

Lab 2

畫面設計與元件使用

本節目的：

- 了解 Android 的畫面設計方式。
- 了解 Android 中主要的三種 Layout 布局。
- 了解如何使用 Android 布局元件。

2.1 版面配置

一個基本的 APP 至少會具備一個畫面來與使用者互動，在 Android 中我們將透過「Xml」語法去描述一個畫面的版面布局，這類檔案我們稱之為「Layout」。

圖 2-1 左下方的「Design/Text」中，Design 是設計畫面的圖形介面，Text 代表相對應的程式畫面的 Xml。

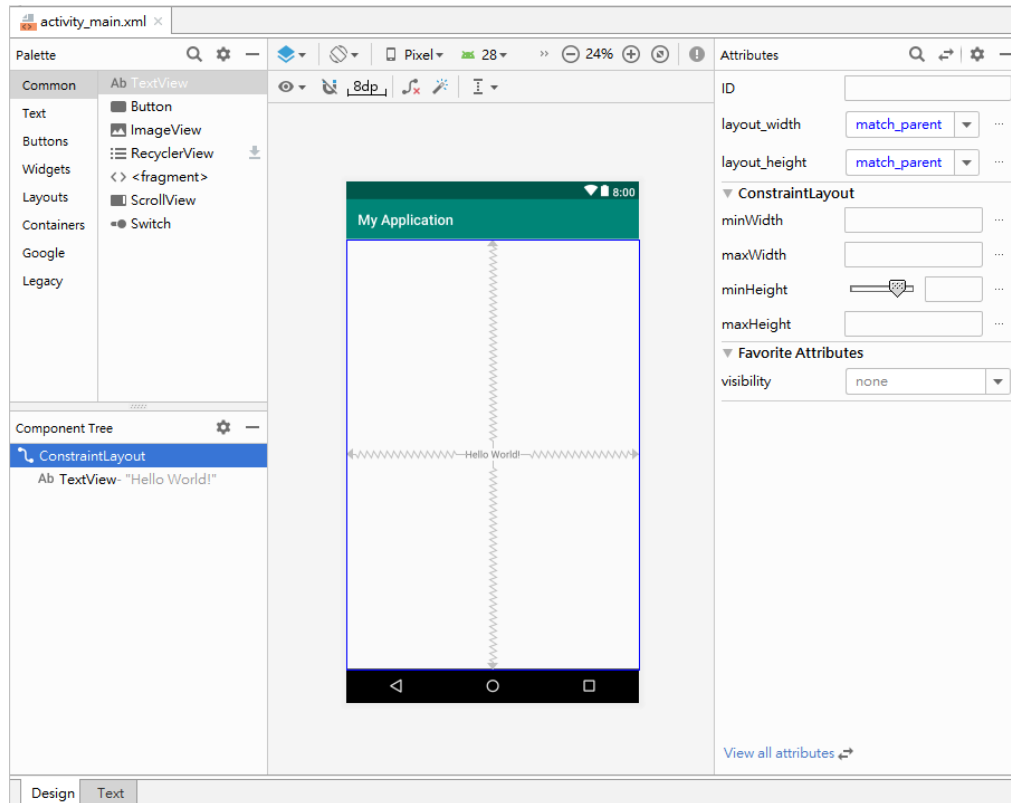


圖 2-1 Android Studio 布局畫面

2.1.1 設計畫面介紹

Layout 檔會被放置在「res/layout」目錄之下，將其打開後，可見到圖 2-2 中的預覽畫面，左側為調色盤/元件盤（palette），我們可以從中挑選 Layout 或是元件直接放到中間的預覽畫面中或是圖 2-4 的元件樹中。

