Java Tutorial

Chapter 4　幾何做圖

4.1 iGeo 函式庫

在本章節中，我們要開始對資料進行視覺化的動作，每當我們有許多資料的時候，對使用者來說最快速了解的方法就是將其視覺化，如此一來就可以一目了然並且快速掌握重點，並且亦可以輕鬆地了解資料的趨勢與變化。從另一方面來說，我們亦可以將一個問題轉數值化之後再來進行視覺化，可以藉由這個過程來進行一些模擬。當然資料視覺化是一個很多樣化的領域，簡單如做出長條圖，複雜卻可以做到一些 3D 建模等等不同的東西。不過在這邊我們要比較注重的是在進行與建築有關的模擬，因此一些針對顏色、向量、曲面等等的基礎知識將會在這邊做介紹。

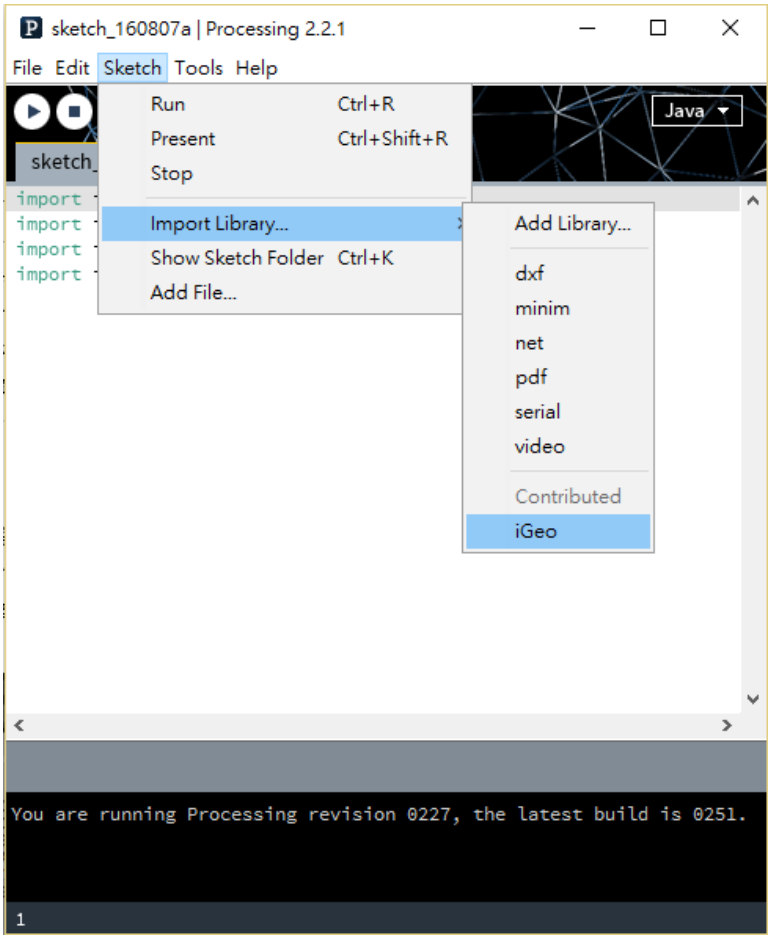
在這之前，我們要先介紹一個函式庫，也是我們往後的章節裡會非常倚重的函式庫：iGeo。iGeo 是一個開園的 3D 模擬函式庫，許多建築模擬、產品模擬，以及互動設計都可以藉著這個函式庫所提供的函式進行設計。因此我們要將這個函式庫匯入到我們的 Processing 裡面。首先我們要先進到 iGeo 的官網裡面下載此函式庫 http://igeo.jp/download.html

下載支援對應 Processing 版本的 iGeo 並且解壓縮，將解壓縮後的資料夾放在 Procssing 的”libraries”的資料夾裡。Processing 的”libraries”資料夾預設會放在

Windows 系統：“My Documents\Processing\libraries\”

Mac 系統：“~/Documents/Processing/libraries/”

