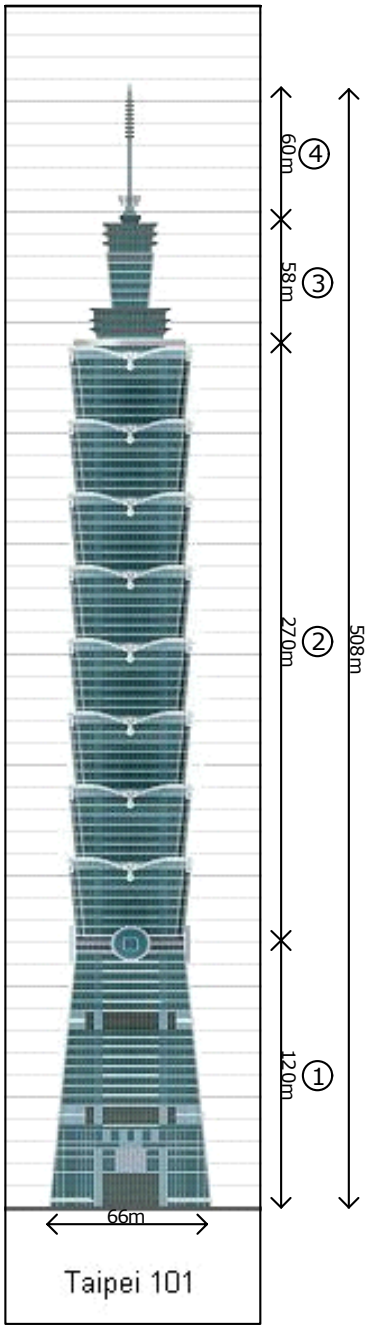
相對位置座標俯視圖



# (十)台北 101 模型建置

## 1 觀察

我們使用數位相機從不同角度拍攝，觀察照片中 101 大樓的形狀及外觀特徵，大致上分成四個部分(圖片 伍-18)。第一部分是 1-26 樓，高約 120M，是整棟樓的基底；第二部分 27-90 樓，爲 101 主要部份，共有 8 節，每節 8 層樓，約 390M，每節稍微傾斜 7 度，四個轉角有 W 型的缺角，寬約 1M；第三部分 91-101 樓，較窄，寬 25M 一底，在上爲寬 18M 長條狀；最頂端的第四部份則是 60M 高的尖塔。



