```
> look

It is dark. You can't see.

> light lamp with match

The lamp glows a warm yellow.

> look

In the light of the lamp you can see before you two closed
doors. To the left is a red, weathered wooden door. To the right
is a rusty, iron door.

The lamp splutters out. It is dark.

> ?
```

Lecture 3a

Interactive Programs and Agent Decision Making

Alan Dorin

FIT3094 Artificial Life, Artificial Intelligence and Virtual Environments

MONASH University
Information Technology

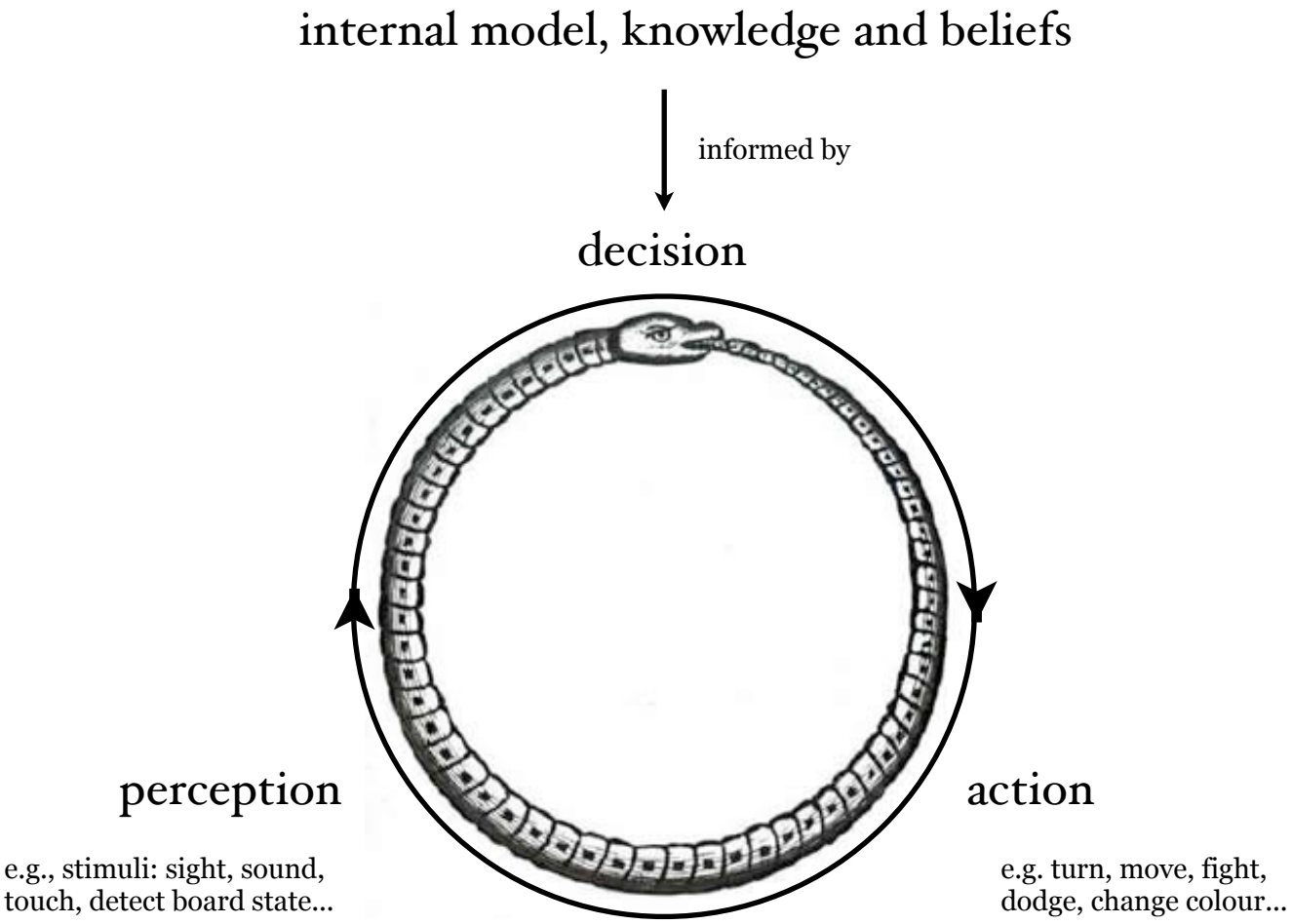
Learning Objectives

To understand the typical cycle of events an agent performs
To appreciate the difference between tactical and strategic behaviours