放對位置之後理應會在打開 Processing 之後，在 Sketch 底下的 Import Library 裡看到 iGeo 出現。



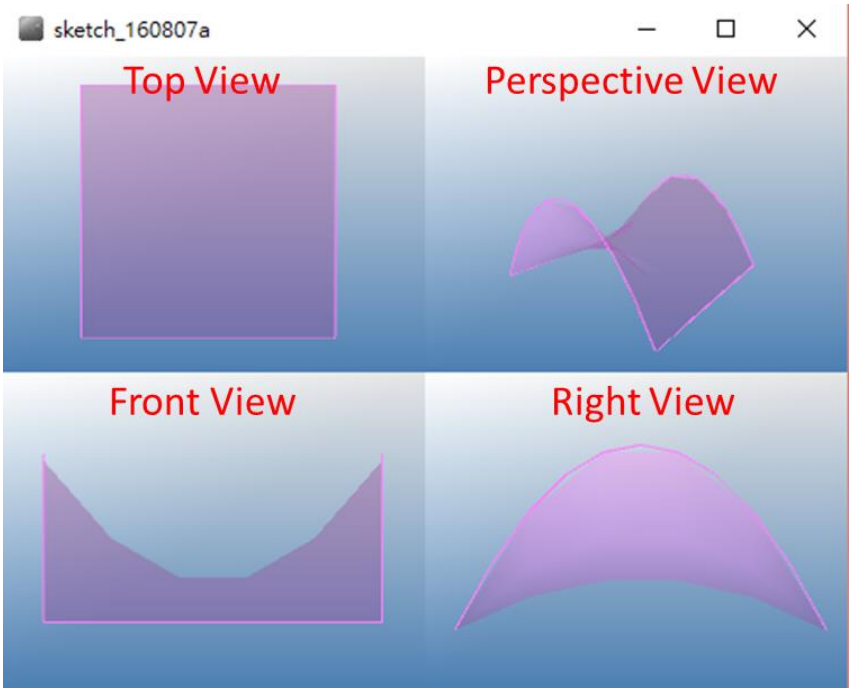
成功的匯入之後，就可以開始運用 iGeo 了。

每當要使用 iGeo 的函式之前，我們通常都會先打上一行”起手式”，也就是要先把我們的畫布建立起來。

```
import igeo.*;
```

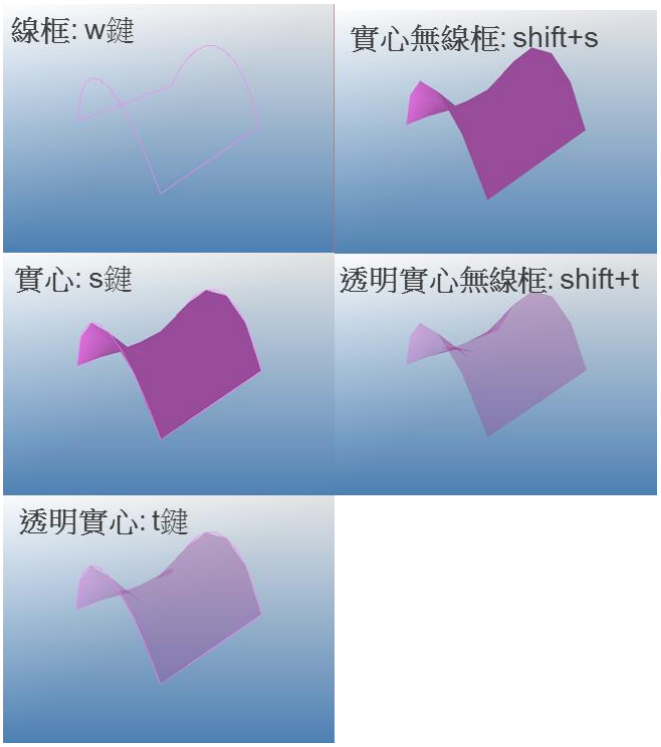
```
size(480,360, IG.GL);
```

這個函式的前兩個參數是用來設定畫布的寬與高，而第三個參數事告訴 Processing 使用 iGeo 並且會在背景啟動 iGeo 的相關系統。iGeo 的畫布與 Processing 本身提供的不一樣，它提供樂四種不同的視角來讓我們可以簡單地看出一個 3D 圖的不同面向。



在獲得這個畫面之後，我們可以對它做一些動作來進行觀察。

旋轉: 方向鍵 或 滑鼠左鍵 或 滑鼠左鍵+alt
移動: 滑鼠中鍵 或 shift+方向鍵 或 shift+滑鼠左鍵
放大縮小: 滑鼠右鍵 或 ctrl+滑鼠左鍵 或 ctrl+方向鍵
觀察特定視角: 雙擊左鍵 或 空白鍵
觀察特定物件: f 鍵
存檔：ctrl+s
關閉：ctrl+w 或 ctrl+q

