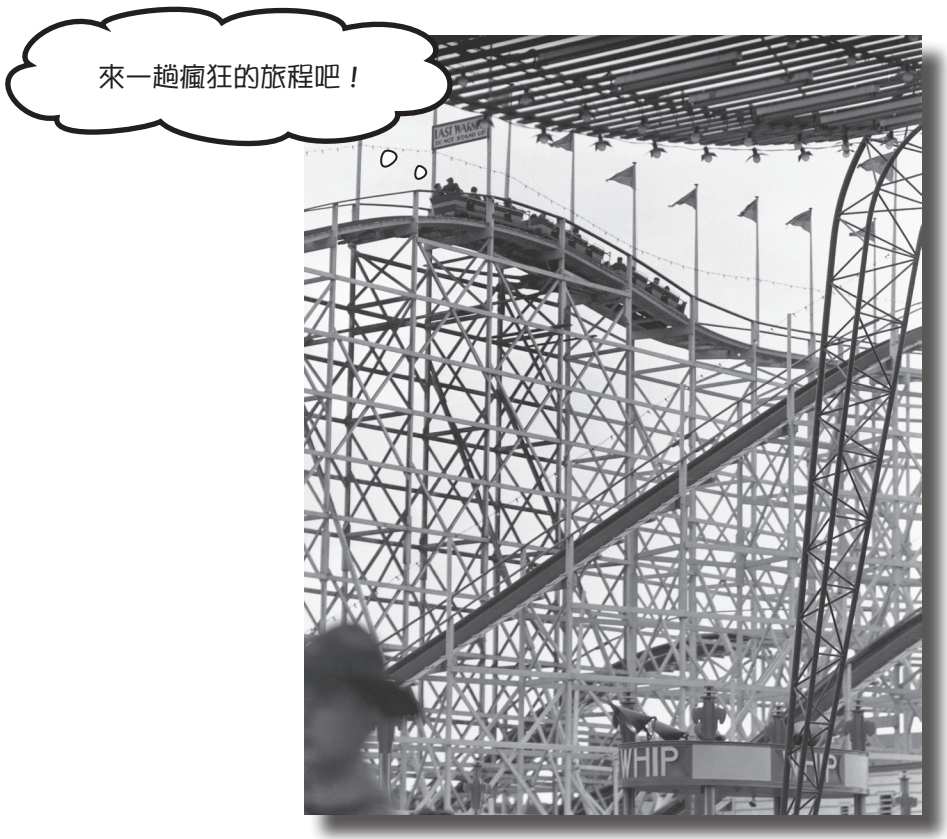
1 開始用 C# 來建構程式

做出很棒的東西…以飛快的速度！



想要寫出偉大的 app 嗎？而且立刻開始？

學會 C#，你就掌握一種**現代的程式語言**，和一種很**寶貴的工具**。而且使用 **Visual Studio** 的話，你就擁有一種**了不起的開發環境**，它具備高度直觀的功能，可讓寫程式的過程盡可能地簡單。Visual Studio 不僅是非常適合用來寫程式的工具，在探索 C# 時，也是**很有價值的學習工具**。聽起來很誘人？現在就翻到下一頁，開始寫程式吧。

為什麼你要學 C#

C# 是一種簡單、現代的語言，可讓你做很多了不起的事情。當你學 C# 時，你學的不是只有一種語言，C# 可以打開整個 .NET 世界的大門，.NET 是一種非常強大的開放原始碼平台，可讓你建構各式各樣的應用程式。

請安裝 Visual Studio，而不是 Visual Studio Code。

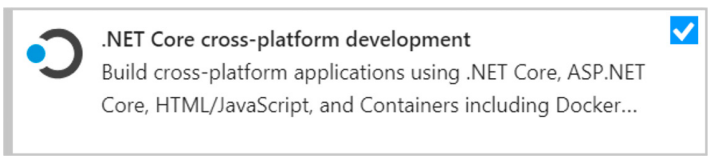
Visual Studio Code 是很了不起的開放原始碼、跨平台的程式編輯器，但是 Visual Studio 是為 .NET 開發量身訂做的，Visual Studio Code 不是，這就是為什麼我們在這本書裡，將 Visual Studio 當成學習和探索的工具。

Visual Studio 是通往 C# 的大門

如果你還沒有安裝 Visual Studio 2019，現在就是安裝它的好時機。前往 https://visualstudio.microsoft.com **下載 Visual Studio Community 版本**。（如果你已經安裝了，那就執行 Visual Studio Installer 來更新你已經安裝的項目。）

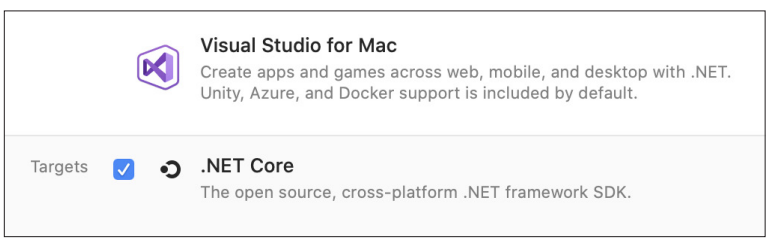
如果你使用 Windows…

務必將「.NET Core 跨平台開發（.NET Core cross-platform development）」和「.NET 桌面開發（.NET desktop development）」這兩個選項打勾，來安裝針對它們的支援。但是不要將使用 *Unity* 進行遊戲開發（*Game development with Unity*）打勾—稍後你會用 Unity 來進行 3D 遊戲開發，但你會獨立安裝 Unity。



如果你使用 Mac…

下載並執行 Visual Studio for Mac 安裝程式，務必將 .NET Core 的 Targets 打勾。



你也可以在 Windows 進行 ASP.NET 專案！只要在安裝 Visual Studio 時，核取「ASP.NET 與網頁程式開發」選項即可。

本書大部分的專案都是 .NET Core 主控台 app，可以在 Windows 和 Mac 上面運行。有些章節裡面的專案是 Windows 桌面專案，例如本章稍後的動物配對遊戲。在遇到這些專案時，請閱讀 Visual Studio for Mac 學習指南附錄。它有完整的第 1 章替代版，以及其他 WPF 專案的 ASP.NET Core Blazor 版本。

Visual Studio 是用來編寫程式和探索 C# 的工具

雖然你可以使用 Notepad 或其他文字編輯器來撰寫 C# 程式，但是有更好的工具可以使用。**IDE**（***integrated development environment*** 的縮寫）是文字編輯器、視覺化設計器、檔案管理器、偵錯工具…很像可以在寫程式時處理任何需求的多功能工具。

Visual Studio 可以提供的協助包括：

1. **建構 app，快速地**。C# 語言很靈活也很容易學習，Visual Studio IDE 可以幫你自動完成許多手工工作，來讓你的學習更簡單。以下這些只是 Visual Studio 可以幫你做的一些事情而已：
   - ★ 管理所有專案檔案
   - ★ 讓你輕鬆地編輯專案的程式碼
   - ★ 記錄專案的圖片、音訊、圖示，還有其他資源
   - ★ 逐行執行程式來幫你進行偵錯
2. **設計很漂亮的使用者介面**。Visual Studio IDE 的 Visual Designer 是最容易使用的設計工具之一。它可以幫你做很多事情，讓你覺得製作使用者介面是開發 C# app 的過程中最滿意的部分。你不需要花好幾個小時調整使用者介面（除非你想這樣做），就可以做出功能完整的專業程式。

如果你使用的是 *Visual Studio for Mac*，你也可以建構一些很漂亮的 app，但不是使用 XAML，而是一起使用 C# 與 HTML。

3. **建構賞心悅目的程式。一起使用 C# 與 XAML**（為 WPF 桌面 app 設計使用者介面的視覺標記語言）等於使用最有效率的視覺程式創作工具之一…你可以用它來建構外觀和功能都很出色的軟體。

> 任何一種 WPF 的使用者介面（或 UI）都是用 XAML（可延伸應用程式標記語言，eXtensible Application Markup Language 的縮寫）來建構的。Visual Studio 可讓你很輕鬆地使用 XAML。

4. **學習和探索 C# 及 .NET**。Visual Studio 是一種世界級的開發工具，幸運的是，它也是一種很棒的學習工具。**我們會用 *IDE* 來探索 *C#***，以便快速地將重要的程式設計概念植入你的大腦。

本書通常會將 *Visual Studio* 簡稱為 *IDE*。

Visual Studio 是了不起的開發環境，但我們也會將它當成學習工具，用來探索 C#。

在 Visual Studio 裡面建立你的第一個專案

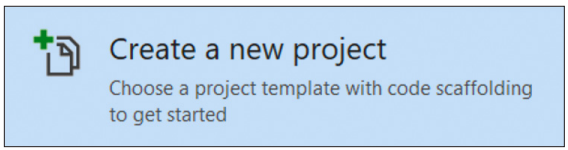
學習 C# 最好的方法就是開始寫程式，所以我們要用 Visual Studio 來**建立一個新的專案**…並且立刻開始寫程式！

