Chapter 13

類別間的關係(二)
幾何範例

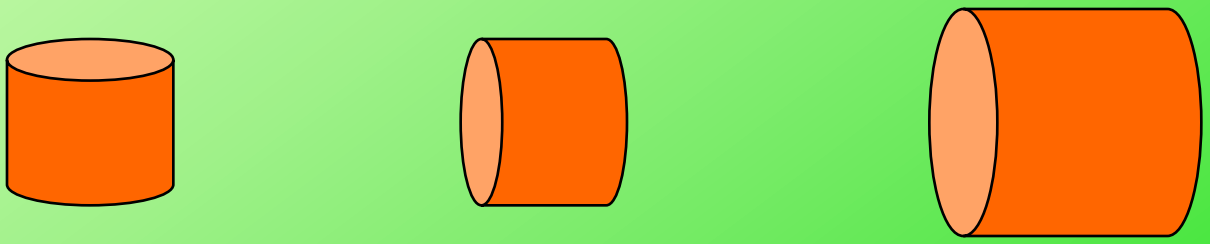
平面點的搬動

- 位置轉換：

  將平面點更動到另一個位置

- 尺度保持：

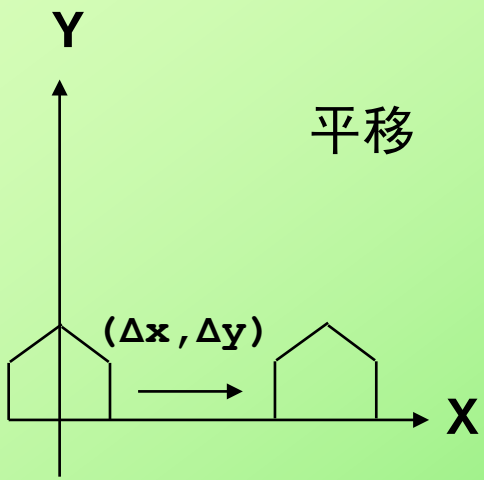
  物體的任兩點距離在經過搬動後仍保持不變



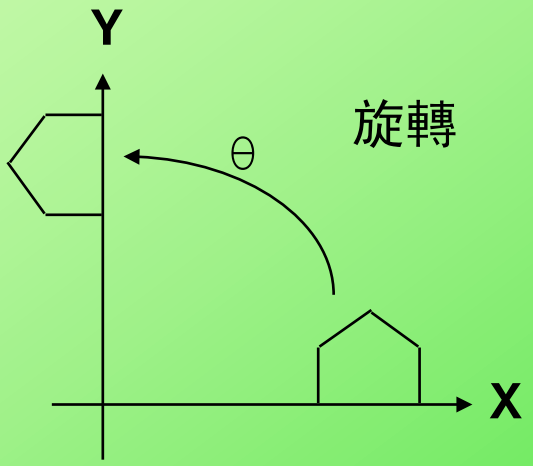
transformation, distance preserving

三種尺度保持的轉換動作（一）

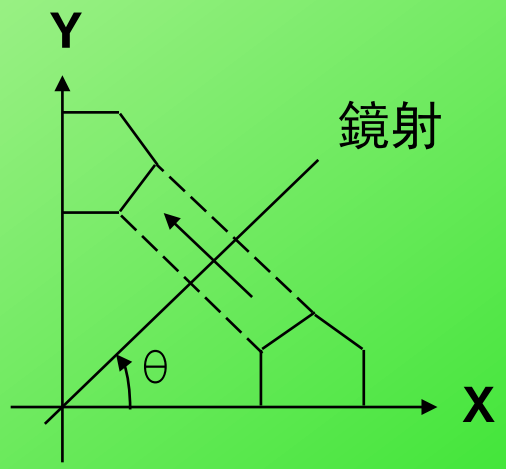
■ 平移



■ 旋轉



■ 鏡射



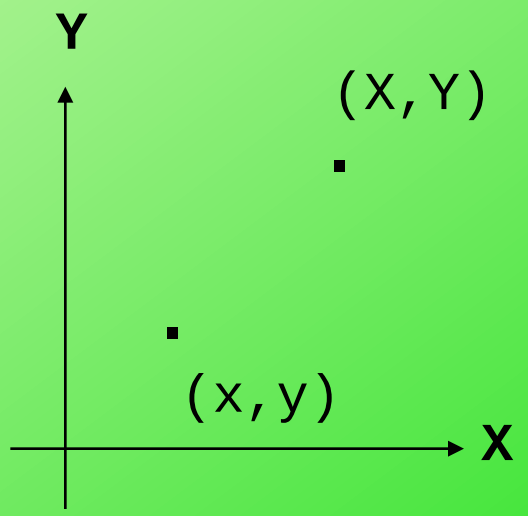
translation, rotation, reflection

三種尺度保持的轉換動作（二）

- 平面點

$$x = \begin{pmatrix} x \\ y \end{pmatrix}$$ ： 點在搬動前的座標

$$X = \begin{pmatrix} X \\ Y \end{pmatrix}$$ ： 點在搬動後的座標



三種尺度保持的轉換動作（三）