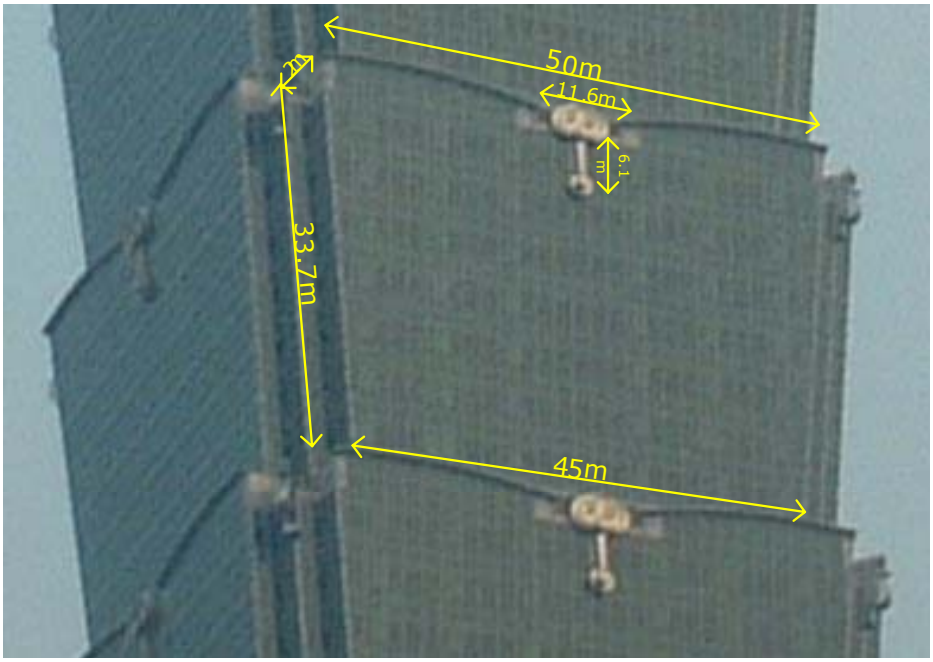

圖片 伍-18 101 整體結構圖

## 2 以照片或設計圖估計長度

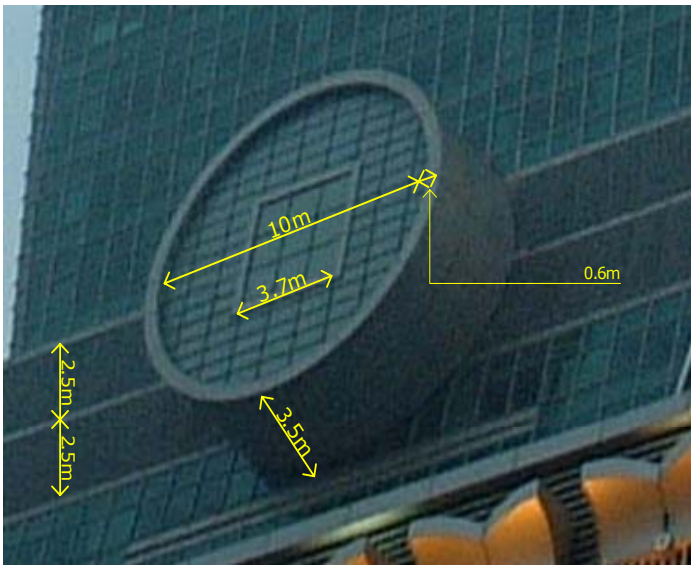
### (1) 第一部分－基底


圖片 伍-19

### (2) 第二部分－竹節

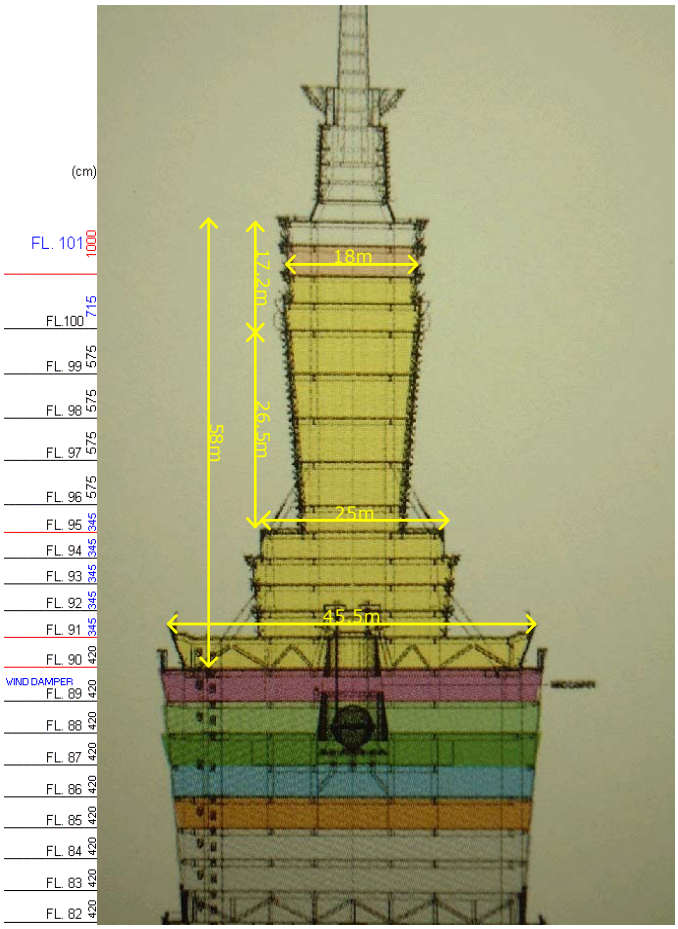



圖片 伍-20　其中一節