動手做！

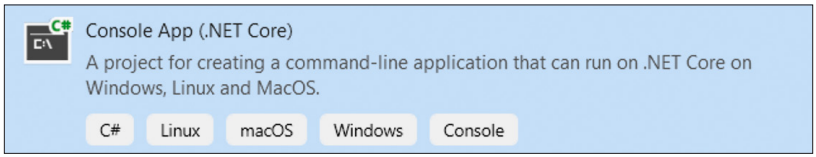
當你看到**動手做！**（或**立刻動手做！**或**解決這個 bug！**…等）時，請打開 Visual Studio，並且跟著操作。我們會告訴你該做些什麼，並指出你需要注意哪些地方，才可以從範例中學到最多東西。

1 **建立新的 Console（主控台）App（.NET Core）專案。**

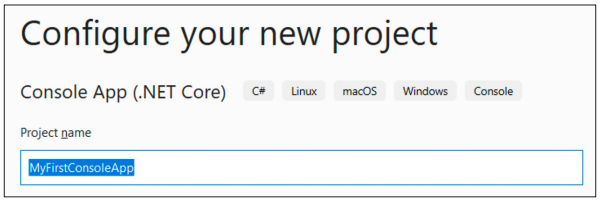
啟動 Visual Studio 2019，第一次啟動它時，它會顯示一個「Create a new project」視窗，裡面有各種不同的選項。選擇 **Create a new project**。如果你關閉這個視窗了，不用擔心，你隨時可以在選單選擇 File >> New >> Project 叫出它。



按下 **Console App (.NET Core)** 來選擇專案類型，然後按下 **Next** 按鈕。



將專案命名為 **MyFirstConsoleApp**，並按下 **Create** 按鈕。



如果你使用 **Visual Studio for Mac**，這個專案（以及本書的所有 .NET Core Console App 專案）的所有程式碼都是一樣的，不過有一些 IDE 功能會不一樣。**Visual Studio for Mac 學習指南**附錄有本章的 Mac 版本。

2 **看看新 app 的程式碼。**

用 Visual Studio 建立新專案時，它會提供一個起點來讓你開始建構程式，當它為 app 建立新檔案之後，它會打開一個稱為 *Program.cs* 的檔案，裡面有這些程式：

```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        Console.WriteLine("Hello World!");
    }
}
```

當 Visual Studio 建立新的 Console App 專案時，它會自動加入 Program 類別。

這個類別的開頭是 Main 方法，它裡面有一個陳述式，該陳述式會將一行文字寫到控制台。第 2 章會更仔細討論類別與方法。

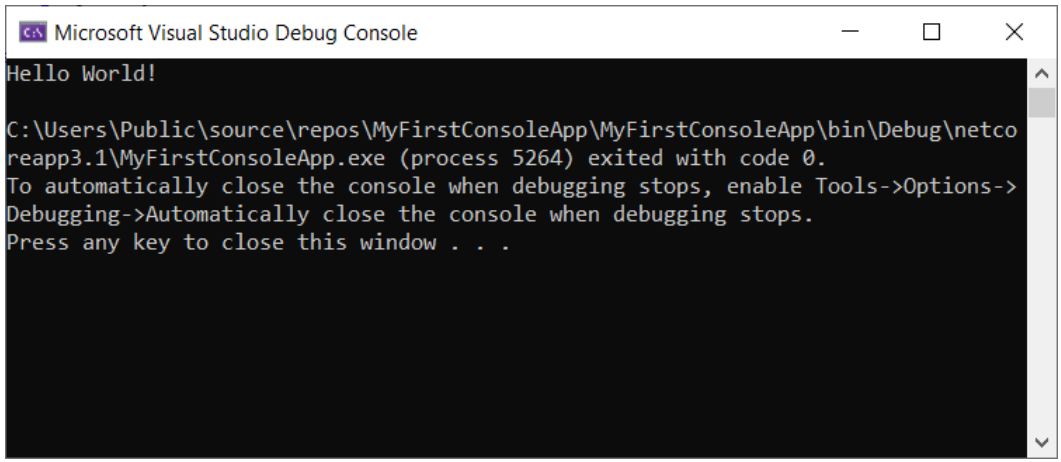
3 **執行新 app。**

Visual Studio 為你建立的 app 可以執行了。在 Visual Studio IDE 的最上面，找到有綠色三角形和你的 app 名稱的按鈕，並按下它：

4 **看看程式的輸出。**

執行程式會出現 Microsoft Visual Studio Debug Console 視窗，顯示程式的輸出：

當你執行 app 時，它會執行 Main 方法，該方法會將這一行文字寫到主控台。



學習語言最好的方法是用它來寫大量的程式，所以你會在這本書裡面建構大量的程式，它們很多都是 .NET Core Console App 專案，我們來仔細地看看你剛才做了什麼。

