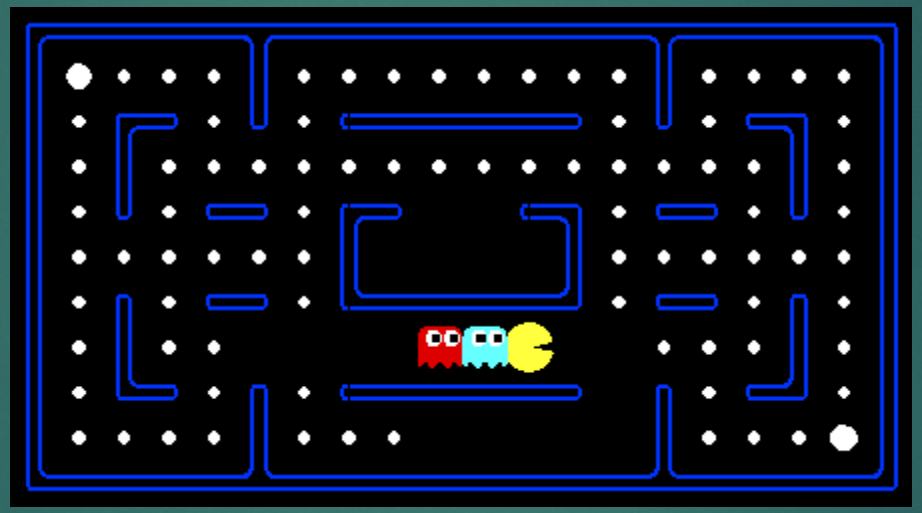
# Artificial Intelligence

FALL 2021

FACULTY OF MATHEMATICS & INFORMATICS, UNIVERSITY OF BARCELONA

# Practical 2 – Multi-Agent Search

- Goal: To design the classic version of the Pacman game.

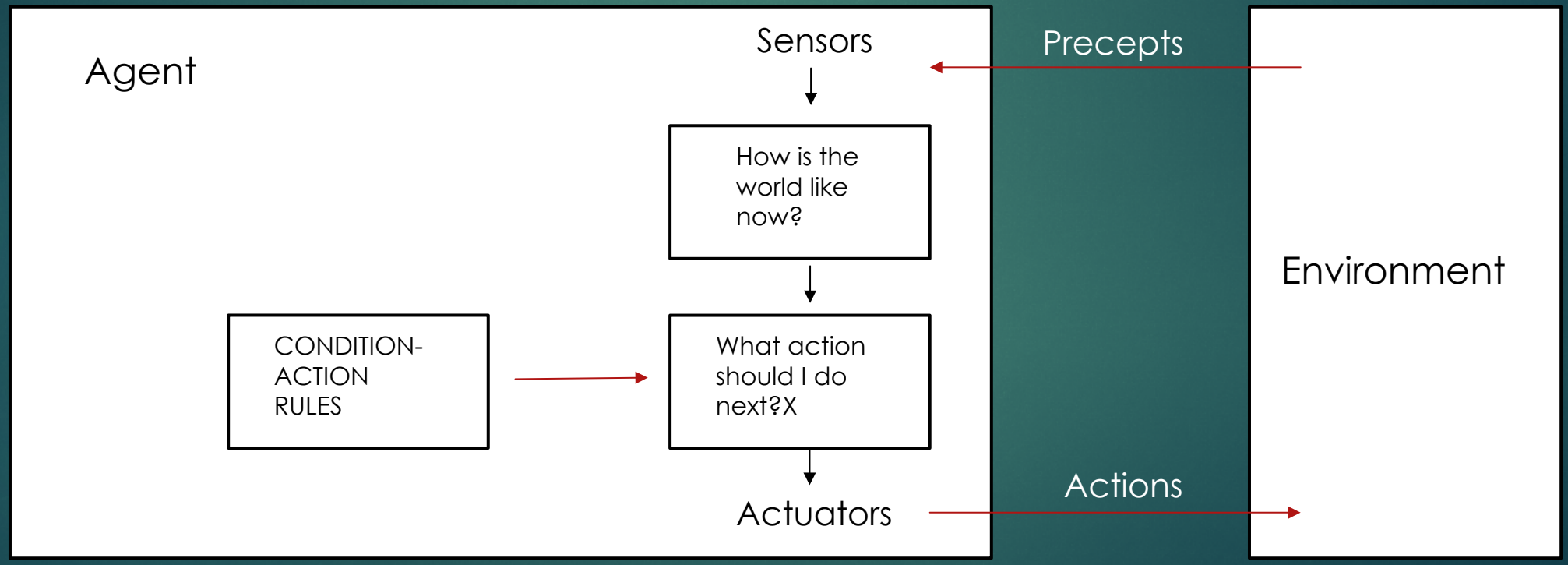


# Practical 2

- Download code
- Files to Edit
  - multiAgents.py  Where all of your multi-agent search agents will reside
- Files you may want to look at:
  - pacman.py  The main file that runs Pacman games. This file describes a Pacman GameState type, which you use in this project.
  - game.py  The logic behind how the Pacman world works. This file describes several supporting types like AgentState, Direction and Grid.
  - util.py  Useful data structures for implementing search algorithms.

# Practical 2 – Multi Agent Search

- Reflex agent -> python pacman.py –p ReflexAgent



# Practical 2 – Multi-Agent Search

- To improve ReflexAgent in multiAgents.py
- The agent must take into account the location of food and of ghosts.
- python pacman.py –frameTime 0 –p ReflexAgent –k 1
  - --frameTime: Time to delay between frames
  - -k: number of agents

# Q1 – Evaluation Function

- 1. Debug your code
  - successorGameState
  - newPos
  - newFood
  - newGhostStates
- 2. Verify distance from ghosts/food
  - How close am I to the ghosts?
  - How close am I from the food?
  - Design a function to assign marks.

# Q2 – Minimax

- We will have one layer per ghost.
- The evaluation function is self.evaluationFunction(state)
- The depth of the tree will be chosen.
- The initial layer is initialized by default.
- Pacman-> Agent 0

# Q2 – Minimax

- Recursive function.
- Function with arguments “gameState” and “depth”
- Verify if the game is over:
  - Depth is 0
  - Has won-> isWin()
  - Has lost-> isLose()
- Find the player: pacman or k phantom (1, 2, 3, 4…)
  - Which possible actions can you do?
- Pacman or ghost?
  - Return the next game with an action that maximizes or minimizes -> (value, action)

# Q3 – Poda Alpha-Beta