圖片 伍-21　主體下方之錢形結構

## (3) 第三部分－塔頂




圖片 伍-22　塔頂結構圖

(4) 第四部分－塔尖

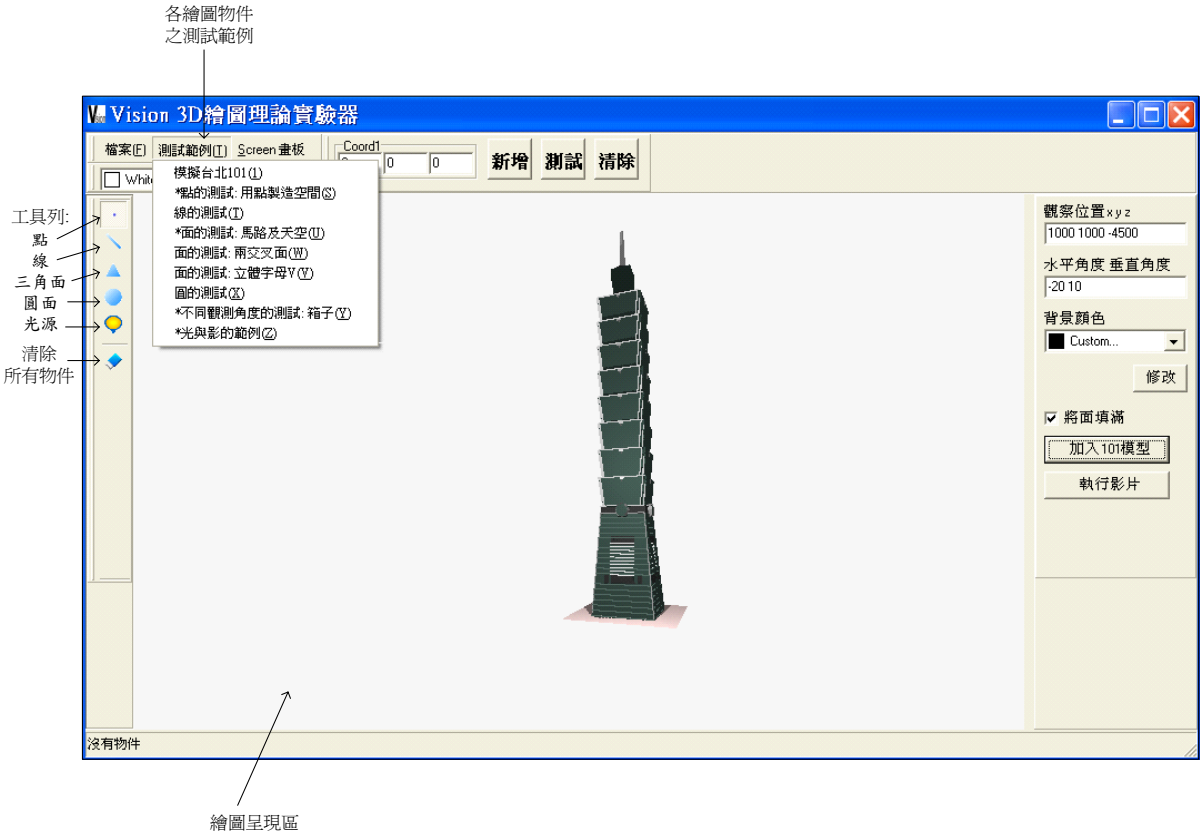

圖片 伍-23

## 3 節點座標換算

將所測量出來的尺寸作依據，作爲模擬 101 所需要的 3 維節點座標的換算依據。在建置模型的過程中我們將比例縮小約 133 倍，使電腦運算較易進行。101 實際高度爲 508 公尺，而我們所建模型高約 3.8 公尺。