顯示在視窗最上面的是**程式的輸出**：

```
Hello World!
```

接著有一行空白，然後是一些其他的文字：

```
C:\path-to-your-project-folder\MyFirstConsoleApp\MyFirstConsoleApp\bin\
Debug\netcoreapp3.1\MyFirstConsoleApp.exe (process ####) exited with code 0.
To automatically close the console when debugging stops, enable Tools->
Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

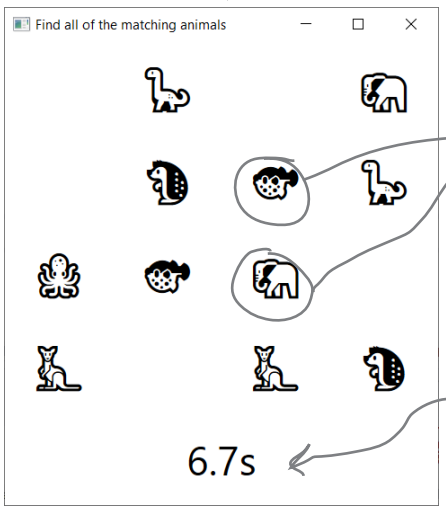
你會在每一個 Debug Console 視窗底下看到同樣的訊息。你的程式印出一行文字（`Hello World!`），然後結束。Visual Studio 會讓輸出視窗維持開啟，直到你按下一個按鈕來關閉它為止，如此一來，你就可以在視窗消失之前看到輸出。

按下一個按鍵來關閉視窗。然後再次執行你的程式。這就是執行你在整本書裡面建構的所有 .NET Core Console App 專案的方法。

我們來建構遊戲！

太棒了！你剛才已經寫出你的第一個 C# app 了！完成它之後，我們來做比較複雜的東西。我們要來做一個**動物配對遊戲**，這個遊戲會顯示一個內含 16 種動物的網格，讓玩家按下成對的動物來移除它們。

這是你要建構的動物配對遊戲。



這個遊戲會顯示 8 對不同的動物，隨機放在視窗的各個位置。玩家要按下兩種動物，如果牠們一樣，牠們就會從視窗消失。

這個計時器會記錄玩家完成遊戲的時間。遊戲的目標是用最短的時間找到所有的配對。

在你的 C# 學習工具箱裡面，建構各種不同的專案是很重要的工具。我們讓這本書的一些專案使用 WPF（即 Windows Presentation Foundation），因為它有一些工具可以讓你設計非常詳細的使用者介面，而且可以在許多不同的 Windows 版本運行，包括很舊的版本，例如 Windows XP。

但是 C# 不是只能在 Windows 上使用！

你使用 Mac 嗎？你很幸運！我們也為你加入一條學習路徑，使用 **Visual Studio for Mac**，請參考本書結尾的 Visual Studio for Mac 學習指南附錄。它有這一章的完整替代專案，以及本書的所有 WPF 專案的 Mac 版本。

WPF 專案的 Mac 版本使用 ASP.NET Core。你也可以在 Windows 建構 ASP.NET Core 專案。

你的動物配對遊戲是 WPF app

如果你只需要輸入與輸出文字，主控台 app 是很棒的選項。如果你想要製作在視窗中顯示的視覺化 app，你就要使用不同的技術。這就是為什麼動物配對遊戲是 **WPF app**。WPF（或 Windows Presentation Foundation）可讓你做出可在任何 Windows 版本上運行的桌面應用程式。本書大部分的章節都有一個 WPF app。這個專案的目標是介紹 WPF，並提供可讓你做出強大視覺效果的桌面應用程式和主控台 app 的工具。

當你完成這個專案時，你會更熟悉你將在本書中用來學習和探索 C# 的工具。

這就是建構遊戲的方法

本章接下來的內容會帶領你建構動物配對遊戲，你會用一系列獨立的部分完成它：

1.首先，在 Visual Studio 裡面建立一個新的桌面應用程式專案。

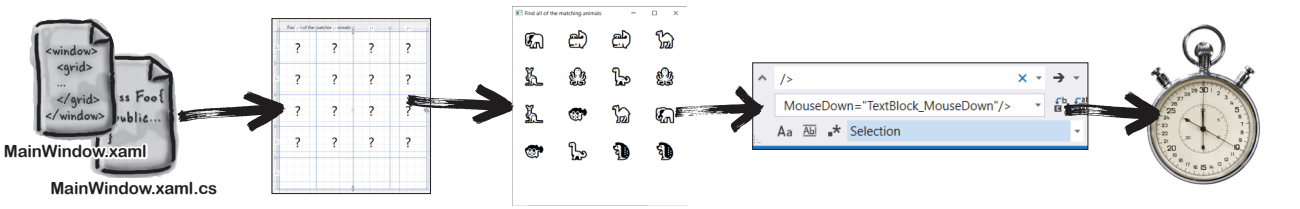
2.然後，使用 XAML 來建構視窗。

3. 編寫 C# 程式來將隨機的動物 emoji 加入視窗。

4. 遊戲會讓用戶按下每一對 emoji 來進行配對。

5. 最後，加入計時器來讓遊戲更刺激。

這個專案可能會耗時 15 分鐘到 1 小時，取決於你的打字速度。慢慢學習有比較好的效果，請給自己充足的時間。



建立專案 **設計視窗** **撰寫 C# 程式** **處理滑鼠按鍵** **加入遊戲計時器**

注意本書的這些「**遊戲設計…漫談**」單元，我們會藉由遊戲設計原則來學習和探索重要的程式設計概念和想法，這些概念和想法適合各種類型的專案，不僅僅是遊戲。

遊戲設計…漫談

什麼是遊戲？

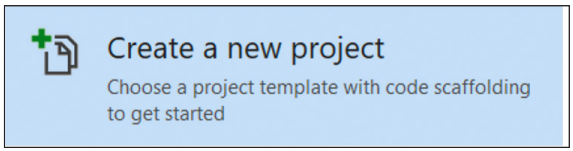
「遊戲是什麼」似乎無需多言，但仔細想想，這個問題不像乍看之下那麼簡單。

- 遊戲都有**贏家**嗎？都會結束嗎？不一定如此。飛行模擬遊戲呢？遊樂園設計遊戲？模擬市民（The Sims）之類的遊戲？
- 遊戲都很**好玩**嗎？不是所有人都有相同的感受。有些玩家喜歡「刷任務」，反覆做同樣的事情，有些人卻覺得這種玩法是在自虐。
- 遊戲一定要**進行決策、有衝突，或一定要解決問題嗎？**不是所有遊戲都是如此。在行走模擬器這種遊戲中，玩家所做的事情只是探索一個環境，通常完全沒有謎題或衝突。
- 事實上，我們很難定義遊戲是什麼。當你閱讀遊戲設計教科書時，你會發現各種不同的定義。因此，出於我們的目的，我們將「**遊戲**」定義成：

遊戲是一種程式，（希望）至少可讓玩家獲得遊戲作者期望提供的樂趣。

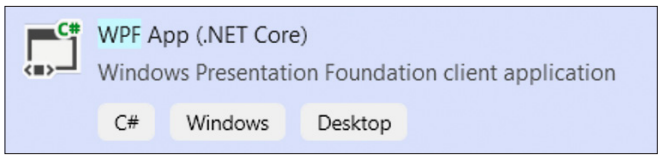
在 Visual Studio 裡建構 WPF 專案

啟動一個 Visual Studio 2019 的新實例，並建立一個新專案：

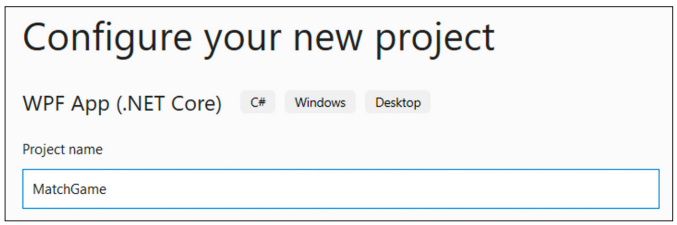


我們完成本章第一個部分的 Console App 專案了，你可以安心地關閉那個 Visual Studio 實例。

我們要用 WPF 來將遊戲做成桌面 app，所以**選擇 WPF App (.NET Core)** 並按下 Next：



Visual Studio 會要求你設置你的專案。在**專案名稱輸入 `MatchGame`**（喜歡的話，你也可以改變建立專案的位置）：



按下 Create（建立）按鈕，Visual Studio 會建立一個稱為 MatchGame 的新專案。

這個檔案裡面有定義主視窗介面的 XAML 程式碼。

MainWindow.xaml

Visual Studio 會幫你建立一個充滿檔案的專案資料夾

當你建立新專案時，Visual Studio 會加入一個稱為 MatchGame 的新資料夾，並且在裡面放入專案需要的所有檔案與資料夾。你要修改其中的兩個檔案，*MainWindow.xaml* 與 *MainWindow.xaml.cs*。

讓遊戲動起來的 C# 程式碼在這裡面。

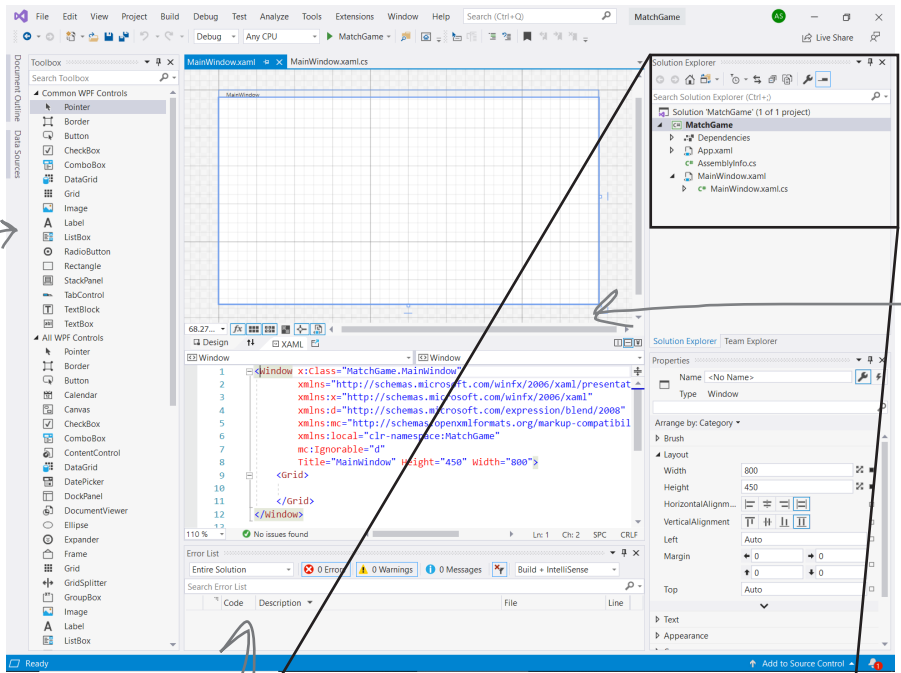
MainWindow.xaml.cs

如果你在進行這個專案時遇到任何問題，可以到我們的 GitHub 網頁尋找引導你操作的影片連結：https://github.com/head-first-csharp/fourth-edition。

削尖你的鉛筆

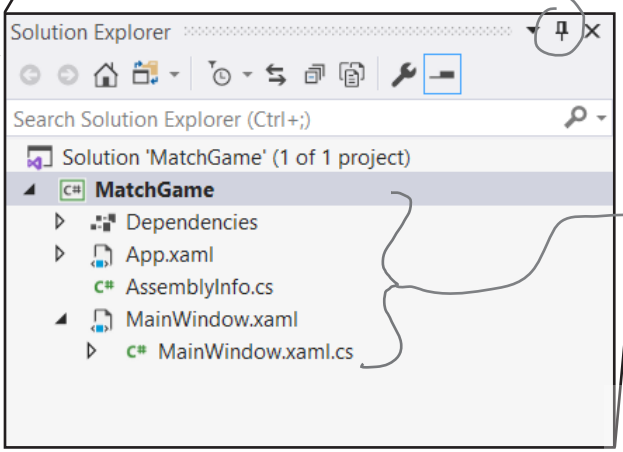
在這本書裡面的這種紙筆練習都不能跳過，它們非常重要，可以幫助你學習、練習和提升 C# 技術。

調整你的 IDE，讓它與下面的畫面一致。首先，在 Solution Explorer 視窗裡面按下 ***MainWindow.xaml*** 兩次，以**開啟**它。然後，**在 View 選單**選擇 *Toolbox* 與 *Error List* 視窗來打開它們。你其實可以透過視窗和檔案的名稱和你的常識來猜出它們的功能！花一分鐘填空一試著填入 Visual Studio IDE 的每一個部分的功能。為了協助你踏出第一步，我們已經完成一個部分了。看看你能不能有根據地猜出其他的功能。



你可以將控制項拉入這個 *Designer*，來編輯使用者介面。

有沒有發現 *Toolbox* 不見了？按下這個圖釘圖示來讓它持續顯示。

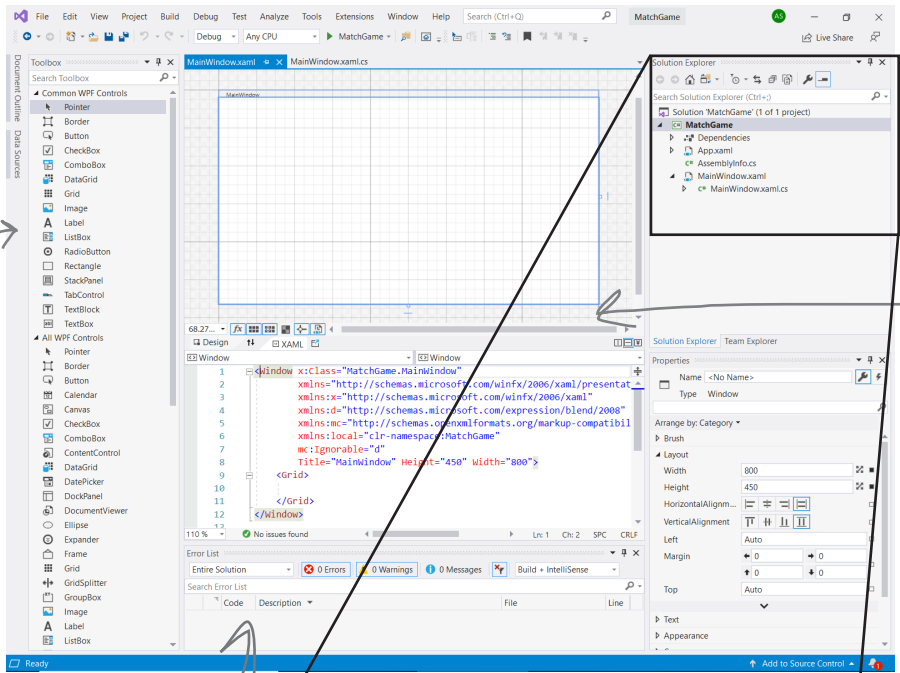


我們使用 Light 色彩佈景主題來讓螢幕截圖更清楚，你可以在 Tools 選單選擇「Options...」，並按下 Environment 來切換各種色彩佈景主題。

削尖你的鉛筆 解答

我們已經填寫 Visual Studio C# IDE 的每個區域的說明了，你寫的東西可能與我們不一樣，希望你可以認出 IDE 的各個視窗與區域的基本用途。別擔心你的答案和我們的略有不同。你會用 IDE 做**很多**練習。

提醒你：這本書會使用「Visual Studio」與「IDE」，包括這一頁。

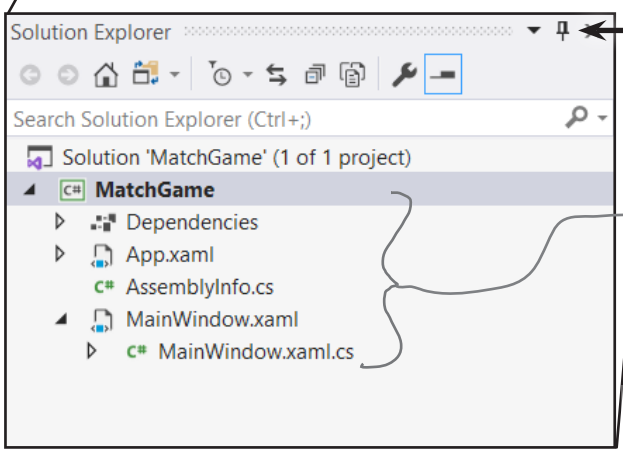


這是 Toolbox，它有許多視覺控制項，可讓你拉入視窗。

你可以將控制項拉入這個 Designer，來編輯使用者介面。

Properties 視窗會顯示你在設計工具 (designer) 裡面選擇的東西的屬性。

這個 Error List 視窗會在你的程式有錯誤時顯示訊息。這個窗格會顯示關於 app 的診斷訊息。



按下這個圖釘圖示會打開或關閉自動隱藏功能。Toolbox 視窗預設自動隱藏。

當你加入新專案時，這個 Solution Explorer 會顯示 IDE 為你建立的 C# 與 XAML 檔案，連同你的解決方案的其他檔案。

你可以使用 IDE 的 Solution Explorer 來切換檔案。

留意這些 Q&A 單元。它們經常回答最要緊的問題，並指出其他讀者正在思考的問題。事實上，有很多問題真的是之前版本的讀者提出來的！

問：既然 Visual Studio 為我寫好全部的程式了，那麼是不是只要學會 Visual Studio，就學會 C# 了？

答：不，雖然 IDE 很棒，可以自動為你產生一些程式碼，但它的能力僅止於此，它很擅長做一些事情，例如為你設置很好的出發點，以及自動改變 UI 控制項的屬性。但任何 IDE 都無法為你代勞程式設計最重要的部分 — 釐清程式需要做什麼，並且讓程式做那些事情。雖然 Visual Studio IDE 是最進階的開發環境，但是它只能做到這樣。寫程式來實際完成工作的人是**你**，不是 IDE。

問：如果 IDE 產生我不想要在專案中使用的程式，該怎麼辦？

答：你可以修改或刪除它。IDE 在設計上，會根據你所拉入或加入的元素最常見的用法來建立程式碼，但有時它不是你要的。IDE 為你做的任何事情（它建立的每一行程式、它加入的每一個檔案）都可以修改，無論是親手編輯檔案，還是透過 IDE 的介面。

問：為什麼你叫我們安裝 Visual Studio Community 版本？你確定本書的任何工作都不需要使用 Visual Studio 的付費版本嗎？

答：本書的任何事情都可以用免費的 Visual Studio 版本來完成（你可以從微軟的網站下載它）。Community 與其他版本的主要差異不會阻礙你編寫 C# 與建立完整的應用程式。

問：你說過一起使用 C# 與 XAML，什麼是 XAML ？如何一起使用 XAML 與 C# ？

答：XAML（X 的發音類似 Z，這是為了與「camel」同韻）是一種**標記語言**，可用來建構 WPF app 的使用者介面。XAML 以 XML 為基礎（所以如果你用過 HTML，你就有一個很好的起點）。這是畫出灰色橢圓的 XAML **標籤**：