- 平移：平移距離為 $(\Delta x, \Delta y)$

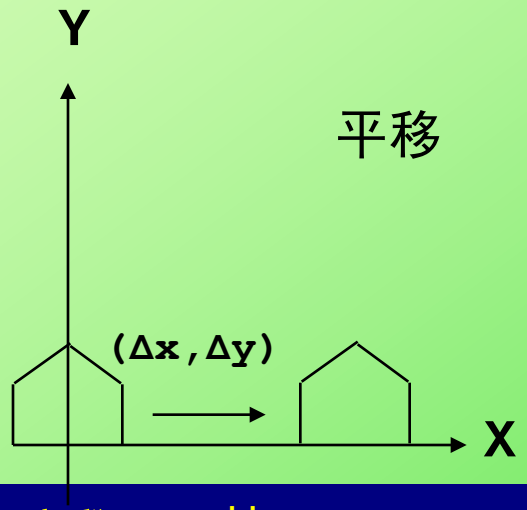
$$X = \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = x + \Delta x$$

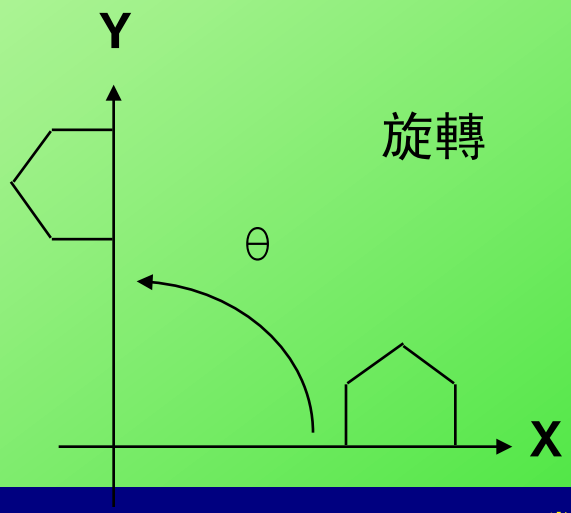
- 旋轉：逆時鐘旋轉角度為 θ

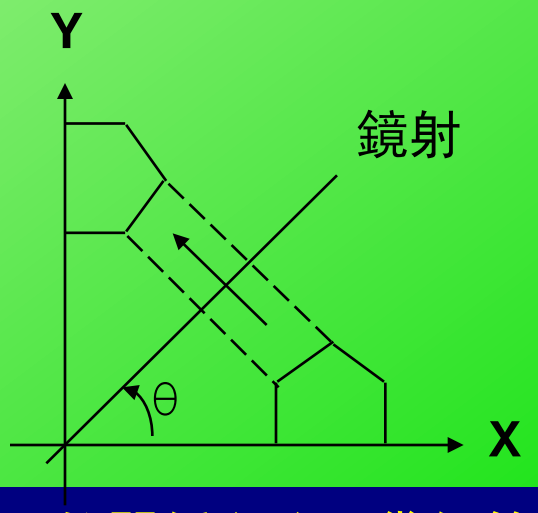
$$X = \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = M_r x \ , \ M_r = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

- 鏡射：鏡射軸角度 θ

$$X = \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = M_f x \ , \ M_f = \begin{bmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{bmatrix}$$