模擬結果將在柒-成果展現呈現。

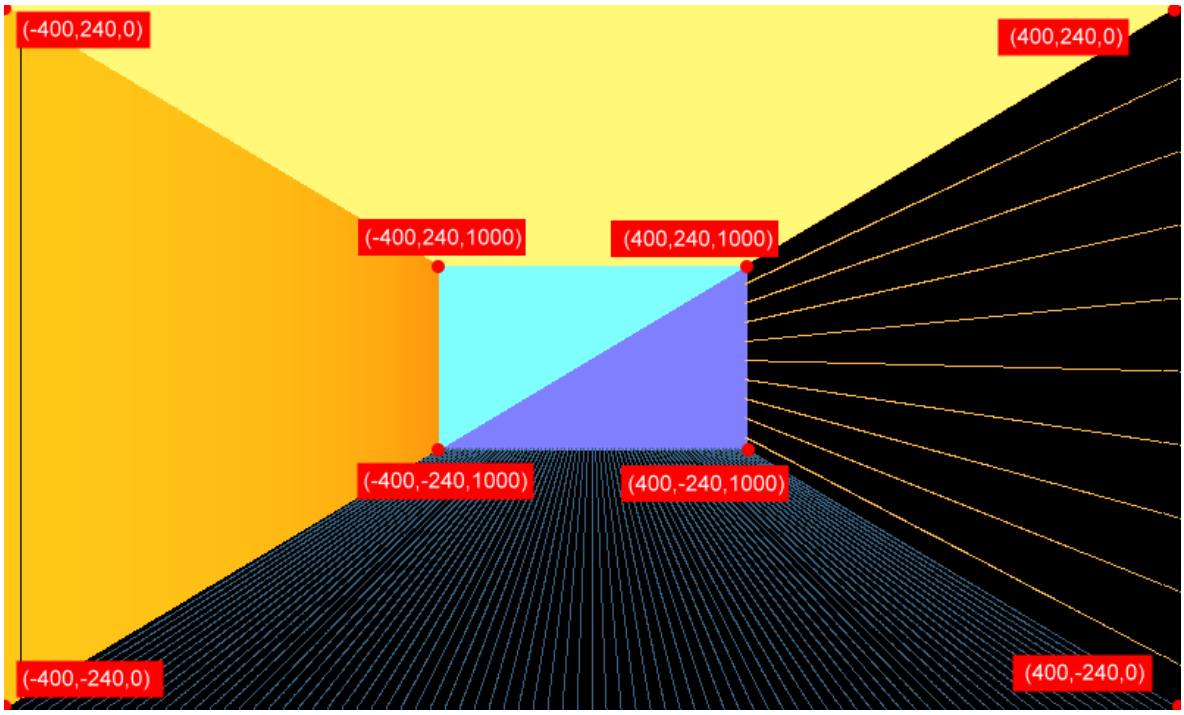
# 陸　軟體介面



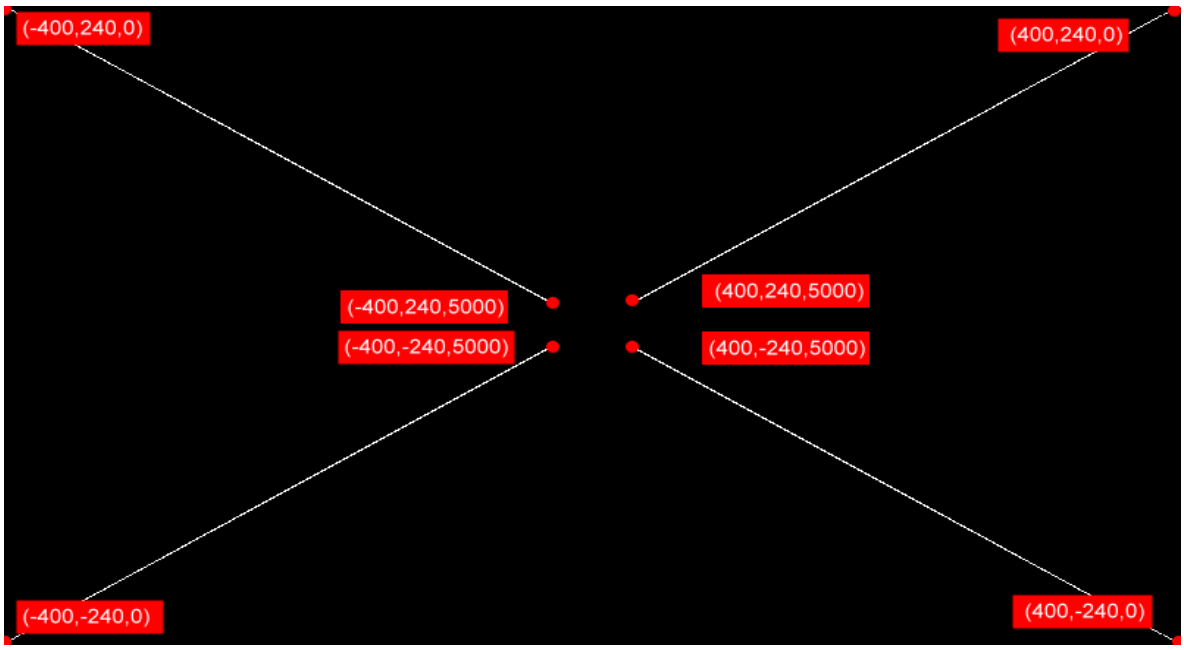

本程式我們將它定位爲實驗器材，用於驗證理論之正確與否，所以外觀及操作便利方面並非研究首先考量之重點。