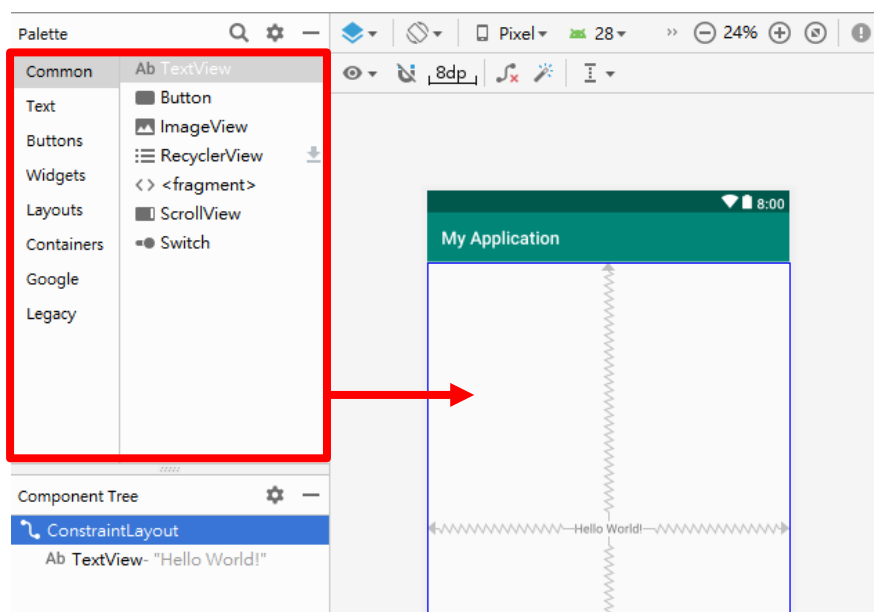


圖 2-2 元件盤位於布局畫面左上方

已經在預覽畫面中的元件可直接點擊，圖 2-3 右方的欄位會顯示所點擊到的元件的屬性表，可以透過屬性表直接更改該元件的相關資訊內容。

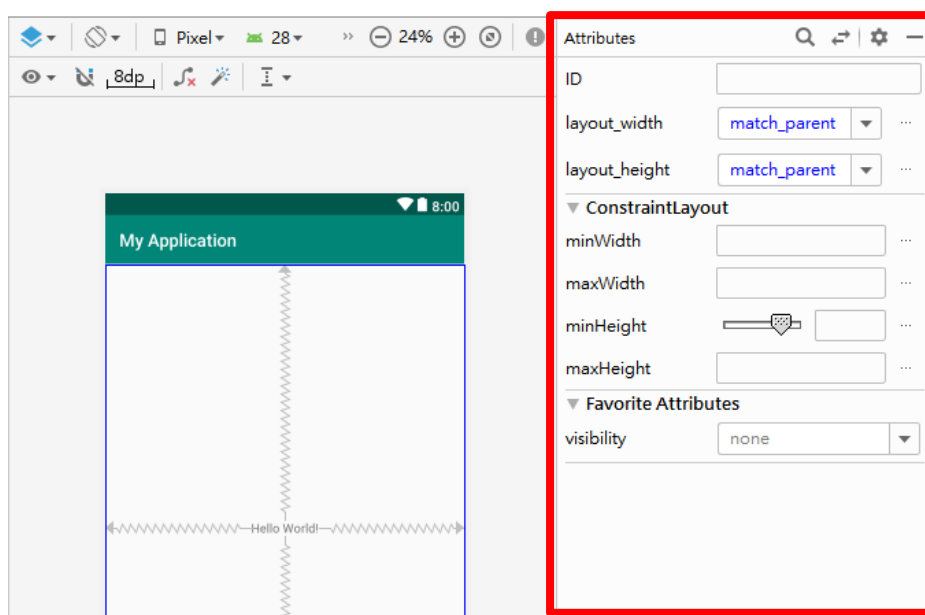


圖 2-3 元件屬性表位於布局畫面右上方

圖 2-4 左下角會顯示畫面對應的元件樹（Component Tree），元件樹可以讓我們知道元件之間的定位關係，透過樹狀關係來描述其位置。

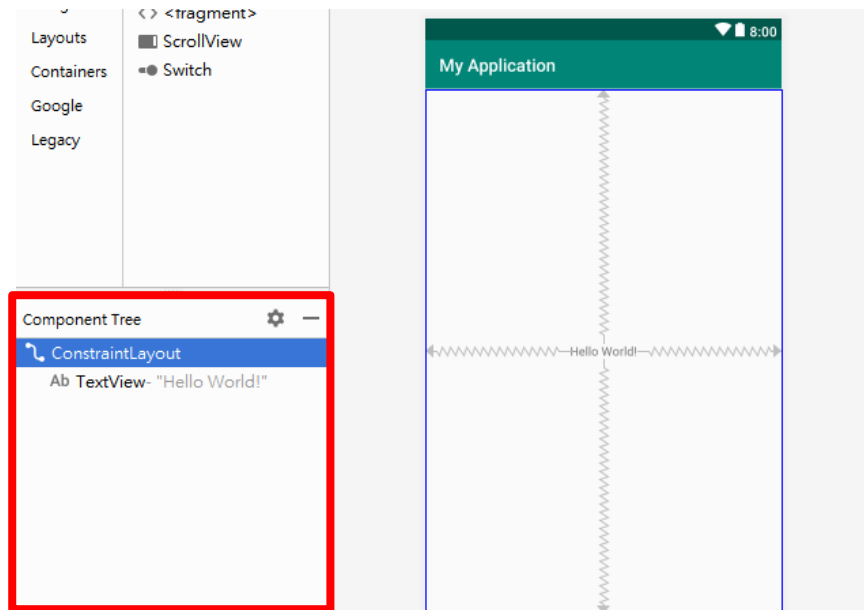


圖 2-4 元件樹位於元件盤的下方

說明

元件盤的元件可直接拖曳到預覽畫面或元件樹中，而當畫面的設計較複雜時則建議直接拖曳至元件樹中，元件樹可以明確的表示元件層級位置，要避免直接拖到預覽畫面，造成元件的層級位置擺放錯誤，細節的層級關係會在後面做介紹。

