國立虎尾科技大學

機械設計工程系

電腦輔助設計實習 bg3 期末報告

專案:捉蝴蝶小遊戲

Preject: Catch butterfly game

組員:

設計一乙 40723206 王冠驊

設計一乙 40723207 吳文翔

設計一乙 40723209 吳耕甫

設計一乙 40723210 李承澤

設計一乙 40723245 蔡育澧

設計一乙 40723246 鄭竣元

指導教授:嚴家銘

摘要

本專案是運用本學期所學的內容，以及老師提供的遊戲範例製作出一個簡單的小遊戲，以及對於寫程式的環境做出些簡單的報告。

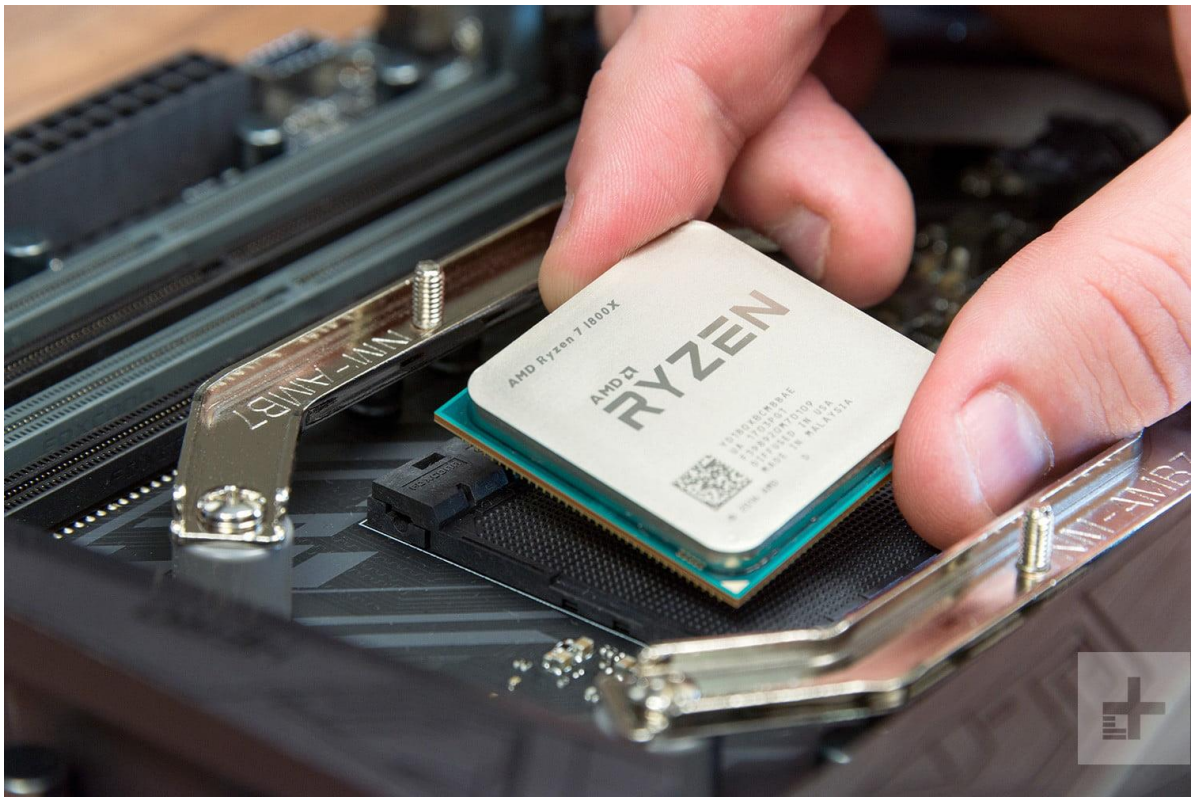
目錄

內容:

電腦硬體

電腦的硬體部分分為處理位元(cpu)、記憶體(RAM)、主機板(motherboard)、記憶位元(硬碟 HDD or SSD)、電源供應器(PSU)安裝在機殼裡面，以及許多非必要的硬體:顯示卡(GPU)、光碟機、各式介面卡等，使用者必須透過輸入介面(鍵盤、滑鼠等)像電腦下指令，電腦會透過輸出介面(螢幕、音響等)回傳結果。

處理位元又稱中央處理器，是電腦中最主要的硬體之一，控制整個電腦主要的算術邏輯單元，使得程式和作業系統可在它上面執行，效能直接影響電腦的處理速度。安裝在主機板上的 CPU 插槽，目前大多由英特爾(Intel)和超微半導體(AMD)兩家公司生產。