```
<Ellipse Fill="Gray"
  Height="100" Width="75"/>
```

回到你的專案，在 XAML 程式中的 `<Grid>` 後面輸入這個標籤，你的視窗中間會出現一個灰色橢圓。因為它的開頭是 `<` 接下來有一個單字（`Ellipse`）（形成一個**開始標籤**），所以你可以認出它是個標籤，這個 `Ellipse` 標籤有三個**屬性**：一個屬性將填充顏色設為灰色，另兩個屬性設定它的高度與寬度。這個標籤的結尾是 `/>`，但有些 XAML 標籤的裡面有其他的標籤。我們可以將 `/>` 換成 `>`，加入其他標籤（這些標籤也可以放入額外的標籤）並且用 `</Ellipse>` 這種結束標籤來關閉它，來將它變成**容器標籤**。

你會在本書學到 XAML 如何運作，以及各種不同的 XAML 標籤。

問：我們的畫面長得跟你的不一樣！它缺少一些視窗，而且有些視窗在不同的位置。我是不是哪裡做錯了？怎麼重設它？

答：在 Window 選單裡面按下 **Reset Window Layout**，IDE 就會恢復成預設的視窗配置。然後在 **View>>Other Windows** 選單，打開 Toolbox 和 Error List 視窗，這樣你的畫面就會跟這一章的一樣了。

Visual Studio 會產生一些程式碼來當成 app 的起點，確保 app 做它該做的事情完全是你的責任。

Toolbox 在預設情況下會收起來，你可以使用 Toolbox 視窗的右上角的圖釘按鈕來讓它保持展開。

XAML 與 camel 同韻

我們會在每一個專案的開頭展示這種「商場地圖」，來協助你掌握大局。

使用 XAML 來設計視窗

讓 Visual Studio 為你建立一個 WPF 專案之後，接下來要開始使用 **XAML** 了。

XAML 是 **Extensible Application Markup Language** 的縮寫，它是一種非常靈活的標記語言，C# 開發者可用它來設計使用者介面。你接下來會用兩種不同的程式來建構 app。首先，你會用 XAML 來設計使用者介面（或 UI），然後，你會加入 C# 程式碼，來讓遊戲動起來。

如果你曾經使用 HTML 來設計網頁，你會發現它與 XAML 有許多相似處。這是使用 XAML 來顯示一個小視窗的小範例：

