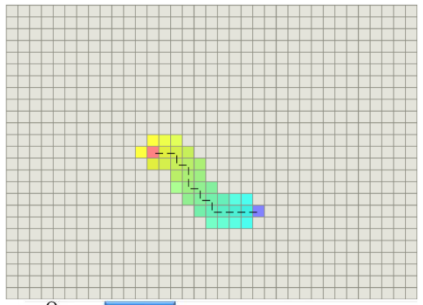
“Inaction breeds doubt and fear. Action breeds confidence and courage. If you want to conquer fear, do not sit home and think about it. Go out and get busy.” -- Dale Carnegie

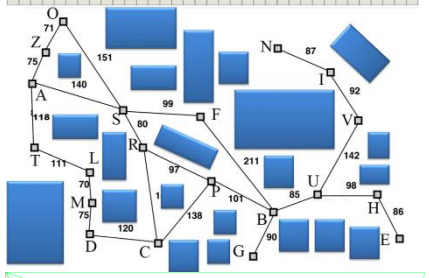
Announcements

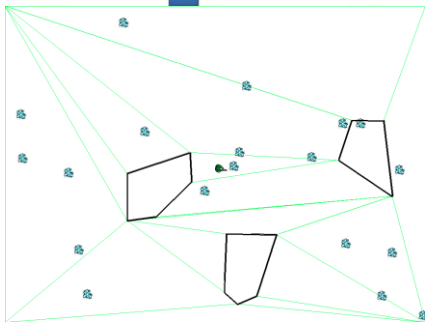
- HW4 is posted, due this Sunday
- Gradescope vs Canvas
- Exam

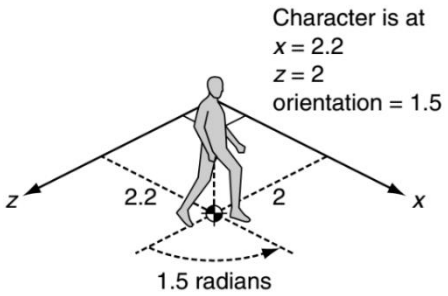
Review

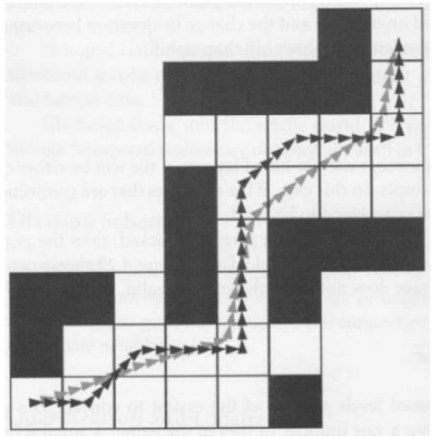
- The story thus far…

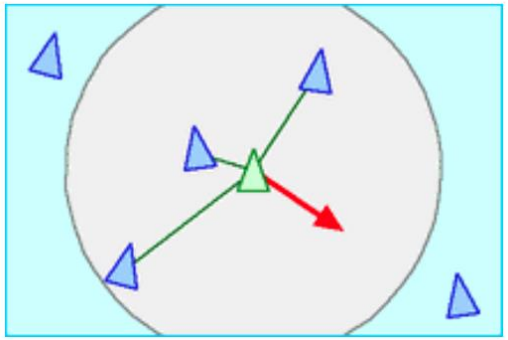






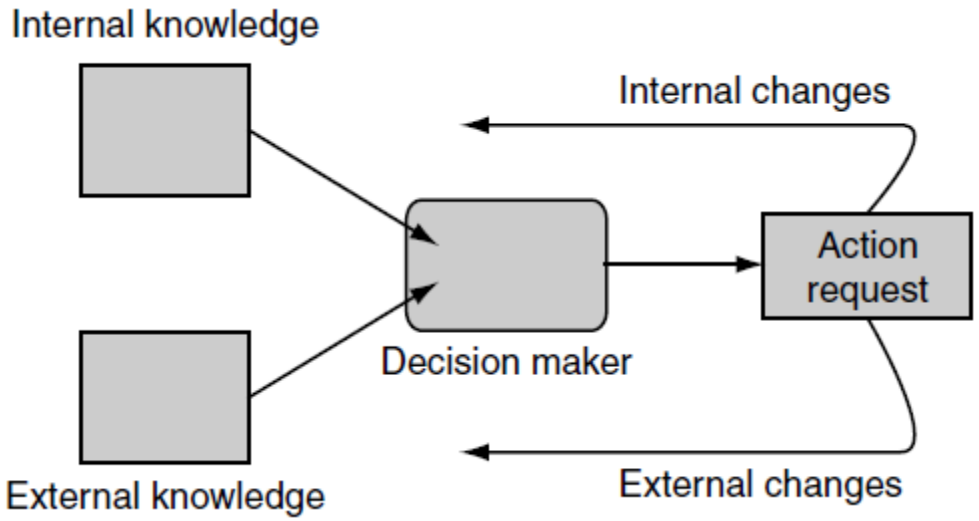






N-1: formations

1. What's the problem with independent shortest distance navigation with groups?
2. What's a simple solution, and its drawbacks? Other workarounds?
3. Is it always sensible to replan paths when problems occur?
4. What are 3 ways of implementing fixed formations?
5. Discuss some problems with entities of different sizes.
6. Do we need complex decision logic to support tactical movement?