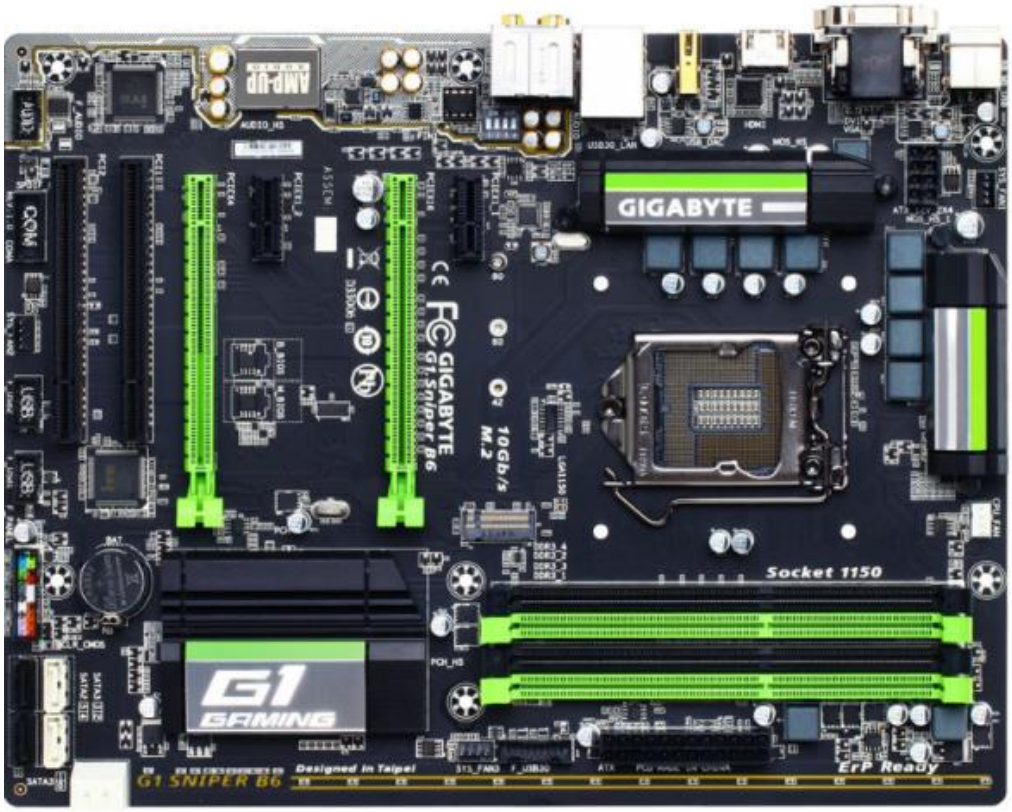


上圖為 AMD 公司的 Ryzen 7 1800X 核心

記憶體即是 RAM，用於存放當前正在執行的程式以及目前所需的資料等，屬於短期臨時記憶。它的讀寫速度遠高於硬碟機或光碟機之類的外部儲存裝置，當系統關閉或無供電時，其中的資料會遺失。

上圖為 crucial 的 DRR4 8G 2400MHz

主機板又稱母板(Motherbroad)是電腦的主要電路板，電腦的其他硬體一般直接插入主機板中來交換資訊，上面有各種周邊裝置的連接孔。主機板的尺寸會隨著需要的功用而改變。



上圖為技嘉的 B85 G1.Sinper B6

硬碟是用於儲存資料資料，與記憶體不同，其中的資料資料在斷電之後不會消失，現今的硬碟大多為 SSD、HDD 以及較新穎的 M2 與 NVMe。

上圖為創見的 SSD

電源供應器作用為將家用的交流電轉為電腦可以用的直流電，在選擇時必須滿足所有零件的需求，以免影響電腦效能。

上圖為海韻的 P-1000W

顯示卡是電腦顯示圖像的重要器件，它負責處理 CPU 傳送來的圖像指令和資料並將處理後的結果輸出至顯示器。雖然是很重要的零件，但現今大多的核心都富有內顯，如果是輕微的文書處理，對顯示卡的需求就比較小了。

上圖為 NVIDIA 的原廠顯卡 GTX1080

參考文獻:

https://zh.wikipedia.org/wiki/%E4%B8%AA%E4%BA%BA%E7%94%B5%E8%84%91

操作系統

系統是用來管理電腦硬體與軟體的操作軟體，現今常見個人電腦的系統為微軟的 Window 以及蘋果的 MacOS。

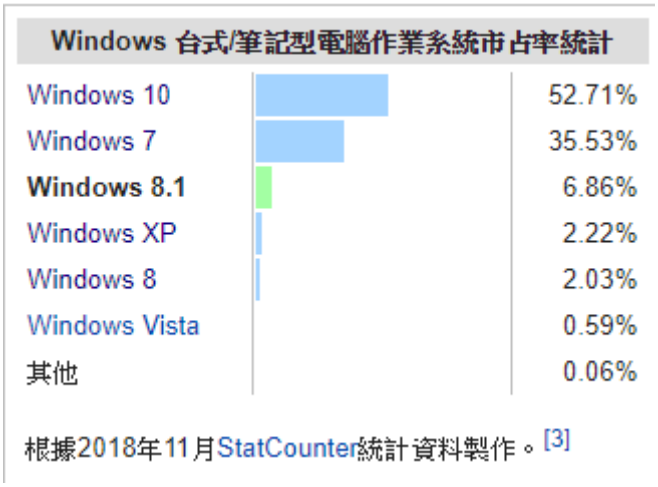
麥金塔作業系統是一套執行於蘋果 Macintosh 系列電腦上的系統，2012 年前稱 Mac OS X，2012 年-2016 年稱 OS X，是個在商業領域上很成功的圖形化使用者介面，電腦的配備都是制定好的，演少變化性，優點有順暢的操作介面、良好的軟體整合性、極佳的性能、穩定的系統等。

微軟的 window 是目前市面上佔有率最大的系統，從早期 1.01 到現在的 Window10，已經有 30 幾年的歷史，起初是 MS-DOS 之下的桌面環境，演變到現在個人操作系統的壟斷地位。

Window7 已經有將近 10 年的歷史，然而直到微軟 2018 年的官方調查，仍然還有 42%的用戶在使用，優點是相容性好、穩定性佳、沒有微軟隨時得強制更新，或是有些人不願意再適應新的系統，但是 Window7 的支援到 2020 年 1 月 14 日便到期，硬體部分則是 2017 年後已經不能在安裝了。

Window8 在 2012 年正式問世，有著新穎的桌面 UI 變化，但由於轉變得過快，刪掉了開始按鈕、沒有鬧鐘、收尋功能難用等，銷量十分低迷，為了挽救

這個場面，微軟推出了 Window8.1 的更新，改善了許多缺點，包跨開式按鈕回歸、個人個人化選項等。



Window8 的佔有率還比 2001 推出的 WindowXP 還低

Window10 是目前最新的系統，打著至今最好的 Windows 的口號，開始初期開放免費升級，以及微軟流氓式的強制更新，前期成效不錯，有著跨平台統一的應用程式商店、全新的網路導覽器、更好的效能、更小的系統空間，以及在 CMD 輸入上加入 Ctrl+V 的貼上功能。

參考文獻：

https://zh.wikipedia.org/zh-tw/Microsoft_Windows

https://zh.wikipedia.org/wiki/Windows_7

https://zh.wikipedia.org/wiki/Windows_8

https://zh.wikipedia.org/wiki/Windows_10

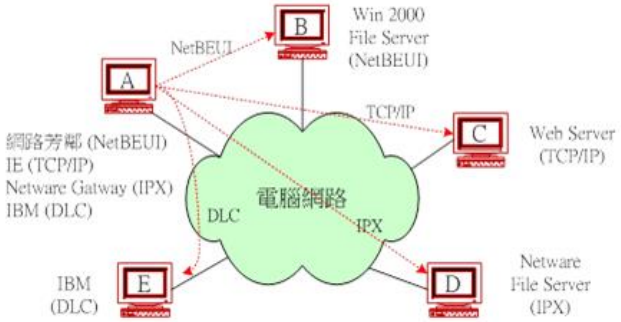
電腦網路

通常也簡稱網路，是利用通訊裝置和線路將地理位置不同的、功能獨立的多個電腦系統連接起來，以功能完善的網路軟體實現網路的硬體、軟體及資源共用和資訊傳遞的系統。 簡單的說即連接兩台或多台電腦進行通訊的系統。