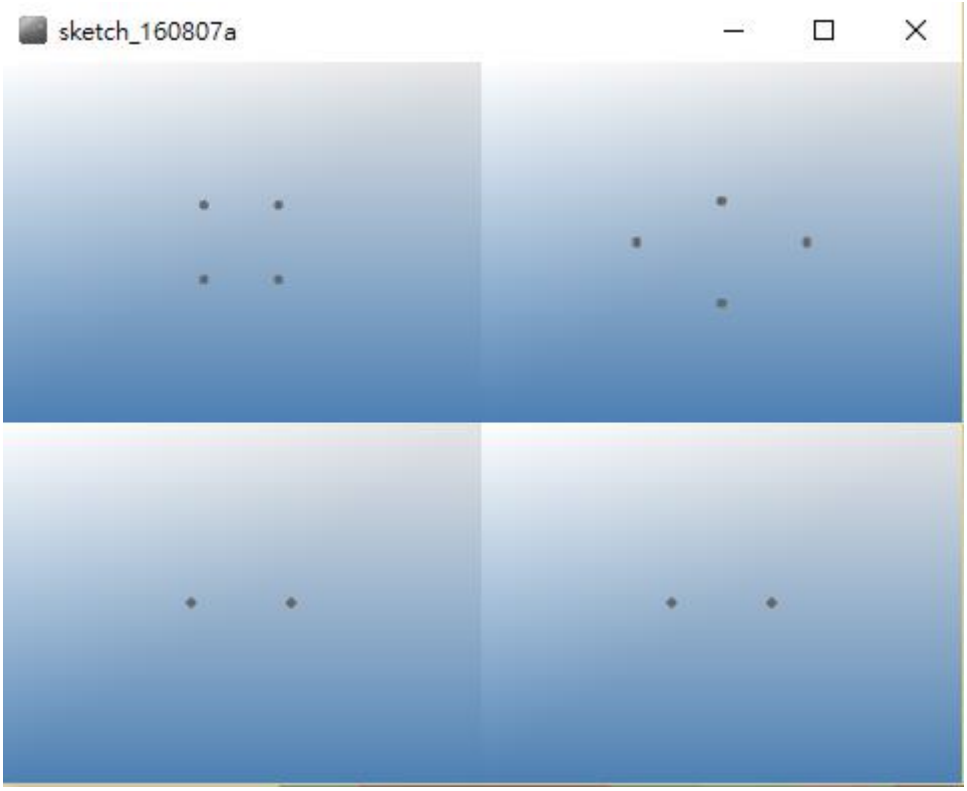


4.2 基礎幾何

(1) 點

首先我們最簡單的就是如何創建一個點，在空間坐標軸(x,y,z)中我們都用三個數字來分別代表在各自軸上與原典的距離，而這也就可以被當作一個點來看。

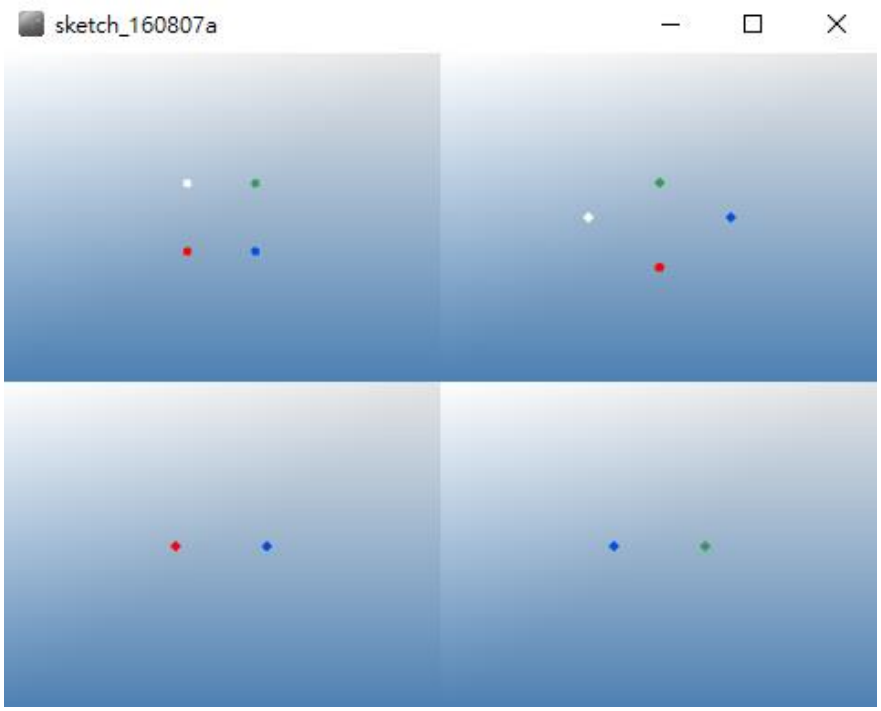
```
size( 480, 360, IG.GL );

new IPoint(0,0,0);
new IPoint(20,0,0);
new IPoint(20,20,0);
new IPoint(0,20,0);
```



(2)顏色

緊接著我們要幫點甚至是往後的圖形加上顏色，在這邊顏色系統我們原則上都使用 RGB，三個參數各自代表著紅綠藍三種顏色，而數值方面都是 0~255，又或是可以用 float 的表示那就是 0.0~1.0。然而，若只是想要 grayscale 的黑白顏色，那就只需要給一個介於 0~255 間的數值即可。