M&F 5.2

All presented techniques are applicable to intra- & inter-character decision making

INTRO TO DECISION MAKING

Complexity Fallacy Alert!

- Decision making is typically a small part of the effort needed to build great game AI
  - Most games use very simple decision making systems: state machines and decision trees.
  - Rule-based systems are rarer, but important
  - Behavior Trees have become popular (increasingly pattern of choice)
    - Halo 2 (2004) one of first AAA game to use and describe in detail
    - Editor built in to Unreal Engine 4, available for Unity
- Recall relationship between predictability and learning

Decision Making

- Historically a “mathematical model of computation” – Wikipedia
- Classic AI:
  - making the optimal choice of action (given what is known or is knowable at the time) that maximizes the chance of achieving a goal or receiving a reward (or minimizes penalty/cost)
- Game AI:
  - choosing the right goal/behavior/animation to support the experience
- Decision-making must connect directly to animation so player can see the results of decision-making directly (explainable AI)
  - What animation do I play now?
  - Where should I move?

What all can agents do?

- Move…
- But also:
  - Gather resources
  - Attack
  - Access computer terminal
  - Open door
  - Etc.
- Steering behavior blending can sometimes address multiple behaviors, but often agents need higher levels of control supported by a decision making process
  - for instance, to avoid steering vectors that negatively interfere, cancel each other out, etc.

Decision Making: Pre-planned behaviors

- Programmer determines *recipe* for agent behaviors
  - In other words, the programmer implements the plan(s)
- All contingencies anticipated as part of this *recipe*
- If something was missed, the agent is unable to adapt
- Decisions are made by conditional logic
- Can include deterministic on non-deterministic conditions

Types of Decision Making

- **Reactive**
  - Response directly to environmental state
  - Pre-planned conditional actions
  - Most RT game AIs are reactive
- **Deliberative / Deductive**
  - Models environment (discretized)
  - Inferences made to determine plan that can achieve goal state
- **Reflective / Inductive**
  - Learn from experience

Pre-planned Behaviors: Types

- Production Rules (Simple version/Fixed Priority Arbiter)
- FSMs (variants)
- Decision & Behavior Trees

Production Rules (Simple Version/Fixed Priority Arbiter)

- Define actions to be executed in response to conditions
- Simple architecture
- Difficult to organize
- Can be difficult to understand behavior from looking at rules
- Rules can conflict (ambiguous situations)
- Action sequences require a series of rules with stateful triggers

Production Rules (Simple Version/Fixed Priority Arbiter)

If(big enemy in sight) run away

If(enemy in sight) fire

If(---) ----

Age of Kings example

```
(defrule
        (building-type-count-total castle less-than 1)
        (can-build castle)
=>
        (build castle)
        (chat-local-to-self "castle"))
```

Production Rules (Simple version/Fixed Priority Arbiter)

- How to select among multiple valid rules?
  - Advanced production rules systems utilize an **arbiter** that selects from matching rules (will be revisited)
  - **Fixed Priority Arbiter**: For simple version, we will just evaluate the first one that matches from an ordered list
  - Priority weighting for rules or sensor events
    - Like firewall rules
  - Random selection
- Can be stateless

Example of random selection with Fixed Priority

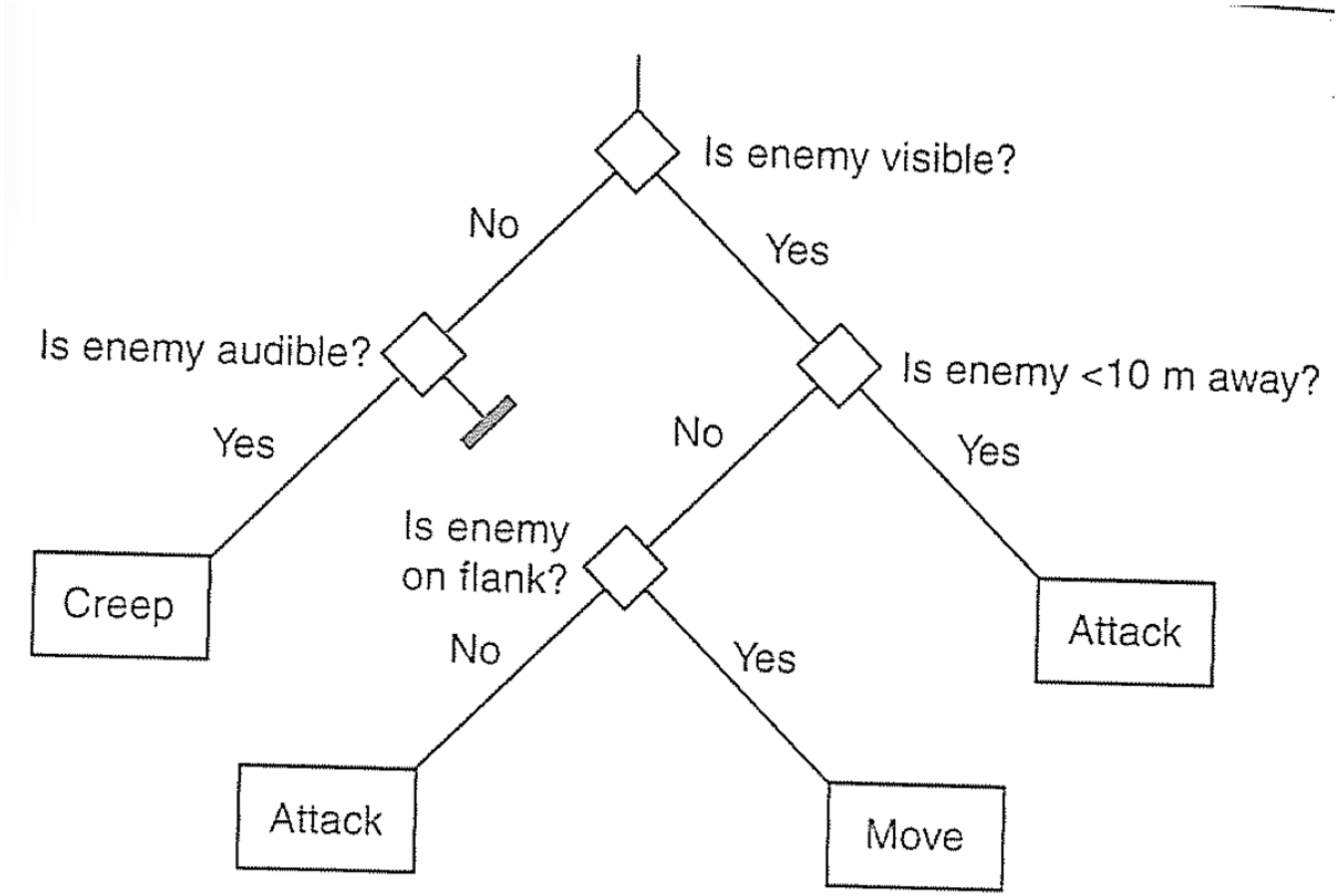
If((big enemy in sight) AND (rand() > thresh1)) run away