上圖為多人網路連接示意圖

通訊協定或稱為傳輸協定（Communications Protocol）在電信中，是指在任何物理媒介中允許兩個或多個在傳輸系統中的終端之間傳播資訊的系統標準，也是指電腦通訊或網路裝置的共同語言。通訊協定定義了通訊中的語法學, 語意學和同步規則以及可能存在的錯誤檢測與糾正。通訊協定在硬體，軟體或兩者之間皆可實現。

一部電腦上可以安裝多種通訊協定，與不同網路之間相互通訊。但如果安裝太多沒有使用到的通訊協定時，將會佔用過多記憶體，而影響電腦的執行速度。



上圖為通訊協定示意圖

乙太網路是一種電腦區域網路技術。IEEE 組織的 IEEE 802.3 標準制定了乙太網路的技術標準，它規定了包括實體層的連線、電子訊號和媒介存取層協定的內容。乙太網路是目前應用最普遍的區域網路技術，取代了其他區域網路標準如令牌環、FDDI 和 ARCNET。

乙太網路的標準拓撲結構為匯流排型拓撲，但目前的快速乙太網路（100BASE-T、1000BASE-T 標準）為了減少衝突，將能提高的網路速度和使用效率最大化，使用交換器來進行網路連接和組織。如此一來，乙太網路的拓撲結構就成了星型；但在邏輯上，乙太網路仍然使用匯流排型拓撲和 CSMA/CD（Carrier Sense Multiple Access/Collision Detection，即載波多重存取/碰撞偵測）的匯流排技術。

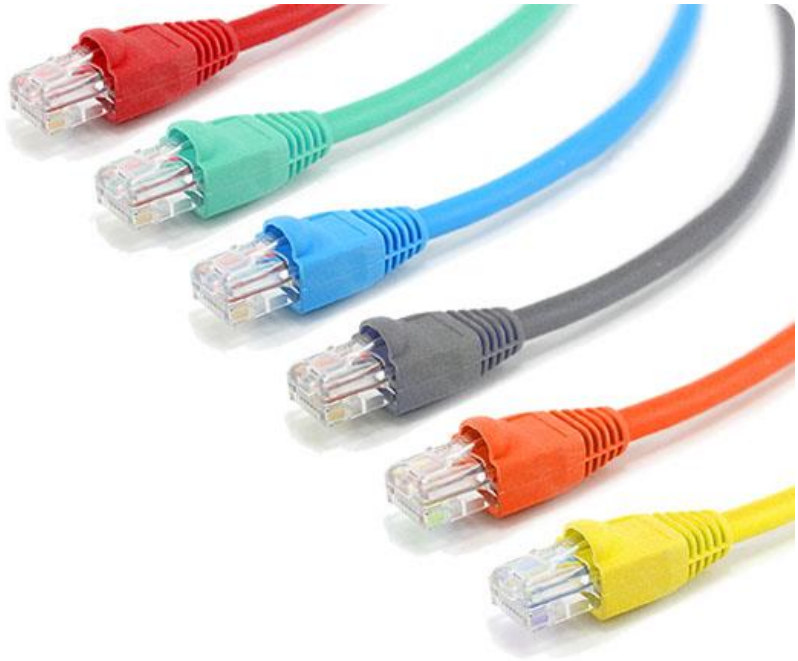
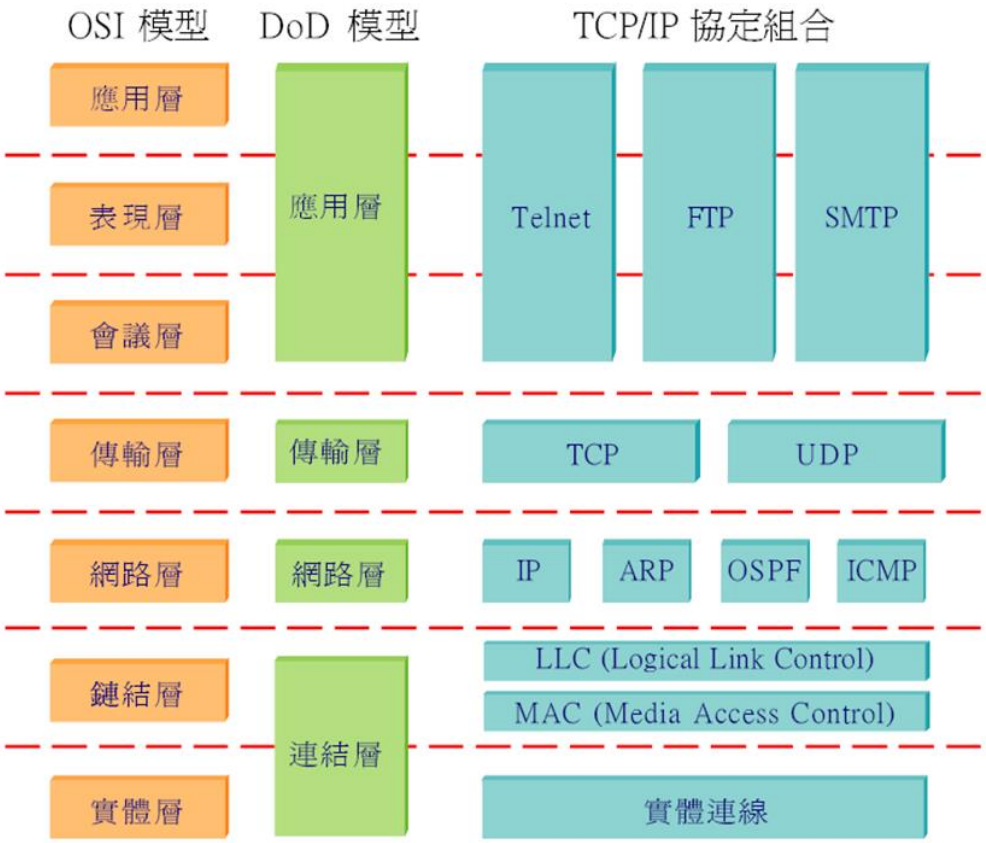