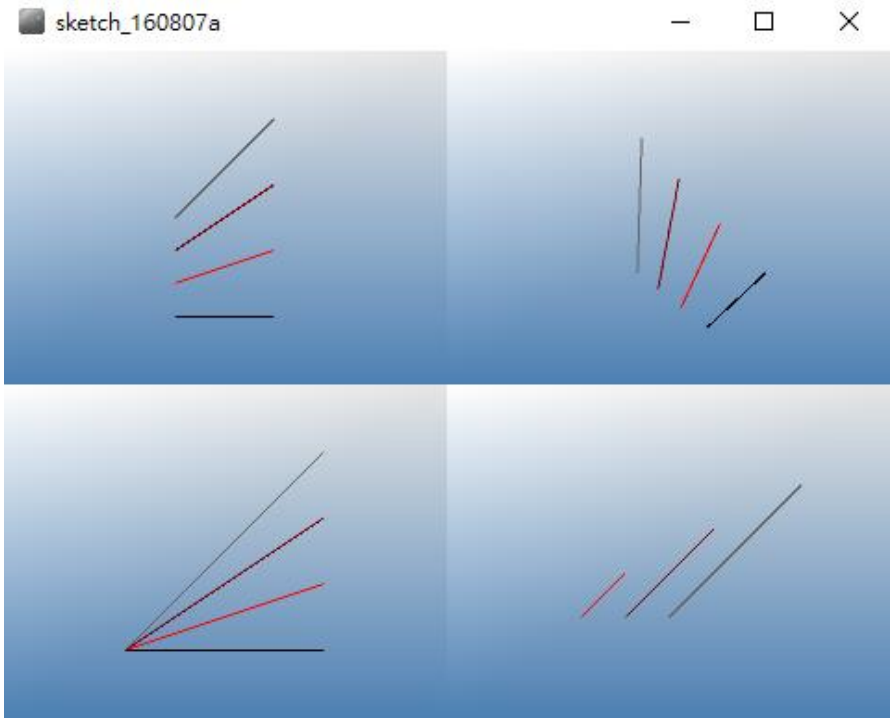
```
size( 480, 360, IG.GL );

new IPoint(0,0,0).clr(255,0,0); // RGB in int
new IPoint(20,0,0).clr(0,0.3,0.9); // RGB in float or double
new IPoint(20,20,0).clr(50,157,80);
new IPoint(0,20,0).clr(255); // grayscale in int, float or double
```



(3) 線

我們都知道兩個點可以形成一條線，因此在這邊要創造一條線也是一樣的概念，使用 ICurve()這個方法並且給予兩個點的座標就可以創造出一條線了。

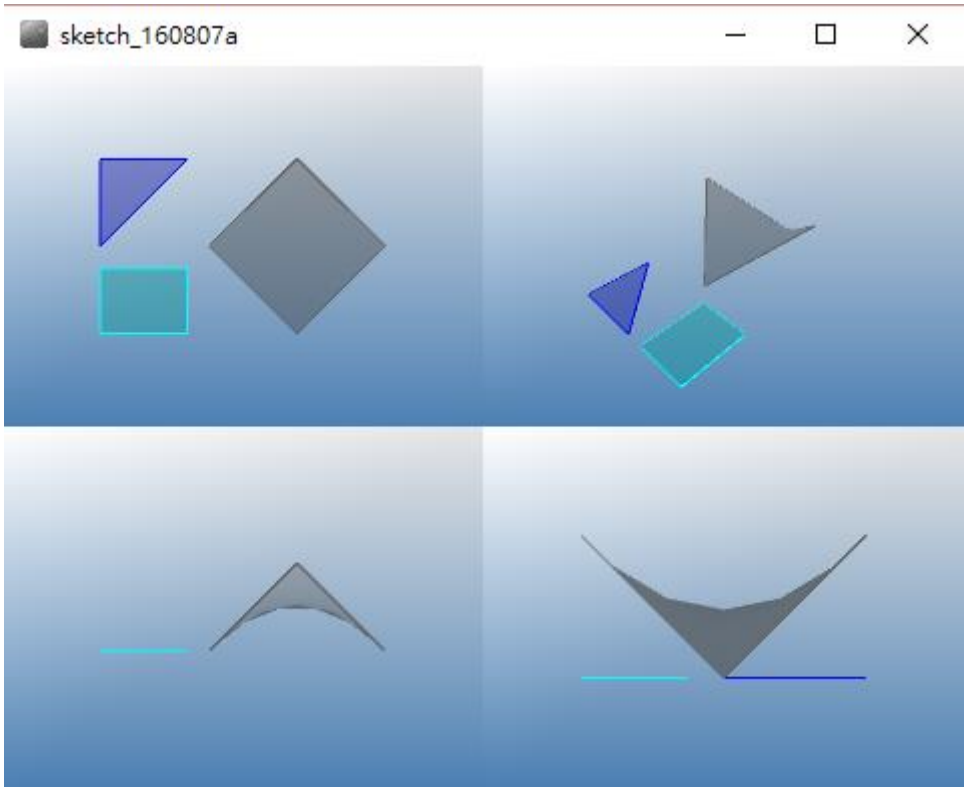
```
size( 480, 360, IG.GL );

new ICurve(0,0,0, 30,30,30);
new ICurve(0,-10,0, 30,10,20).clr(0.5,0,0);
new ICurve(0,-20,0, 30,-10,10).clr(1.0,0,0);
new ICurve(0,-30,0, 30,-30,0).clr(1,0,0);
```



