Game AI Part 1

Admin

- Schedule and Quizzes
- Questions?

Game AI vs AI

- Game AI often not real AI
  - Real AI art of “making computers do things only people are good at”
    - Self defeating definition
  - Lots of processing power /computation
  - Overkill for most games
  - exceptions?

Game AI vs AI

- Most of the “AI” in games not really Intelligence at all
  - Artificial Stupidity
  - Don't want an intelligent opponent
    - Would cause difficulty level issues
  - Want one just good enough.
  - Games full of AlphaStar or Pluribus would be no fun to play
    - https://builtin.com/artificial-intelligence/lessons-ai-games
  - Even Dark souls III changed death to be less punishing than Dark Souls II (no more shrinking health bar)
    - Elden Ring?
  - LLMs? Or even Small Language Models?

So what is good enough for Game AI?

- Standard programming techniques, adapted to games usually good enough
  - Most common
    - Finite state automata/finite state machines
    - Simple is often good enough
    - Quick review of FSM for undergrads

Of course, for big strategy games, something more might be needed.

But often lots of smaller techniques are good.

FSM from a game designers point of view

- simple look at FSM :
  - Model of behavior of game entity
    - With limited number of defined modes
    - Mode transitions change with circumstance
  - Examples and details to follow

Example simple FSA

- Example:(pseudo code)
  - if self.state == "exploring":
    - self.random_heading()
    - if self.can_see(player):
      - self.state = 'seeking'
  - elif self.state == 'seeking':
    - self.head_towards('player')
    - If self.in_range_of(player):
      - self.fire_at(player)
    - If not self.can_see(player):
      - self.state = 'exploring'
- Now lets draw the diagram for this machine

Elements of FSM

- Elements of FSM
  - states
    - which define behavior and may produce actions
  - state transitions
    - which are movement from one state to another
  - rules or conditions
    - which must be met to allow a state transition
  - input events
    - which are either externally or internally generated, which may possibly trigger rules and lead to state transitions

