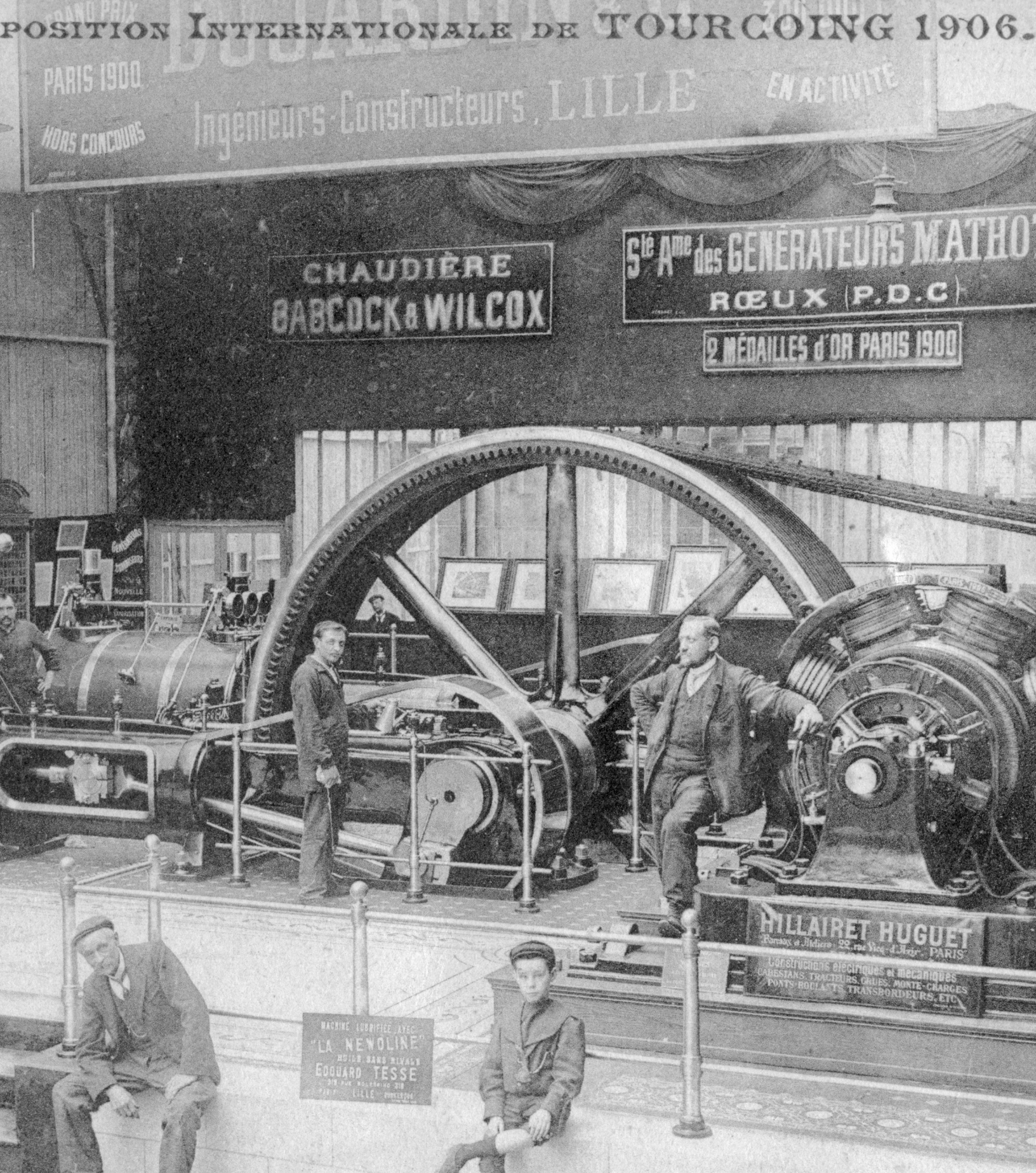
POSITION INTERNATIONALE DE TOURCOING 1906.