大部分操作，如測試範例都直接於程式碼完成。

# 柒　成果展現[3]

## (一)基本繪圖工具測試

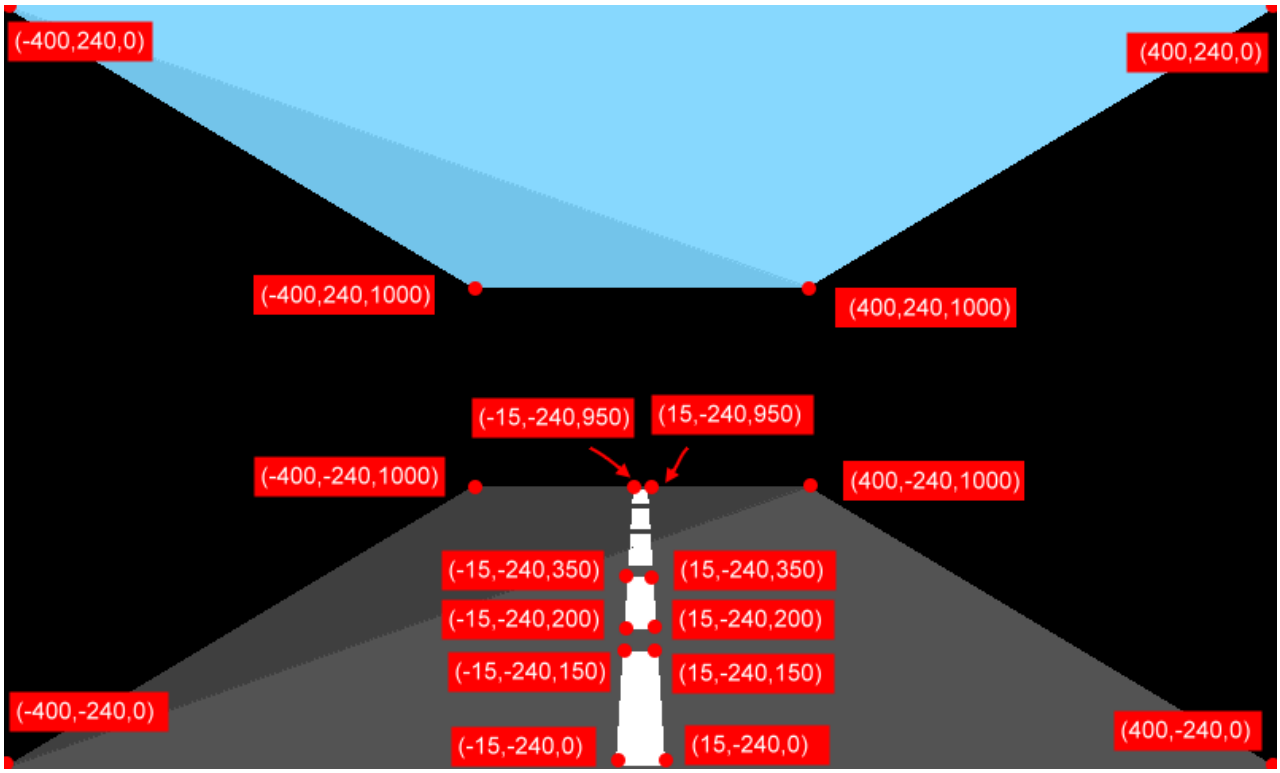



圖片 柒-1　點所形成的空間感，在這張圖中呈現了繪圖理論中最基本的近大遠小概念。