均勻座標

- 均勻座標系統：

  將二維空間上的平面點看成在三維空間 **Z = 1** 平面上的座標點

(x , y) → (x , y , 1)

homogeneous coordinate

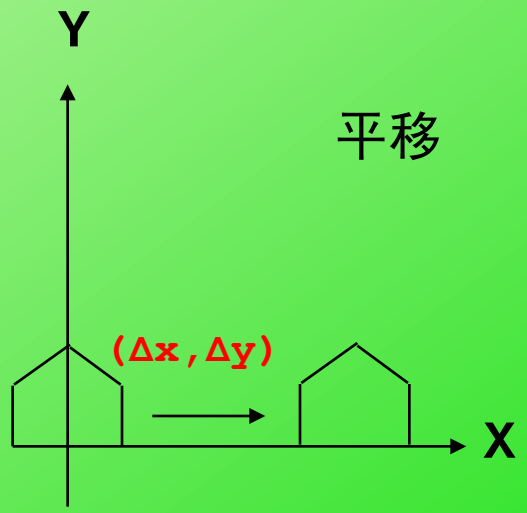
均勻座標的轉換動作（一）

■ 平移：平移距離為 $(\Delta x, \Delta y)$

$$X = \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$= M_t\, x$$

這裡 $M_t = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix}$



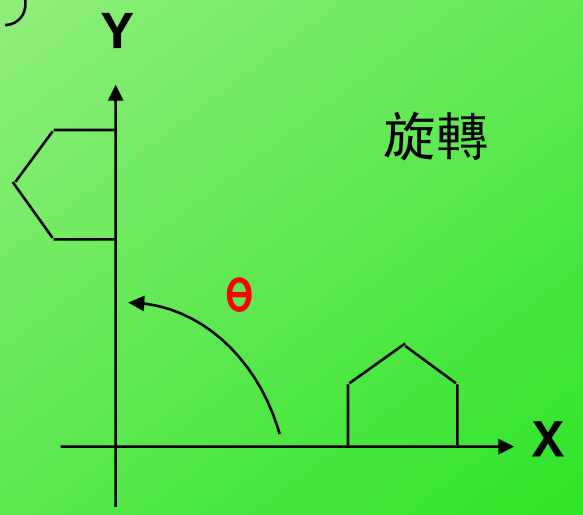
均勻座標的轉換動作（二）

- 旋轉：逆時鐘旋轉角度為θ

$$X = \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$= M_r\, x$$

這裡 $M_r = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$



均勻座標的轉換動作（三）

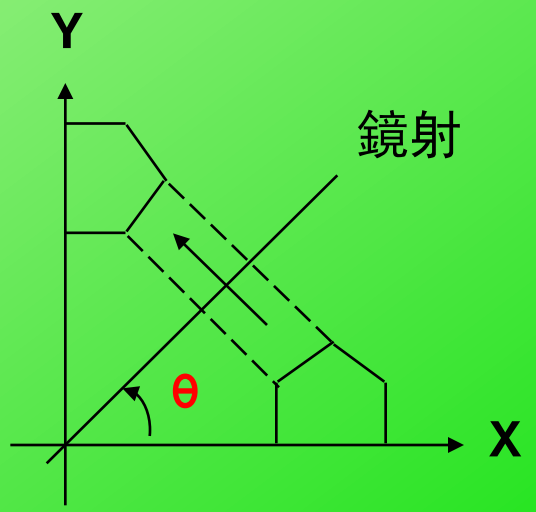
- 鏡射：鏡射軸角度 θ

$$X = \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ \sin\theta & -\cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$= M_f\, x$$

這裡 M_f =

$$\begin{pmatrix} \cos\theta & \sin\theta & 0 \\ \sin\theta & -\cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



轉換矩陣（一）

- 轉換矩陣：

  轉換矩陣乘上搬動前點的均勻座標即可求得在搬動後的座標

transformation matrix

轉換矩陣（二）

- X_0 旋轉 45° 後對 Y 軸鏡射，再往上平移 5

$$x_1 = M_r x_0 \quad x_2 = M_f x_1 \quad x_3 = M_t x_2$$

$$M_r = \begin{pmatrix} \cos 45^\circ & -\sin 45^\circ & 0 \\ \sin 45^\circ & \cos 45^\circ & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad M_t = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{pmatrix}$$

$$M_f = \begin{pmatrix} \cos 90^\circ & \sin 90^\circ & 0 \\ \sin 90^\circ & -\cos 90^\circ & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