(4) 面

三點成面，當然如果要做出一個例如方形的話也是需要四個點，因此我們使用 ISurface()加上 3~4 個點的參數來做圖。

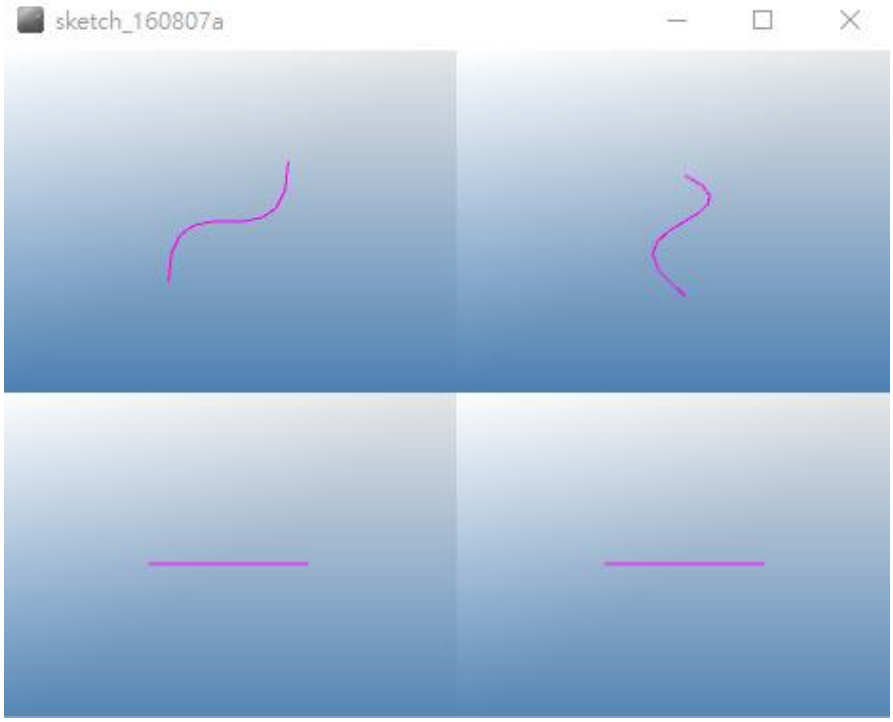
```
size( 480, 360, IG.GL );

new ISurface(0,0,0, 40,40,40, 80,0,0, 40,-40,40);

// 3 corner points (triangle)
new ISurface(-50,40,0, -10,40,0, -50,0,0).clr(0,0,1.);

// rectangle
new ISurface(-50,-10,0, -10,-10,0, -10,-40,0, -50,-40,0).clr(0,1.,1.);
```



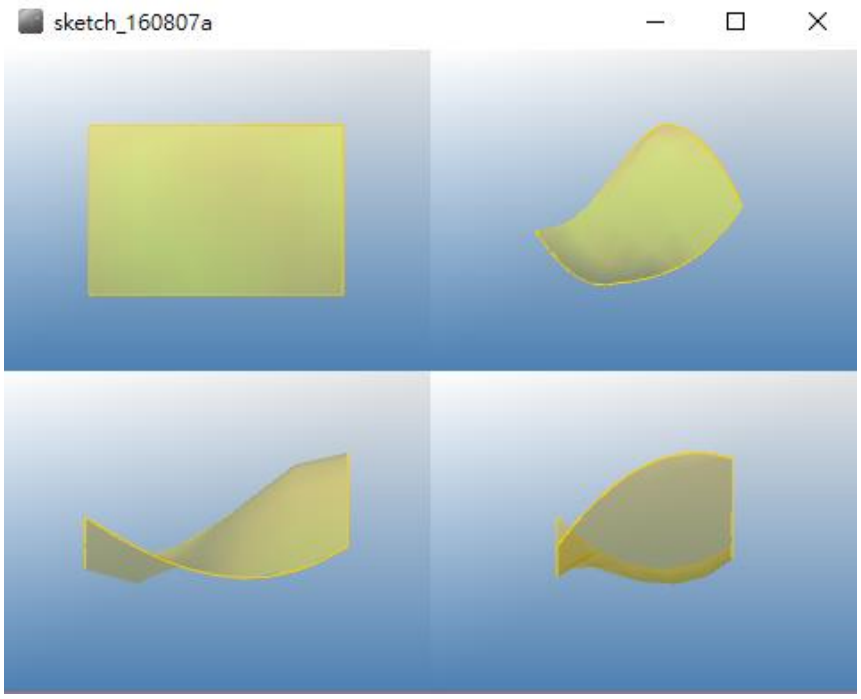
(5) 曲線、曲面

先前我們都只有畫出直線與平面，現在我們要開始畫出一些曲線甚至是曲面，而實作的方法就是在呼叫 ICurve()或 ISurface()時給予的參數是一連串陣列的點，並且加上要"彎"幾次的參數 degree。當然，若此線需要彎兩次那就需要給予三個點；若需要讓它轉三次則需要四個點，以此類推。至於在實作曲面的時候，會有兩個 degree 參數需要給予，分別是 u 與 v degree，而點的數目也是分別 u 與 v 的參數來決定要給予多少個點量。

```
size( 480, 360, IG.GL );

// degree 3 curve
double[][] controlPoints =
  {{30,-30,0}, {30,-10,0}, {50,-30,0}, {50,-10,0}};
new ICurve(controlPoints,3).clr(1.,0,1.);
```




```
size( 480, 360, IG.GL );

// 4 points in u direction, 3 points in v direction
double[][][] controlPoints2 =
  {{{-30,-30, 10}, {-30,  0,-20}, {-30, 30,  0}},
   {{  0,-30,-10}, {  0, 20,-50}, {  0, 30,-20}},
   {{ 30,-30,-20}, { 30, 20, 60}, { 30, 30, 30}},
   {{ 60,-30,  0}, { 60,  0, 40}, { 60, 30, 30}}};

// u degree = 3, v degree = 2
new ISurface(controlPoints2, 3, 2).clr(1,.8,0);
```