If((enemy in sight) AND (rand() > thresh2)) fire

If(---) ----

Simple Decision Tree (Manually Created)

- Nested if statements organized logically (for now)
- Good for simple decision making
- But quickly becomes a mess to manage manually
- (To be revisited later)



Decision Making – FSMs

2019-09-23

B Ch2, M 5.1,5.3

Most game agents do more than move

- Shoot
- Patrol
- Farm
- Sleep
- Hide
- Hunt
- …
- We will call these “states”

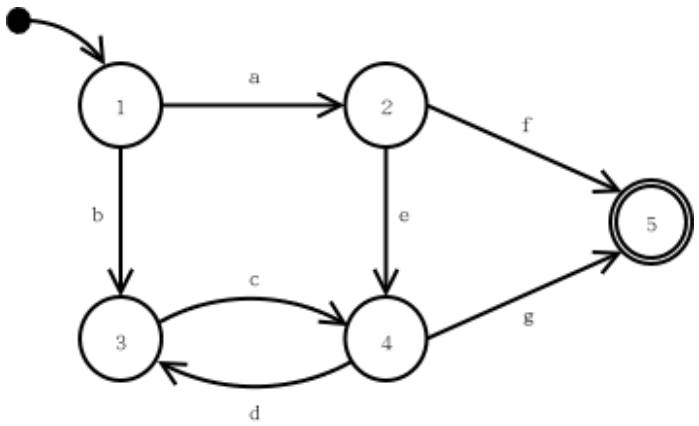
Most game agents go through more than one state

Game agents will have different sets of states, and different ways of moving through those states based on their purpose in the game:

- Soldier: Patrol -> Shoot -> Hunt
- Farmer/Villager: Farm -> Patrol -> Sleep

FSM theory

- A (model of a) device which has
  - a finite number of states (S)
  - an input vocabulary (I)
  - a transition function T(s,i) → s'
  - a start state ∈ I
  - zero or more final states ⊂ S



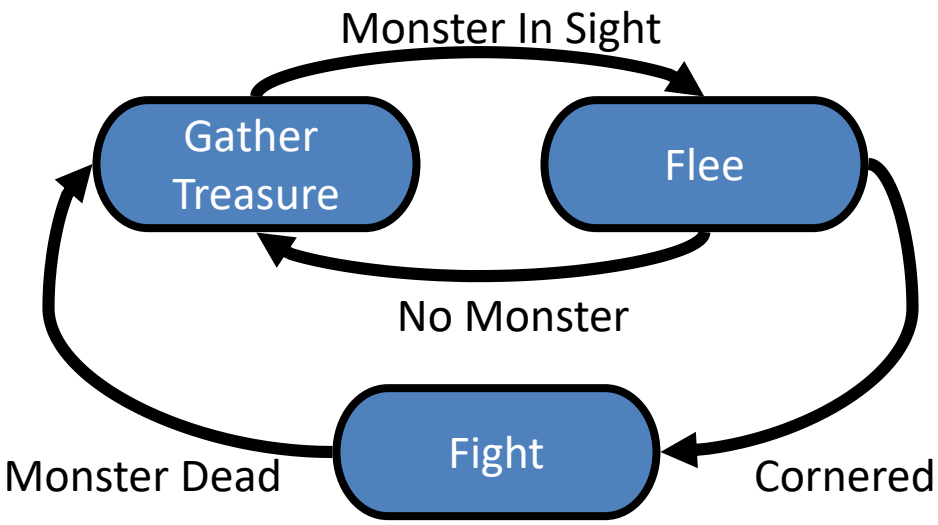
- Behavior
  - Can only be in one state at a given moment in time
  - Can make transitions from one state to another or to cause an output (action) to take place.