轉換矩陣（三）

$$x_3 = M_t x_2$$
$$= M_t M_f x_1$$
$$= M_t M_f M_r x_0$$
$$= M x_0$$

其中 $M = M_t M_f M_r$

$$M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos 90^\circ & \sin 90^\circ & 0 \\ \sin 90^\circ & -\cos 90^\circ & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos 45^\circ & -\sin 45^\circ & 0 \\ \sin 45^\circ & \cos 45^\circ & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 5 \\ 0 & 0 & 1 \end{pmatrix}$$

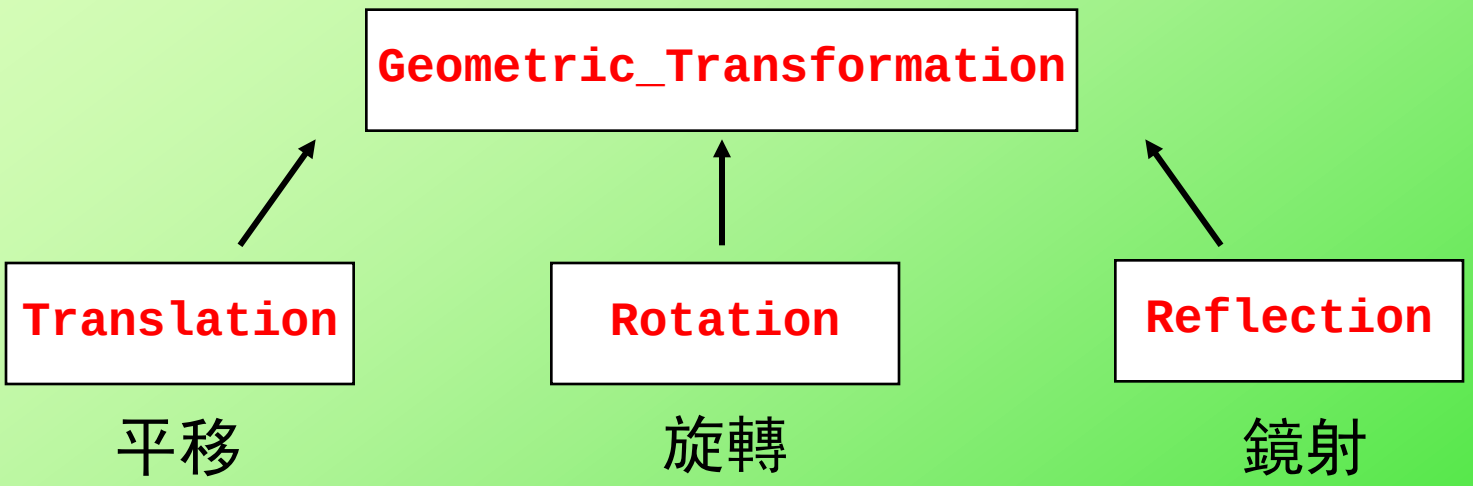
轉換矩陣（四）

$$X_3 = \begin{pmatrix} X_3 \\ Y_3 \\ 1 \end{pmatrix} = Mx_0 = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 5 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix}$$

- 轉換矩陣 **M** 的第三列一直保持不變，因此在程式設計上 **M** 可以定義為 **2x3** 的矩陣

轉換基礎類別（一）

- 平移、旋轉與鏡射「都是一種」幾何轉換類別



- **Geometric_Transformation** 並無任何實體，故定義其為一抽象類別

轉換基礎類別（二）

- 基本形式

```
// 設定轉換類別：平移、旋轉、鏡射
enum  Transformation_Type  { TRA , ROT , REF } ;

// 幾何點轉換：抽象基礎類別
class  Geometric_Transformation {
   protected :
      double m[2][3] ;  // 轉換矩陣
   public :
      // 計算新的轉換矩陣 A = M A 兩轉換矩陣乘積
      void update_transformation_matrix( double a[2][3] ) ;

      // 回傳轉換的類型，為一純虛擬函式
      virtual Transformation_Type get_transformation_type()
         const = 0 ;
} ;
```

轉換基礎類別（三）

- 平移衍生類別

```
class Translation : public Geometric_Transformation {
   private :
      Distance dx , dy ;  // 儲存 x 與 y 的平移距離
   public :
      // 設定轉換矩陣
      Translation( Distance x , Distance y ) : dx(x) , dy(y) {
         m[0][0] = 1. ; m[0][1] = 0, ; m[0][2] = dx ;
         m[1][0] = 0. ; m[1][1] = 1. ; m[1][2] = dy ;
      }
      virtual Transformation_Type get_transformation_type()
         const {
         return Transformation_Type( TRA ) ;
      }
} ;
```