由於 Layout 檔是「Xml」格式，可以透過圖 2-5 左下角的「Text」按鈕切換查看 Xml 程式碼，「Design」按鈕切換查看設計畫面的圖形介面。

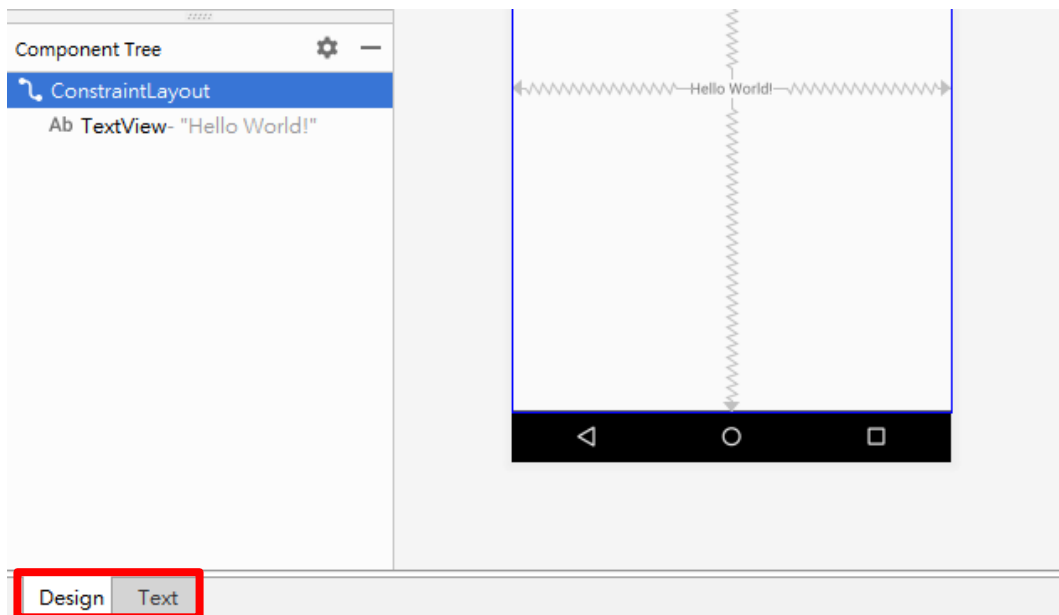


圖 2-5 按鈕切換布局模式

切換至「Text」後，我們可以看到其原始碼格式。

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>
</android.support.constraint.ConstraintLayout>
```

2.1.2 Layout－版面布局

在畫面編排時最重要的便是要決定每一個元件在畫面上的位置。若要使用元件被 layout 所控制，我們需要把元件放到 layout 之內，形成父子層級的關係，如圖 2-6 所示。

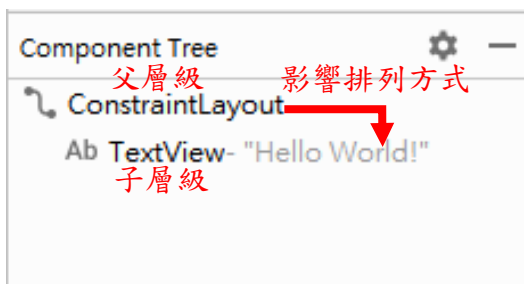


圖 2-6 元件的層級排序

子層級的元件排列方式就會被父層級的 layout 所影響，當然子層級也可以擺放新的 layout，這樣「孫」層級就會同時受到子層級與父層級影響，而 Android 基本提供三種主要定位方式來對畫面進行布局。

- **LinearLayout**：顧名思義就是依照順序逐一排列介面元件，也是最常被使用的布局方式。依照方向（orientation）的不同 LinearLayout 分為垂直排列（vertical）與水平排列（horizontal）兩種呈現方式，透過 orientation 屬性切換，如圖 2-7 所示。

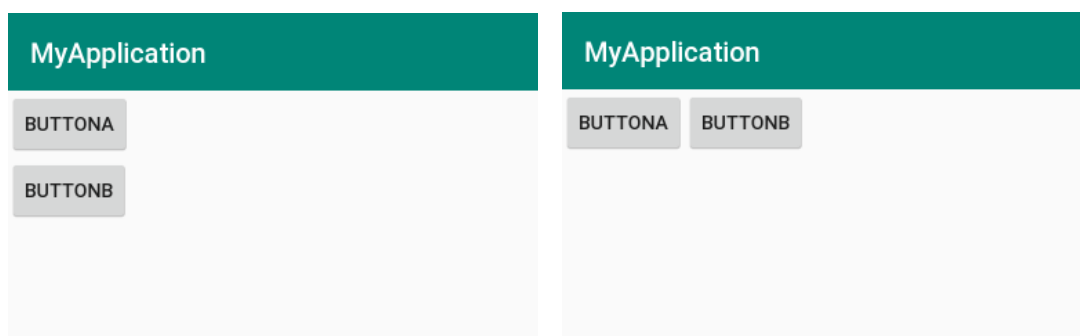


圖 2-7 垂直布局 vertical（左）與水平布局 horizontal（右）

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
```

Orientation 可以決定 LinearLayout 是 vertical 或 horizontal，如果沒有此屬性預設是 vertical。

而編排布局上除了三種布局方式之外，每個畫面元件的屬性中也可以用 layout:width 與 layout:height 設定本身的寬與高。

```
android:layout_width="match_parent"
android:layout_height="match_parent"
```