FSMs in Practice

- Each state represents some desired behavior
  - See “chunking” e.g. https://en.wikipedia.org/wiki/Chunking_(psychology)
- Transition function often resides across states
  - Each state determines subsequent states
- Can poll the world, or respond to events (more on this later)
- Support actions that depend on state & triggering event (Mealy) as well as entry & exit actions associated with states (Moore)

FSM as GAI

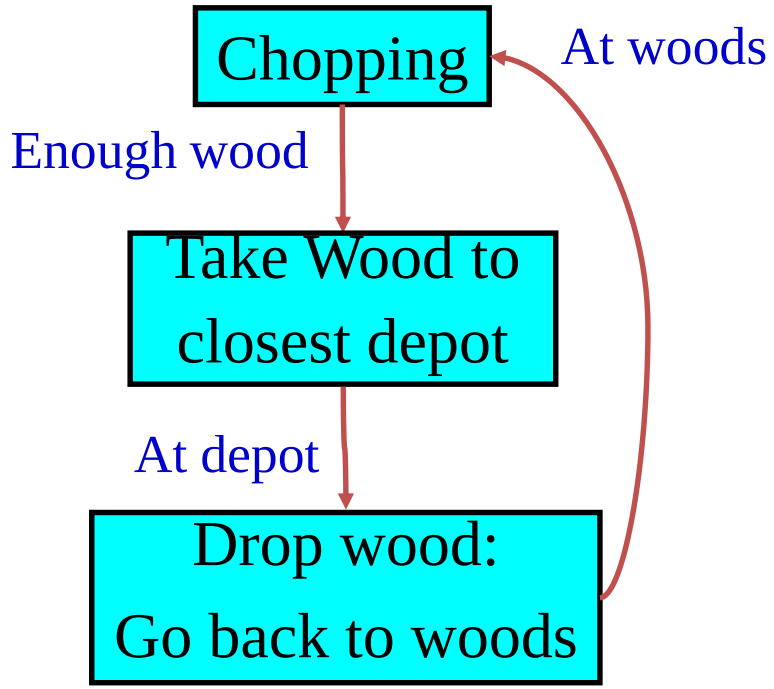
- Character AI modeled as sequence of mental states
- World events (can) force a change in state
- Mental model easy to grasp (for all)



Jarret Raim

Finite State Machines (FSMs)

- States: Action to take
  - Chase
  - Random Walk
  - Patrol
  - Eat
- Transitions:
  - Time
  - Events
  - Completion of action