La Machine Dujardin donnant la force motric

NEWOLINE » EDOUARD TESSE, huiles. - L

Carte Officielle (déposé). — J. Bauchart, ph

Create Your Own Awesome State Machine

”from design to build”

Patrick Chiu

This book is for sale at http://leanpub.com/createyourownawesomestatemachine

This version was published on 2014-05-09

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

Chapter 1 : State Machine 的定義

「有限狀態機 (Finite State Machine) 是由一組狀態、一個起始狀態、輸入事件、將輸入事件與現在狀態轉換為下一個狀態的轉換函數所組成」

使用狀態機來設計系統，一個重要的觀念是把一個系統分成很多的當下 (狀態)，在那個當下應該要做什麼事，然後什麼事情讓你離開了那個當下，又到另外一個當下。

舉例來說一個哲學家的一天，可以分成三個當下 (狀態)，「睡覺、吃飯、思考」

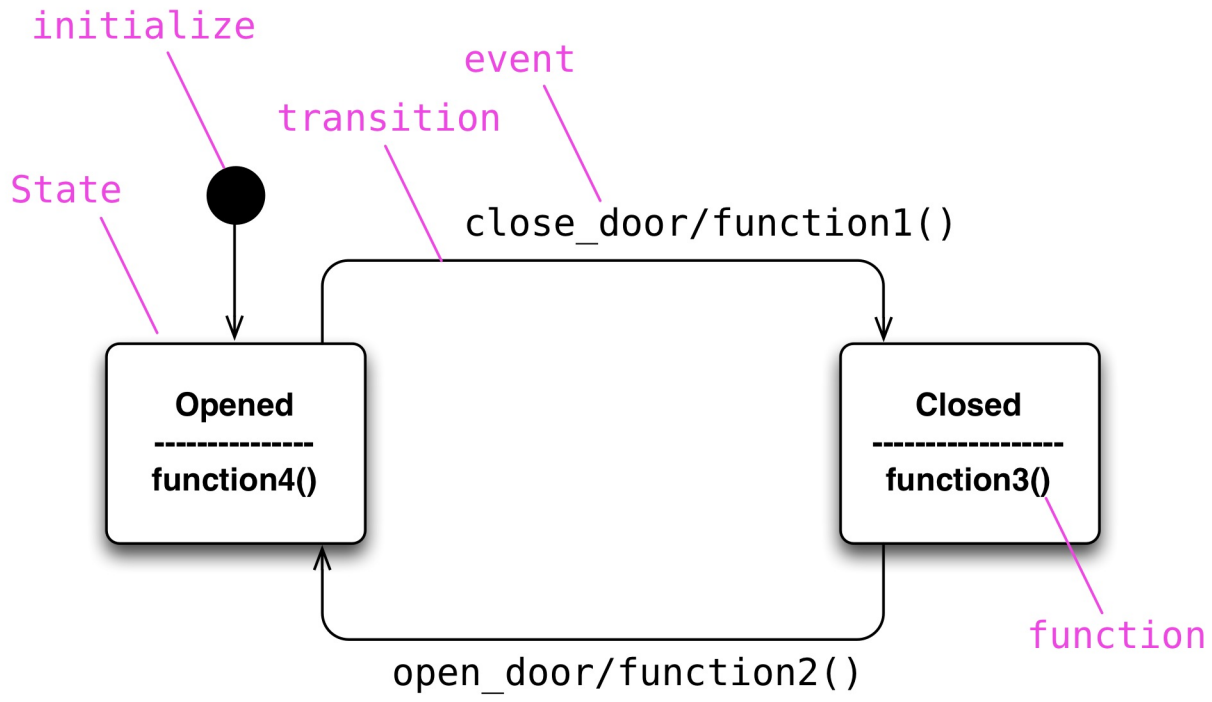
如果一開始哲學家是睡覺 (狀態)，鬧鐘一響 (事件)，起來就吃飯 (狀態)，沒事無聊 (事件)，就思考了起來 (狀態)

