指令式程式設計 基礎概念

本投影片圖片內容取材自 scratch.mit.edu

指令式程式設計

- 每個程式是由一連串的指令構成。
- 甚麼是程式?
 - 描述完成一件事情的方法。
- 甚麼是指令?
 - 一件可以做得到的事。
- 生活中的範例:
 - 食譜
 - 摺紙
 - 電視節目表

指令的樣貌

• 通常是個會改變狀態的動作



- 會有受詞
 - 拿起麥克風



- 可能還有副詞
 - 向前走 10 步



- 偶而還會有主詞
 - Scratch Cat 順時鐘轉 15 度





- 機器能夠做的指令
 - 簡單、規範明確、機器可以快速反覆進行
 - 做一件複雜的事情要透過許多指令來完成

Scratch 積木的分類:看造型



流程控制

- 指令式程式設計的核心,是流程設計。
- 流程控制的結構:
 - 循序
 - 選擇
 - 重複
- 事件驅動程式設計
 - 事件
- 程序式程式設計
 - 子程序

循序

一個指令做完之後,就做下一個指令,直到沒有 指令為止。

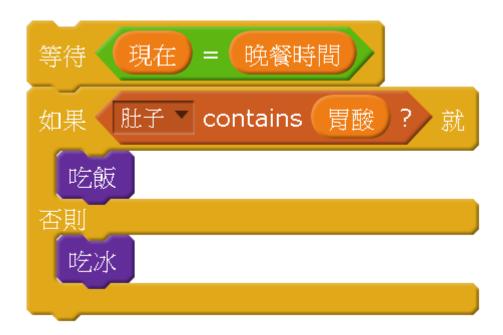


• 我們都沒有下一個,完了就完了



選擇

• 依據某個條件是否成立,決定要不要做某些事。



- 當你要做一件事情很多次的時候:畫個 5 邊形
- 用複製的好像還行
- 複製幾次就行了?

```
72
移動
  (~ 旋轉 72 度
移動 72 步
  (* 旋轉 72 度
移動 (72) 步
  (* 旋轉 72) 度
移動 72 步
  (* 旋轉 72) 度
移動 72 步
  (~ 旋轉 72 度
```

- 當你要做一件事情很多次的時候: 畫個 10 邊形
- 用複製的好像還行
- 複製幾次就行了?
- 看起來有點多
- 可是一個螢幕還是裝得下

```
移動 (36) 步
  (聲 旋轉 36) 度
移動 (36) 步
  【 旋轉 36 度
移動 36 步
  【 旋轉 36 度
移動 (36) 步
  【 旋轉 36 度
移動 (36) 步
  € 旋轉 36 度
移動 (36) 步
  【 旋轉 36 度
移動 (36) 步
  【 旋轉 36 度
移動 (36) 步
  【 旋轉 36 度
移動 (36) 步
  (* 旋轉 36 度
移動 (36) 步
  【 旋轉 36 度
```

- 當你要做一件事情很多次的時候: 畫個 360 邊形 (看起來很圓)
- 能夠一樣用之前的方法兜嗎?
 - 可以
- 可是一個螢幕還是裝得下嗎?
 - 我覺得有困難
- 你知道你到底重複了幾次嗎?
- 所以你該用看看重複結構



- 有些事情你必須在程式結束之前,不斷地做,那就需要無窮迴圈。
- 當你需要畫個類似奧迪廠徽的圖案時,你可能會需要雙層迴圈。
- 當你需要進行到特定條件成立, 像是 Scratch Cat 碰到滑鼠指標, 可以這樣做。

```
當 被點一下

重複執行直到 touching 滑鼠游標 → ?

等待 0.05 秒

面向 滑鼠游標 → 移動 10 步

說 我抓到老鼠了!
```



```
當 被點一下
隱藏
移到 x: -150 y: 100
面向 90 方向
清除所有筆跡
重複 4 次
下筆
重複 360 次
移動 1 步
向 (* 旋轉 1 度
停筆
移動 80 步
```

事件驅動

•程式的起點:事件



運算

- 不會直接改變狀態的動作
- 依據資料推算出新的資料
- 可以是構成指令的一部分





常見語法語意陷阱

- a < b < c 是否成立:-3 < -2 < -1
 - 語法跟語意上都跟上面的數學式吻合, 結果也一樣



• 語法上看起來一樣,但語意上不同!結果也不同



- 【 | 表示:請不要餵我吃長得很尖銳的怪東西。
- 學習一種程式語法語意,就能用它來做程式設計。

子程序

- 簡化流程
- 增加可讀性
- 好維護

```
被點一下
           初始化
           重複 4 次
             畫圈圈
             移動 80 步
定義 畫圈圈
下筆
重複(360)
               定義 初始化
 移動 1 步
   ( 旋轉 1 度
               隱藏
               移到 x: -150 y: 100
停筆
               面向 90▼ 方向
               清除所有筆跡
```