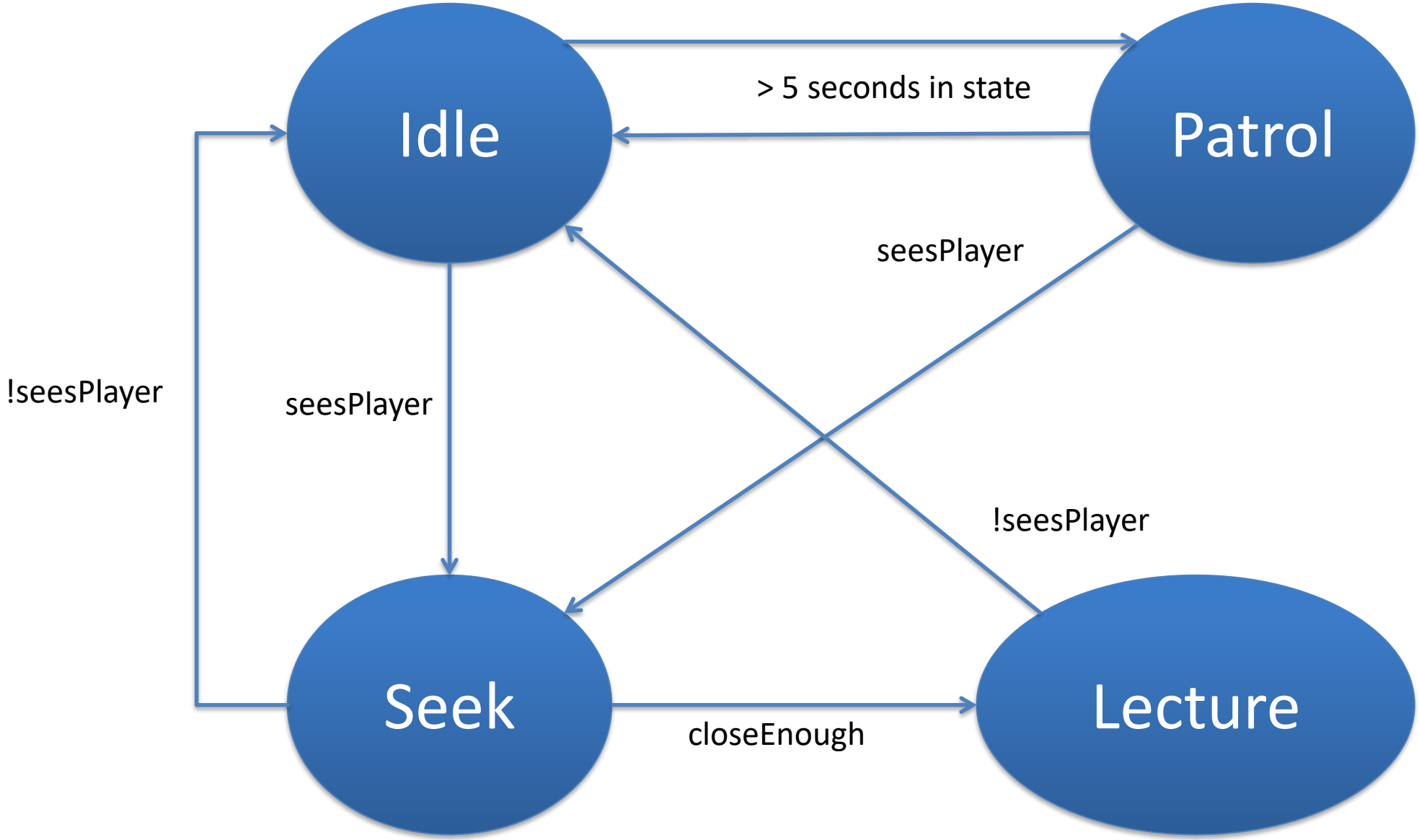


Question 1

Using the graph form, design/visualize an FSM for a security guard that patrols a school after hours. If it sees the player it needs to seek the player and give them a lecture till the player can slip away. Otherwise switch between Idle and Patrol.

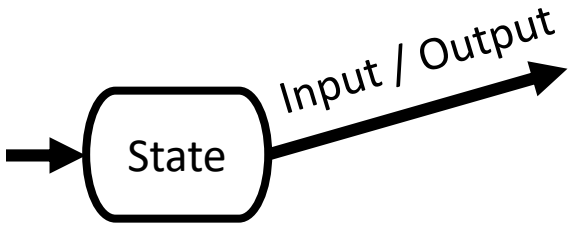
States = {Idle, Patrol, Seek, Lecture}

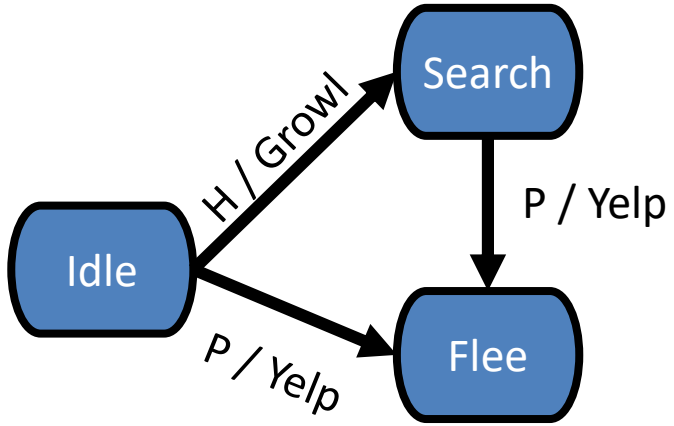
Feel free to make up variables.


Idle
Patrol
Seek
Lecture
> 5 seconds in state
seesPlayer
seesPlayer
!seesPlayer
!seesPlayer
closeEnough

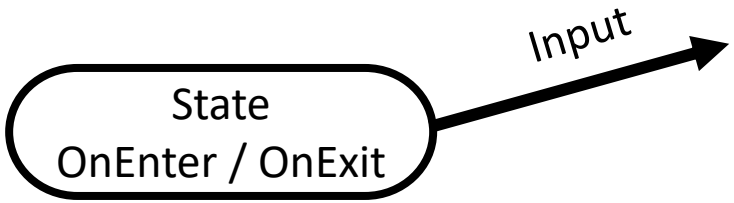
Mealy & Moore

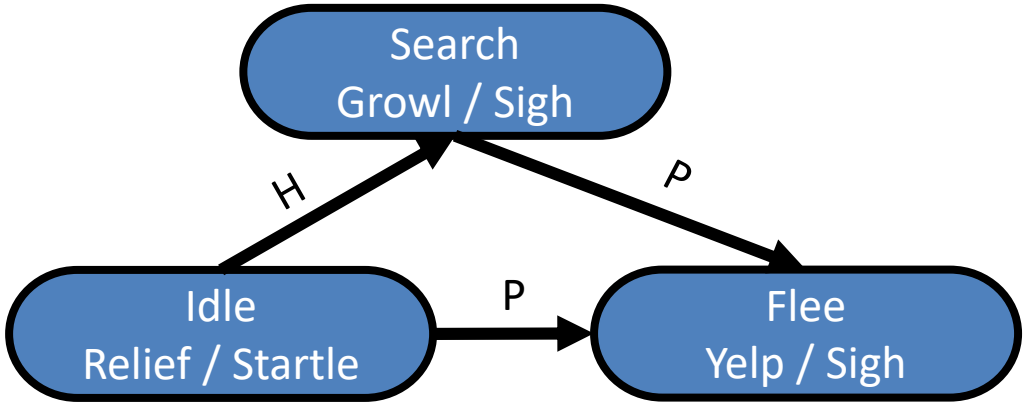
Mealy Output =
F(state, input)