width 與 height 可以寫一個固定數值強制定義大小，由於裝置大小不同，建議要能以裝置尺寸做動態調整。

- match_parent：元件的寬度與高度擴展到最大，但最大只能等同父層級的大小，如圖 2-8 所示。

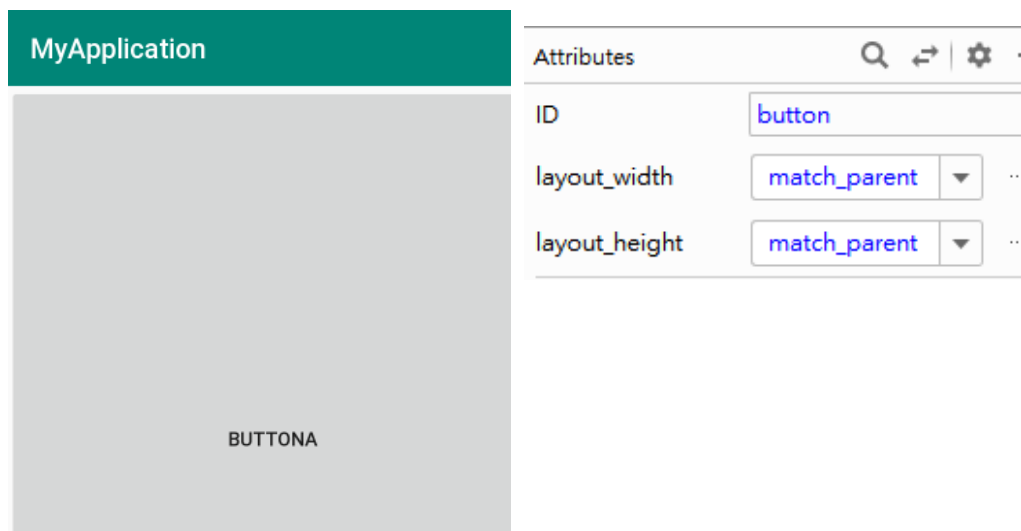


圖 2-8 最大化按鈕顯示

- wrap_content：元件的寬度與高度依據內容自動調整，而內容定義包含文字、圖片或子層級，如圖 2-9 所示。

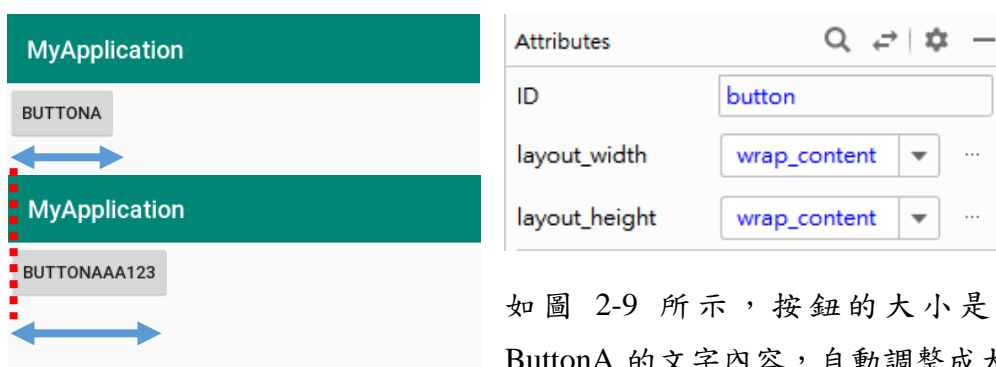


圖 2-9 最小化按鈕顯示

如圖 2-9 所示，按鈕的大小是依照 ButtonA 的文字內容，自動調整成大小。

- **FrameLayout**：FrameLayout 是以堆疊方式呈現，如果使用這種介面布局檔，子層級的元件皆會重疊，重疊順序會依照元件樹，下面的元件會覆蓋上面的元件。圖 2-10 中，ButtonB 覆蓋於 ButtonA 上，可看到元件樹中 ButtonB 排列在 ButtonA 下。

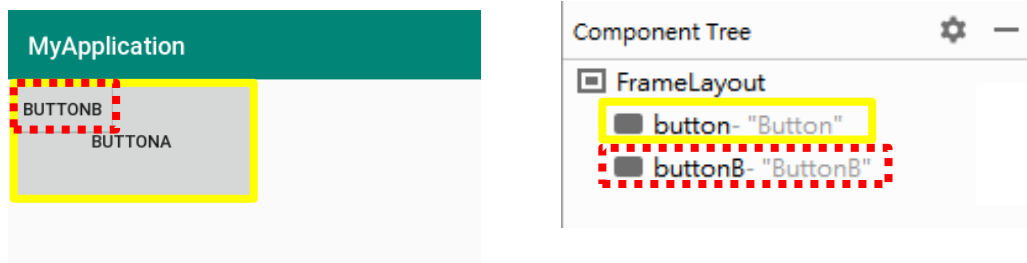


圖 2-10 按鈕 B 覆蓋於按鈕 A 上

- **ConstraintLayout**：ConstraintLayout 是 Android Studio 2.2 中新加入的布局，結合了 FrameLayout 與 RelativeLayout 的特性，可以有效地解決布局層級過多的問題。ConstraintLayout 採用堆疊的方式呈現，定位上需要明確描述參考的對象，以及與該對象的具體距離單位量，否則元件會以畫面左上角作為基準。與傳統 Layout 不同，ConstraintLayout 非常適合使用圖形化的方式來編輯介面。

Step1 設定邊框範圍，上下左右的 padding 各為 16dp，如圖 2-11 所示，實線框是 ConstraintLayout 的尺寸，上方程式碼設定 ConstraintLayout 中內容到 ConstraintLayout 邊緣的距離（實線到虛線的距離為 16dp），虛線框的範圍為可放置元件的區域。

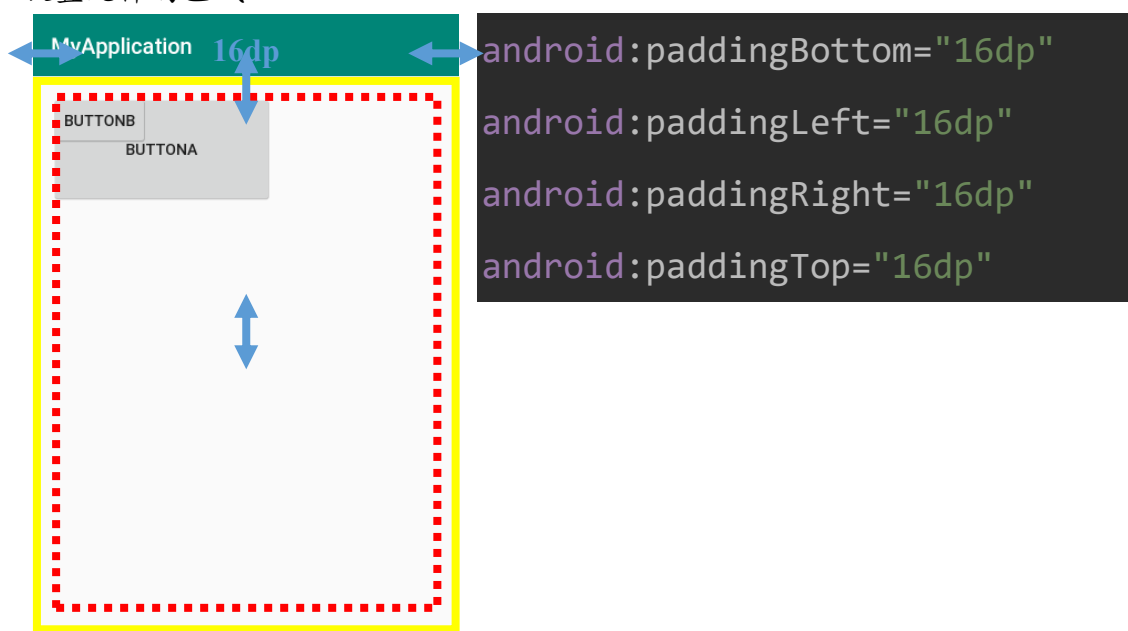
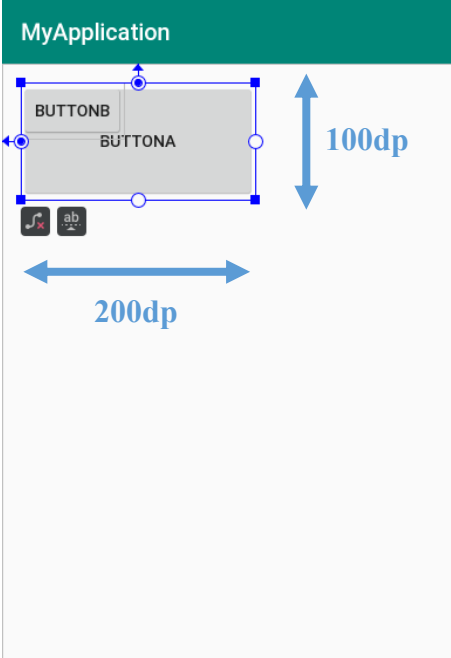