相信到這邊，你已經有狀態與事件的觀念了。

所有接下來提到的 State Machine 皆是指有限狀態機

State Machine 實作上最重要的四個元素 events, states, transitions, functions。

State Machine 狀態圖



如上圖，我們看到一個開關門的 state machine，其中

- State(狀態)：指的是系統有哪些可能性的情形，以這個門來說，就只有開門跟關門兩種狀態。上圖中標示方框的部份
- Initialize(一開始的狀態)：要定義一個系統剛開始的狀態。上圖中黑色實心圓加上箭頭
- Transition(轉態)：當在一個狀態下，被某個事件觸發，而轉向另外一個狀態的。上圖中在兩個狀態中互相指向對方的線
- Event(事件)：事件乃是觸發轉態的的條件。上圖中會從 state(Opened) 轉向 state(Closed) 的原因是 event(close_door) 被觸發
- Function(程式功能)：在轉態或是狀態上，都是可以附加程式功能 (function)，也就是你希望在這個時候電腦程式要做什麼。在上圖中，function1(), function2() 都是轉態過程中會被呼叫的程式功能，function3(), function4() 則是在轉至那個狀態時呼叫的程式功能。

所以你可以用這方式來看待狀態圖

- 原本初始的 state(Opened)，經由 event(close_door) 的觸發，轉態至 state(Closed)
- 在 state(Closed) 的狀態下，經由 event(open_door) 的觸發，轉態至 state(Opened)
- 在當前的狀態下，只要看往外轉態的線上的 event 才是會被觸發的，其它的 event 是不作動的，所以假設你的狀態目前是在 state(Opened)，這時收到 event(open_door) 是不作動的，只有在收到 event(close_door) 才會被觸發與轉態。