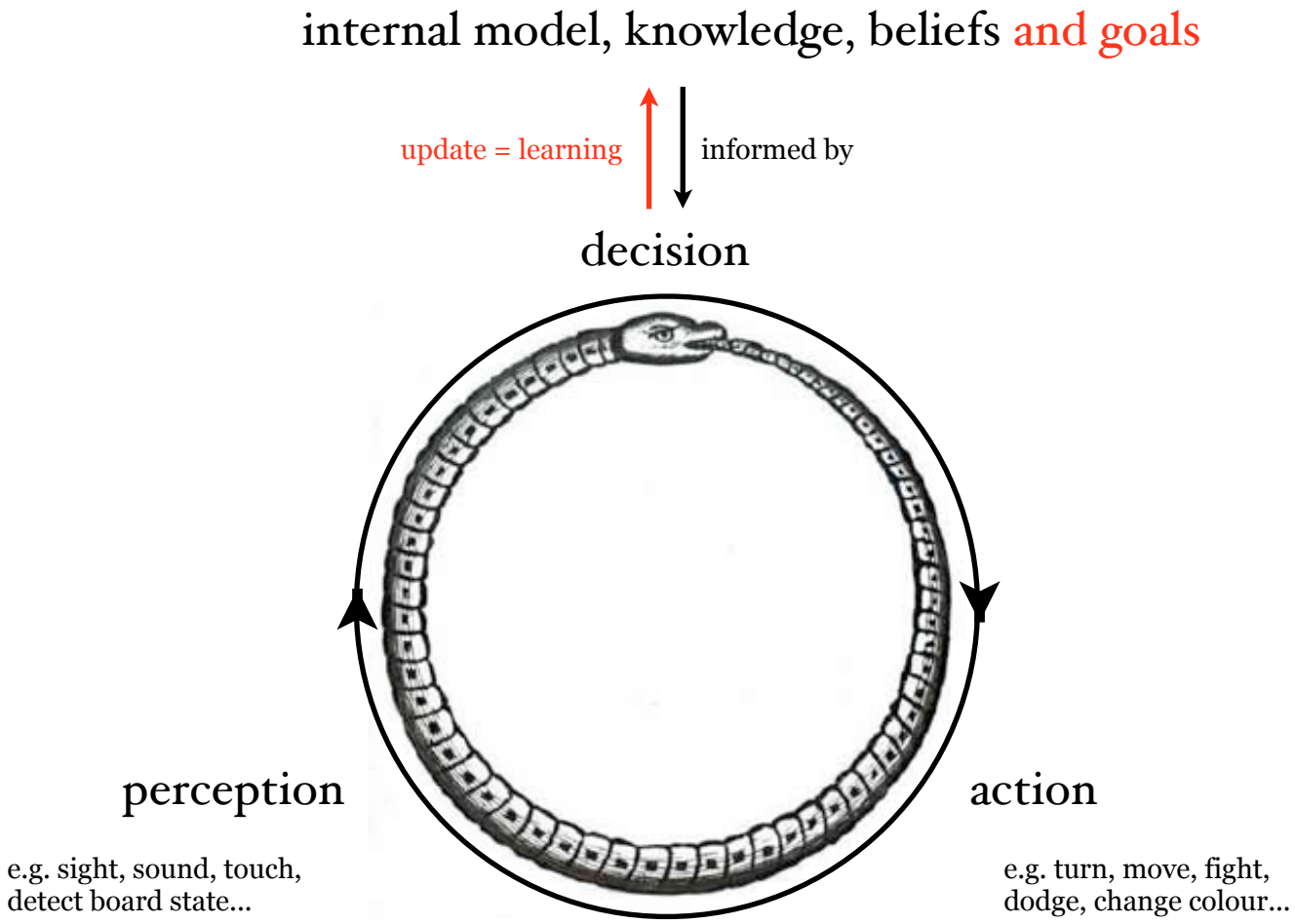
To learn how an AI unit fits within an interactive software loop
To learn how to set up an interactive software loop using OpenGL/GLUT

To understand how to construct a basic AI using Finite State Automata

An Agent's Behavioural Cycle

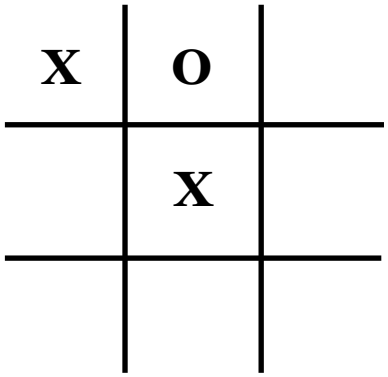


A Learning Agent's Behavioural Cycle



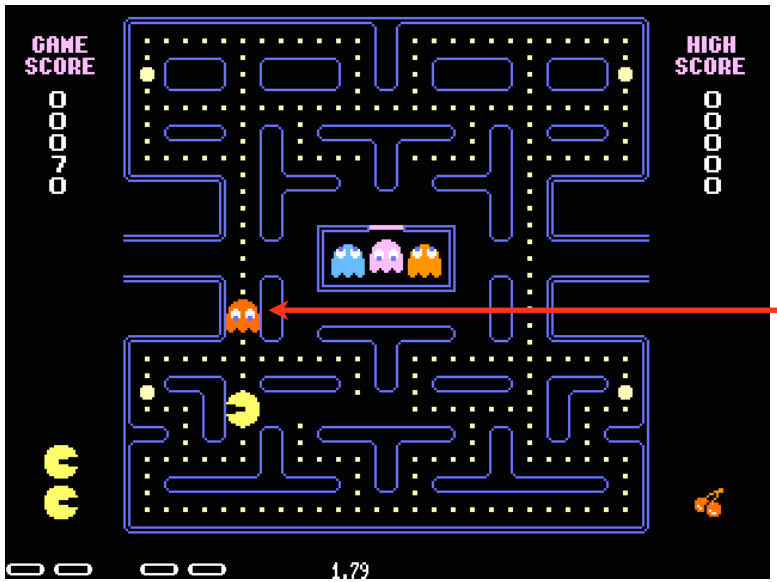
Perception

The state of a game or virtual environment *from the perspective of the agent* must be encoded for interpretation by the decision making software.



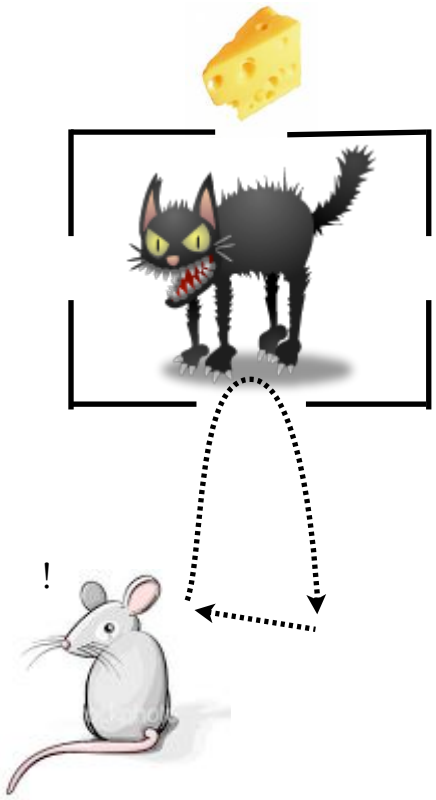

```
 1   0 -1
-1   1 -1
-1 -1 -1
```

How much should a software agent know?