平面座標點類別

- 平面點類別

```
class Point {
   private :
      double x , y ;      // 平面點的座標
   public :
      Point(){}
      Point( double a , double b ) : x(a) , y(b) {}
      // 回傳在平移，旋轉，鏡射後的座標點
      Point rotation( Angle angle ) const ;
      Point reflection ( Angle angle ) const ;
      Point translation( Distance dx , Distance dy ) const ;

      // 回傳座標點經過一連串的幾何轉換的座標
      Point transformation(
            const vector<Geometric_transformation*>& gt ) const ;
} ;
```

平面點座標轉換

- 一平面點可以經過一連串的轉換動作搬動到某處

```
Point Point::transformation(
      const vector<Geometric_transformation*>& gt ) const {

   // 設定初始的轉換矩陣為單位矩陣
   double m[2][3] = { {1,0,0} , {0,1,0} } ;

   // 重複計算每次幾何轉換後的轉換矩陣 m
   for( int i = 0 ; i < gt.size() ; ++i ){
      gt[i]->update_transformation_matrix(m) ;
   }

   //  將轉換矩陣乘上原始點座標位置後 產生新點輸出
   return
      Point( m[0][0]*pt.getx() + m[0][1]*pt.gety() + m[0][2] ,
             m[1][0]*pt.getx() + m[1][1]*pt.gety() + m[1][2] ) ;
}
```

平面點移動

- 輸入格式：

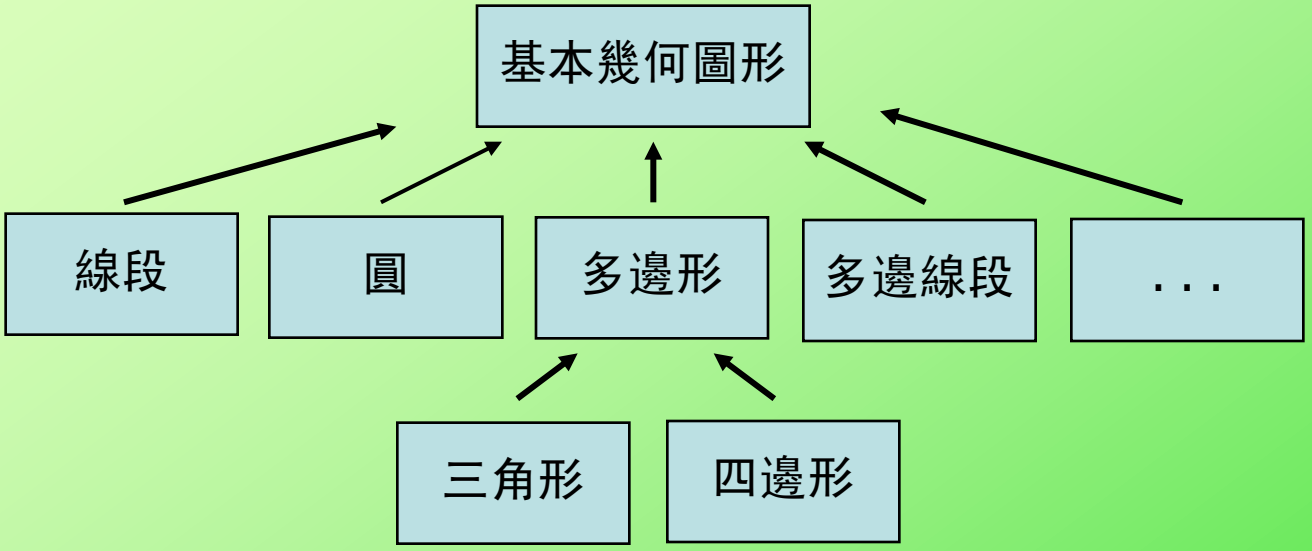
點座標　　轉換次數　　個別轉換的方式與其相關資料

```
1 1        3              TRA 2 , 0   ROT 90   REF 45
```