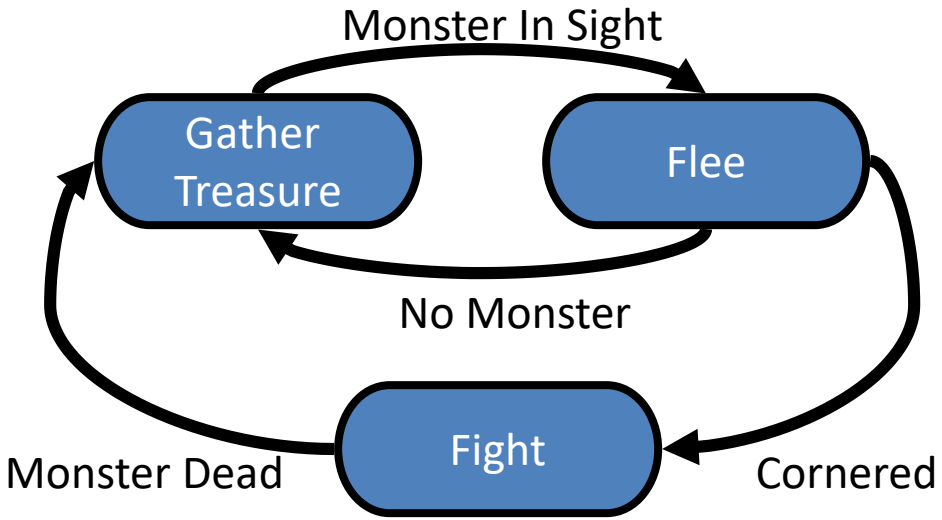
Moore Output =
F(state)





State Transition Table

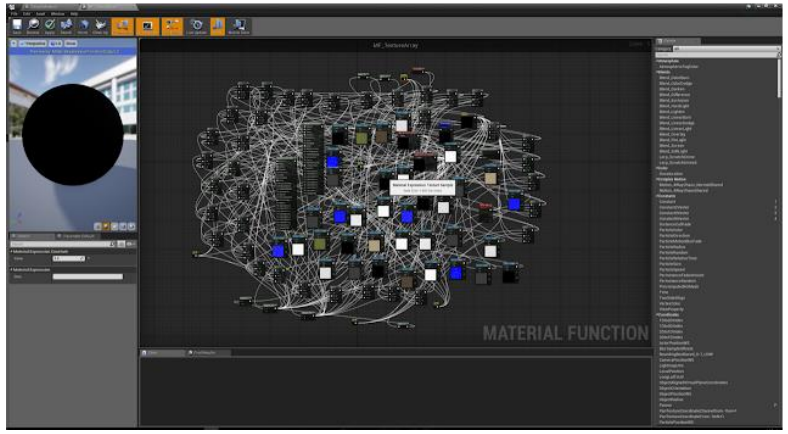
Current State	Condition	State Transition
Gather Treasure	Monster	Flee
Flee	Cornered	Fight
Flee	No Monster	Gather Treasure
Fight	Monster Dead	Gather Treasure



Advantages

- Predictable (can be disadvantage)
- Ubiquitous (not only in digital games)
- Quick and simple to code
- (can be) Easy* to debug
- Fast: Small computational overhead
- Intuitive
- Flexible
- Easy for designers without coding knowledge

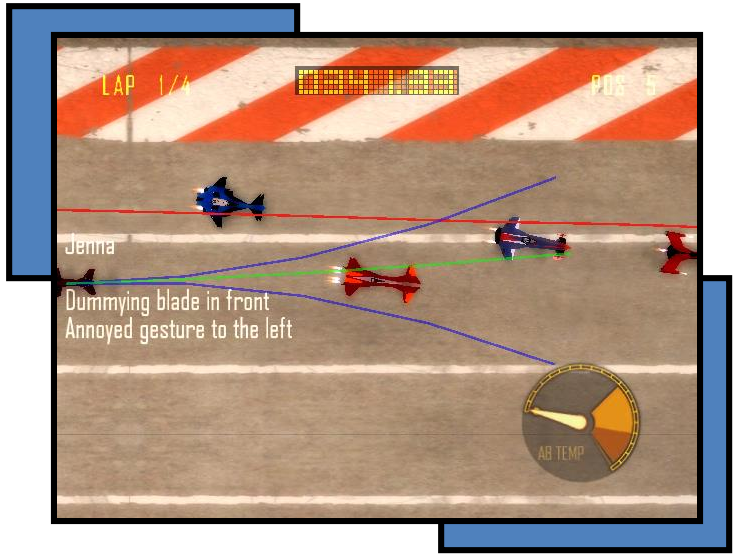
Disadvantages



- When it fails, fails hard:
  - A transition from one state to another requires forethought (get stuck in a state or can’t do the “correct” next action)
- Number of states can grow fast
  - Exponentially with number of events in world (multiple ways to react to same event given other variables)
- Number of transitions can grow even faster
- Doesn’t work with sequences of actions/memory

Debugging FSM's

- Offline Debugging
  - Logging
  - Verbosity Levels
- Online Debugging
  - Graphical representation is modified based on AI state
  - Command line to modify AI behavior on the fly.