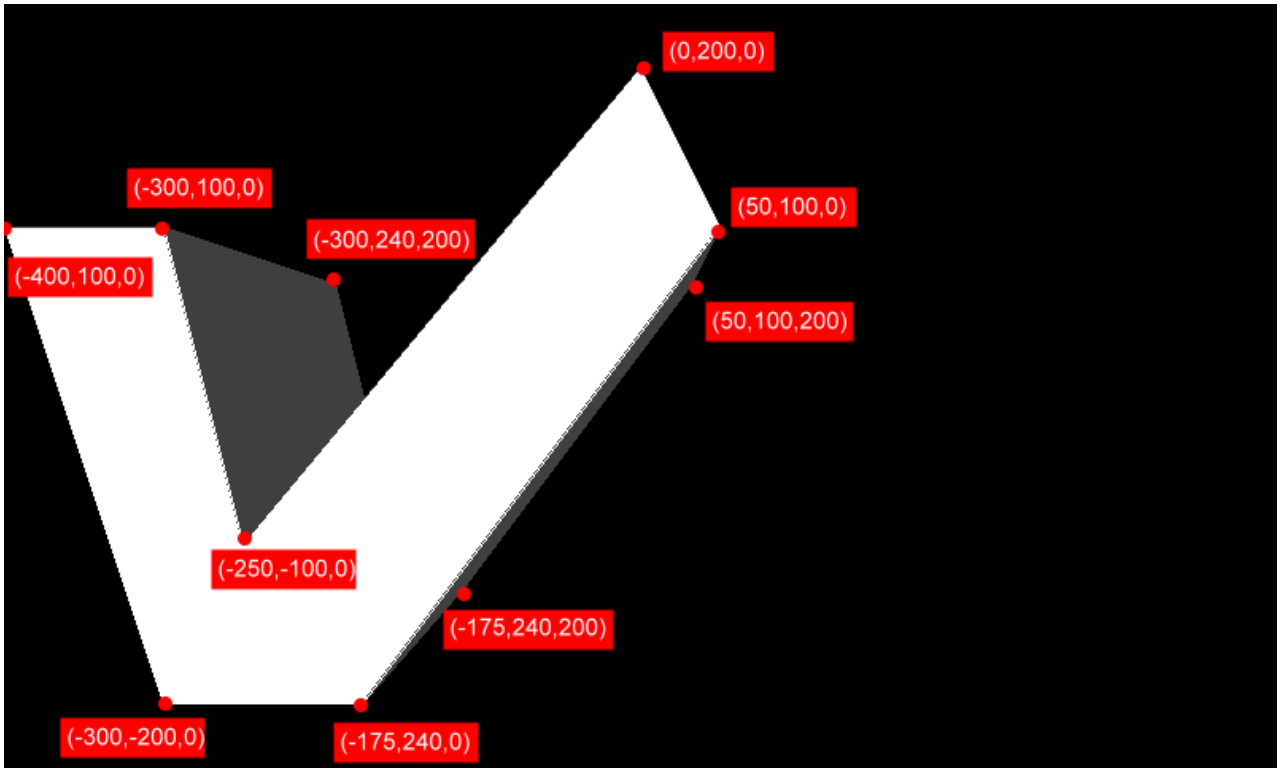



圖片 柒-2　線向觀察者前方延伸，呈現出一點透視概念。

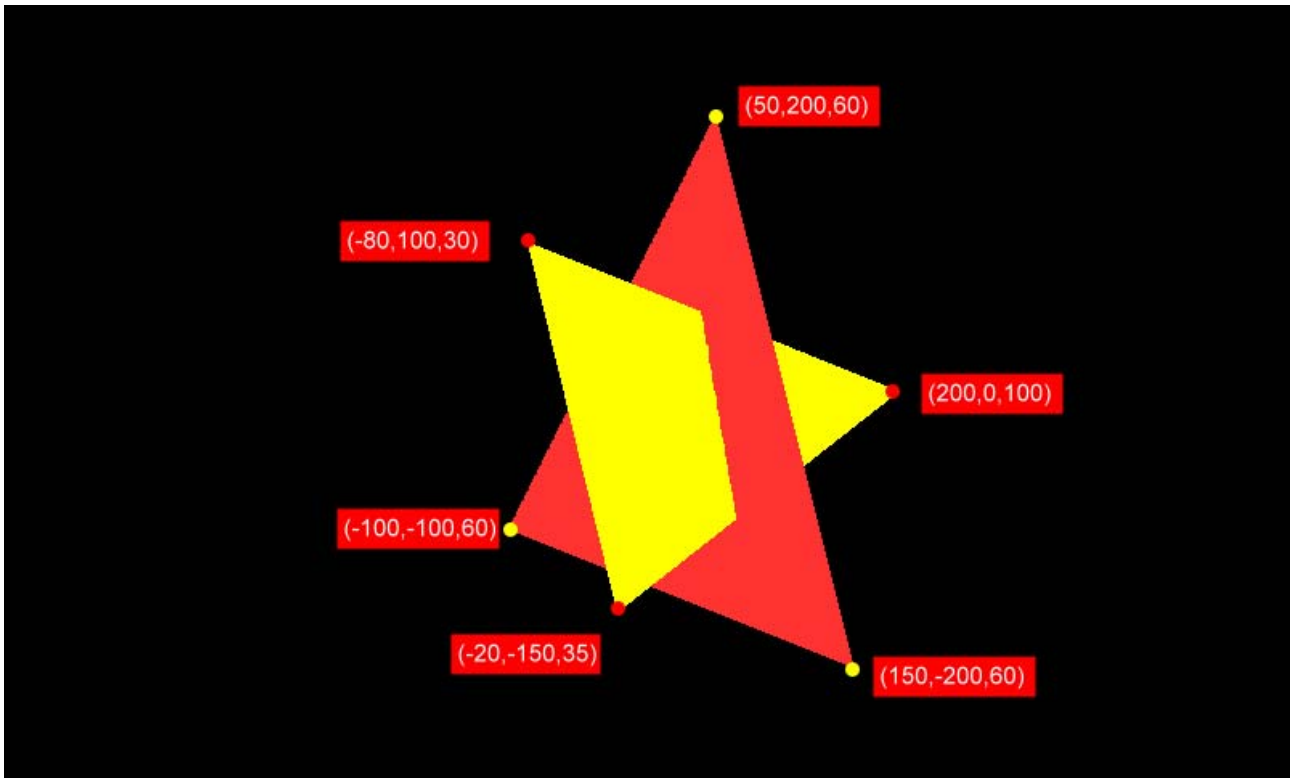
[3] 圖中紅色點爲額外標出的三維像素座標，1 像素=7.5E$^{-4}$ 公尺