```
<Window x:Class="MyWPFApp.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="This is a WPF window" Height="100" Width="400"> (1)
    <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
        <TextBlock FontSize="18px" Text="XAML helps you design great user interfaces."/> (2)
        <Button Width="50" Margin="5,10" Content="I agree!"/> (3)
    </StackPanel>
</Window>
```

我們幫定義文字的 XAML 加上編號。

注意下面的螢幕畫面中對應的編號。

下面是當 WPF **轉譯**這個視窗（或是在螢幕上畫出它）時的樣子。它會畫出一個視窗，裡面有兩個可見的**控制項**，包含一個顯示文字的 TextBlock 控制項，以及一個讓用戶按下的 Button 控制項。它們是用不可見的 StackPanel 控制項來排版的，StackPanel 會將其中一個顯示在另一個上面。你可以先在視窗截圖裡面找出想要比對的控制項，再到 XAML 找出它的 `TextBlock` 與 `Button` 標籤。

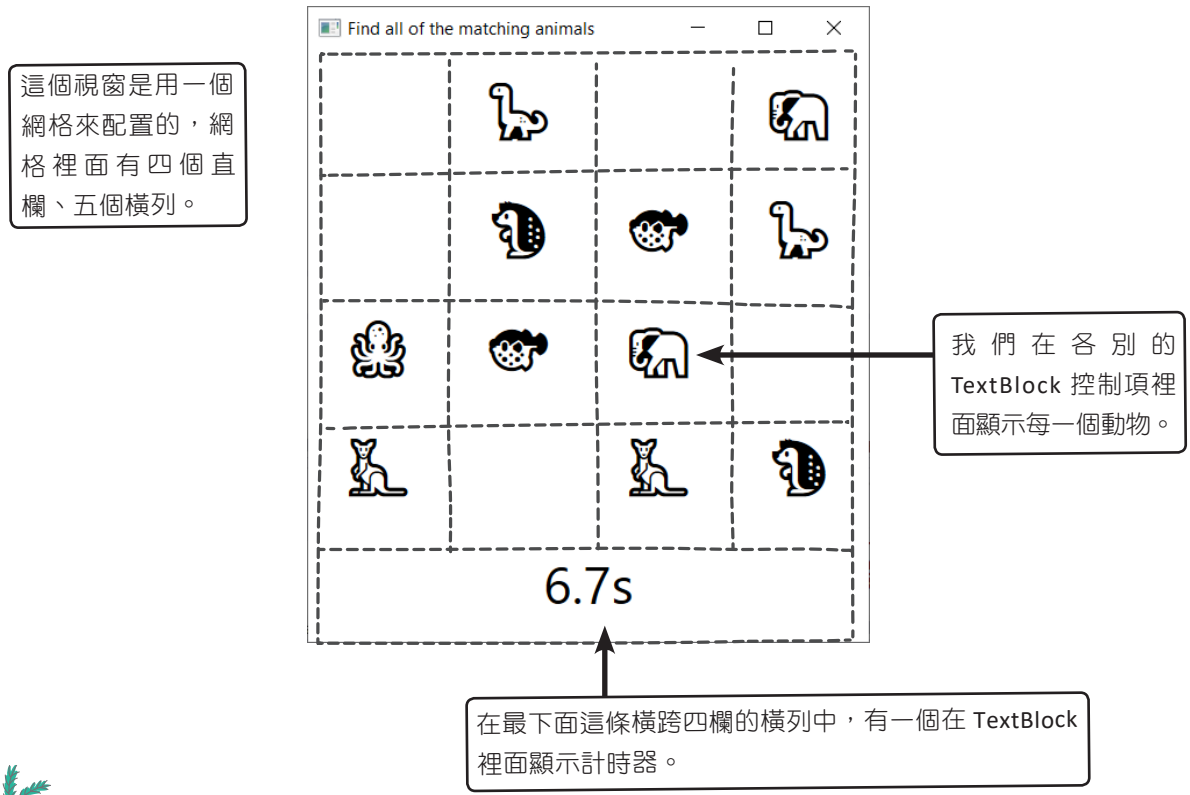
TextBlock 控制項的功能從名字就可以知道 — 顯示文字區塊。

在螢幕畫面中的這些數字對應 XAML 程式碼的同一個數字。

設計遊戲視窗

我們的應用程式需要一個圖形化的使用者介面、讓遊戲動起來的物件，以及一個可執行檔，才可以執行。看起來我們有很多事情要做，但你會在本章其餘的內容建構所有東西，最後，你將完全知道如何用 Visual Studio 來設計漂亮的 WPF app。

這是我們要製作的 app 的視窗配置：



對 C# 開發者來說，XAML 是重要的技能。

你可能在想：「不是吧！這是《深入淺出 C#》，為什麼要花這麼多時間學習 XAML ？我們應該把精力放在 C# 才對吧？」

WPF 應用程式使用 XAML 來設計使用者介面，其他類型的 C# 專案也一樣。你不但可以用它來製作桌面 app，也可以運用同一種技術，使用 Xamarin Forms（採用 XAML 的變體，具有稍微不同的控制項組合）來建構 C# Android 與 iOS 行動 app。這就是為什麼對每位 C# 開發者來說，使用 XAML 來建構使用者介面都是很重要的技能，也是你將在這本書學到許多 XAML 的原因。我們會**一步一步地**帶領你建構 XAML — 你可以在 Visual Studio 2019 XAML 設計工具裡面使用拖曳工具來建立使用者介面，而不需要打太多字。**把話講白：**

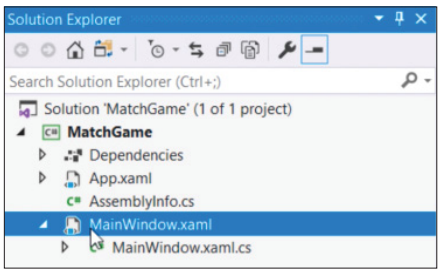
XAML 是定義使用者介面的程式，C# 是定義行為的程式。

使用 XAML 屬性來設定視窗尺寸與標題

我們來為動物配對遊戲建構 UI。你的第一個工作是把視窗變窄，並且改變它的標題，你將會更熟悉 Visual Studio 的 XAML 設計工具，它是一種強大的工具，可幫助你為 app 設計出很漂亮的使用者介面。

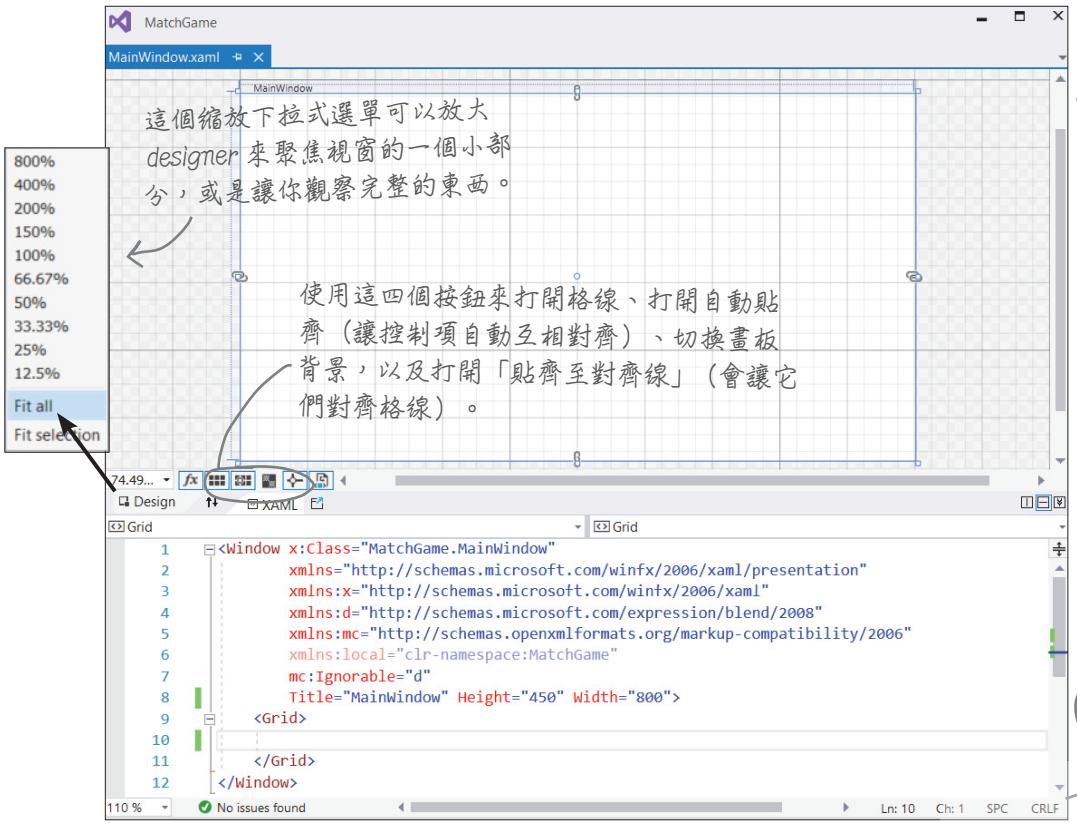
1 **選擇主視窗。**

在 Solution Explorer 裡面按兩下 *MainWindow.xaml*。



在 Solution Explorer 裡面的檔案按兩下可以用適當的編輯器打開它。結尾為 .cs 的 C# 程式檔案會在程式碼編輯器裡面開啟。結尾為 .xaml 的 XAML 檔案會在 XAML 設計工具裡面開啟。

按兩下之後，Visual Studio 會在 XAML 設計工具裡面打開它。



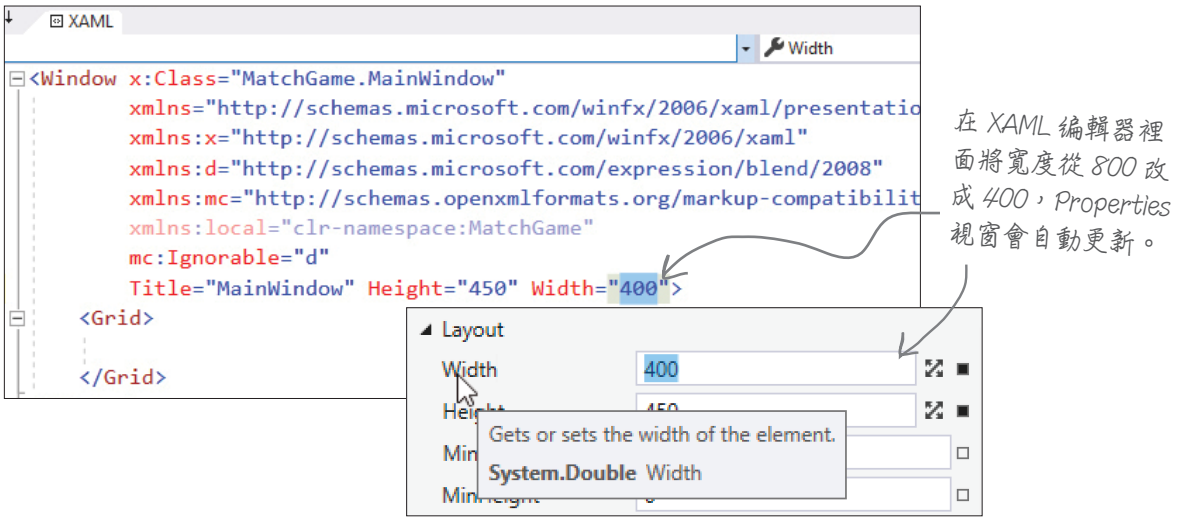
設計工具會顯示你正在編輯的視窗的預覽畫面。在這裡做的任何更改都會更改下面的 XAML。

你可以修改這裡的 XAML，並且在上面的視窗中即時看到更新。

2 **改變視窗的尺寸。**

將滑鼠移到 XAML 編輯器，在 XAML 程式碼的前八行隨便按下一個地方，你應該可以在 Properties 視窗看到視窗的屬性。

展開版面配置區域，**將寬度改成 400**。在設計窗格裡面的視窗會立刻變窄。仔細地看一下 XAML 程式 — Width 屬性變成 400 了。



3 **改變視窗標題。**

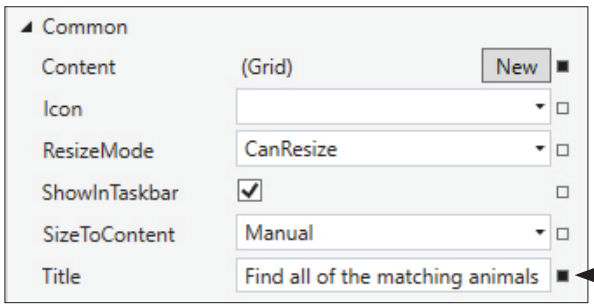
找到 `Window` 標籤最後面的這一行 XAML 程式碼：

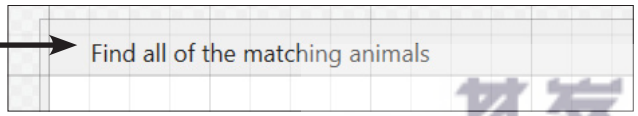
```
Title="MainWindow" Height="450" Width="400">
```

將標題改成 **`Find all of the matching animals`**，變成這樣：