圖 2-11 ConstraintLayout 布局

Step2 點擊 ButtonA 後可以看見元件四周的基準點，透過滑鼠拖曳基準點，可讓元件對齊畫面邊緣或是其它元件，設定 ButtonA 對齊畫面上緣與左側，如圖 2-12 所示。



```
<Button
    android:id="@+id/btn_A"
    android:layout_width="200dp"
    android:layout_height="100dp"
    android:text="ButtonA"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

圖 2-12 拖曳按鈕 A 對齊畫面

Step3 將 ButtonB 的基準點對齊 ButtonA 的右側與下緣，並設定距離 16dp，如圖 2-13 所示。

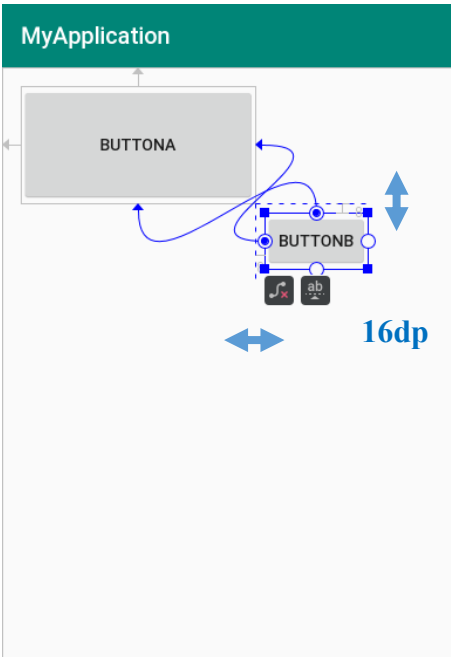


圖 2-13 的實線框代表 ButtonB 的位置，下方為程式碼設定：

- marginTop 設定距離上緣基準 16dp。
- marginStart 設定距離左側基準 16dp。
- ButtonB 的上緣對齊 ButtonA 的下緣。
- ButtonB 的左側對齊 ButtonA 的右側。

圖 2-13 拖曳按鈕 B 對齊按鈕 A

```
<Button
    android:id="@+id/btn_B"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="ButtonB"
    距離上緣基準點 16dp
    android:layout_marginTop="16dp"
    距離左側基準點 16dp
    android:layout_marginStart="16dp"
    以 ButtonA 的下緣作為 ButtonB 的上緣基準點
    android:layout_constraintTop_toBottomOf="@id/buttonA"
    以 ButtonA 的右側作為 ButtonB 的左側基準點
    android:layout_constraintStart_toEndOf="@id/buttonA"/>
```

2.1.3 Widget—視窗元件

了解如何編排元件的位置後，下一步要了解元件盤中的元件（Widget），元件可於圖 2-14 左側的原件盤中選擇。

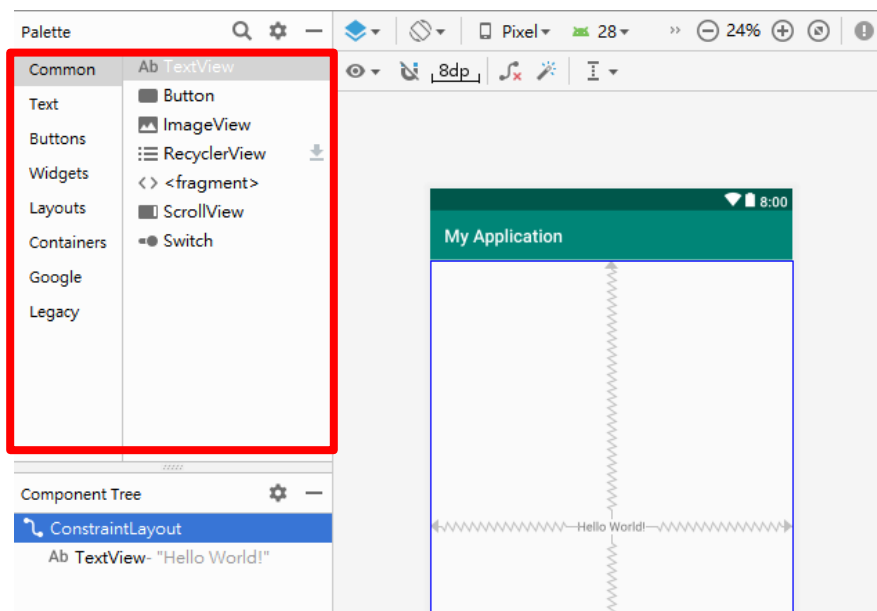


圖 2-14 元件盤

下列簡單介紹四種常用的元件：

- TextView：顯示文字的文字元件，如圖 2-15 所示。

<TextView

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Hello Word"      顯示的文字
android:id="@+id/textView"    元件 id
android:textColor="#ff0000"   文字顏色
android:background="#ffff00"  背景顏色
android:textSize="20dp" />   文字大小
```



