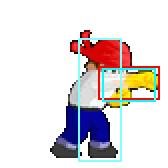
雙itr雙bdy的使用：  
雙itr及雙bdy的使用時機有兩種：  
１調整範圍  
由於座標x: y: w: h: zwidth: 所構成的區域是方形的  
對於一些有弧形的圖案，若用一個bdy概括全部，在角落就會被攻擊就變得不太常理  
因此就會用兩個以上的bdy來修正區域  
如圖，這是bandit的punch動作61  
bdy:  
  kind: 0 x: 39 y: 19 w: 21 h: 61  
bdy:  
 kind: 0 x: 51 y: 34 w: 26 h: 16  
  
青色的部分是bdy，分佈在身體和拳頭上；紅色部分是itr  
   
２指定到不同動作  
如下：  
bdy:  
kind: 1070 x: ...  
bdy:  
kind: 1075 x: ...  
此id為300時，就可以用kind決定被打的動作  
第一個bdy被打時就跳到動作70，第二個bdy被打時就跳到動作75…  
   
同理，itr也可用catch來指定動作  
只要在不同的頻道使用不同的catchingact:及caughtact:，就能在某時機下將人指定到不同動作  
另外還可以用fall或injury來指定動作，這就請自行鑽研囉

決定事件：  
type 5的6個動作，可以用不同的fall來指定  
type 3的動作，可以用不同的injury來指定，而其中還關係著hit\_a, hit\_d的使用  
   
後決定事件  
如下，當物件被攻擊後，只要透過特殊方式，就可知道injury數值中是要它跳到哪個動作：  
frame 20 next: 21  
frame 21 next: 22 wait: 0 hit\_a: 20 hit\_d: 50  
frame 22 next: 23 wait: 0 hit\_a: 20 hit\_d: 51  
frame 23 next: 24 wait: 0 hit\_a: 20 hit\_d: 52  
frame 24 next: 25 wait: 0 hit\_a: 20 hit\_d: 53 ...  
   
↑以此為例，攻擊力為480，就會跳到動作50；攻擊力為460，就會跳到動作51…  
只是，放在越後面的事件，則需要花點時間才能得到結果  
   
先決定事件  
開頭就已知要發生哪個事件  
一開始先用hit\_a: 扣好血量，然後再用一個攻擊力0的攻擊來做觸發  
再照著上述的方式設計，就可以決定要跳到哪個動作  
起點在60 ( id: 251 type: 3 )

<frame> 20 event

pic: 999 state: 15 wait: 1 next: 21 hit\_a: 1 hit\_d: 30

<frame\_end>

<frame> 21 event1

pic: 999 state: 15 wait: 0 next: 22 hit\_a: 20 hit\_d: 80

<frame\_end>

<frame> 22 event2

pic: 999 state: 15 wait: 0 next: 23 hit\_a: 20 hit\_d: 82

<frame\_end>

<frame> 23 event3

pic: 999 state: 15 wait: 0 next: 24 hit\_a: 20 hit\_d: 84

<frame\_end>

<frame> 24 event4

pic: 999 state: 15 wait: 0 next: 30 hit\_a: 20 hit\_d: 86

<frame\_end>

<frame> 30 dis

pic: 999 state: 9998 wait: 0 next: 1000

<frame\_end>

<frame> 60 start

pic: 999 state: 15 wait: 1 next: 61

<frame\_end>

<frame> 61 s

pic: 999 state: 15 wait: 0 next: 62 hit\_a: 440 hit\_d: 30

<frame\_end> #先決定事件為3

<frame> 62 s

pic: 1 state: 10 wait: 10 next: 0

opoint:

kind: 1 x: 0 y: 0 action: 70 oid: 251

opoint\_end:

bdy:

kind: 0 x: -5 y: 2100 w: 10 h: 10

bdy\_end:

<frame\_end>

<frame> 70 find\_man

pic: 999 state: 3005 wait: 2 next: 1000

itr:

kind: 8 x: -5 y: -20 w: 10 h: 10 dvx: 71 vrest: 10

itr\_end:

<frame\_end>

<frame> 71 find\_man

pic: 999 state: 3005 wait: 2 next: 1000

itr:

kind: 0 x: -5 y: 2100 w: 10 h: 10 vrest: 10

itr\_end:

<frame\_end>

<frame> 80 e1

pic: 999 state: 15 wait: 1 next: 81

<frame\_end>

<frame> 81 e1

pic: 999 state: 15 wait: 1 next: 1000

opoint:

kind: 1 x: 0 y: 0 action: 0 oid: 998 facing: 2

opoint\_end:

<frame\_end> #come

<frame> 82 e2

pic: 999 state: 15 wait: 1 next: 83

<frame\_end>

<frame> 83 e2

pic: 999 state: 15 wait: 1 next: 1000

opoint:

kind: 1 x: 0 y: 0 action: 2 oid: 998 facing: 2

opoint\_end:

<frame\_end> #stay

<frame> 84 e3

pic: 999 state: 15 wait: 1 next: 85

<frame\_end>

<frame> 85 e3

pic: 999 state: 15 wait: 1 next: 1000

opoint:

kind: 1 x: 0 y: 0 action: 4 oid: 998 facing: 2

opoint\_end:

<frame\_end> #move

<frame> 86 e4

pic: 999 state: 15 wait: 1 next: 87

<frame\_end>

<frame> 87 e4

pic: 999 state: 15 wait: 1 next: 1000

opoint:

kind: 1 x: 0 y: 0 action: 6 oid: 998 facing: 2

opoint\_end:

<frame\_end> #join

   
計數回歸  
通常某些時候，需要重覆使用決定事件  
將已產生結果的動作導引到一個動作後，造出物件，並攻擊自己-500hp (fall: -1)  
(請使用state: 3005, 3006等一些被攻擊不會跳到別的動作的狀態)  
  
最後再牽回原始動作就行了

停止連續：  
可停止連續造出物件的狀態，分為  
１阻止後物件  
其效果為，連續造出物件的行為變成一段時間才做一次  
假設有東西會連續造出此物件：  
frame 50 state: 10 wait: 5 next: 51  
bdy:  
  kind: 0 x: 0 y: 1000 w: 10 h: 10  
frame 51 state: 15 wait: 40 next: 52  
itr:  
  kind: 0 x: 0 y: 1000 w: 10 h: 10 vrest: 20  
   
其先出來的動作51會攻擊慢出來的動作50，使其消失，直到51的動作跳到52…  
   
２阻止前物件  
其效果為，停止連續造出的行為，直到主要物件消失或中斷連續造出狀態，就會出現一個效果  
假設有東西會連續造出此物件：  
frame 50 state: 15 wait: 5 next: 51  
itr:  
  kind: 0 x: 0 y: 1000 w: 10 h: 10 vrest: 20  
frame 51 state: 10 wait: 15 next: 52  
bdy:  
  kind: 0 x: 0 y: 1000 w: 10 h: 10  
   
較慢造出來的物件會攻擊先造出來的物件，使其消失