```
Title="Find all of the matching animals" Height="450" Width="400">
```

你會在 Properties 視窗的 Common 區域看到這個改變，更重要的是，現在視窗的標題列顯示出新文字了。





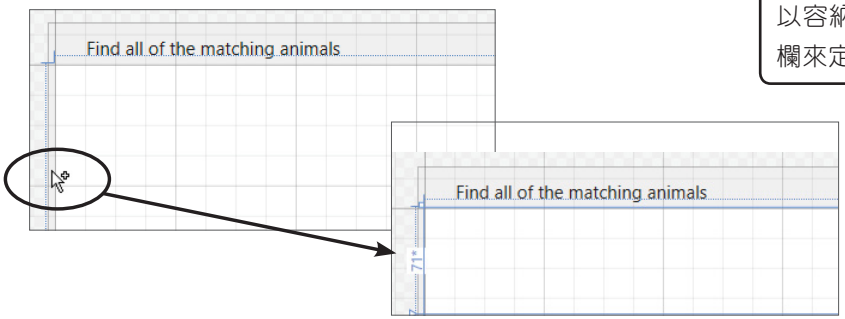
當你修改 XAML 標籤裡面的屬性時，修改的部分會立刻出現在 Properties 視窗裡面。當你使用 Properties 視窗來修改 UI 時，IDE 會更新 XAML。

在 XAML 格線裡面加入橫列與直欄

乍看之下，你的主視窗是空的，但是仔細看一下 XAML 的最下面，有沒有看到 <Grid>，還有它下面的 </Grid> ？你的視窗其實是**格線**，你看不到任何東西是因為它沒有任何橫列與直欄。我們來加入一列。

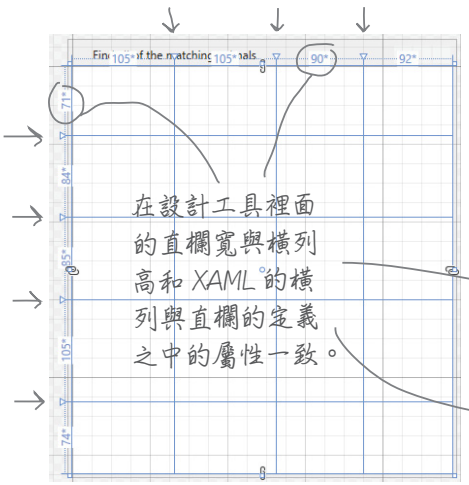
將你的滑鼠移到設計工具裡面的視窗的左側。當游標出現加號時，按下滑鼠來加入一列。

WPF app 的 UI 是用按鈕、標籤、確認方塊等控制項來建構的。格線是一種特殊的控制項，稱為容器（container），可以容納其他的控制項。它使用橫列與直欄來定義版面配置。



你會看到一個數字還有它旁邊的星號，以及跨越視窗的橫線，這樣你就在格線中加入一列了！現在要加入橫列與直欄：

- 再重複做四次，總共加入五列。
- 把滑鼠移到視窗的最上面，按下按鍵，加入四個直欄。此時你的視窗會變成下面的畫面（但是你的數字會不同，這不是問題）。
- 回到 XAML。現在裡面有一組 ColumnDefinition 與 RowDefinition 標籤，符合你剛才加入的橫列與直欄。