圖 3.乙太網路電纜線

網際網路協定是一個網路通訊模型，以及一整個網路傳輸協定家族，為網際網路的基礎通訊架構。它常被通稱為 TCP/IP 協定套組（英語：TCP/IP Protocol Suite，或 TCP/IP Protocols），簡稱 TCP/IP。因為該協定家族的兩個核心協定：TCP（傳輸控制協定）和 IP（網際網路協定），為該家族中最早通過的標準。由於在網路通訊協定普遍採用分層的結構，當多個層次的協定共同工作時，類似電腦科學中的堆疊，因此又被稱為 TCP/IP 協定疊（英語：TCP/IP Protocol Stack）。這些協定最早發源於美國國防部（縮寫為 DoD）的 ARPA 網專案，因此也被稱作 DoD 模型（DoD Model）。這個協定套組由網際網路工程任務組負責維護。

TCP/IP 提供點對點的連結機制，將資料應該如何封裝、定址、傳輸、路由以及在目的地如何接收，都加以標準化。它將軟體通訊過程抽象化為四個抽象層，採取協定

堆疊的方式，分別實作出不同通訊協定。協定套組下的各種協定，依其功能不同，被分別歸屬到這四個階層之中，常被視為是簡化的七層 OSI 模型。



上圖為 TCP/IP 協定

參考文獻:

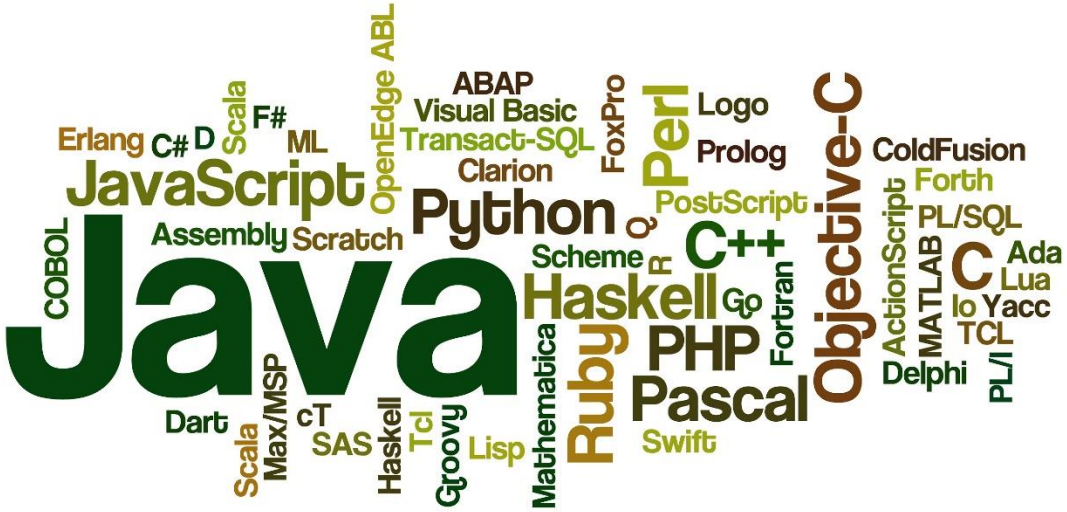
https://zh.wikipedia.org/wiki/%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%BD%91%E7%BB%9C#%E7%BD%91%E7%BB%9C%E5%9F%BA%E7%A1%80%E7%90%86%E8%AE%BA

計算機程式

是用來定義電腦程式的形式語言。它是一種被標準化的交流技巧，用來向電腦發出指令。一種電腦語言讓程式設計師能夠準確地定義電腦所需要使用的資料，並精確地定義在不同情況下所應當採取的行動。

現代電腦內部的資料都只以二元方式儲存，即開-關模式（on-off）。現實世界中代表資訊的各種資料，例如名字、銀行帳號、度量以及同樣低端的二元資料，都經由程式設計語言整理，成為高階的概念。一個程式中專門處理資料的那個系統被稱為程式語言的型態系統（type system）；對型態系統的研究和設計被稱為型態理論（type theory）。語言可以被分為靜態型態系統（statically typed systems），例如 C++和 Java，和動態型態系統（dynamically typed systems），例如 Lisp，JavaScript，Tcl 和 Prolog。前者可被進一步分為包含宣告型態（manifest type）的語言，即每一個變數和函式的型態都清楚地宣告，或 type-inferred 語言（例如 MUMPS，ML）。

真正的程式語言遠比上面列出的這些指令還要嚴謹，程式語言在此就是描述電腦要做事情的一種語言，還是不懂嗎？沒關係，我們舉一些實際的例子，例如作曲家、樂譜、樂團之間的關係，就如程式設計師、程式、電腦之間的關係，作曲家寫下樂譜，用的語言是音符和五線譜，當作曲家將樂譜寫完之後交給樂團，樂團就照樂譜上的指示將樂曲演奏出來，同樣的，程式設計師以程式語言將想執行的任務寫成程式，交由電腦執行。



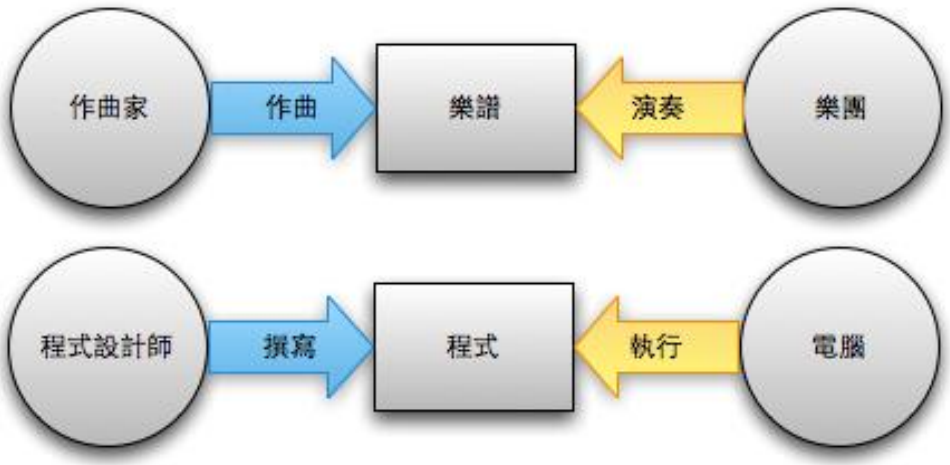
上圖為眾多程式語言的集合

某一種程式語言可能需要較多的時間進行開發，另一種程式語言可能產生的程式碼較長，第三種程式語言可能在 CPU 中執行的時間較長。有些語言在特定的領域仍十分受歡迎，例如 COBOL 在公司的資料中心仍相當常用，多半是在大型電腦上執行，Fortran 常用在科學及工程應用，C 語言常用在嵌入式應用及作業系統中等。