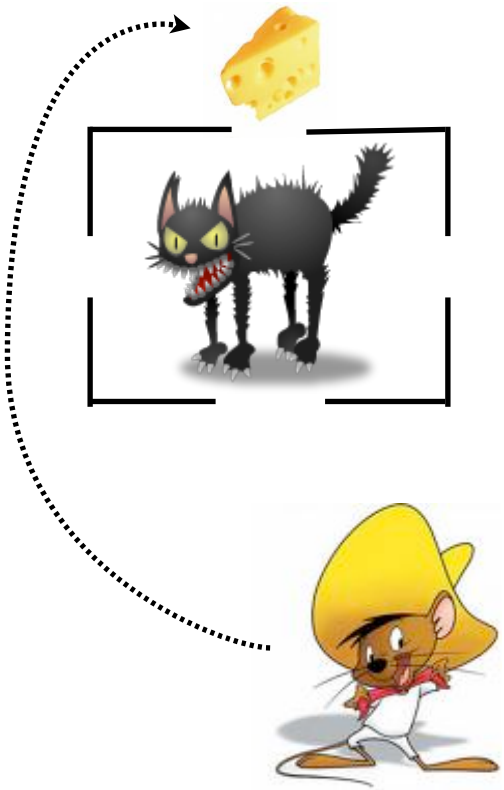


available paths	Y+, Y-
target position	6, 22
current position	6, 17
current mode	chase

Decision: Reactive and Strategic Behaviour

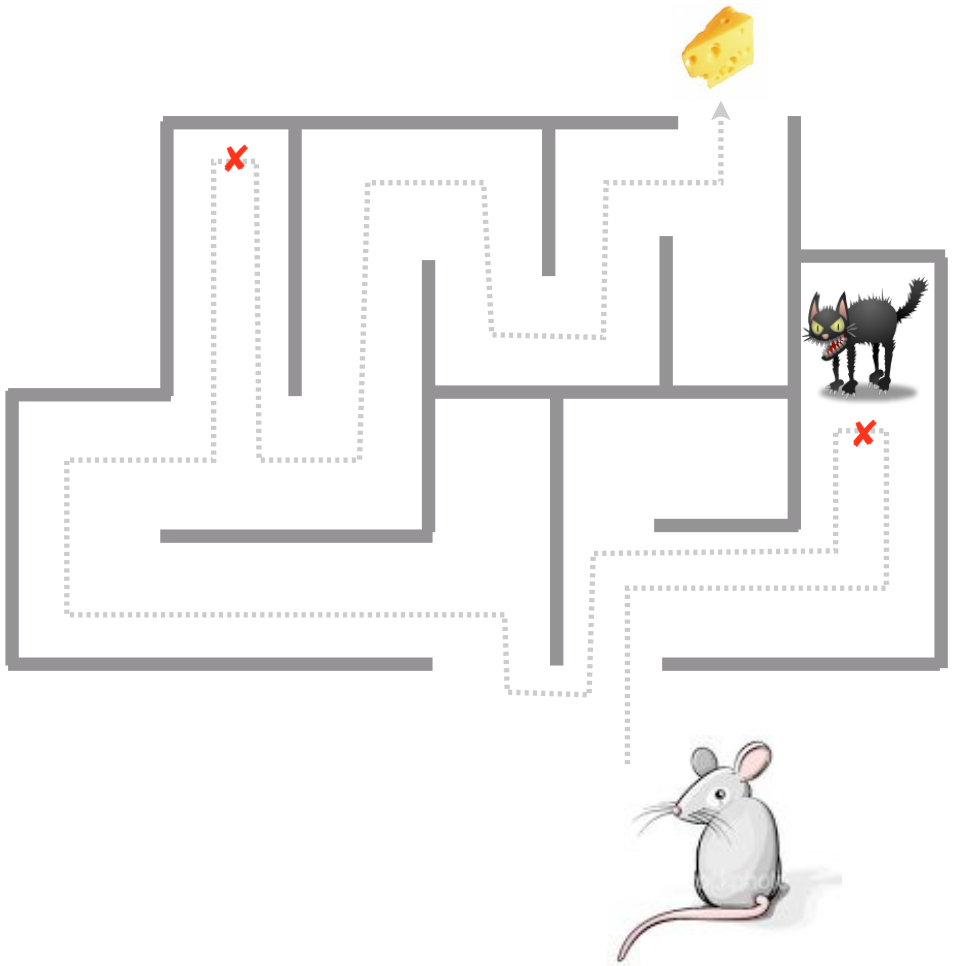


A reactive agent examines the current state of the world and responds to it tactically.



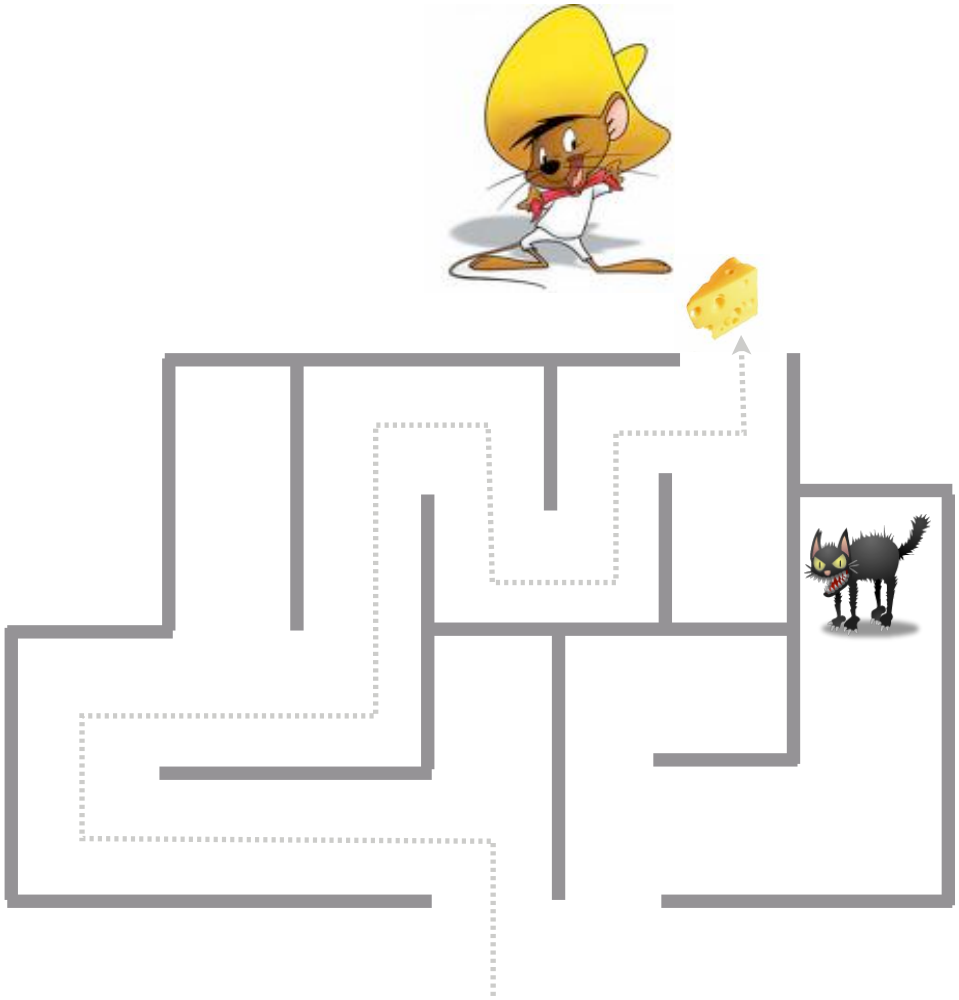
Strategic behaviour considers long term goals... it may even forfeit a battle to win a war.