圖 2-15 TextView 範例

- Button：觸發點擊事件的按鈕元件，自帶有額外有按下的動畫，如圖 2-16 所示。

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    按鈕上顯示的內容
    android:text="New Button"
    android:id="@+id/button" />
```

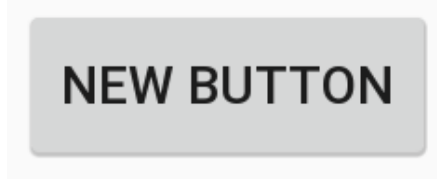


圖 2-16 Button 範例

- EditText：可輸入訊息的輸入元件。當 EditText 被點擊後會自動彈出小鍵盤讓使用者對「android:text」做輸入，如圖 2-17 所示。

```
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/editText"
    沒有文字輸入時顯示提示訊息
    android:hint="請輸入電話號碼"
    輸入類型[phone:只能輸入 0~9]
    android:inputType="phone" />
```



圖 2-17 EditText 範例

- RadioGroup 與 RadioButton：單選框的群體與單選元件。一個 RadioGroup 中可放置多個 RadioButton，如圖 2-18 所示。

checked 屬性決定該 RadioButton 為開啟或是關閉狀態（預設為 false）。若透過使用者去點擊其中一個 RadioButton 並將其變為 true（開啟），而其他的 RadioButton 會被應用程式給轉為 false（關閉）。

```
<RadioGroup
    android:layout_width="wrap_parent"
    android:layout_height="wrap_parent">

    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="AAA"
        android:id="@+id/radioButton"
        按鈕點擊狀態，預設為 false
        android:checked="true" />

    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="BBB"
        android:id="@+id/radioButton2" />
</RadioGroup>
```



圖 2-18 RadioGroup 範例

2.2 猜拳遊戲畫面設計

- 使用 ConstraintLayout 實作圖 2-19 的畫面布置。



圖 2-19 猜拳遊戲預覽畫面（左）與布局元件樹（右）

說明

觀察布局與畫面之間的關聯性，可以注意到元件的擺放是受到 layout 影響的。因此需要去理解 layout 的用法。

2.2.1 元件布局與排版

Step1 開啟 activity_main.xml 檔，由於預設 Layout 會有一個「Hello World!」文字的 TextView 元件，此處需要我們手動將它刪除，如圖 2-20 所示。

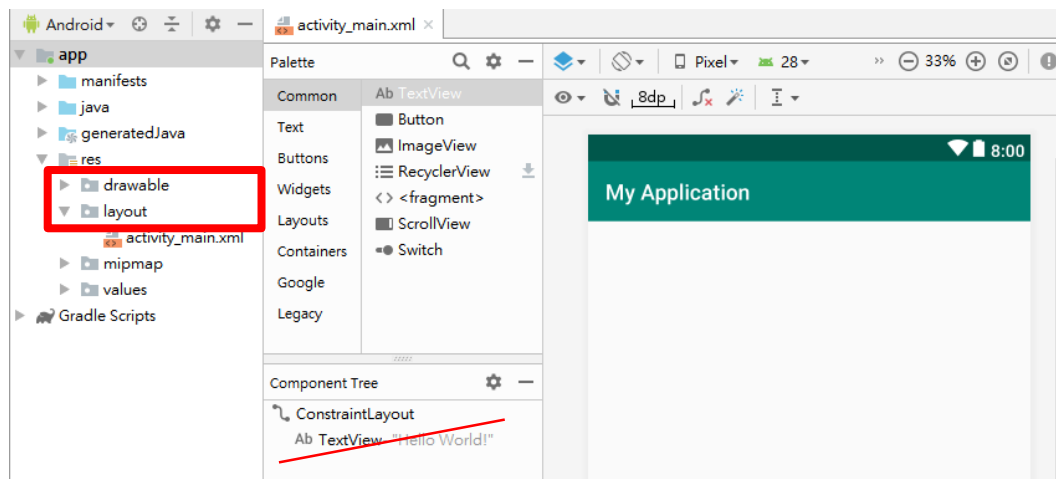


圖 2-20 移除預設的 TextView

Step2 在左上方搜尋框輸入「Text」，將 TextView 與 EditText 拉入 Component Tree 中，並將 TextView 對齊畫面上緣與左側，而 EditText 則對齊 TextView 左側與下緣，如圖 2-21 所示。

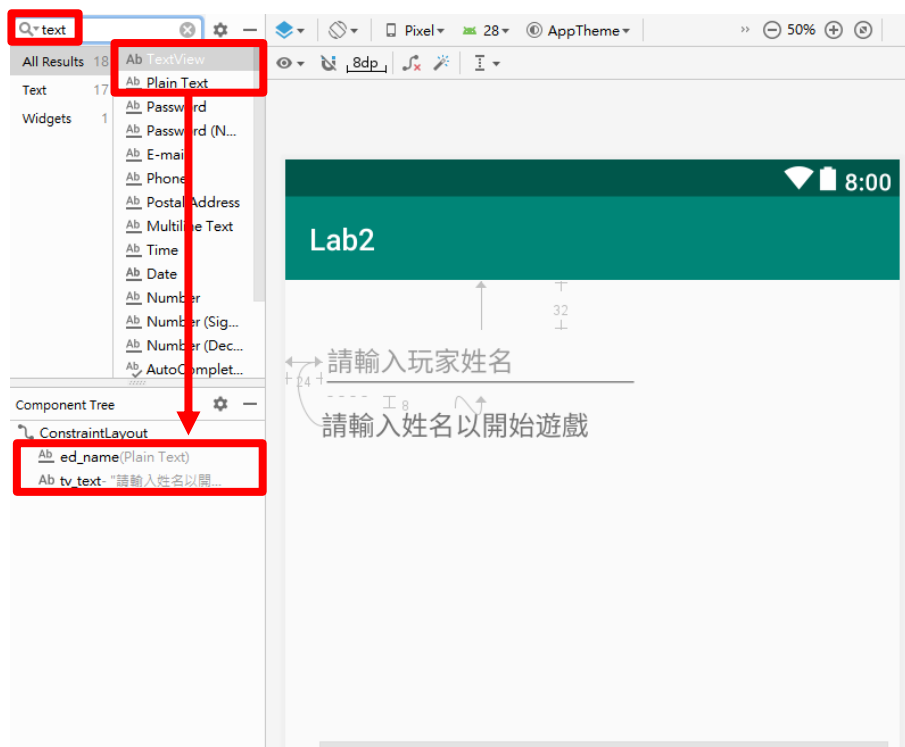


圖 2-21 加入顯示文字與輸入框

元件放置完成後，系統會產生對應的 Xml，之後額外可修改 id 與增加屬性。

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/ed_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="32dp"
        android:ems="10"
        android:hint="請輸入玩家姓名"
        android:inputType="textPersonName"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/tv_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="請輸入姓名以開始遊戲"
        android:textSize="18sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintStart_toStartOf="@+id/ed_name"
        app:layout_constraintTop_toBottomOf="@+id/ed_name" />
</android.support.constraint.ConstraintLayout>
```