有許多不同的量測方式不同程式語言使用的程度，不同的量測方式也有其各自的誤差：

- 計算徵才廣告中提到各程式語言的次數。
- 計算教授或描述各程式語言書籍賣出的數量。
- 佔計各程式語言目前仍在使用程式碼的長度，不過可能會低估一些公開搜尋不容易找到的程式語言。
- 利用搜尋引擎計算找到不同程式語言的次數。

根據 TIOBE 指數在 2018 年 9 月，十大最受歡迎的語言如下：Java、C、Python、C++、Visual Basic .NET、C#、PHP、JavaScript、SQL 及 Objective-C.。



上圖為程式語言是意圖

參考資料：https://progressbar.tw/posts/5

https://zh.wikipedia.org/zh-tw/%E7%BC%96%E7%A8%8B%E8%AF%AD%E8%A8%80

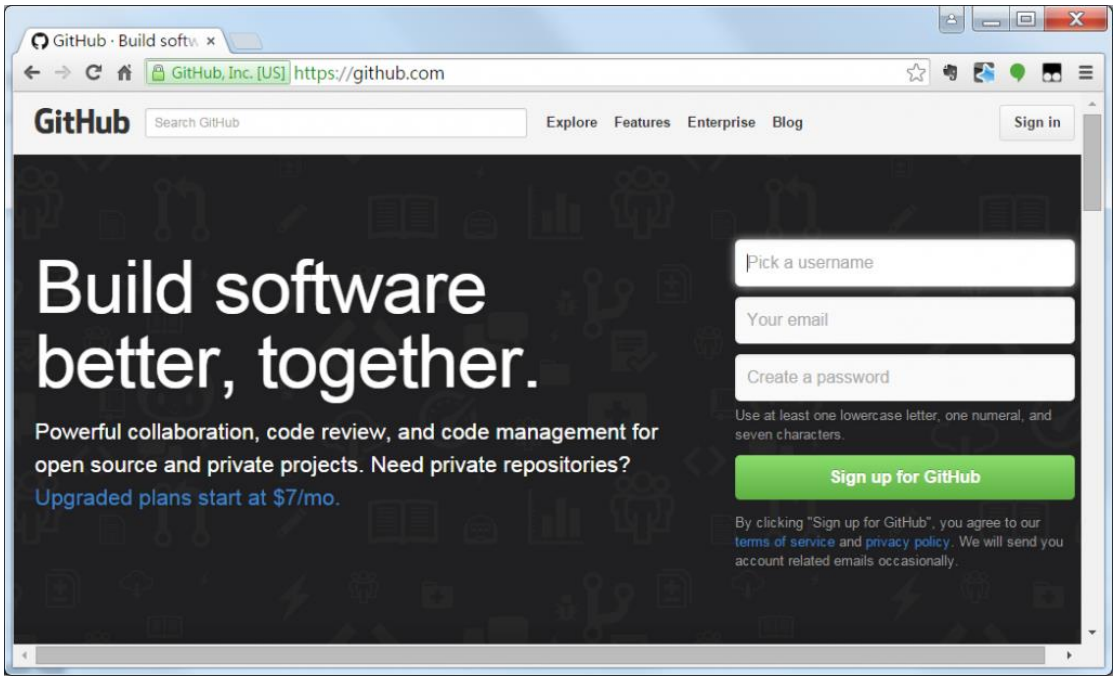
Github

GitHub 平台於 2007 年 10 月 1 日開始開發。網站於 2008 年 2 月以 beta 版本開始上線，4 月份正式上線。

GitHub 裡面的專案可以透過標準的 Git 命令進行存取和操作。同時，所有的 Git 命令都可以用到 GitHub 專案上面。GitHub 開發了針對 Microsoft Windows 和 macOS 作業系統的桌面用戶端。此外，也可以使用第三方外掛程式來實現 Git 功能。網站提供了一系列社群網路具有的功能，例如讚(star)、跟隨(follow)、評論。用戶可以透過複刻(fork)他人專案的形式參與開發，並可透過共同作業示意圖來檢視有多少開發者參與了開發並追蹤最新的複刻版本。此外網站還有 Wiki（透過一個名為 gollum 的軟體實現）等功能。

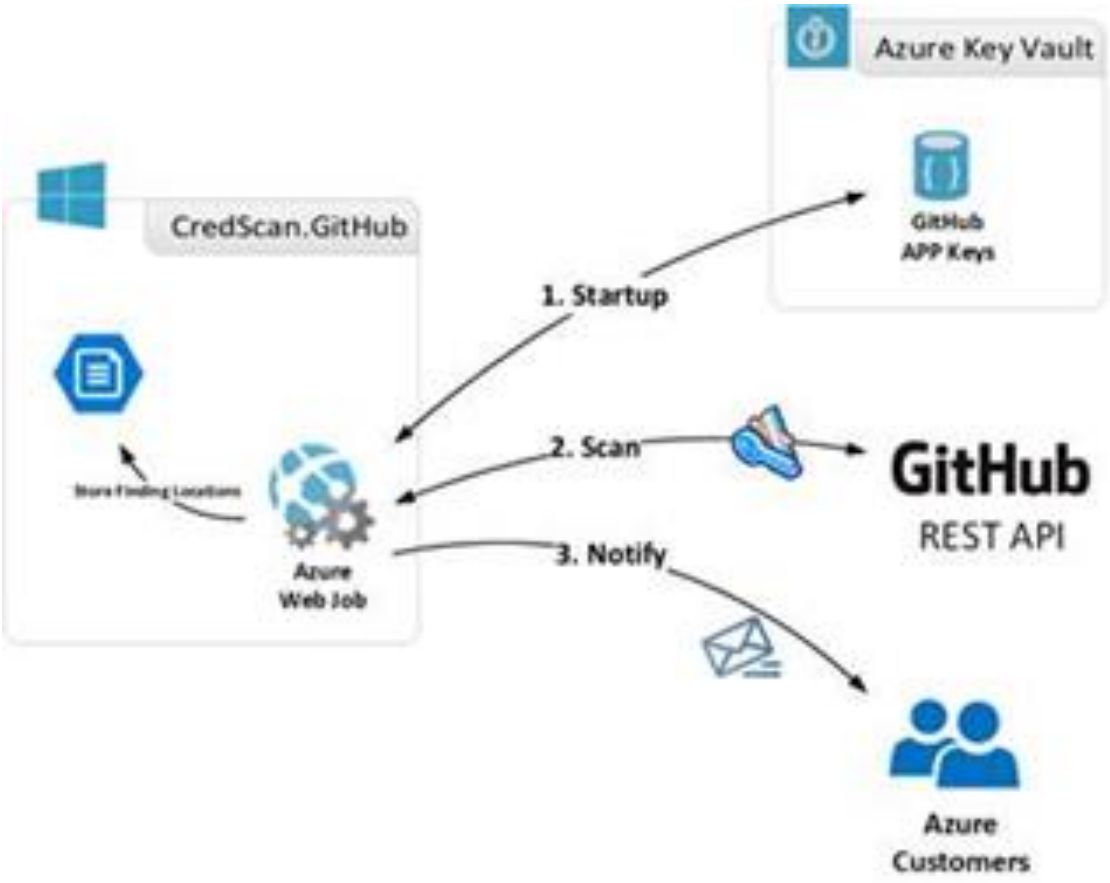
GitHub 同時允許註冊用戶和非註冊用戶在網頁中瀏覽專案，也可以以 ZIP 格式打包下載。但是用戶必須註冊一個帳號然後才能進行討論、建立並編輯專案、參與他人的專案和程式碼審查。