Decision: Realistic...



Strategies must be computed keeping in mind an individual agent's abilities, internal model, knowledge and intelligence...

...and *Unrealistic*

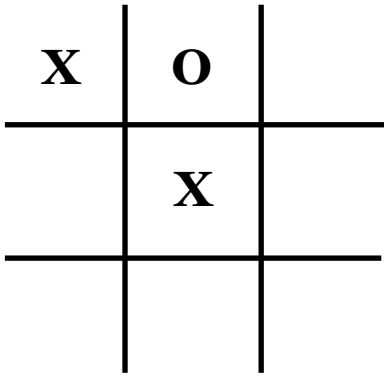


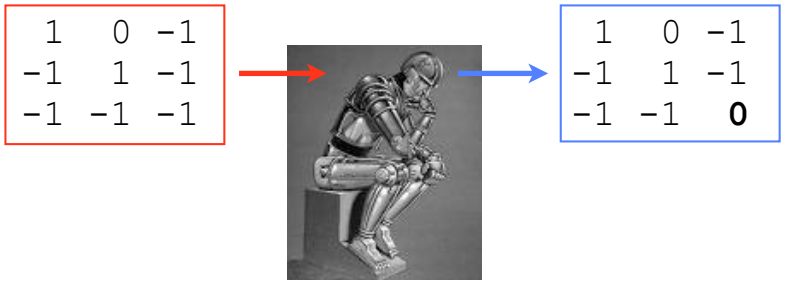
...or the agent's behaviour will be completely unconvincing!

Action

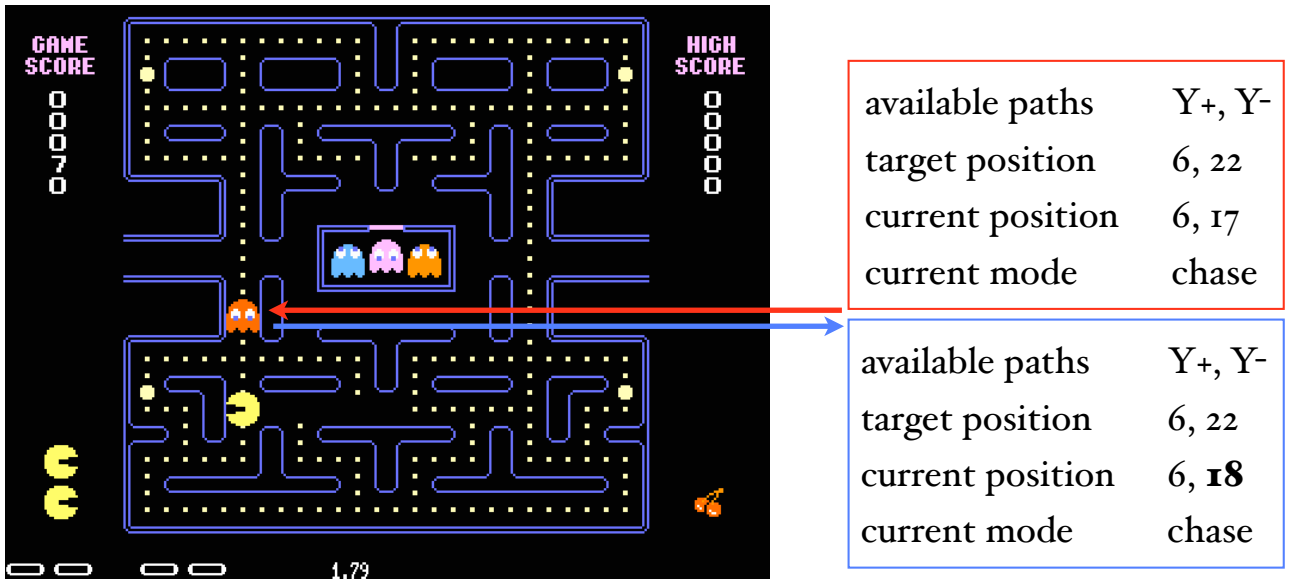
Apply the Decision System to the perceptions of the world.

Carry out the behaviour the decision system recommends.