```
<Grid.ColumnDefinitions>
    <ColumnDefinition Width="105*"/>
    <ColumnDefinition Width="105*"/>
    <ColumnDefinition Width="90*"/>
    <ColumnDefinition Width="92*"/>
</Grid.ColumnDefinitions>
<Grid.RowDefinitions>
    <RowDefinition Height="71*"/>
    <RowDefinition Height="84*"/>
    <RowDefinition Height="85*"/>
    <RowDefinition Height="105*"/>
    <RowDefinition Height="74*"/>
</Grid.RowDefinitions>
```

這些「照過來！」單元會提醒你一些重要卻經常令人困惑的事情，它們可能會讓你犯錯，或減緩你的速度。

照過來！

你的 IDE 看起來可能會不一樣。

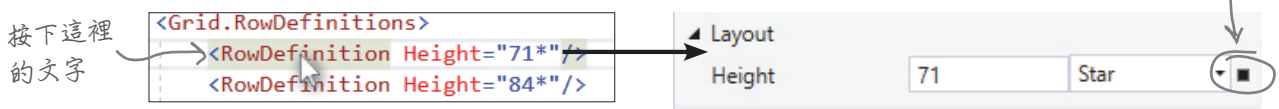
本書的螢幕畫面都來自 **Visual Studio Community 2019 for Windows**，如果你使用 Professional 或 Enterprise 版本，你可能會看到一些小地方不一樣。

別擔心，一切事物的運作方式都一模一樣。

把橫列與直欄變成相同的尺寸

我們希望遊戲將動物整齊排列來讓玩家配對，我們會把動物放在格線的格子裡面，而且格線會被自動調整成視窗的大小，所以我們要讓橫列與直欄都有相同的尺寸。幸好，XAML 可讓我們輕鬆地調整橫列與直欄的尺寸。**在 XAML 編輯器裡面按下第一個 RowDefinition 標籤**，讓 Properties 視窗顯示它的屬性：

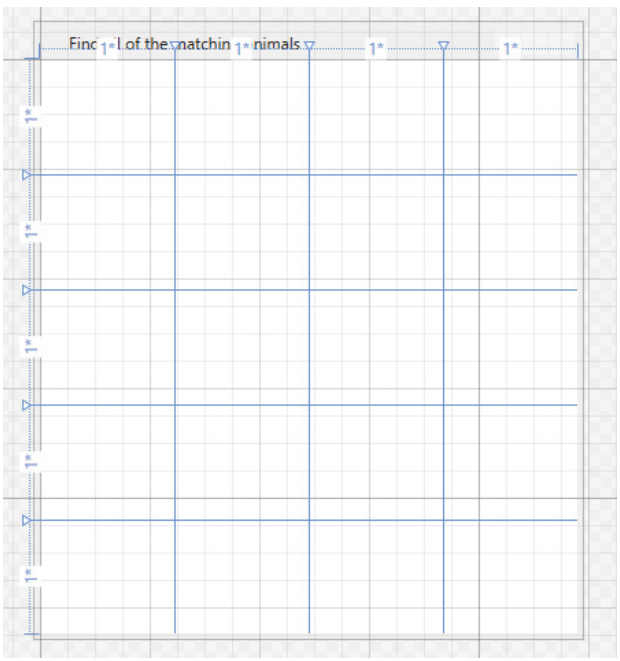
實心的方塊代表屬性值不是預設值，按下方塊，然後在選單裡面選擇 Reset 可將它重設為預設值。



在 Properties 視窗**按下** Height 屬性右邊的**方塊**，然後在跳出來的**選單裡面選擇 Reset**。嘿！等一下！這樣做之後，designer 裡面的橫列消失了。事實上，它沒有消失，只是變得很窄，繼續**重設**每一列的 **Height 屬性**。然後**重設**所有直欄的 **Width 屬性**。現在你的格線應該是四條大小相同的直欄，與五條大小相同的橫列。

試著認真閱讀 XAML，如果你沒有用過 HTML 或 XML，在一開始可能會覺得它們看起來像一堆 < 括號 > 與 / 斜線，你閱讀它的次數越多，你就越了解它。

你會在設計工具裡面看到這個畫面：



你會在 XAML 編輯器裡面的 <Window ... > 開始標籤與 </Window> 結束標籤之間看到這些東西：

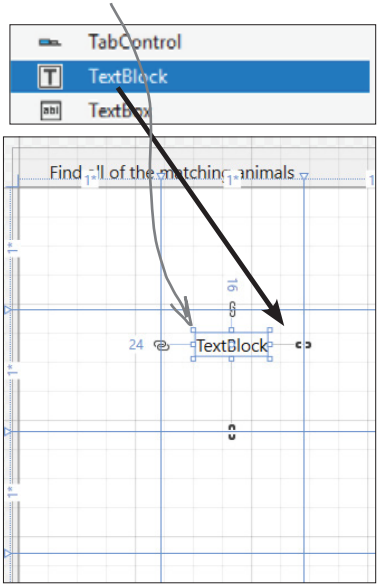
```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition/>
        <ColumnDefinition/>
        <ColumnDefinition/>
        <ColumnDefinition/>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
    </Grid.RowDefinitions>
</Grid>
```

這是建立四條同樣大小的直欄，與五條同樣大小的橫列的 XAML 程式碼。

在格線裡面加入 TextBlock 控制項

WPF app 使用 **TextBlock 控制項**來顯示文字，我們要用它們來顯示動物，讓玩家可以尋找和配對。我們在視窗中加入一個。

當你從工具箱拉一個控制項到格子裡面時，IDE 會在你的 XAML 裡面加入 TextBlock，並設定它的橫列、直欄與邊界。



在 Toolbox 裡，展開 Common WPF Controls，**將一個 TextBlock 拉到第二欄、第二列的格子裡**。IDE 會在 `Grid` 的開始與結束標籤之間加入一個 `TextBlock`：

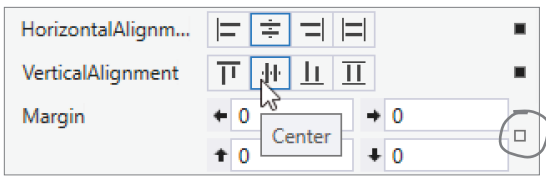
```
<TextBlock Text="TextBlock"
    HorizontalAlignment="Left" VerticalAlignment="Center"
    Margin="560,0,0,0" TextWrapping="Wrap" />
```

這個 TextBlock 的 XAML 有五個屬性：

- `Text` 可讓 TextBlock 知道要在視窗裡顯示什麼文字。
- `HorizontalAlignment` 可以把文字對齊左邊、右邊或中央。
- `VerticalAlignment` 將它對齊格子的頂部、中間或底部。
- `Margin` 設定它與容器的頂部、側邊或底部之間的距離。
- `TextWrapping` 設定是否加入分行符號，來將文字分成多行。

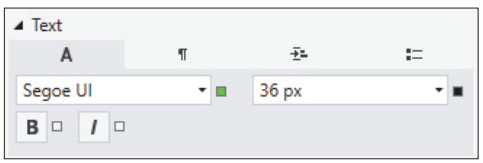
你的屬性可能有不同的順序，而且 Margin 屬性會有不同的數字，因為它們取決於你把它拉到格子裡面的哪裡。你可以用 IDE 的 Properties 視窗來修改或重設以上所有屬性。

我們想要把所有動物置中。在設計工具裡面**按下標籤**，然後在 Properties 視窗按下 Layout 來展開 **Layout 區域**。為直向對齊與橫向對齊屬性按下 **Center**，然後使用視窗右邊的方塊來**重設 Margin 屬性**。



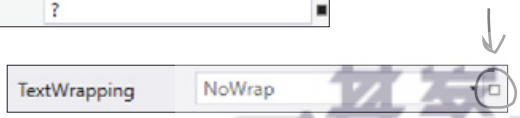
按下這個方塊並選擇「Reset」來重設邊界。

我們也想要讓動物大一些，所以在 Properties 視窗裡面**展開 Text 區域**，將**字型大小改成** `36 px`。然後在 Common 區域將 Text 屬性改成 **?** 來讓它顯示一個問號。



Text 屬性（在 Common 底下）可設定 TextBlock 的文字。

在 Properties 視窗最上面**按下搜尋欄**，然後輸入 `wrap` 來尋找符合它的屬性。使用視窗右邊的方塊來重設 TextWrapping 屬性。



問：**當我重設前四列的高（height）時，它們消失了，但是當我重設最後一列的高之後，它們又出現了，為什麼會這樣？**

答：那幾列看起來消失的原因是在預設情況下，WPF 格線會**按比例調整**橫列與直欄的**大小**。如果最後一列的高是 74*，當你將前四列改成預設高度 1* 時，格線會改變列的大小，讓前四列都占格線高度的 1/78（或 1.3%），讓最後一列占 74/78（或 94.8%），導致前四列看起來非常小。當你將最後一列重設成預設高 1* 時，格線會將每一列的大小平均調整為格線高的 20%。

問：**我們將視窗的寬設成 400，它的單位是什麼？400 有多寬？**

答：WPF 使用**非關設備的像素**，它永遠是一英寸的 1/96。也就是說，在未縮放的畫面上，96 個像素一定等於 1 英寸。但如果你拿尺來測量你的視窗，你可能會發現它並非剛好是 400 像素（或大約 4.16 英寸）寬。這是因為 Windows 有一個很有用的功能，可讓你更改畫面的尺度，如此一來，當你將房間另一頭的電視當成電腦螢幕來使用時，你的 app 看起來就不會太小。非關設備的像素可協助 WPF 讓 app 在任何尺度之下都很漂亮。

你會在這本書看到類似這樣的練習。它們可以提升你的程式技術。偷看答案絕對是 OK 的！

你有一個 TextBlock 了一這是很棒的起點！但是我們需要 16 個 TextBlock 來顯示所有的動物。你可以想出如何加入更多 XAML，將相同的 TextBlock 加入格線的前四列的每一格裡面嗎？

先看一下你剛才建立的 XAML 標籤。它應該長這樣一裡面的屬性可能有不同的順序，我在這裡將它分成兩行（如果你想要讓 XAML 更容易閱讀，你也可以這樣做）：