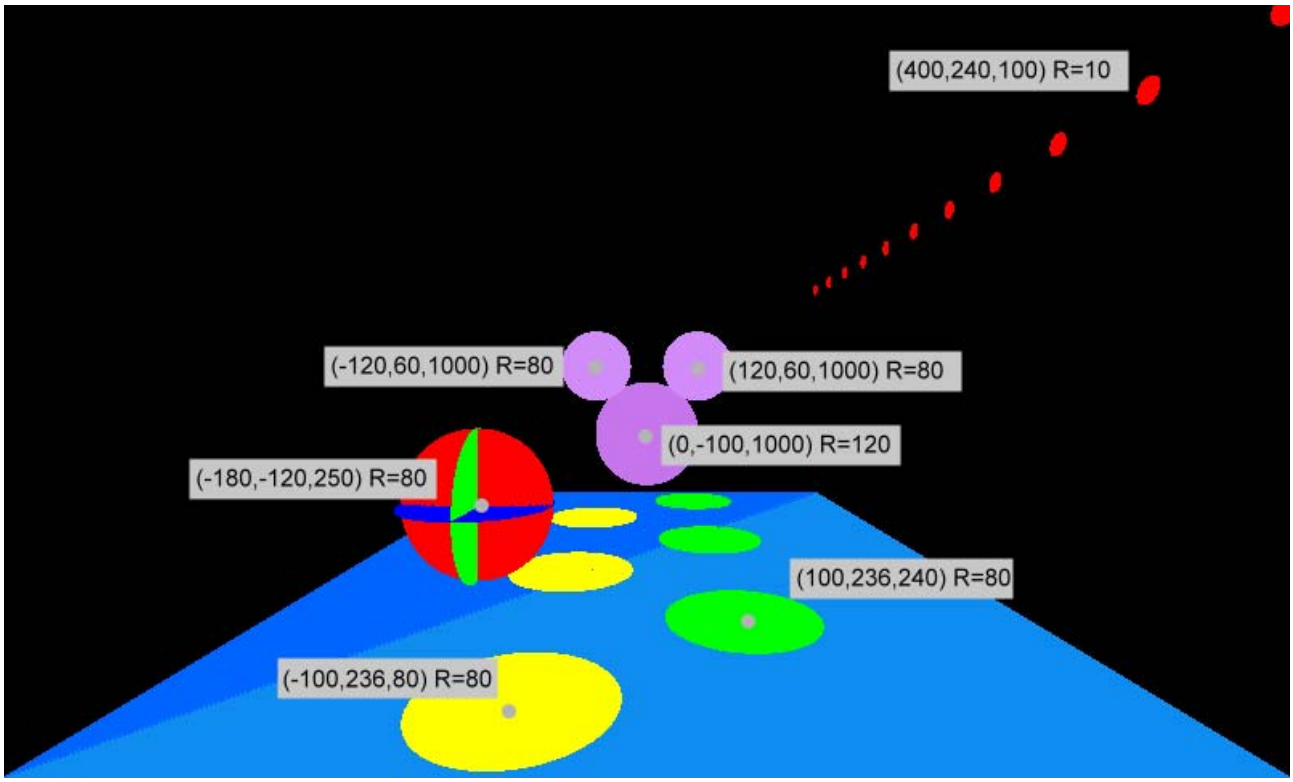

圖片 柒-3　以三角面所拼成的馬路(下方灰色部分)及天空(藍色部分)，此爲三角面的測試。




圖片 柒-4　字母 V 之立體樣式。




圖片 柒-5　兩面相交時的情況。




圖片 柒-6　不同距離及方向之圓形。

## (二)101 大樓模擬成果

A. 觀察者位於空中以 30 度角俯視

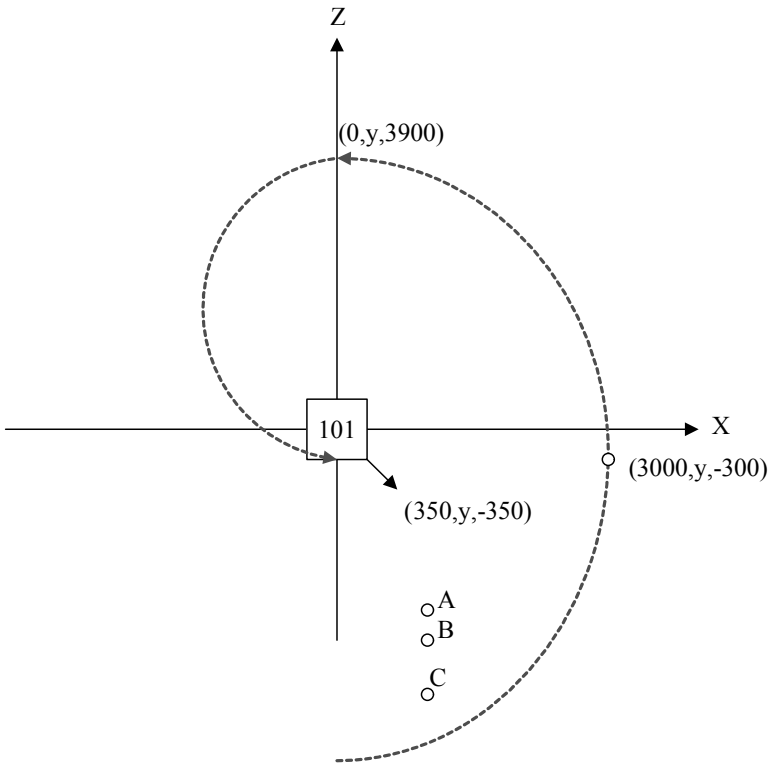
觀察者像素座標 A(1000,5000,-2200)　旋轉角度(-20,-30)

B. 觀察者位於 101 高度一半處 10 度角俯視

觀察者像素座標 B(1000,2500,-3000)　旋轉角度(-20,-10)

C. 觀察者位於地面以 20 度角仰視

觀察者像素座標 C(1000,20,-4500)　旋轉角度(-20,20)




上圖爲座標俯視圖，標示以上三種觀測位置與 101 之相對方位。
虛線軌跡爲展示影片中觀察者的動態位置。

# 捌　結論

## (一)研究成果

1. 將立體感的近大遠小表現在平面的螢幕上。
2. 研究出在平面上呈現空間中點、線、三角面、圓面的方法，並嘗試正確的填色方法。
3. 使用暫存記憶體處理像素重複的問題，並加快繪圖速度。
4. 有不同觀測方式的效果：可以讓觀察者在水平、鉛直、前後方向自由移動。並且透過旋轉的處理可以達到從不同角度觀察效果。
5. 從一般對光的認知，推導出產生光的規則，造成光與影的效果。
6. 實地觀測 101 並輸入電腦模擬，產生連續的影格，造成連續的動畫效果。

以上的技巧皆爲一些平日相當容易觀察到的自然現象，但將他們綜合後，並運用物件導向的程式設計，便可以達到一個簡易的模擬實物的效果。平日一般人無法自由地升至高空中，享受居高臨下俯看台北 101 的樂趣，但透過我們的研究使它變成一件可能的事情。