Are these example behaviours likely to require tactical or strategic responses?



available paths	Y+, Y-
target position	6, 22
current position	6, 17
current mode	chase

available paths	Y+, Y-
target position	6, 22
current position	6, **18**
current mode	chase

```
IF((target is directly ahead)
&& (target distance <= maximum range)
&& (current ammo > 0)
&& (current state == aim)
&& (current health > 50%))
THEN
{
  set (current state, shoot)
  shoot (1)
  current ammo--
}
ELSE
{
 ...?
}
```

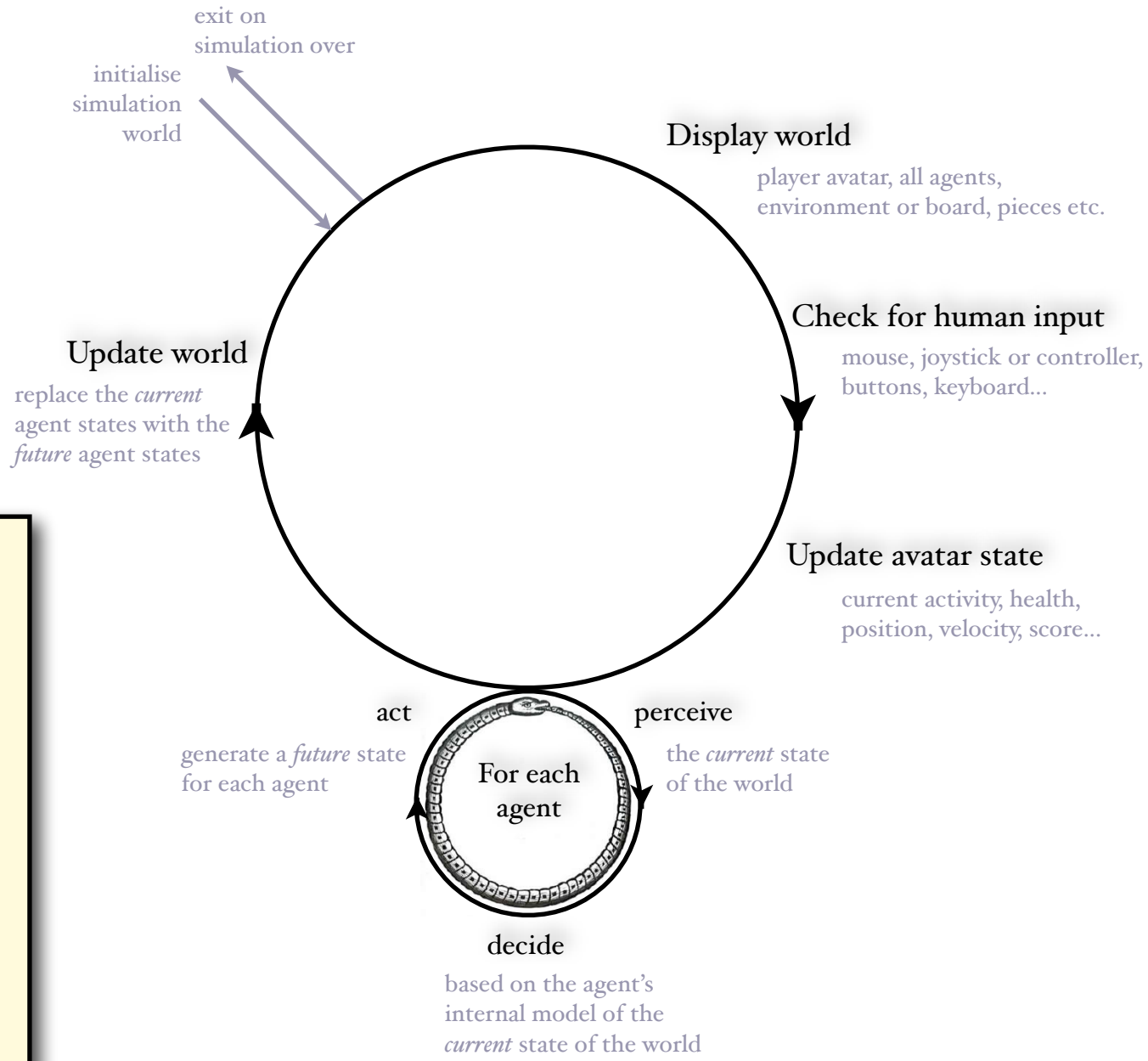
available paths	N, S, E, W
target distance	5
target direction	0, 0, 0
current state	**shoot**
current health	100%
current ammo	**16**
target state	shoot
target health	84%
target ammo	27

available paths	N, S, E, W
target distance	5
target direction	0, 1
current state	aim
current health	100%
current ammo	17
target state	shoot
target health	84%
target ammo	27

Is the illustrated behaviour tactical or strategic?
How quickly should the agent return fire?
How accurately should the agent shoot? What if it is injured?
How well should the agent predict the player's behaviour?

How should the agent behave if the "THEN" clause is not activated?

A Simple Interactive Algorithm Loop




```
BEGIN
initialise world
WHILE (!simulation over)
{
   display world
   check for human input
   update avatar state
   FOR EACH agent DO
   {
      perceive
      decide
      act
     update agent state
   }
   update world
}
END
```

Noughts and Crosses

a basic interactive algorithm

```
BEGIN
clear board
WHILE (!gameOver)
{
   draw board and pieces
   WHILE (!player moved)
   {
      check for human move
   }
   read human move
   check for human win
   draw board and pieces

   do 3-in-a-row(col/row/diag) if avail. gameOver = true
   else do block if avail.
   else do fork if avail.
   else do corner if avail.
   else do centre if avail.
   else do random if avail.
   else gameOver = true
}

END
```

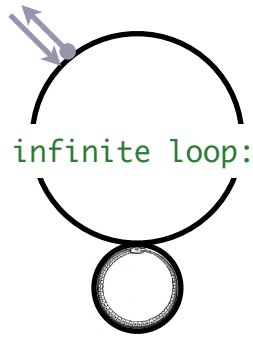
```
BEGIN
initialise game world
WHILE (!game over)
{
    display game world
    check for player input
    update game world
    FOR EACH agent DO
    {
        perceive
        decide
        act
    }
    update game world
}
END
```