自我挑戰

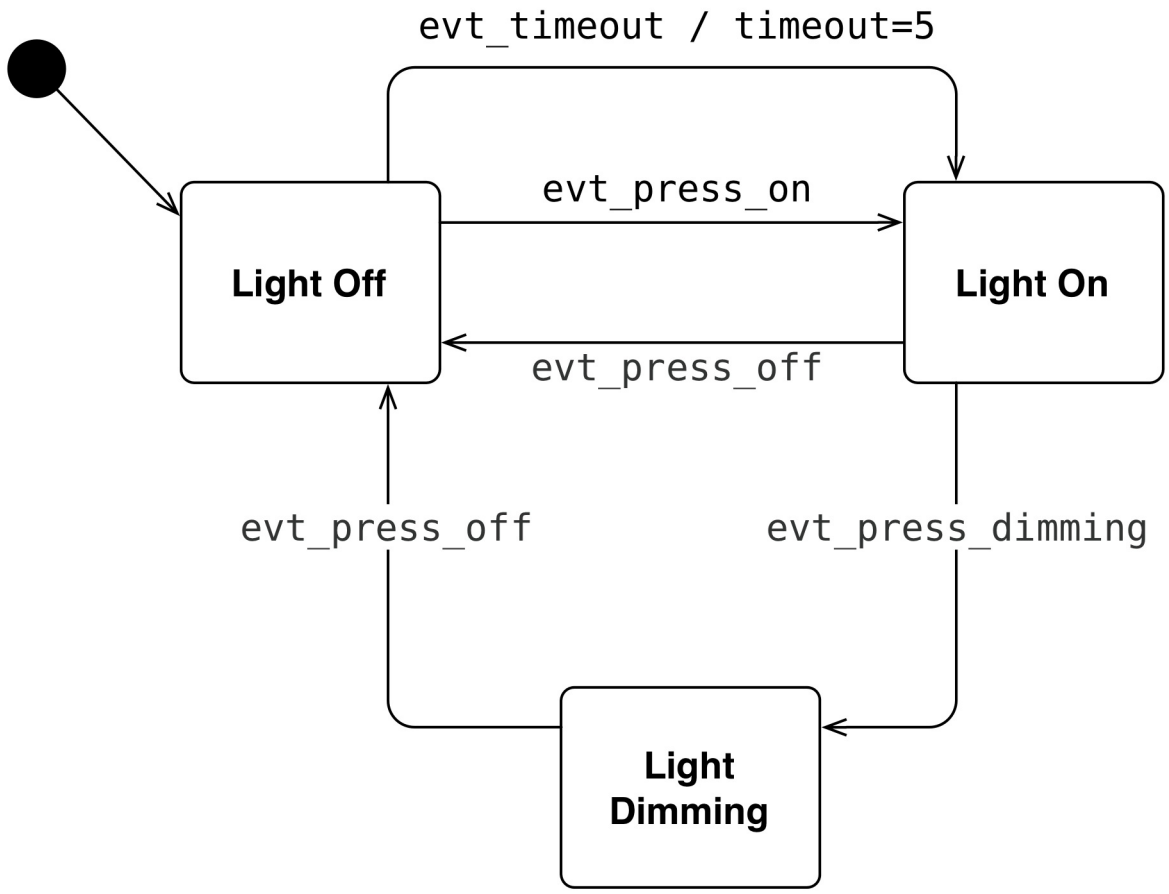
請利用上述技巧，劃出一個狀態機，情境如下

有一個立燈，上面只有一個燈跟三個按鈕 (分別是「開」「關」「閃爍」功能)

1. 一開始立燈是關的：當按下「開」，立燈馬上開，沒按任何按鈕，立燈五秒後也是會自動打開，沒錯！這是一個怪立燈，一插上電話 5 秒後自己會打開。
2. 在立燈開的時候：當按下「關」，立燈馬上關。或是在開的狀態下，按下「閃爍」立燈會立即會閃爍
3. 在立燈閃爍的時候：當按下「關」，立燈馬上關。