## (二)未來展望

1. 物件分層群組，可以使繪圖更有條理。如汽車可分爲輪胎、車殼、方向盤，只需要將各部份元件分工完成，組成複雜模型將會更容易進行。
2. 人性化介面-製作更易理解的介面，如建立座標系，讓使用者能輕易掌握物體的實際位置。由滑鼠做視覺操控也可以讓使用者更容易建置模型。

## (三)應用

1. 這些繪圖技術，可以用於立體概念的學習，對於學習美術，數學，三角測量者皆具有相當程度的幫助。
2. 因爲我們的技術可從各種角度，各種方向來觀測我們所建構的模型，而且也能夠使用程式產生一連串模擬出來的影像，現在即可以將所製成的連續的實體模擬如台北 101，利用定時撥放的方式，將所有的圖片一張一張地撥放，製成動畫。

# 玖　參考文獻

## (一)繪圖原理


1. David F. Rogers & J. Alan Adams(1983).Mathematical Elements for Computer Graphics. 台北市:科技圖書股份有限公司
2. 南一書局。高級中學數學〔甲〕上冊，台北市。


## (二)電腦技術


1. Stanley B Lippman & Josee Lajoie 譯者：侯捷。C++ Primer 中文版。碁峰
2. Time 研究室。C++ Builder 6 完全攻略。金禾。


## (三)台北 **101** 相關資料來源


1. TAIPEI 101. C.I.S.的 TAIPEI 101 介紹網站
   http://home.kimo.com.tw/cis_taipei101/home00.htm
2. 台北金融大樓股份有限公司網站
   http://www.tfc101.com.tw

# 評語

研究此問題深具實用價值，但目前尚有幾處關鍵點尚待突破，例如：數處取像的位置距離和遮蓋處如何取像，上、下圖像取得方式？建議應先研讀相關文獻，了解目前技術上瓶頸，再尋求創新突破的機會。