程式

輸出

基本幾何圖形資料庫



- 以執行效率考量：

  常見的三角形與四邊形也可以直接繼承自基礎類別

- 在此類別架構下，新的圖形可直接掛上而不須更改已有的任何程式碼

基本幾何圖形抽象類別

- 基本幾何圖形類別並無幾何實體，故定義為抽象類別

```
class Basic_Geometric_Entity {
   private :
      Color fgcolor , bgcolor ;   // 物體的前景與背景顏色
      Name entity_name ;          // 物體名稱
   public :
      // (1) 幾何轉換： 用來平移，旋轉，鏡射
      virtual Basic_Geometric_Entity* transformation(
          const vector<Geometric_Transformation*>& geotrs )
             const = 0 ;
      // (2) 複製物件
      virtual Basic_Geometric_Entity* clone() const = 0 ;
      // (3) 列印物件
      virtual ostream& print( ostream& out ) const = 0 ;
      // (4) 覆載輸出運算子
      friend ostream& operator<< (
             ostream& out , const Basic_Geometric_Entity& bge ){
           return bge.print(out) ;
      }
} ;
```

- 第(1)個純虛擬函式用於移動衍生類別的圖形；第(2)個純虛擬函式用於複製衍生類別物件；第(3)個純虛擬函式用於列印衍生類別物件

線段衍生類別

■ 線段衍生類別

```
class Line : public Basic_Geometric_Entity{
   private :
      Point  p , q ;  // 線段的兩個端點
   public :
      // 建構函式
      Line( const Point& a , const Point& b ) : p(a) , q(b) {}
      // 複製物件，並回傳其指標
      virtual Line* clone() const { return new Line(*this) ; }
      // 列印線段
      virtual ostream& print( ostream& out ) const {
         return out << "> Line : "
                    << "[ " << p << " , " << q << " ]" ;
      }
};
```

虛擬複製建構函式（一）

- 虛擬複製建構函式：
  帶有複製建構函式特性的一般虛擬成員函式

```
class Basic_Geometric_Entity {
   ...
   // 複製物件
   virtual Basic_Geometric_Entity* clone() const = 0 ;
};

class Line : public Basic_Geometric_Entity {
   ...
   // 使用 Line 類別自己的複製建構函式來複製物件，並回傳其指標
   virtual Line* clone() const { return new Line(*this); }
};
```

virtual copy constructor

虛擬複製建構函式（二）

- 使用方式

```
// 定義兩個各包含 10 個指標的陣列
Basic_Geometric_Entity  *foo[10] ,  *bar[10] ;

// 動態產生各衍生類別物件存入 foo 陣列
foo[0] = new Line(...) ;
foo[1] = new Circle(...) ;
...

// 複製 foo 指標陣列的每個物件到 bar 指標陣列
for( int i = 0 ; i < 10 ; ++i )
   bar[i] = foo[i]->clone() ;
```

❖ 同樣的方式也可以應用於移除動態空間的解構函式

非成員函式虛擬化（一）

- 若要列印類別架構內的衍生類別物件

```
for ( int i = 0 ; i < 10 ; ++i ) cout << *foo[i] << endl ;
```

若定義 **operator<<** 為

```
// 錯誤的設計方式
friend ostream& operaotr<< ( ostream& out ,
                      const Basic_Geometric_Entity& bge ){
   return out << bge ;
}
```

上式將會將 **operator<<** 列印函式變成遞迴函式引發錯誤

非成員函式虛擬化（二）

- 列印函式虛擬化

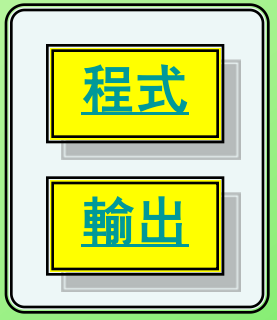
```
// 列印基本圖型
class Basic_Geometric_Entity {

   ...
   // 列印物件
   virtual ostream& print(ostream& out) const = 0 ;

   // 覆載輸出運算子
   friend ostream& operator<<
     ( ostream& out , const Basic_Geometric_Entity& bge ){
       return bge.print(out) ;
   }

} ;
```