4.3 空間向量

在空間中，不論是點、線、面，甚至是曲面全部都是由向量所組成，每個幾何圖形其實說穿了就是由一堆不同的向量各自拓展並且組合而成，因此，在我們往後所做的許多模擬，其實說穿了也就是一連串的向量讓它進行改變。

(1) 向量表示

我們這邊不講太多數學性的東西，原則上來說向量就是一組(x,y,z)空間坐標軸上各一個數字所組成的，它代表了一個在空間上的方向。而在程式裡面，我們會有一個 IVec 的方法它可以建立出一個向量：

```
import igeo.*;

size(480,360, IG.GL);

IVec v1 = new IVec(0, 0, 0);
IVec v2 = new IVec(1, 2, 3);
```

想要建立向量，IG class 裡也提供了一些方法：

```
import igeo.*;

size(480,360, IG.GL);

IVec v1 = IG.vec(0, 0, 0);
IVec v2 = IG.vec(10, 20, 30);
```

```
import igeo.*;

size(480,360, IG.GL);

IVec v1 = IG.v(0, 0, 0);
IVec v2 = IG.v(10, 20, 30);
```

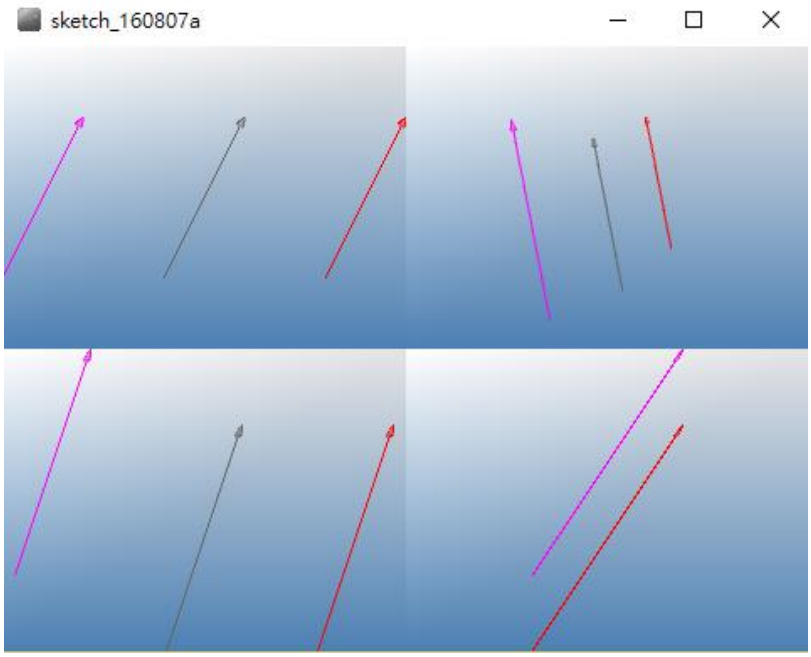
建立出向量在執行程式並不會在視窗中顯現出來，因為它只是一個代表，並沒有實際圖形的表示。

(2) 視覺化向量

雖然向量本身沒有實際圖形的代表，我們還是可以使用箭頭來當作向量的表示，因此我們可以將向量”畫”在我們的畫布上。

IGeo 非常貼心地給了我們一個很方便的函數，也就是 show()，這個函數可以讓我們輕易的在畫布上表示向量，而向量長度的部分就是看我們一開始在宣告向量的時候所給予的數字來決定。

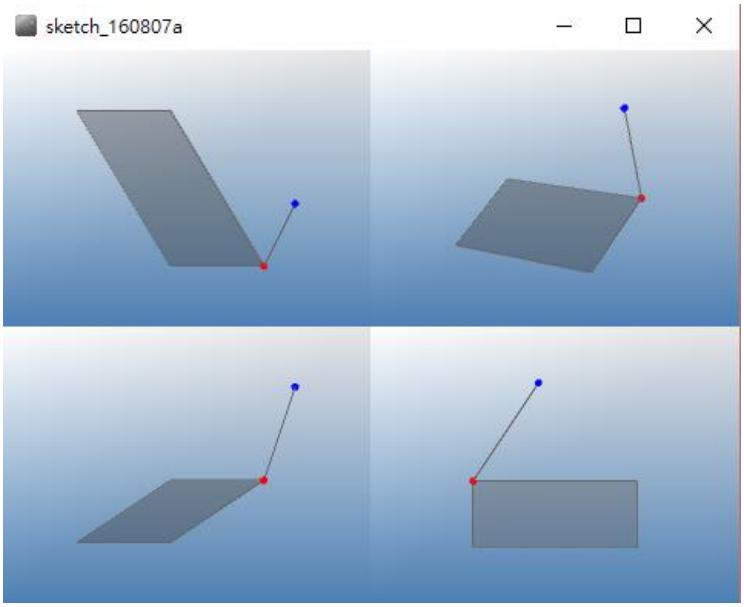
```
size( 480, 360, IG.GL );

IVec v1 = new IVec(10, 20, 30);

// showing the vector at (0,0,0) (gray)
v1.show();
// showing the vector at (20,0,0) (red)
v1.show( new IVec(20, 0, 0) ).clr(1.,0,0);
// showing the vector at (-20,0,10) (magenta)
v1.show(new IVec(-20, 0, 10) ).clr(1.,0,1.);
```