Initialisaing an Interactive Loop

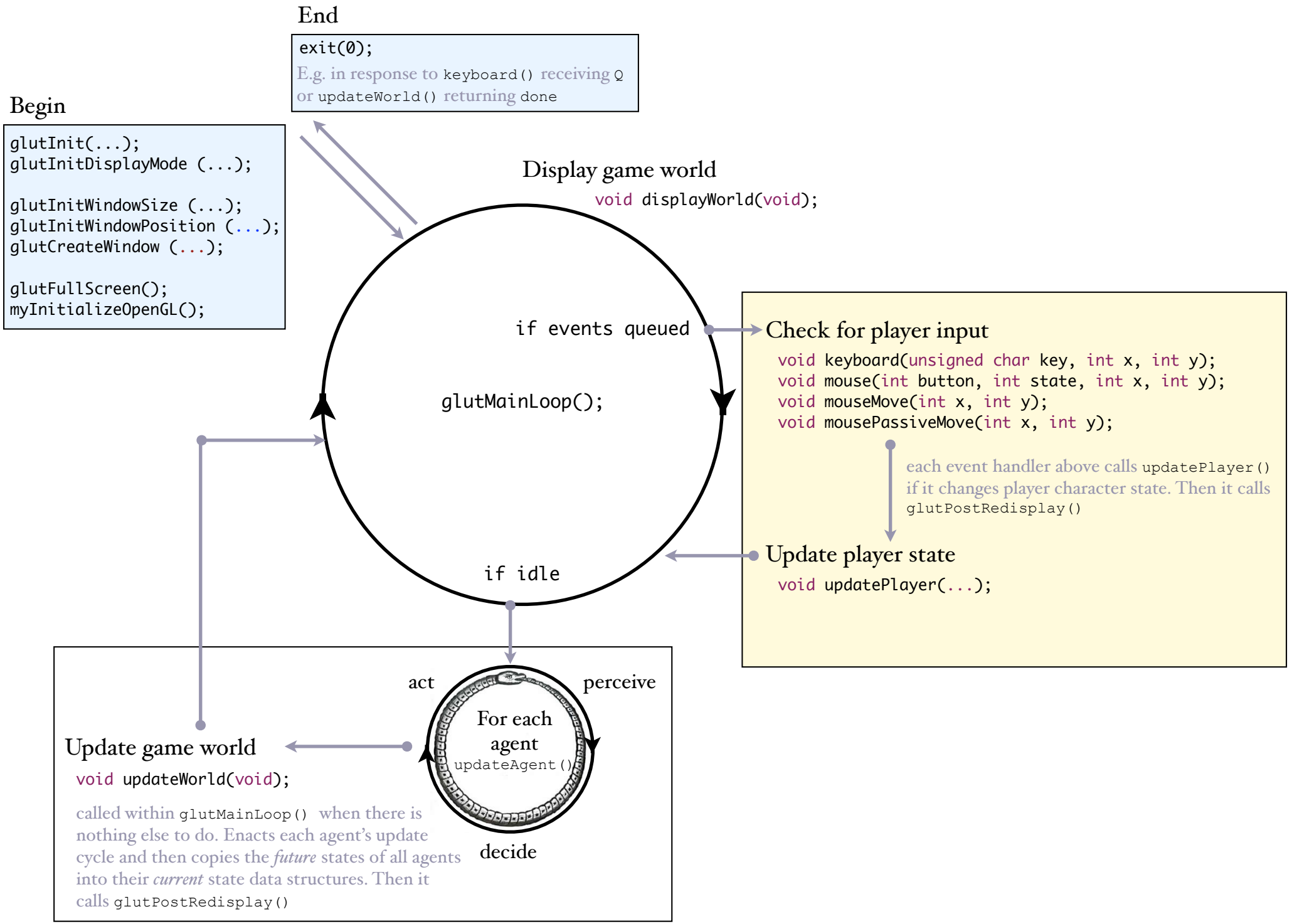
GLUT : (Open GL) Graphics Library Utility Toolkit

http://www.opengl.org/resources/libraries/glut/

```
#include <GLUT/glut.h>

int main(int argc, char **argv)
{
  glutInit(&argc, argv);                                          // Initialize OpenGL/GLUT (only do this once)
  glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);     // Use double buffering, RGB mode, depth-buffer

  glutInitWindowSize (gWinCols, gWinRows);                        // Set up the window and open it...
  glutInitWindowPosition (0, 0);
  glutCreateWindow ("Your Window Name");

  glutFullScreen();                                               // Make the graphics window occupy the whole screen
  myInitializeOpenGL();                                           // Do some of your own initializations

  glutDisplayFunc(displayWorld);                                  // Register all of the event handlers
  glutVisibilityFunc(visible);
  glutReshapeFunc(reshape);
  glutKeyboardFunc(keyboard);
  glutMouseFunc(mouse);
  glutMotionFunc(mouseMove);
  glutPassiveMotionFunc(mousePassiveMove);
  glutIdleFunc(updateWorld);

  glutMainLoop();                                                 // Start infinite loop: poll events update state
    return 0;
}
```



Running an Interactive Loop



Agent Decision Making

Recall this simple algorithm for agent decision making?

```
IF((target is not shooting at me)
&& (target is directly ahead)
&& (target distance <= maximum range)
&& (current ammo > 0)
&& (current state == aim)
&& (current health > 50%))
THEN
{
  set (current state, shoot)
  shoot (1)
  current ammo--
}
ELSE
{
 ...?
}
```

Note this in particular...

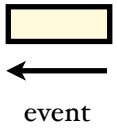
...and this!