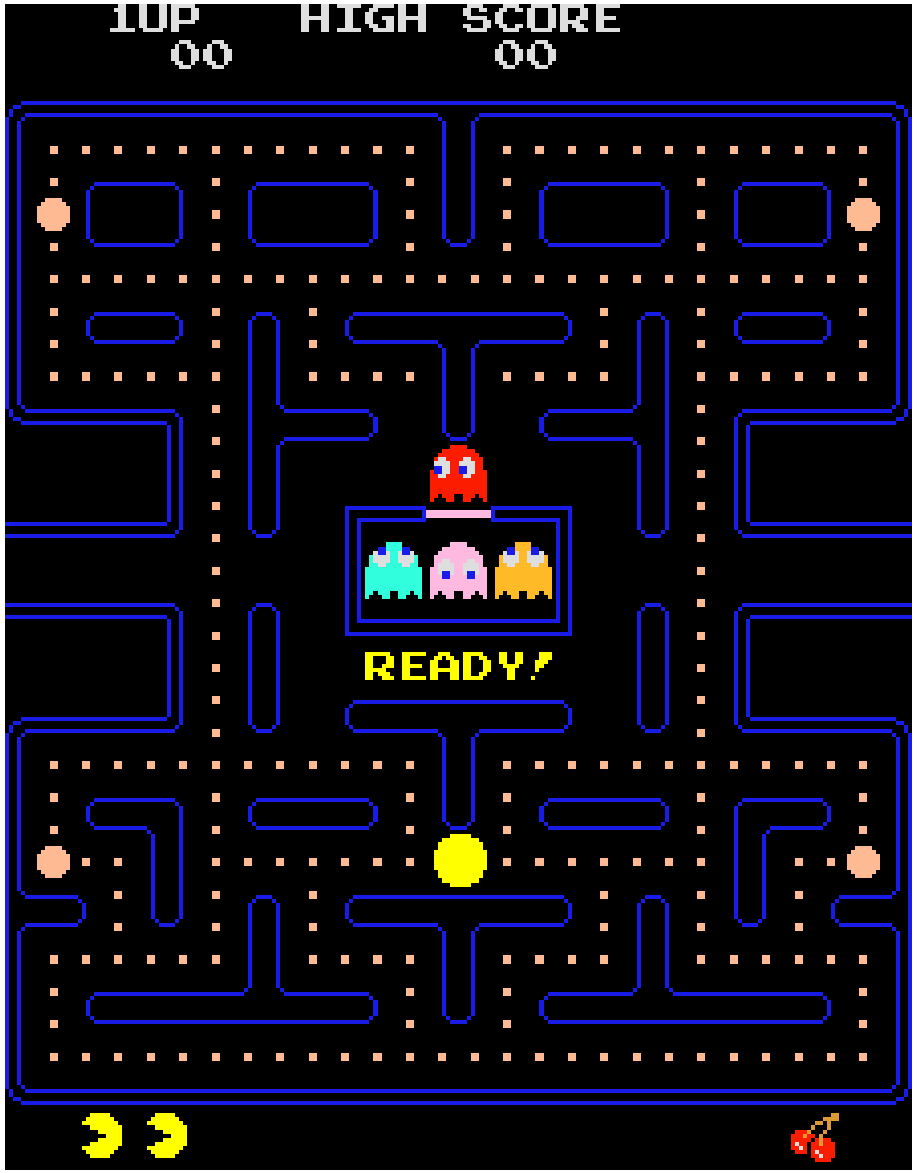


Where we see that they’re incredibly common, effective, and useful

FSM EXAMPLES

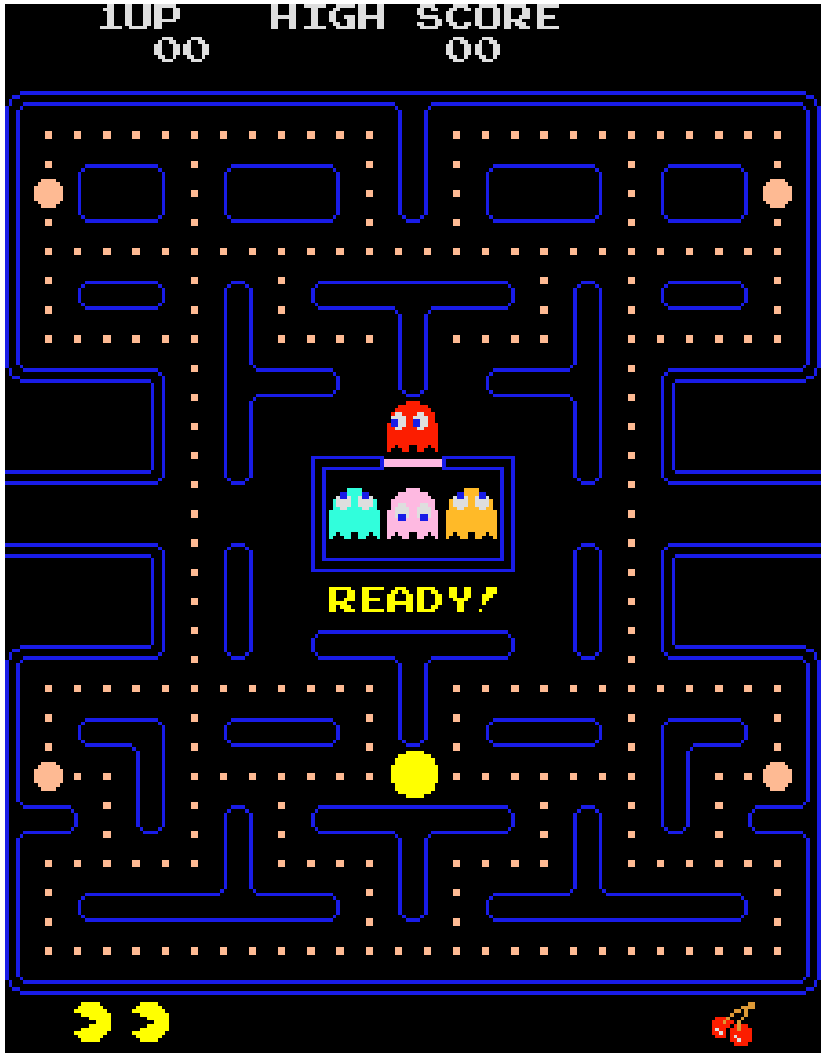
Pac-Man Ghosts

- Maze filled with food/pellets
  - Tile/grid world
  - Eat all pellets to advance levels
  - Ghost navigation to “target tile”
- Four ghosts pursue pman
  - Contact is -1 life for pman, ghost and pman positions reset
- Pman eats energizer pellet
  - Ghosts become vulnerable, flee pman
  - Contact returns ghost to monster pen / ghost house