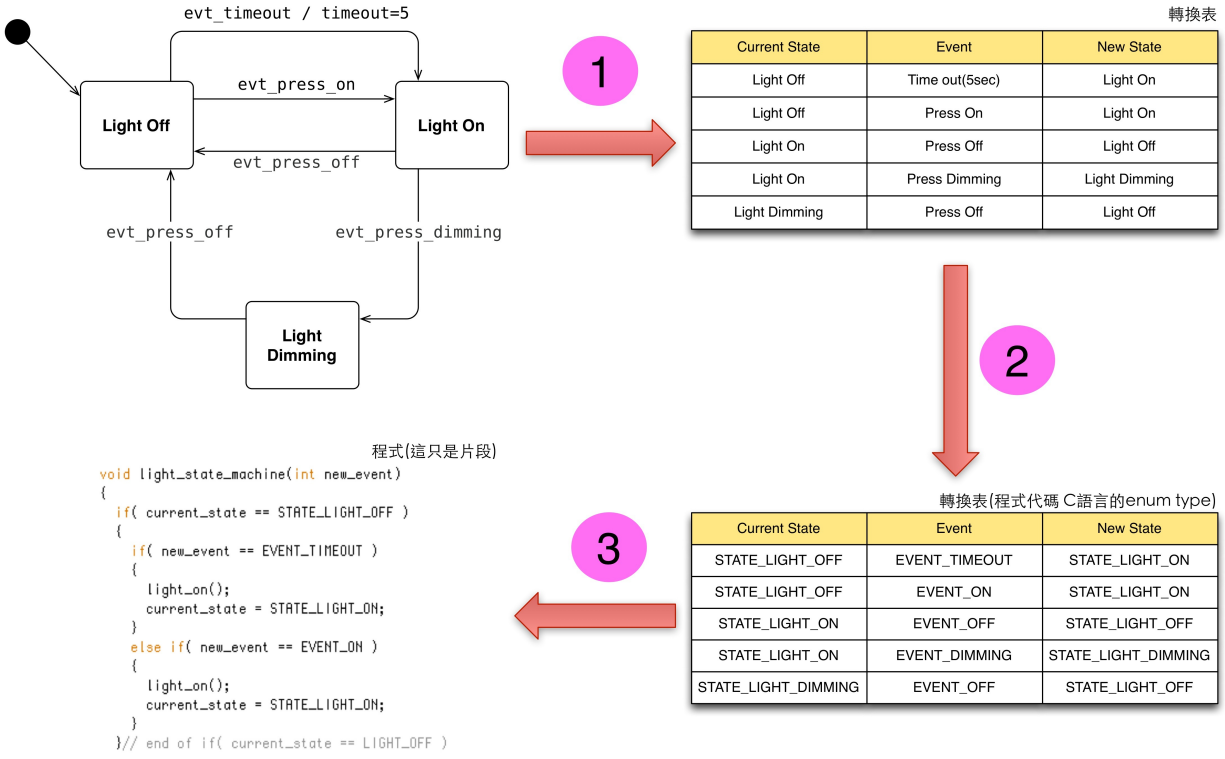
你請可以找一張白紙，開始畫出這個狀態圖

答案如下，你是否跟我的差不多呢?



Chapter 2：狀態圖如何轉成程式

會經由三個步驟 (當你熟的話，就能直接由狀態圖就能轉成程式碼了)



上圖 3 步驟

1.「狀態圖」轉成「轉態表」

表上 Title 你可以看到 Current State 是什麼，經由什麼 Event 轉到 New State。

狀態圖上有五個 Transition，分別對映至轉態表上的五個列上面，所以有幾個 Transition 就會有幾列，列表的方式，方便你檢查。

2.「轉態表」轉成「程式代碼轉態表」

直接定義變數名稱，從「轉態表」對映下來就可以了, 通常 State 前面我會加上 STATE, 而 EVENT 前面會加上 EVENT 或是 EVT 後面的數字，只要不重覆就可以了，因為電腦是認數字來做比較是不同的 State 或是 Event

```
8  // 定義狀態
9  #define STATE_LIGHT_OFF 1       // 關燈
10 #define STATE_LIGHT_ON 2        // 開燈
11 #define STATE_LIGHT_DIMMING 3   // 燈閃爍
12
13 // 定義事件
14 #define EVENT_ON 101            // 觸發「ON」
15 #define EVENT_OFF 102           // 觸發「OFF」
16 #define EVENT_DIMMING 103       // 觸發「Dimming」
```