GitHub 是透過 Git 進行版本控制的軟體原始碼代管服務，由 GitHub 公司（曾稱 Logical Awesome）的開發者 Chris Wanstrath、PJ Hyett 和 Tom Preston-Werner 使用 Ruby on Rails 編寫而成。



上圖為 github 網站

GitHub 同時提供付費帳戶和免費帳戶。這兩種帳戶都可以建立公開的程式碼倉庫，但是付費帳戶還可以建立私有的程式碼倉庫。根據在 2009 年的 Git 用戶調查，GitHub 是最流行的 Git 存取站點。[3]除了允許個人和組織建立和存取保管中的程式碼以外，它也提供了一些方便社會化共同軟體開發的功能，即一般人口中的社群功能，包括允許用戶追蹤其他用戶、組織、軟體庫的動態，對軟體程式碼的改動和 bug 提出評論等。GitHub 也提供了圖表功能，用於概觀顯示開發者們怎樣在程式碼庫上工作以及軟體的開發活躍程度。



上圖為 github 的連接系統

參考資料: http://blog.kevinlinul.idv.tw/?p=369

https://zh.wikipedia.org/zh-tw/GitHub

Eric6

eric 是一個用於計算機編程的免費集成開發環境（IDE）。是一個功能齊全的 IDE，默認提供編寫代碼和軟件項目專業管理所需的所有必要工具。是用 Python編程語言編寫的，它主要用於開髮用 Python 編寫的軟件。儘管如此，eric 還支持許多其他編程語言，Ruby 的支持程度幾乎與 Python 一樣高。可用於在 Linux，macOS 和 Microsoft Windows 平台上開發 Python 3 或 Python 2，Qt 5 或 Qt 4 以及 PyQt5 或 PyQt 4 項目的任何組合。eric6 目前提供英語、法語、德語、俄語、捷克語、西班牙語、義大利語、土耳其語和簡體中文的翻譯。

上圖為 eric 的啟動畫面

特色為:

- 同時提供 Python2/3、PyQt4/5 的開發環境，開發者可以根據實際需求自由選擇；
- 與版本控制系統整合，目前版本內建支援 Mercurial 和 Subversion，可通過可選外掛程式支援 Git；
- 與 Qt 整合，可以直接呼叫 Qt Creator 中包含的 Qt 設計師、Qt 語言家等 Qt 開發工具；
- 在 IDE 窗口內提供了內建的通訊工具，包括一個 IRC 用戶端和一個輕量級的共同作業工具；
- 提供豐富而實用的獨立小工具，如截圖工具、網頁瀏覽器、SQL 資料庫瀏覽器、迷你文字編輯器、圖示編輯器、系統托盤圖示、等等。

參考資料: https://zh.wikipedia.org/zh-tw/Eric_Python_IDE

https://tw.saowen.com/a/76f2a9d36bcd5f338ce2508e2f962e8cb6e0c77f55090e133a01ea74c28c00a2

Leo Editor

Leo 是一個開源文本編輯器/大綱，其特色是 clone（大綱節點的虛擬副本）作為組織、導航、定制和腳本的核心工具。Leo 已經積極開發了 20 多年，擁有一群活躍的開發人員和用戶。

Leo 是：

- 功能齊全的 IDE，具有許多受 Emacs 啟發的功能。
- 大綱。Leo 的一切都是大綱。
- 數據管理員和個人信息管理員。
- 強大的腳本環境。
- 用於組織和學習計算機程式的工具。
- 通過簡單的插件架構擴展。
- 與 IPython、Vim 和 Emacs 配合使用的工具。

語言:

Leo 可以用任何人類或計算機編程語言，例如:Python、C、C ++和 Java 的操縱文本或代碼，因為 Leo 是一種獨立於語言或“適應性強的 LPE”（有文化的編程環境）。Leo 是用 Python 編寫的，可以使用 Python 編寫的插件進行擴展。GUI 使用 Qt 工具包；語法識別編輯器基於 Scintilla。Leo 輪廓存儲為 XML 文件。

數據結構 (Trees)、複製 (clones)和視圖 (views):

除了文本編輯之外，Leo 最重要的功能是大綱的功能，通過 GUI 和鍵盤命令通過拖放實現用於構建和管理輪廓的各種便利功能。Leo 的大綱窗格顯示了一個數據節點樹。節點包含標題、正文和其他信息。標題即作為正文的描述。例如，@file 節點是標題以@file 開頭的節點。Leo 數據結構(Trees)實際上是個”有向無環圖”；意指節點可能有多個父節點。Leo 稱這樣的節點為複製(clones)。複製顯示在大綱窗格的多個位置。查看只是其子節點包含複製的節點。單個輪廓

可以包含其中包含的節點的任意多個視圖(views)。

外部文件:

根標題以@file、@auto 或@edit 開頭的數據結構在文件系統上創建外部文件。以下是一些示例標題：

```
@file myClass.py
@auto ../graphics/circles.cpp
@edit~ / .leo / .leoID.txt
```

如您所見，這些節點指定文件名，它可以是絕對路徑或相對於包含 Leo 輪廓的目錄的路徑。

腳本 (Scripting):

Leo 的大綱或層次結構不同於與經典文字編輯工具相關的交錯程序和文檔“塊 (chunk)”的網絡。任何 Leo 節點的正文可能包含一個 Leo 腳本，一個在 Leo 大綱的上下文中執行的 Python 腳本。一個簡單的 API 使 Leo 腳本可以完全訪問已加載輪廓中的所有數據，以及對 Leo 自己的源代碼的完全訪問權限。API 包括 Python 迭代器，允許腳本輕鬆遍歷輪廓。腳本可以由任何節點數據結構組成。

參考文獻

http://leoeditor.com/
https://en.wikipedia.org/wiki/Leo_(text_editor)

Onshape

現代 CAD。
從任何地方做最好的工作。
沒有崩潰。沒有數據丟失。沒有設計僵局。

上圖表示 onshap 可在多樣化的的硬體上使用

Onshape 的現代 CAD 系統使工程師能夠專注於做最好的工作。與舊的 CAD 系統不同，Onshape 將建模工具和設計數據管理結合在一個可在任何設備上訪問的安全雲工作空間中，從不丟失數據，並消除了設計僵局。

Onshape 特色

1.讓使用者更易上手的建模策略

Onshape 是用規劃圖來建立 2D 的概念，而不是單一斷面，這表示使用者可以在一張 2D 圖上看出各個面向的設計，這對機構工程師是基本能力，但對大部份沒有受過圖學訓練的一般使用者，要他們把一空間的物件拆成單一的幾何元素，再組合成一個物體，才會達到需要的外型，很有難度，但這就是繪製 3D 需要做的事。

3D 建模大部份是由 2D 截面長成 3D，這時 2D 的邊界就要很明確，因長出 3D 的範圍就是由線所建立的封閉面積，但 Onshape 改用類似塗色塊的方式，選出要長肉的區塊。

2. 線上協同設計，特別適合使用者

協同設計不是新觀念，在很多軟體上早就有這樣的設計模式，Onshape 也強調自己是一個協同設計平台，提供的功能與前述軟體差不多，在操作上規劃了方便的工具或指令，以一般使用者在機構設計的使用上是該夠用。

不過，因為 Onshape 是在網路平台上使用的系統，所以在做產品設計時，當

電子零件的數目跟資料量變多的時候，操作起來會比較慢，這時就要看自己的網路速度快慢。不過也有解決的辦法，就是把連在一起的電子零件畫成一件，而細節略過不畫，那這問題差不多就可以解決了。

3. Maker 可得到專業的支援

現在使用者做出的創意作品非常多樣，以最近很夯的四軸直升機來說，如有特殊的需求，例如航空動力學或結構分析，一般這種資料在網路上也找的到，但真要做分析的話，還是必須使用分析軟體會比較精準，但大部份使用者應該沒有財力去購買這類軟體。

而 Onshape 這個推出算很年輕的平台，在網路上已可找到不少直接支援用戶進行這些分析的資源，這部分可以去看 YouTube，上面有不少教學影片，這使得使用者在完成自己的想法時，可以得到專業的支援，減少關於技術方面的問題。

參考文獻

https://www.onshape.com/
https://makerpro.cc/2016/10/why-onshape-important/

遊戲製作

創建一個新的應用程序類

對於本教程的第一部分，我們希望 App 通過創建一個名為從標準類繼承其基本行為的全新應用程序類 MyApp 來自定義標準 App 類的行為。

將以下代碼段粘貼到該 myapp = ...行之前：

```
class SpaceGame(App):
    """
    Tutorial4 space game example.
    """
    def __init__(self):
        super().__init__()
```

然後剪切創建背景的四條線並將它們粘貼到 super()...線下方並縮進它們以匹配。最後，改變說零件myapp.width 和 myapp.height 是 self.width 和 self.height 分別。現在，您的新代碼應該如下所示：

```
class SpaceGame(App):
    """
    Tutorial4 space game example.
    """
    def __init__(self):
        super().__init__()
        # Background
        black = Color(0, 1)
        noline = LineStyle(0, black)
        bg_asset = RectangleAsset(self.width, self.height, noline, black)
        bg = Sprite(bg_asset, (0,0))
```

是的，你可以從一開始就粘貼它，但我希望你能清楚這段代碼的來源。

有關變化的注意事項：

- class SpaceGame(App):定義一個名為的新類，SpaceGame 它繼承了標準 App 類的所有功能。
- 下一行定義__init__了類的方法。在這種情況下，它不期望任何參數。

- 該 super().__init__(...行強制新的 SpaceGame 類__init__在開始自己的初始化之前調用標準 App 類的函數。如果您希望新類完全繼承父類的行為，請始終執行此操作。
- 最後，由於此代碼初始化遊戲，因此__init__在遊戲類的方法中放置用於創建黑色背景的代碼是有意義的。

目前，您的計劃已經破產。為了使新的 SpaceGame 類生效，我們必須實例化它而不是實例化 App 類。將程序的下一行更改 myapp = App()為myapp = SpaceGame()。
嘗試運行該程序。你應該看到黑色背景。

創建一個新的 Sprite 類:

在 SpaceGame 類定義的上方，粘貼這個新代碼：

```
class SpaceShip(Sprite):
    """
    Animated space ship
    """
    asset = ImageAsset("images/four_spaceship_by_albertov_with_thrust.png",
        Frame(227,0,65,125), 4, 'vertical')

    def __init__(self, position):
        super().__init__(SpaceShip.asset, position)
```

運行你的程序。它不應該做任何與以前不同的事情。創建一個新的 Sprite類實際上並不創建任何 sprite。它所做的就是創造一個製作精靈的藍圖。

通過在 SpaceGame__init__方法的末尾添加以下行來添加單個 SpaceShip 精靈（當然，正確縮進）：

```
        SpaceShip((100,100))
```

現在運行你的代碼。涼。嘗試在 SpaceGame__init__方法的末尾添加一些 SpaceShip 實例：

```
        SpaceShip((150,150))
        SpaceShip((200,50))
```