非成員函式虛擬化（三）

- 列印線段

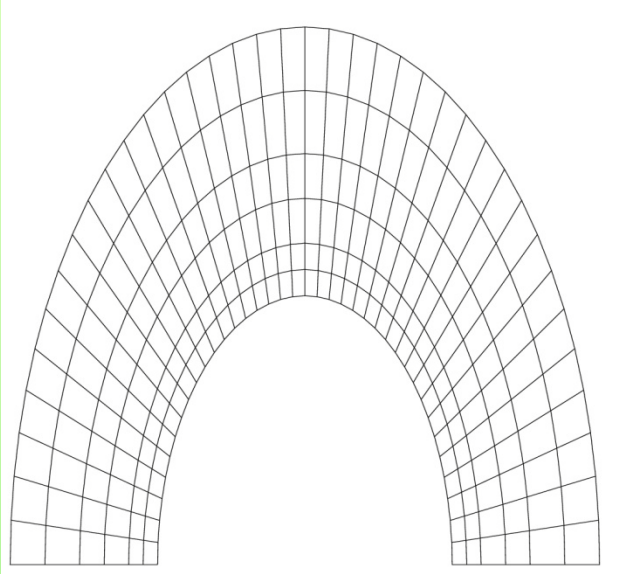
```
// 線段類別
class Line : public Basic_Geometric_Entity {

   private :

      Point p , q ;   // 線段的兩個端點

   public :
      ...
      // 列印線段
      virtual ostream& print( ostream& out ) const {
          return out << "> Line : "
                     << "[ " << p << " , " << q << " ]" ;
      }
} ;
```