```
<TextBlock Text="?" Grid.Column="1" Grid.Row="1" FontSize="36"
      HorizontalAlignment="Center" VerticalAlignment="Center"/>
```

你的工作是**複製那個 TextBlock**，讓格線上面的 16 格都有一模一樣的 TextBlock — 為了完成這個練習，你*必須在 app 再加入 15 個 TextBlock*。請記得這些事情：

- 橫列與直欄的編號是從 0 開始的，其預設值也是 0。所以如果你沒有設定 `Grid.Row` 或 `Grid.Column` 屬性，TextBlock 會出現在最左邊那一列或最上面那一欄。
- 你可以在設計工具裡面編輯 UI，或複製貼上 XAML。嘗試這兩種做法，看看哪一種適合你！

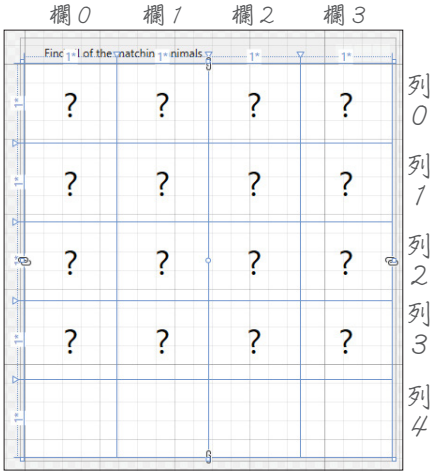
這是讓玩家配對的動物的 16 個 TextBlock 的 XAML 程式碼－它們除了 Grid.Row 與 Grid.Column 屬性之外都是一致的，這段程式在一個 5 列 4 欄的格線上方的 16 個格子裡面各放入一個 TextBlock。（*Window* 標籤保持不變，所以在這裡沒有列出它。）

```
<Grid>
  <Grid.ColumnDefinitions>
    <ColumnDefinition/>
    <ColumnDefinition/>
    <ColumnDefinition/>
    <ColumnDefinition/>
  </Grid.ColumnDefinitions>

  <Grid.RowDefinitions>
    <RowDefinition/>
    <RowDefinition/>
    <RowDefinition/>
    <RowDefinition/>
    <RowDefinition/>
  </Grid.RowDefinitions>
```

當你將各列與各欄的大小都設為相同時，橫列與直欄的定義就是這樣。

這是加入所有 TextBlock 之後，在 Visual Studio 設計工具裡面的視窗。




```
  <TextBlock Text="?" FontSize="36" HorizontalAlignment="Center" VerticalAlignment="Center"/>
  <TextBlock Text="?" FontSize="36" HorizontalAlignment="Center" VerticalAlignment="Center" Grid.Column="1"/>
  <TextBlock Text="?" FontSize="36" HorizontalAlignment="Center" VerticalAlignment="Center" Grid.Column="2"/>
  <TextBlock Text="?" FontSize="36" HorizontalAlignment="Center" VerticalAlignment="Center" Grid.Column="3"/>

  <TextBlock Text="?" FontSize="36" HorizontalAlignment="Center" VerticalAlignment="Center" Grid.Row="1"/>
  <TextBlock Text="?" FontSize="36" Grid.Row="1" Grid.Column="1"
        HorizontalAlignment="Center" VerticalAlignment="Center"/>
  <TextBlock Text="?" FontSize="36" Grid.Row="1" Grid.Column="2"
        HorizontalAlignment="Center" VerticalAlignment="Center"/>
  <TextBlock Text="?" FontSize="36" Grid.Row="1" Grid.Column="3"
        HorizontalAlignment="Center" VerticalAlignment="Center"/>
```

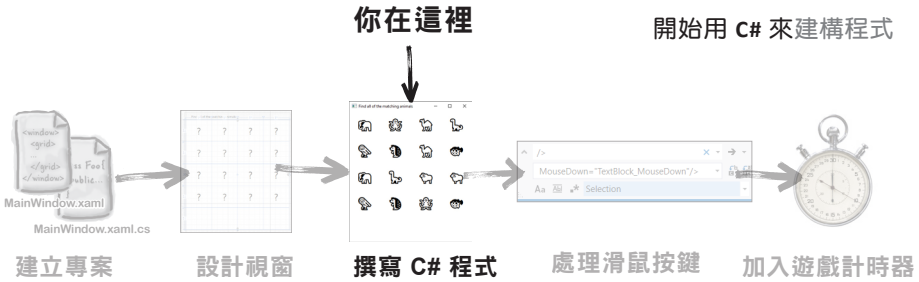
這四個 TextBlock 控制項的 Grid.Row 屬性都設為 1，所以它們位於上面算下來的第二列（因為第一列是 0）。

```
  <TextBlock Text="?" FontSize="36" Grid.Row="2" HorizontalAlignment="Center" VerticalAlignment="Center"/>
  <TextBlock Text="?" FontSize="36" Grid.Row="2" Grid.Column="1"
        HorizontalAlignment="Center" VerticalAlignment="Center"/>
  <TextBlock Text="?" FontSize="36" Grid.Row="2" Grid.Column="2"
        HorizontalAlignment="Center" VerticalAlignment="Center"/>
  <TextBlock Text="?" FontSize="36" Grid.Row="2" Grid.Column="3"
        HorizontalAlignment="Center" VerticalAlignment="Center"/>
```

加入設為 0 的 Grid.Row 或 Grid.Column 屬性是 OK 的，我們不加入它們是因為 0 是預設值。

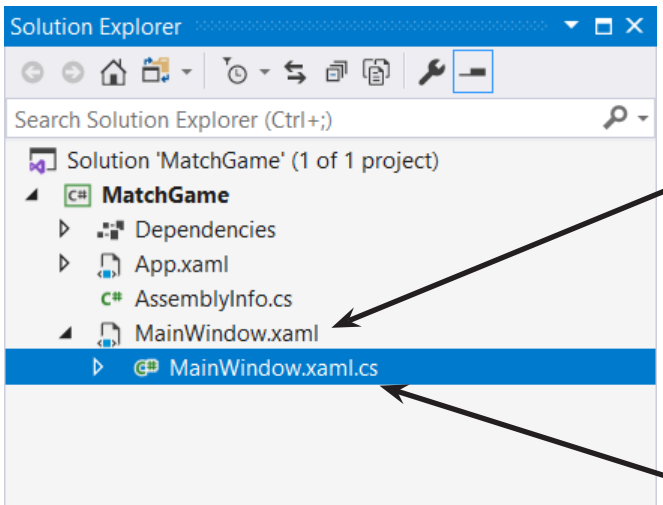
```
  <TextBlock Text="?" FontSize="36" Grid.Row="3" HorizontalAlignment="Center" VerticalAlignment="Center"/>
  <TextBlock Text="?" FontSize="36" Grid.Row="3" Grid.Column="1"
        HorizontalAlignment="Center" VerticalAlignment="Center"/>
  <TextBlock Text="?" FontSize="36" Grid.Row="3" Grid.Column="2"
        HorizontalAlignment="Center" VerticalAlignment="Center"/>
  <TextBlock Text="?" FontSize="36" Grid.Row="3" Grid.Column="3"
        HorizontalAlignment="Center" VerticalAlignment="Center"/>
</Grid>
```

這裡的程式很多，但是它其實只是將同一行程式碼複製16次並稍微修改。開頭為<TextBlock的每一行都有相同的四個屬性（Text、FontSize、HorizontalAlignment 與 VerticalAlignment）。它們只是有不同的 Grid.Row 與 Grid.Column 屬性。（屬性可以用任何順序排列。）



現在你可以開始寫遊戲程式了

剛才你已經設計好視窗了，或者說，至少足以進入遊戲製作的下一個部分了。接下來要加入 C# 程式，來讓遊戲可以運作。



你剛才在 *MainWindow.xaml* 裡面編輯 XAML 程式，它是存放視窗的所有設計元素的地方—這個檔案內的 XAML 定義了視窗的外觀和版面配置。

現在你要開始撰寫 C# 程式碼，它們位於 *MainWindow.xaml.cs* 裡面。它稱為視窗的 code-behind（程式碼後置），因為它是與 XAML 檔裡面的標記連結的。正因為如此，除了結尾的「.cs」之外，它們的名稱是相同的。你接下來要在這個檔案裡面加入 C# 程式碼來定義遊戲的行為，包括在格線中加入 emoji、處理滑鼠按鍵，以及讓計時器運作的程式碼。

你輸入的 C# 程式碼必須完全正確。

有人說，除非你曾經花了好幾個小時找出一個擺錯位置的句點，否則不會成為真正的開發者。大小寫非常重要：`SetUpGame` 與 `setUpGame` 不一樣。多餘的逗號、分號、括號…等可能會破壞你的程式碼，更糟糕的情況是，它們可能改變你的程式，讓程式可以通過組建，但產生出乎意外的行為。雖然 IDE 的 **AI 輔助 IntelliSense** 可以協助你避免這些問題…但是它沒辦法幫你做每一件事。