看起來你正在建造一支艦隊！

我們添加的代碼非常簡單，但有一行需要一些解釋：

```
asset = ImageAsset("images/four_spaceship_by_albertov_with_thrust.png",
        Frame(227,0,65,125), 4, 'vertical')
```

該 asset 變量被創建的類中，但在任何方法以外。這使它成為一個類屬性，可供該類的所有實例使用。我們用它來調用父類雪碧__init__使用語法的方法：SpaceShip.asset。這種方法允許我們根據需要創建盡可能多的 SpaceShip 實例，但不創建多個資產。有一個物體代表宇宙飛船圖像，但有多個物體代表精靈。

這個創建一個 ImageAsset 參數的調用比我們在上一個教程中使用的參數多。這是他們的意思：

- 所述 Frame(227,0,65,125)參數指定的矩形截面內的圖像文件。如果你看一下 Github 中的圖像文件，你會發現它實際上由十六個不同的航天器圖像組成，一些有火箭推力，有些沒有。幀參數指的是我們想要的子圖像左上角的水平和垂直位置（227 和 0 像素），後面是它的寬度和高度（65 和 125 像素）。
- 該 4 論證意味著該資產實際上將包括與第一個相同大小的四個子圖像，並且.....
- 該'vertical'參數意味著這四個圖像垂直排列在圖像文件中。回到 github 存儲庫，看看這個圖像文件，看看我的意思是“四個圖像.....垂直排列”。

所有這些附加信息意味著此資產已準備好動畫！它由一個沒有推力的宇宙飛船和三個包含爆炸推力的宇宙飛船圖像組成。通過選擇我們想要在任何給定時間顯示哪些幀，我們可以在精靈本身內給出運動的外觀。

動畫一步！：

首先，讓我們為 SpaceShip 類添加一些屬性。在 SpaceShip 類__init__方法的末尾添加以下行：

```
        self .vx =   1
        self .vy =   1
        self .vr =   0.01
```

這些將設置初始水平，垂直和旋轉速度。

然後，step 向 SpaceShip 類添加一個方法。這應該出現在 SpaceShip 類__init__方法之後（但在它們之間留一個空格）：

```
    def step(self):
        self.x += self.vx
        self.y += self.vy
        self.rotation += self.vr
```

這將剛才添加的速度下精靈的位置屬性x，y 和 rotation，它們是被自動 Sprite 類的內置屬性繼承由飛船類。
如果您不確定粘貼這些代碼段的位置，請查看本頁末尾的完整列表。

不幸的是，只是向 stepsprite 類添加一個方法並不意味著它將被調用。所以我們必須為 step 應用程序本身添加一個方法。__init__在 SpaceGame 類的方法下面添加以下代碼（但在它們之間留一個空格）：

```
def step(self):
    for ship in self.getSpritesbyClass(SpaceShip):
        ship.step()
```

您需要為 step 自己的標準 App 類自定義添加方法。這種 step 方法是自動與遊戲中的每個視頻幀更新調用。
此方法體使用 for 循環來訪問 SpaceShip 類的每個實例，然後調用其 stepmethod（ship.step()）。由於 step 每次視頻幀更新都會調用 SpaceGame函數，這意味著每次更新視頻幀時 step 都會調用每個 SpaceShip函數。

更改精靈圖像:

現在來看動畫細節。當用戶按下空格鍵時，我們希望推力圖像具有動畫效果。所以這是我們要做的事情：

- 聽當空間按鈕被按下了下來。
- 釋放空格按鈕時監聽。
- 使用該 step 方法更改精靈圖像，具體取決於空間是否已關閉或已釋放。

首先，讓我們添加用於管理推進狀態的代碼，並聽取相應的密鑰。在 SpaceShip__init__方法的末尾添加以下內容：

```
        self .thrust =  0
        self .thrustframe =  1
        SpaceGame.listenKeyEvent（ “ keydown ” ，“ space ” ，self .thrustOn）
        SpaceGame.listenKeyEvent（ “ keyup ” ，“ space ” ，self .thrustOff）
```

然後將 thrustOnand 和 thrustOff 方法添加到 SpaceShip 類中。在 SpaceShipstep 方法之後立即添加以下內容：

```
def  thrustOn（self，event）：
     self .thrust =  1

def  thrustOff（self，event）：
```

```
            self .thrust =    0
```
這些簡單的功能將跟踪空格鍵是否向下（推力為 1）或向上（推力為 0）。

最後，在 SpaceShipstep 方法的末尾添加以下內容：

```
        # manage thrust animation
        if self.thrust == 1:
            self.setImage(self.thrustframe)
            self.thrustframe += 1
            if self.thrustframe == 4:
                self.thrustframe = 1
        else:
            self.setImage(0)
```

如果 self.thrust 設置為 1，則表示按下空格按鈕，精靈圖像設置為任何 self.thrustframe 值（記住我們在類__init__方法中將其初始化為 1）。圖像編號 0 是第一個圖像，1 是第二個圖像，依此類推。接下來的三行增加了推力幀屬性，檢查它是否超出了我們的圖像列表的末尾（只有三個），並在必要時將其設置為 1。

最後，如果 self.thrust 設置為 0，則意味著空間按鈕被釋放，我們應該只顯示無推力的太空船圖像 self.setImage(0)。

最後細節

我們的遊戲還有很多改進，但這些改進留給學生練習！

你可能已經註意到宇宙飛船的精靈以一種非常奇怪的方式旋轉。這是因為精靈的默認“中心”實際上是它的左上角。您可以通過設置改變中心 fxcenter 和 fycenter 雪碧的屬性（或在我們的情況下，飛船）類。通過將此最後一行添加到 SpaceShip__init__方法來執行此操作：

```
        self .fxcenter =    self .fycenter =    0.5
```
再次運行你的程序，享受它的精彩！

問題：

1. 擴展教程以使用'left arrow'和'right arrow'鍵左右旋轉船隻。
2. 擴展教程以使用 AWSD 鍵來控制船舶運動。
3. 高級：擴展教程以創建 Blast 使用文件夾中 blast.png包含的圖像的精靈 /images。使用該'enter'鍵在屏幕上隨機位置創建 Blast 精靈（或從太空飛船精靈“射擊”）。
4. 高級：擴展教程以動畫 Blast 精靈。

參考文獻:
https://github.com/HHS-IntroProgramming/Space-Shooter