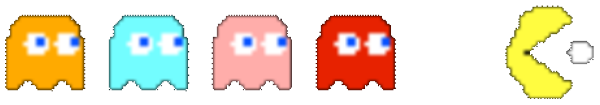
FIT3094 AI, ALife and Virtual Environments, by Alan Dorin.

Practical sheet: Pacman and FSMs

Part A : Pacman

1. Log on to an online *Pacman* game e.g. http://www.webpacman.com/



   Watch over the shoulder of a class mate as they play the first game level a few times. Carefully observe the behaviour of the ghosts. Do they all behave in the same way? Observe your classmate's game play too. Then swap roles.

2. Discuss with your classmate, then experiment and write down alone, for each ghost, the algorithms you think they might be following. (Some brief information is provided on the pages of the webpacman.com website that might help.) What does the designer, Toru Iwatani, say about the reasons for making these algorithms as he did?

3. Construct a Finite State Machine (FSM) for your favourite ghost. (Blinky is provided in the lecture notes as an example.) Choose a *different* ghost to your classmate and the lecturer's notes. What are the triggers that govern changes in your ghost's behaviour?

4. Discuss with your classmate their observations of your game play. Then, write down (alone) the algorithm that you employed to complete a level of the game.

5. Construct a FSM for your own game play strategy.

Part B : FSM Implementation

1. Implement the FSM you devised for your ghost in C++ class. You don't need to make any graphics code to visualise the ghost's state. Instead, every time the FSM changes state, just print out the name of the new state to the screen. For example, "Running away", "Chasing Pacman", "Returning home".

2. Discuss and design with your classmate a consistent way to implement a test-harness for your ghost FSMs. This should allow a human user to enter ASCII characters to generate the triggers that cause the FSMs to change state. For example, C => Chase ghost, T => Touch ghost, F => Flee!

3. Together, implement your test harness.

4. Incorporate both of your ghost FSMs into one C++ program and play Textman!