These refer to internal state data that must be stored in the enemy solider class. It changes, if certain conditions are met, like this:

Finite State Machines (FSM)

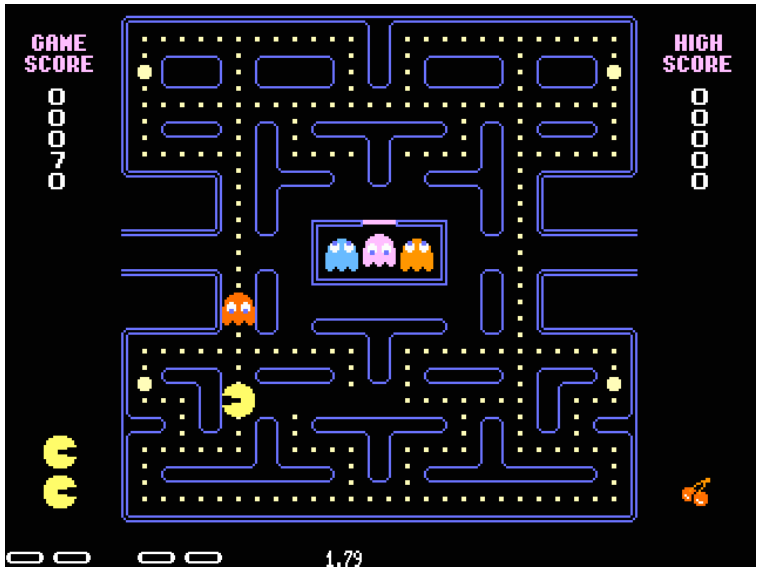
The FSM* is an extremely common, simple and powerful way to encode the behaviour of game or simulation agents.

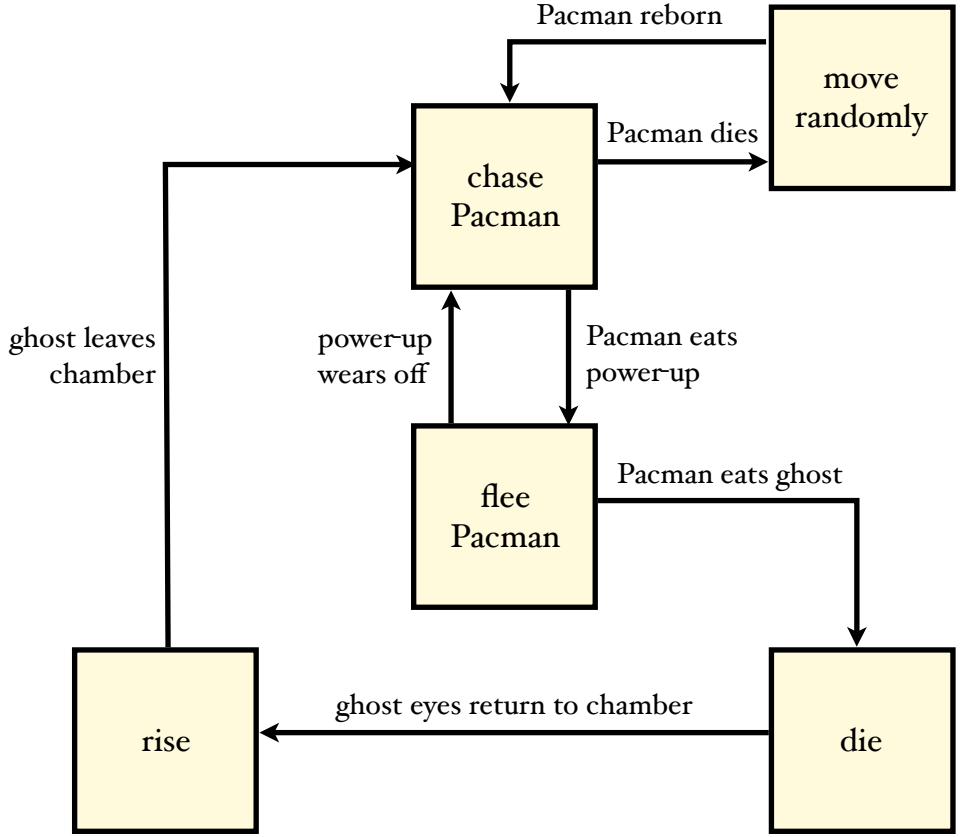
A state machine contains:



A set of states
A set of transitions between states
A set of conditions by which transitions are triggered