Finite State Machine for Pac Man Red Ghost

- Let's build part of the pseu

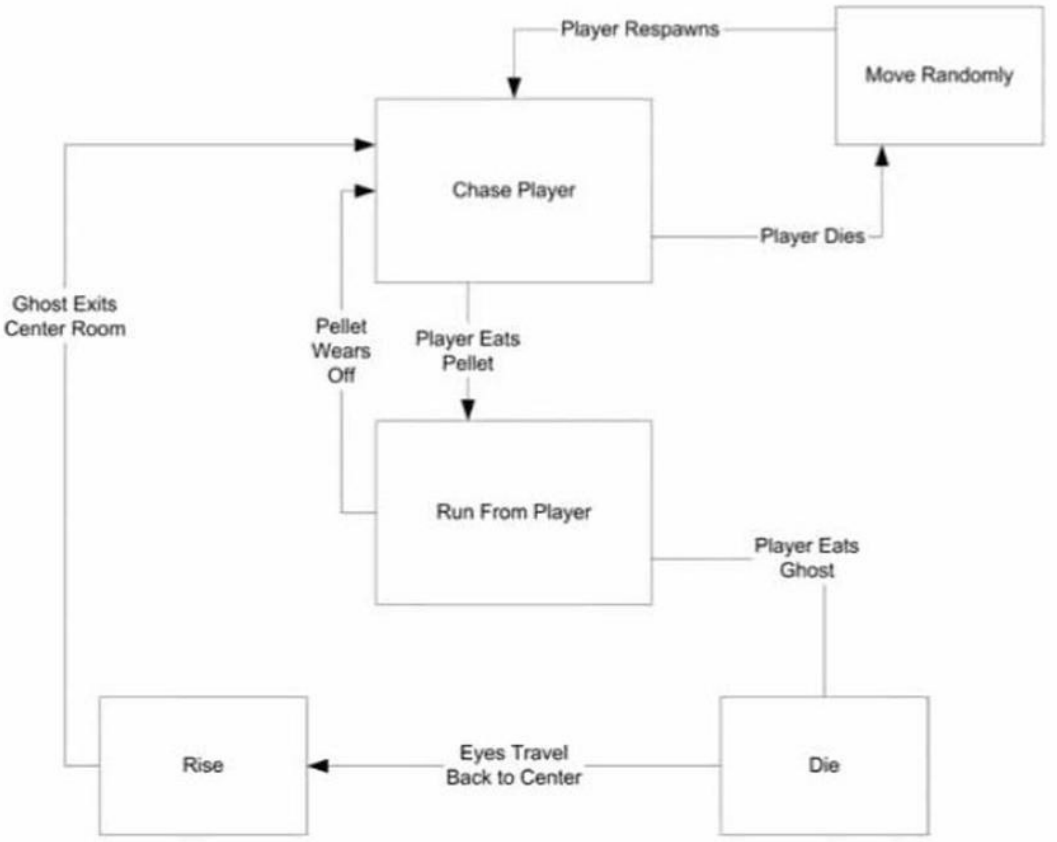
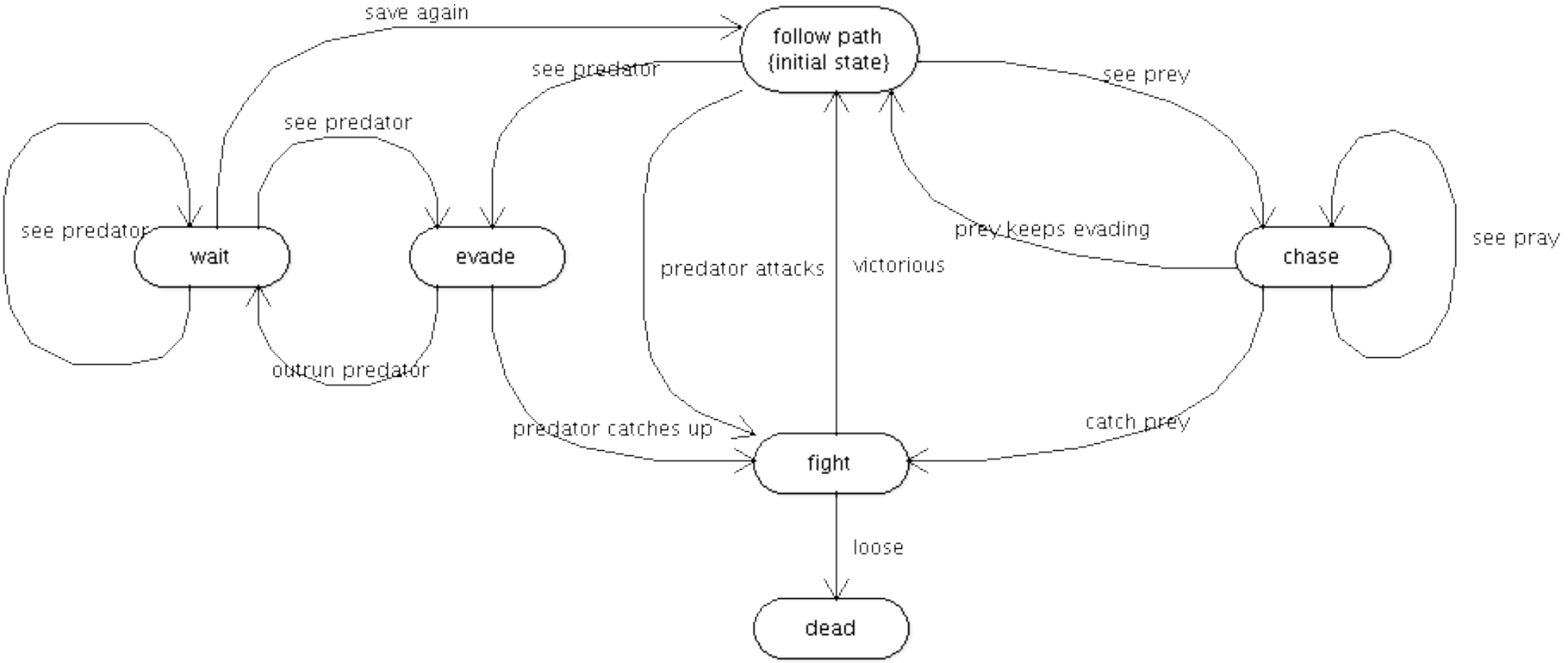


FIGURE 15.2 Simple FSM diagram of the red ghost from *Pac-Man*.

A slightly more complex FSM for a game entity



Implementing a Finite State Machine

- States in FSM generally define 2 things
  - What the game entity is doing at moment
  - When the entity needs to switch to new state
- If I were to ask about this in a quiz or exam, you would want both
- Where is that in examples so far?

Advantages of FSMs for Game dev

- * simplicity: easy for inexperienced developers to implement
- * Predictability: given a set of inputs and a known current state, the state transition can be predicted
- FSMs are quick to design, quick to implement and quick in execution
- FSM been around for a long time: well proven
- FSMs are relatively flexible: number of ways to implement
- Easy to transfer from meaningful abstract representation to code
- Low processor overhead; Only the code for the current state need be executed,
- Easy determination of reachability of a state, when represented in an abstract form, it is immediately obvious whether a state is achievable from another state, and what is required to achieve the state

Disadvatages of FSMs

- Disadvantages
  - The predictable nature of deterministic FSMs can be undesirable
  - * Larger systems implemented using a FSM can be difficult to manage
  - Not suited to all problem domains: should only be used when
    - a systems behavior can be decomposed into separate states with well defined conditions for state transitions.
    - all states, transitions and conditions need to be known up front and be well defined

Non Deterministic FSM

- Some of the problems overcome by using Nondeterministic FSMs
  - Brief description of NonDeterministic FSMs for undergrads
  - Lets diagram non-deterministic finite state machine for game entity

Implementing Non-deterministic FSA

- When more than one transition is valid
  - Randomly choose between them.
  - Lets write a sample
- Fu FSA
  - Fuzzy FSA
    - Like non-deterministic fsa except transitions are weighted rather than random

Stop here

- Flocking continues here, but go look at path planning at this point