(3) 利用向量做圖

除了上一節都使用點來畫圖，這邊也可以使用給予向量來畫圖。

```
size( 480, 360, IG.GL );

IVec v1 = new IVec(0, 0, 0);
IVec v2 = new IVec(10, 20, 30);

new IPoint(v1).clr(1.,0,0);
new IPoint(v2).clr(0,0,1.);

new ICurve(v1,v2);

IVec v3 = new IVec(-30,0,-20);
IVec v4 = new IVec(-60,50,-20);
IVec v5 = new IVec(-30,50,0);

new ISurface(v1,v3,v4,v5);
```

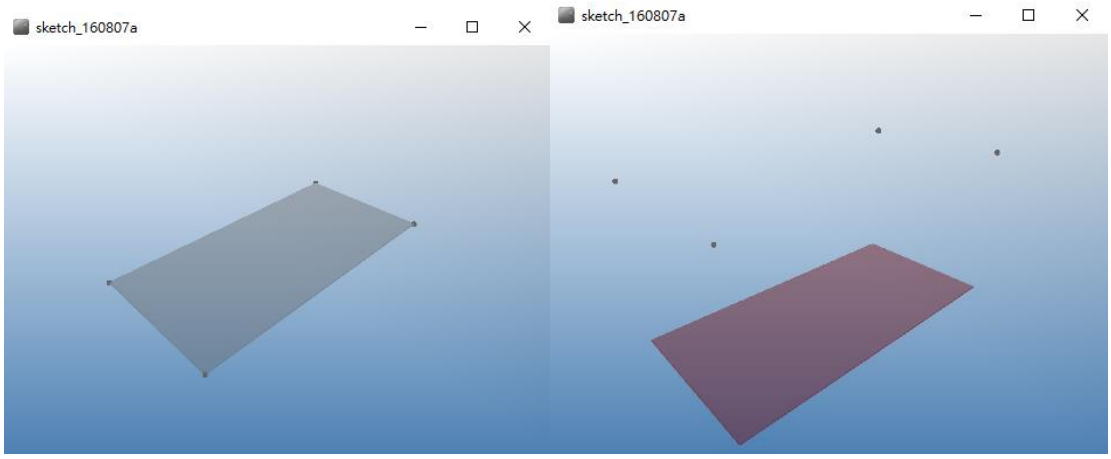


(4)幾何圖形變化

(a) 移動

第一個先進行的是移動，直接對該物件使用 mv()的方法，裡面的參數就是加上一個移動的向量。

```
size(480, 360, IG.GL);

IVec pt1 = new IVec(0,0,0);
IVec pt2 = new IVec(40,0,0);
IVec pt3 = new IVec(40,20,0);
IVec pt4 = new IVec(0,20,0);

//showing original location
new IPoint(pt1);
new IPoint(pt2);
new IPoint(pt3);
new IPoint(pt4);

ISurface surface1 = new ISurface(pt1.dup(), pt2.dup(), pt3.dup(), pt4.dup());
//surface1.mv(new IVec(0,0,-20)).clr(125,0,0);
```