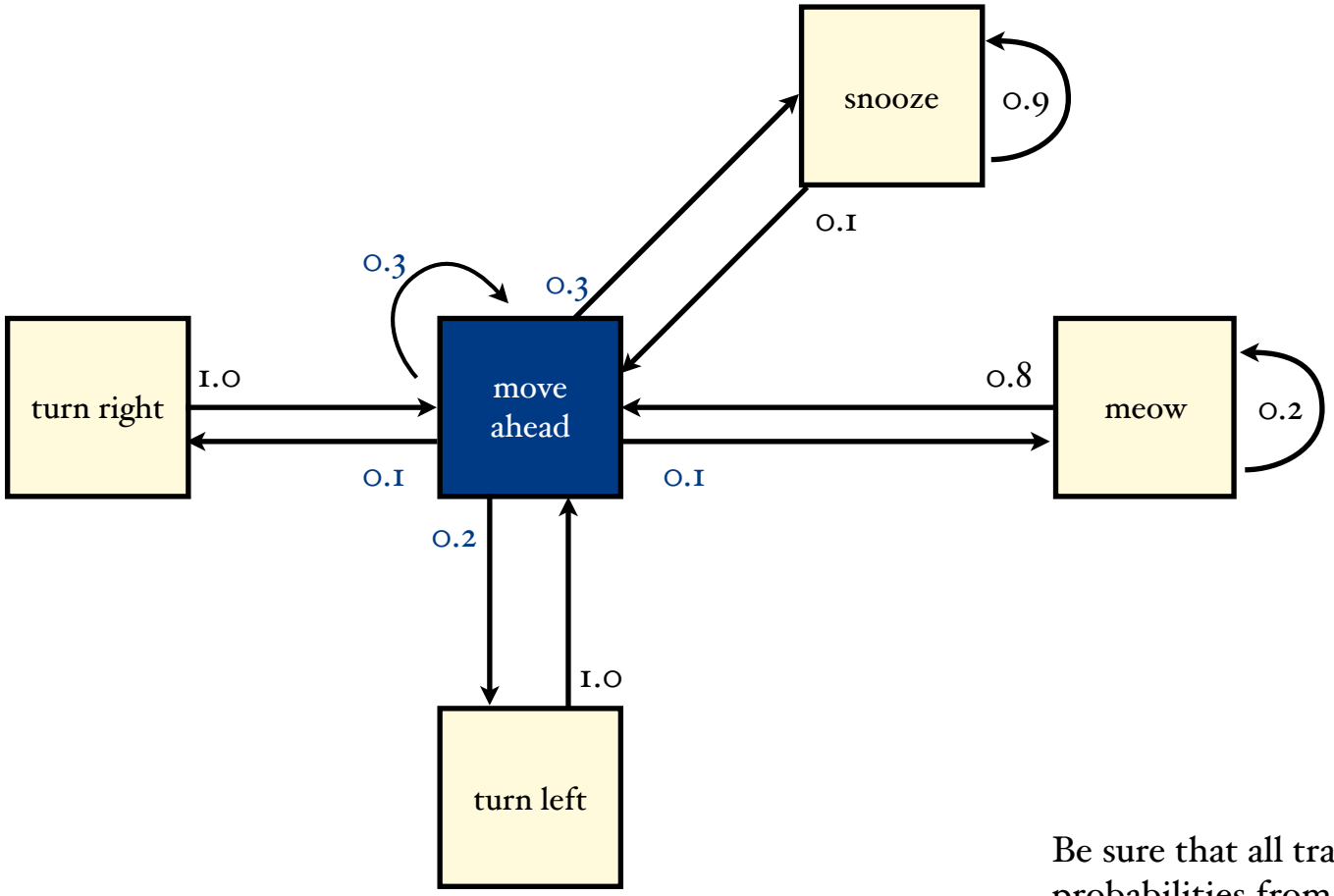
Blinky, the red ghost and his FSM





* also called a *Finite State Automaton* (FSA)

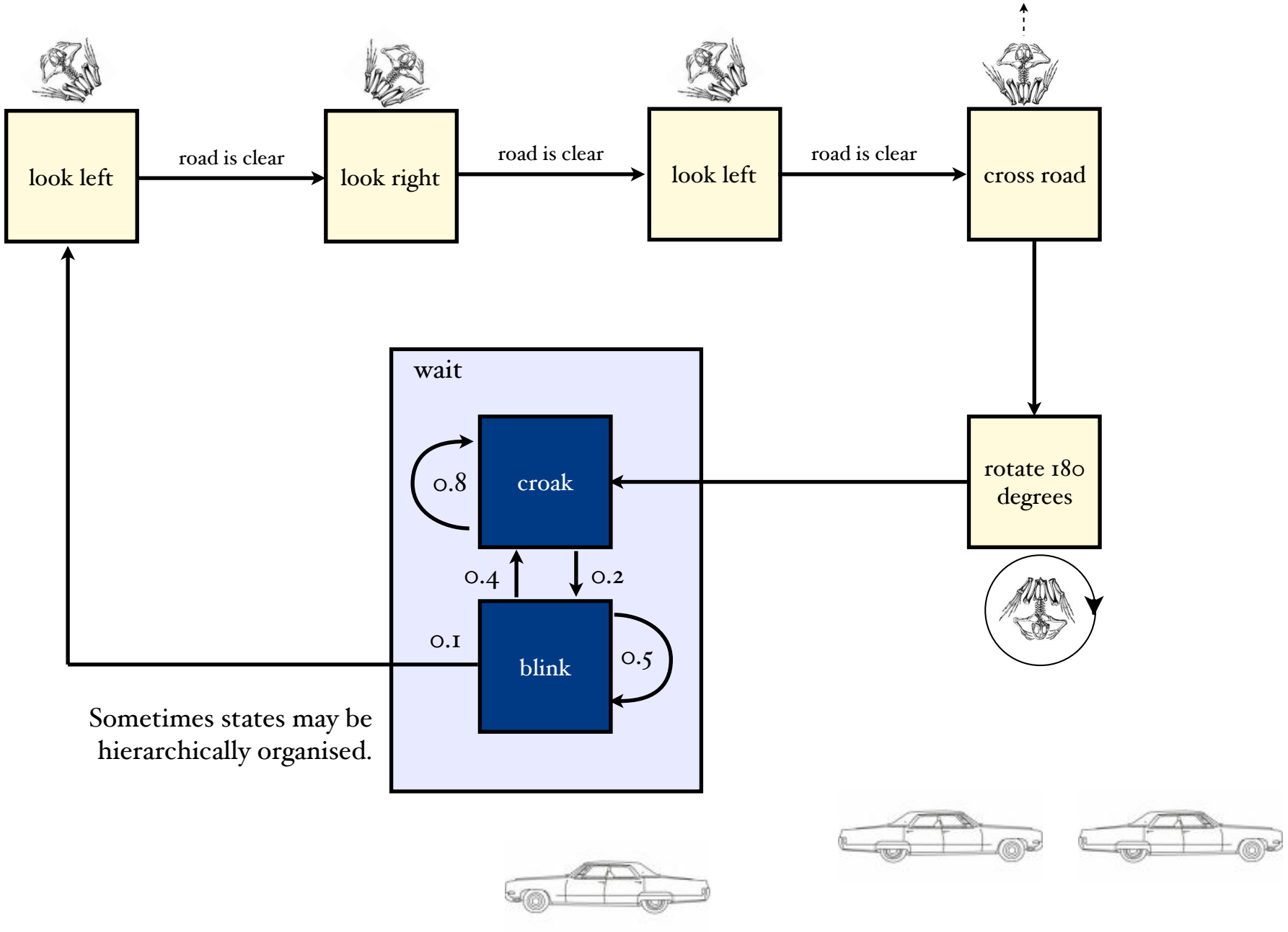
A Probalistic Finite State Machine (FSM)



Be sure that all transition probabilities from a state sum to 1.0!

A Hierarchical Finite State Machine with Linear Sections

Sometimes an agent (such as a frog trying to cross a busy road) needs a series of transitions to occur one after another...



Transition Triggers

State transitions can be triggered:

- Automatically at the conclusion of a state's behaviour
- Stochastically
- By human/player action
- On timer elapse
- When certain model conditions are met

What are some examples of each?