3. 「程式代碼轉態表」轉成「程式」

我們先看到 light_state_machine_dispatch()，這是 state machine dispath 的重要機制。

你可以看到裏面就是一堆巢狀結構的 if-else，先判斷目前的狀態，再看新的觸發 event 是不是要處理的 event。

如果是需要處理的 event，處理完就轉態至新的狀態 (配合上述的轉態表來設計出來的)。

列 35,53,71 都是判斷目前的狀態 (current_state)，進入該判斷式，會判斷是否有要處理的 Event。

如果有，執行要做的動作，接著將目前的狀態 (current_state) 設定成新的狀態。

```
33 void light_state_machine_dispatch(int new_event)
34 {
35   if( current_state == STATE_LIGHT_OFF )
36   {
37     // 在 OFF 的狀態下，只有 TIMEOUT 跟 ON 這兩個 Event 觸發，才進行處理
38
39      if( new_event == EVENT_TIMEOUT )
40     {
41       light_on();
42       tick = TICK_DISABLE;
43       current_state = STATE_LIGHT_ON;
44     }
45     else if( new_event == EVENT_ON )
46     {
47       light_on();
48       tick = TICK_DISABLE;
49       current_state = STATE_LIGHT_ON;
50     }
51
52   }// end of if( current_state == LIGHT_OFF )
```

```
53   else if( current_state == STATE_LIGHT_ON )
54   {
55     // 在 OFF 的狀態下，只有 OFF 跟 DIMMING 這兩個 Event 觸發，才進行處理
56
57     if( new_event == EVENT_OFF )
58     {
59       light_off();
60       tick = TICK_COUNT;
61       current_state = STATE_LIGHT_OFF;
62     }
63     else if( new_event == EVENT_DIMMING )
64     {
65       light_dimming();
66       tick = TICK_DISABLE;
67       current_state = STATE_LIGHT_DIMMING;
68     }
69
70   }// end of else if( current_state == LIGHT_ON )
71   else if( current_state == STATE_LIGHT_DIMMING )
72   {
73     // 在 OFF 的狀態下，只有 OFF Event 觸發，才進行處理
74
75     if( new_event == EVENT_OFF)
76     {
77       light_off();
78       tick = TICK_COUNT;
79       current_state = STATE_LIGHT_OFF;
80     }
81
82   }// end of else if( current_state == LIGHT_DIMMING )
83 }
```

主程式說明

在主程式中，會有一個 for(;;) 迴圈，裏面會有一個 100 毫秒的 sleep，用來避免消耗無謂的 cpu 效能。

在這個迴圈中，每一次進來，都會去減少一個 tick，直到 tick 被減少至 0 的時候，就會觸發一個 Timeout 的 Event(列 116)。

另外會檢查 User 是否有輸入，有輸入的話 (列 121 至 122)，會把它從輸入的文字轉成 event，然後分派至狀態機中 (列 124 至 134)

```
103   for (;;) {
104
105       // 每 1/10 秒處理一次
106       usleep(100000);
107
108       // 如果 timer 是啟動的, tick 則是每次遞減
109       if( tick >= 0 )
110       {
111         tick--;
112       }
113
114       // tick 已經由 50 次遞減至 0, 則發出 EVENT_TIMEOUT
115       if (tick == 0) {
116           light_state_machine_dispatch(EVENT_TIMEOUT);
117           ShowState(current_state);
118       }
119
120       // 檢查 User 是不是有輸入任何的字元
121       if (__kbhit()) {
122           switch (__getch()) {
123               case 'o': { // User 按了 ON
124                   light_state_machine_dispatch(EVENT_ON);
125                   break;
126               }
127
128               case 'f': { // User 按了 OFF
129                   light_state_machine_dispatch(EVENT_OFF);
130                   break;
131               }
132
133               case 'd': { // User 按了 Dimming
134                   light_state_machine_dispatch(EVENT_DIMMING);
135                   break;
136               }
137               case '\33': { // User 按了 ESC
138                   printf("\nESC : exit");
139                   _exit(0);
140                   break;
141               }
142           }
143           ShowState(current_state);
144       }
```

範例執行：

如果是在 Linux 跟 MAC OS X 底下，請打開你的終端機，底下黑色斜體字的部份，就是輸入與程式互動的部份

Windows 底下使用 Code::Blocks，只要按 F9(Build & Run) 就可以執行了

你可以輸入 o，模擬按下 on

輸入 f，模擬按下 off

輸入 d，模擬按下 dimming

在不同的狀態下，你分別都可以按下'o','f','d'，對照狀態圖看，看它是不是在該處理的狀態下才會作動，其它狀態一率是不理你的

「中括號」秀的是目前「State Machine 的狀態」，「星號」開頭的是「機器底層會執行的事情」，目前使用文字的方式做回應而已

```
$ ./stand_lamp
Stand Lamp
Press 'o' for push on button
Press 'f' for push off button
Press 'd' for push dimming button
Press 'Esc' to exit.
[LIGHT_OFF]
*Light On
[LIGHT_ON]
f
*Light Off
[LIGHT_OFF]
o
*Light On
[LIGHT_ON]
d
*Light Dimming
[LIGHT_DIMMING]
```