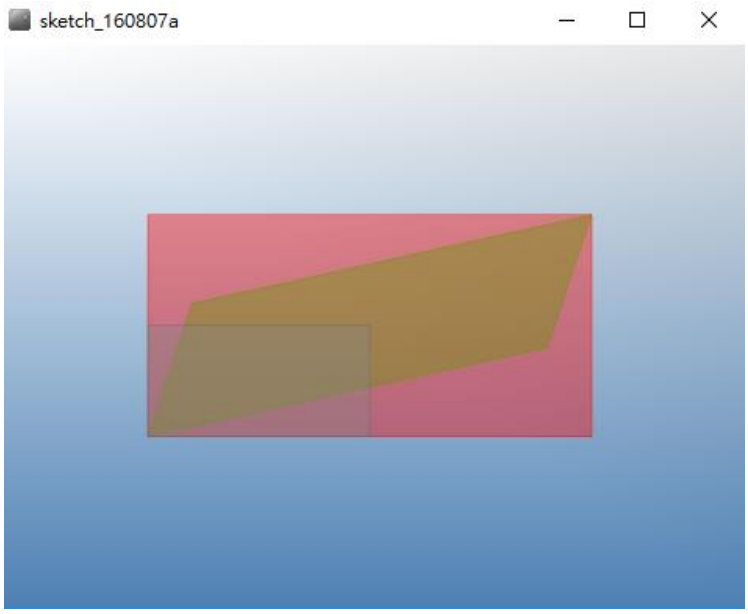


上面兩個圖左邊是尚未移動時候的平面，而右邊就是加上了 mv()函式移動過後的平面。

(b) 比例尺放大/縮小

再來就是將物體等比例放大縮小，使用 scale()的方法並且給予一個數值參數表示要放大多少倍率。同時，也有方法 scale1d()可以向特定一個方向來進行放大。

```
IVec pt1 = new IVec(0,0,0);
IVec pt2 = new IVec(40,0,0);
IVec pt3 = new IVec(40,20,0);
IVec pt4 = new IVec(0,20,0);

ISurface surface1 = new ISurface(pt1.dup(), pt2.dup(), pt3.dup(), pt4.dup());
ISurface surface2 = surface1.dup().scale(2).clr(255,0,0);
ISurface surface3 = surface1.dup().scale1d(new IVec(2,1,0), 2).clr(0,255,0);
```

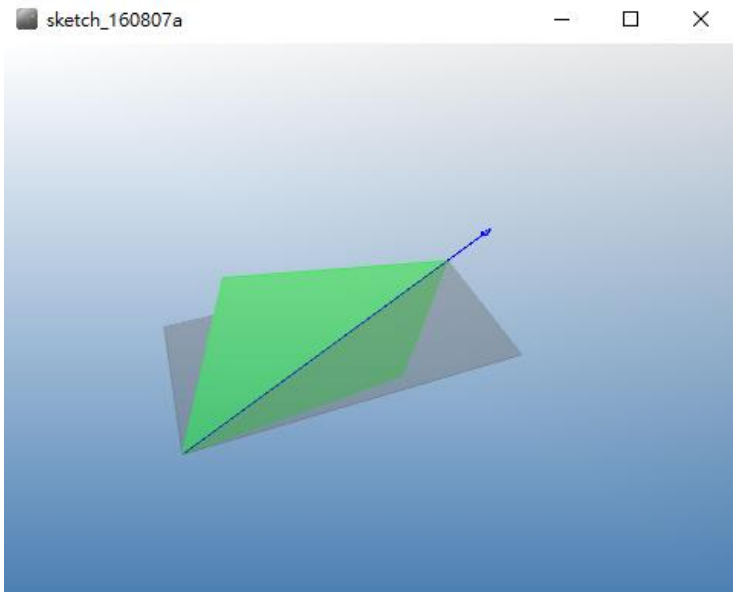


根據這個範例，最一開始的 surface1 是左下角灰色的，而 surface2 向兩方面都放大兩倍成為紅色的，至於最後的 surface3 則是往向量(2,1,0)的方向拉長兩倍而得到綠色的。

(c) 旋轉

接下來是旋轉，對一個物件旋轉我們會使用 rot()的方法，並且在裡面加上兩個參數(Ivec, angle)。第一個參數是一個向量，這個向量會作為旋轉的基準軸；而第二個參數則是旋轉的角度，原則上是以弧度為單位。

```
IVec pt1 = new IVec(0,0,0);
IVec pt2 = new IVec(40,0,0);
IVec pt3 = new IVec(40,20,0);
IVec pt4 = new IVec(0,20,0);

ISurface surface1 = new ISurface(pt1.dup(), pt2.dup(), pt3.dup(), pt4.dup());
ISurface surface3 = surface1.dup().rot(new IVec(2,1,0),PI/2).clr(0,255,0);
new IVec(2,1,0).show(new IVec(0,0,0)).mul(25).clr(0,0,255);
```



在這個範例中我們對這個面以向量(2,1,0)為軸進行旋轉，並且旋轉 pi/2 的角度，進而獲得綠色的面。

(5)映射

最後是映射，也就是將一個物件依照一個平面做映射。使用 ref()的方法，並且給予一個向量的參數，而這個向量就是一個面的法向量，物件就會照著這個法向量的面去進行映射。

```
IVec pt1 = new IVec(0,0,0);
IVec pt2 = new IVec(40,0,0);
IVec pt3 = new IVec(40,20,0);
IVec pt4 = new IVec(0,20,0);

ISurface surface1 = new ISurface(pt1.dup(), pt2.dup(), pt3.dup(), pt4.dup());
ISurface surface3 = surface1.dup().ref(new IVec(1,0,0)).clr(0,255,0);
```

這個範例就是照著 yz 平面來映射，綠色的就是映射過後的結果。

本章節介紹了一些幾何上面的設計與處理，然而這些必須要有一些對空間座標的概念，擁有這些概念可以比較容易去對這些圖形進行處理以及變化，這些東西在往後要處理更複雜的題目的時候可以比較容易去想像並解決。而在下一章，我們會先介紹一些隨機的概念，而後就會講解一些較複雜的圖形處理的技巧。