基本幾何圖形資料庫

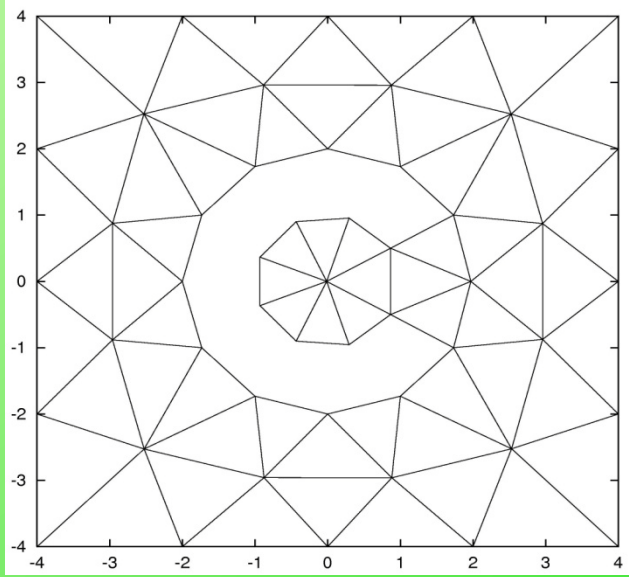


簡易網格生成法

- 結構性網格



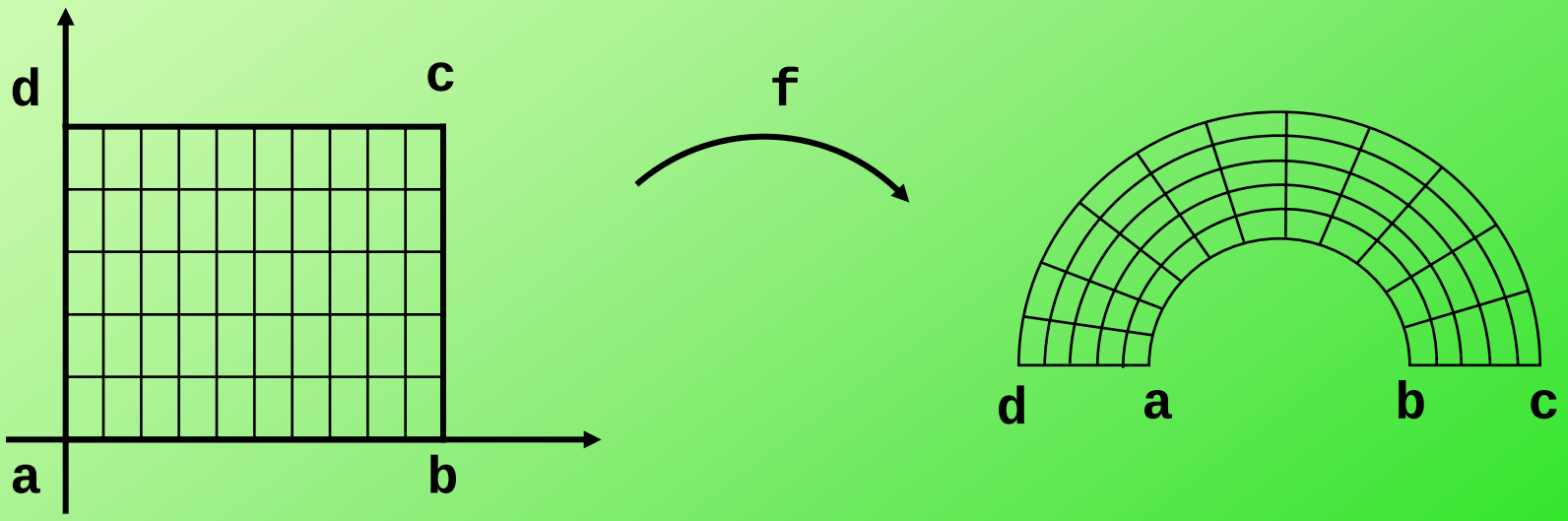
- 非結構性網格



結構性網格

- 結構性網格：

  可透過數學函式將在 [0,1]x[0,1] 的方形區域的簡單格子 ， ——映射到較複雜的幾何區域內

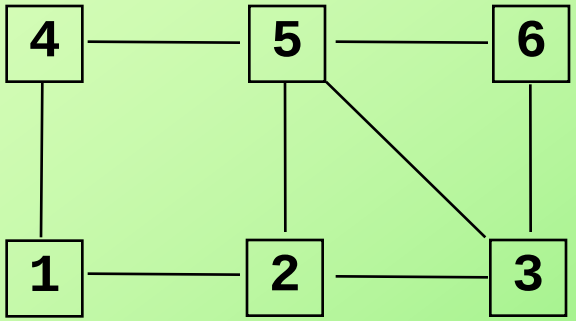


gunplot 繪圖軟體（一）

- 最普遍的科學繪圖程式
- 繪圖功能強大且多樣，同時也是免費
  - 程式網址 (www.gnuplot.org)

gunplot 繪圖軟體（二）

- 網格範例



網格與格點

```
1 : ( 0 , 0 )  2 : ( 1 , 0 )
3 : ( 2 , 0 )  4 : ( 0 , 1 )
5 : ( 1 , 1 )  6 : ( 2 , 1 )
```

格點座標

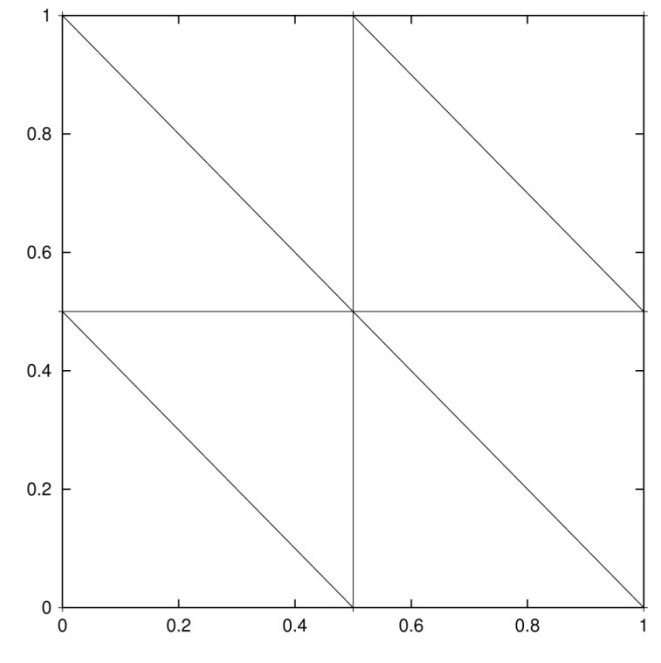
gunplot 繪圖軟體（三）

■ gunplot 檔案儲存格式

```
# 四邊形格子 1254
0   0   # 點 1
1   0   # 點 2
1   1   # 點 5
0   1   # 點 4
0   0   # 點 1

# 三角形格子 235
1   0   # 點 2
2   0   # 點 3
1   1   # 點 5
1   0   # 點 2
```

- ❖ 繪圖指令 plot ”檔名” with linespoint
- ❖ gunplot 以 ’#’ 代表註解，相當於 C++ 的 //

簡易網格生成法



程式

輸出