Step3 在 TextView 下方加入 RadioGroup 元件，並在 RadioGroup 中增加三個 RadioButton，如圖 2-22 所示。

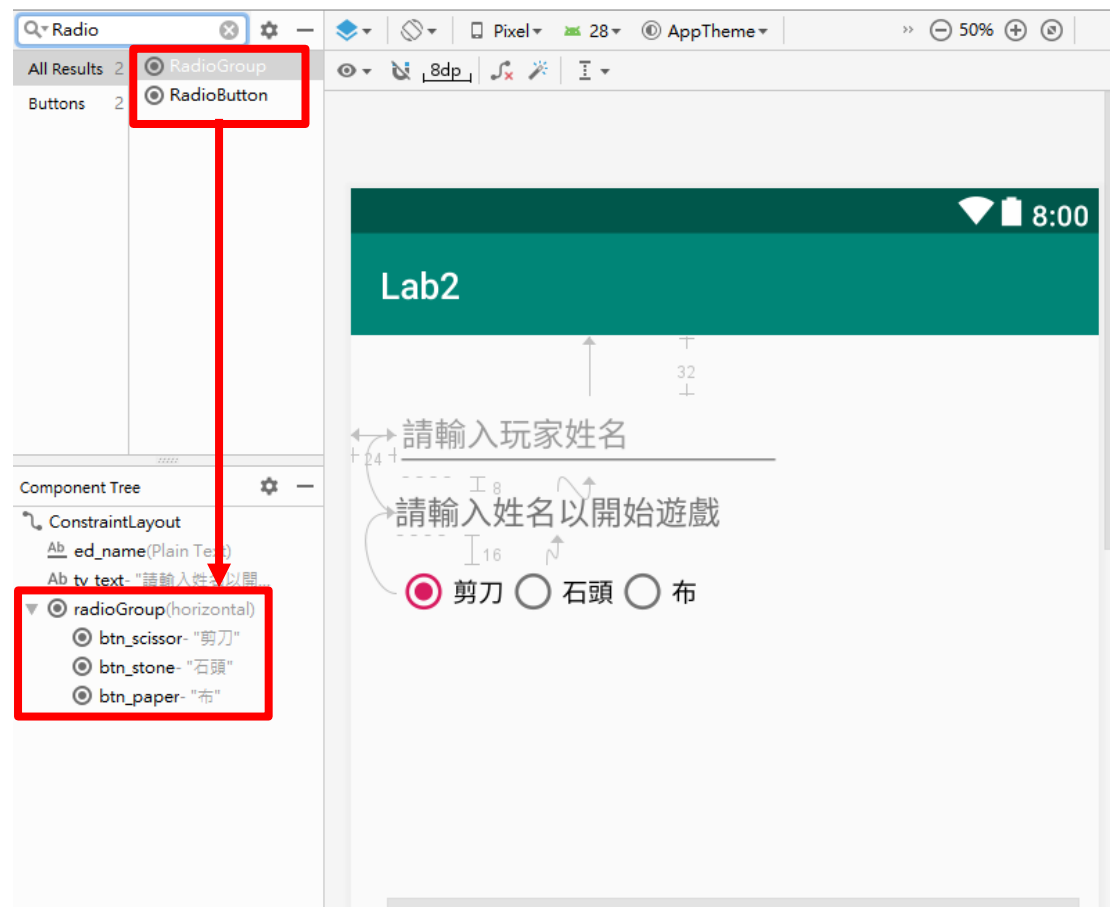


圖 2-22 加入 RadioButton

```
<RadioGroup
    android:id="@+id/radioGroup"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:orientation="horizontal"
    app:layout_constraintStart_toStartOf="@+id/tv_text"
    app:layout_constraintTop_toBottomOf="@+id/tv_text">

    <RadioButton
        android:id="@+id/btn_scissor"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:checked="true"
        android:text="剪刀" />

    <RadioButton
```

checked 設定按下狀態
text 設定顯示內容

```

        android:id="@+id/btn_stone"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="石頭" />

<RadioButton
    android:id="@+id/btn_paper"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="布" />
</RadioGroup>

