在2D平台使用的find path

* 使用 find path 工具匯出 Path array (如C++ array)
* 敵人的行動模式「巡邏」及「路徑」
* 巡邏 --- 在指定區域往來行走
* 路徑 --- 主動決定目的位置，並制定及遵循路徑走到位置
* 範例示範在2D平台路徑行走，省略巡邏部份

重新思考行動

路徑

段落

已到達

核心使用state machine方法，僅用3個狀態建立AI：

**思考行動**

指示要巡邏或按路徑走到目的位置

**執行路徑**

例如路徑 (0🡺16)：0⏵3⏵4⏵9⏵10⏵16

把路徑分成段落，安排順序執行

如走到路徑終點 🡪 返回思考state

**執行段落**

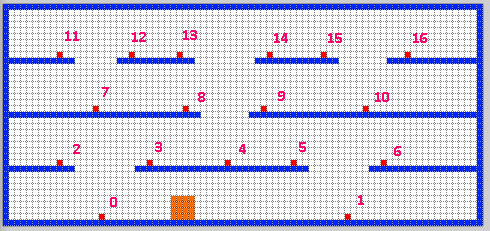
例如：

* (0⏵3) 行走方法 (先跨跳，再行走)
* (3⏵4) 行走方法 (直接行走)

行走方法是按開始、目的位置而定 (如：直走，跨跳，墮下)

如到達目的位置 🡪 返回路徑state

如走錯路 🡪 返回思考state



思考結果：由0至16

路徑：0⏵3⏵4⏵9⏵10⏵16

各個段落：(0⏵3) 🡪 (3⏵4) 🡪 (4⏵9) 🡪 (9⏵10) 🡪 (10⏵16)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | ⏵ | 3 | ⏵ | 4 | ⏵ | 9 | ⏵ | 10 | ⏵ | 16 |

**路徑、段落、位置點、行走方法**

路徑：例如由0到16的

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 3 | 4 | 9 | 10 | 16 |

段落：路徑段落及位置點(共5組)

|  |  |
| --- | --- |
| 由 | 到 |
| 0 | 3 |
| 3 | 4 |
| 4 | 9 |
| 9 | 10 |
| 10 | 16 |

段落位置之間的行走方法

|  |  |  |
| --- | --- | --- |
| 由 | 到 | 行走方法 |
| 0 | 3 | 向目標跨跳 |
| 3 | 4 | 直接行走 |
| 4 | 9 | 向目標跨跳 |
| 9 | 10 | 直接行走 |
| 10 | 16 | 向目標跨跳 |

0⏵3⏵4⏵9⏵10⏵16