```

Step4 將 Button 與 TextView 放入 RadioGroup 的下方，如圖 2-23 所示。

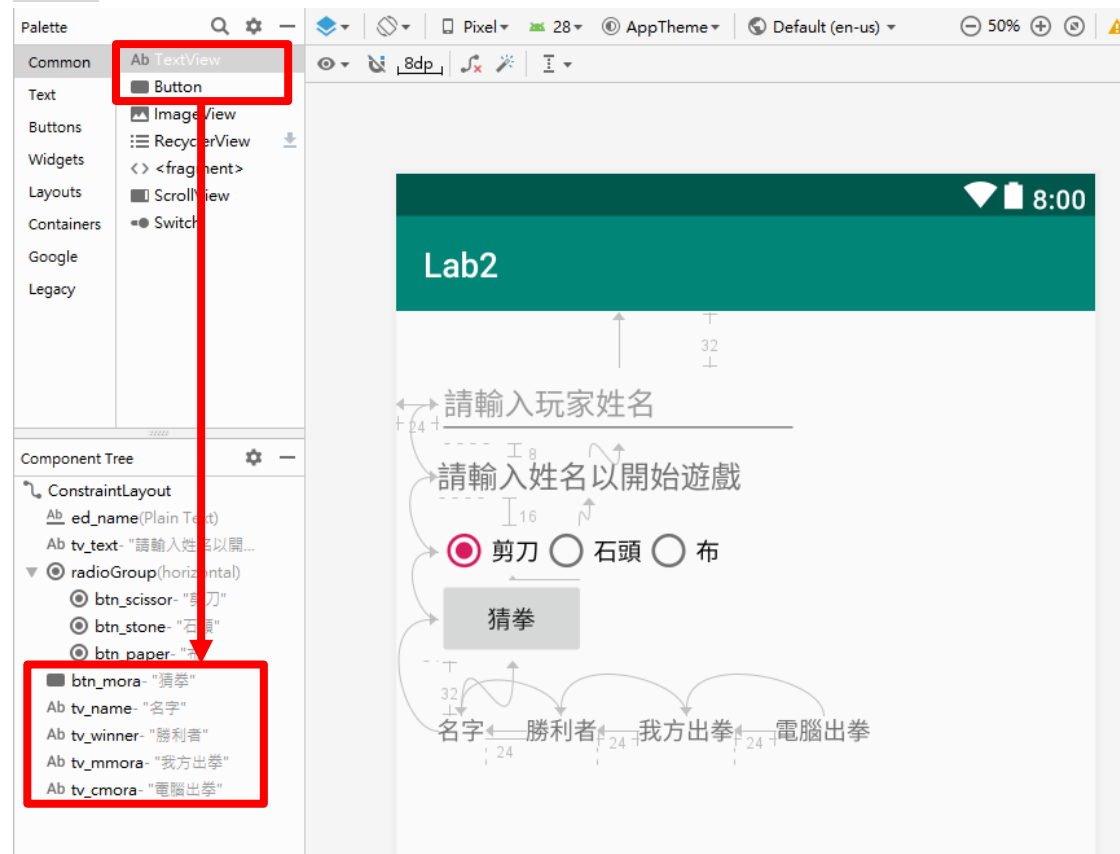


圖 2-23 放入猜拳鈕與狀態文字

<Button

```
    android:id="@+id/btn_mora"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="猜拳"
    app:layout_constraintStart_toStartOf="@+id/radioGroup"
    app:layout_constraintTop_toBottomOf="@+id/radioGroup" />
```

<TextView

```
    android:id="@+id/tv_name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="32dp"
    android:text="名字"
    app:layout_constraintStart_toStartOf="@+id/btn_mora"
    app:layout_constraintTop_toBottomOf="@+id/btn_mora" />
```

<TextView

```
    android:id="@+id/tv_winner"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:text="勝利者"
    app:layout_constraintStart_toEndOf="@+id/tv_name"
    app:layout_constraintTop_toTopOf="@+id/tv_name" />
```

<TextView

```
    android:id="@+id/tv_mmora"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:text="我方出拳"
    app:layout_constraintStart_toEndOf="@+id/tv_winner"
    app:layout_constraintTop_toTopOf="@+id/tv_winner" />
```

<TextView

```
    android:id="@+id/tv_cmora"
    android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"  
android:layout_marginStart="24dp"  
android:text="電腦出拳"  
app:layout_constraintStart_toEndOf="@+id/tv_mmora"  
app:layout_constraintTop_toTopOf="@+id/tv_mmora" />
```

Step5 完成後的畫面，如圖 2-24 所示。

The screenshot displays the 'Lab2' application interface. At the top, there is a teal header with the text 'Lab2'. Below the header, the interface is light gray. It features a text input field with the placeholder '請輸入玩家姓名'. Below this is a label '請輸入姓名以開始遊戲'. There are three radio buttons for selecting a move: '剪刀' (Scissors) with a selected red circle, '石頭' (Rock), and '布' (Paper). A gray button labeled '猜拳' (Rock-Paper-Scissors) is positioned below the radio buttons. At the bottom, there is a table with four columns: '名字' (Name), '勝利者' (Winner), '我方出拳' (My Move), and '電腦出拳' (Computer Move). The table is currently empty.

圖 2-24 猜拳遊戲實機畫面