Activity

- States:
- Transitions:

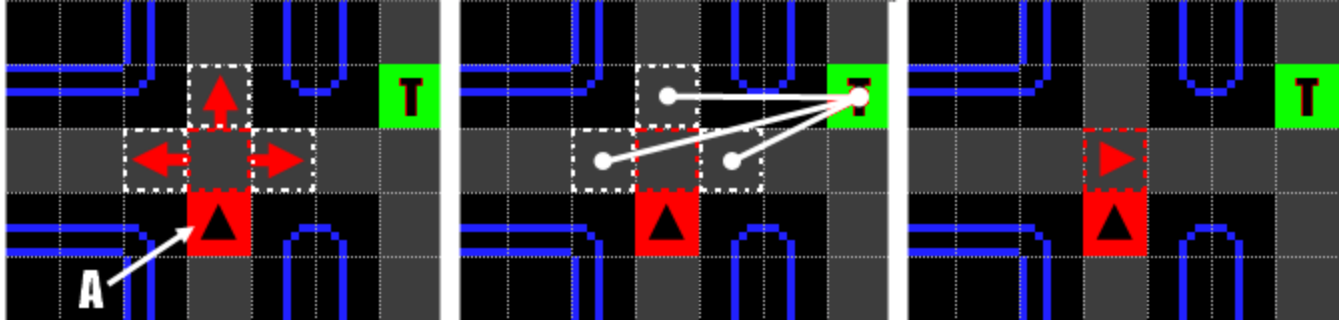


See also https://www.gamasutra.com/view/feature/3938/the_pacman_dossier.php?print=1

Target Tiles

- Most of the time, each ghost trying to reach a specific tile
  - Behavior revolves around trying to get from current to target
  - All ghosts use identical methods to travel towards their targets
  - Different ghost personalities derived from individual way each ghost selects target tile
    - Note: no restrictions that target tile must actually be possible to reach
    - many of the common ghost behaviors are a direct result of this possibility.

Movement: Seek

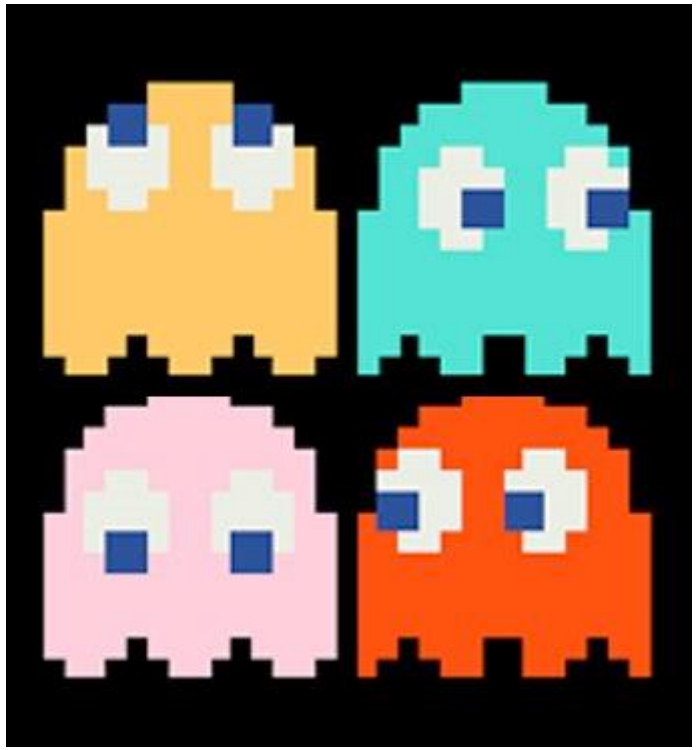


- Ghosts only think one step into future as they move
  - Enter new tile: look ahead to the next tile along current direction of travel. Decide which way to go from that tile.
    - Select test tile w/ shortest Euclidian distance to target tile
    - Tie-break: up, left, down, right
  - Upon reaching next tile, change direction to what was decided
  - Repeat (mode change updates target)
- Ghosts never voluntarily reverse direction
  - Tile w/ two exits → continue in the same direction
- Change from Chase/Scatter: must reverse direction upon entering next tile
  - Signals to player that ghost mode changed
  - Note: does not reverse when leaving Frightened (Why? Color reverts)

https://www.gamasutra.com/view/feature/3938/the_pacman_dossier.php?print=1

Mode: Chase

- Red: Shadow, blinky
  - “pursuer” or “chaser”
  - Oikake means “to run down or pursue”
  - Chase target: pac-man’s tile
- Pink: Speedy, pinky
  - “ambusher”
  - Machibuse means “to perform an ambush”
  - Chase target: pac-man’s tile + 4 ahead
- Blue: Bashful, inky
  - “whimsical”
  - Kimagure means “a fickle, moody, or uneven temper”
  - Chase target: double vector from red ghost to (pac + 2 ahead)
- Orange: Pokey, Clyde
  - Otoboke means “feigning ignorance”
  - Guzuta means “one who lags behind”
  - Chase target: dist < 8 ? scatter tile : pac-man’s tile

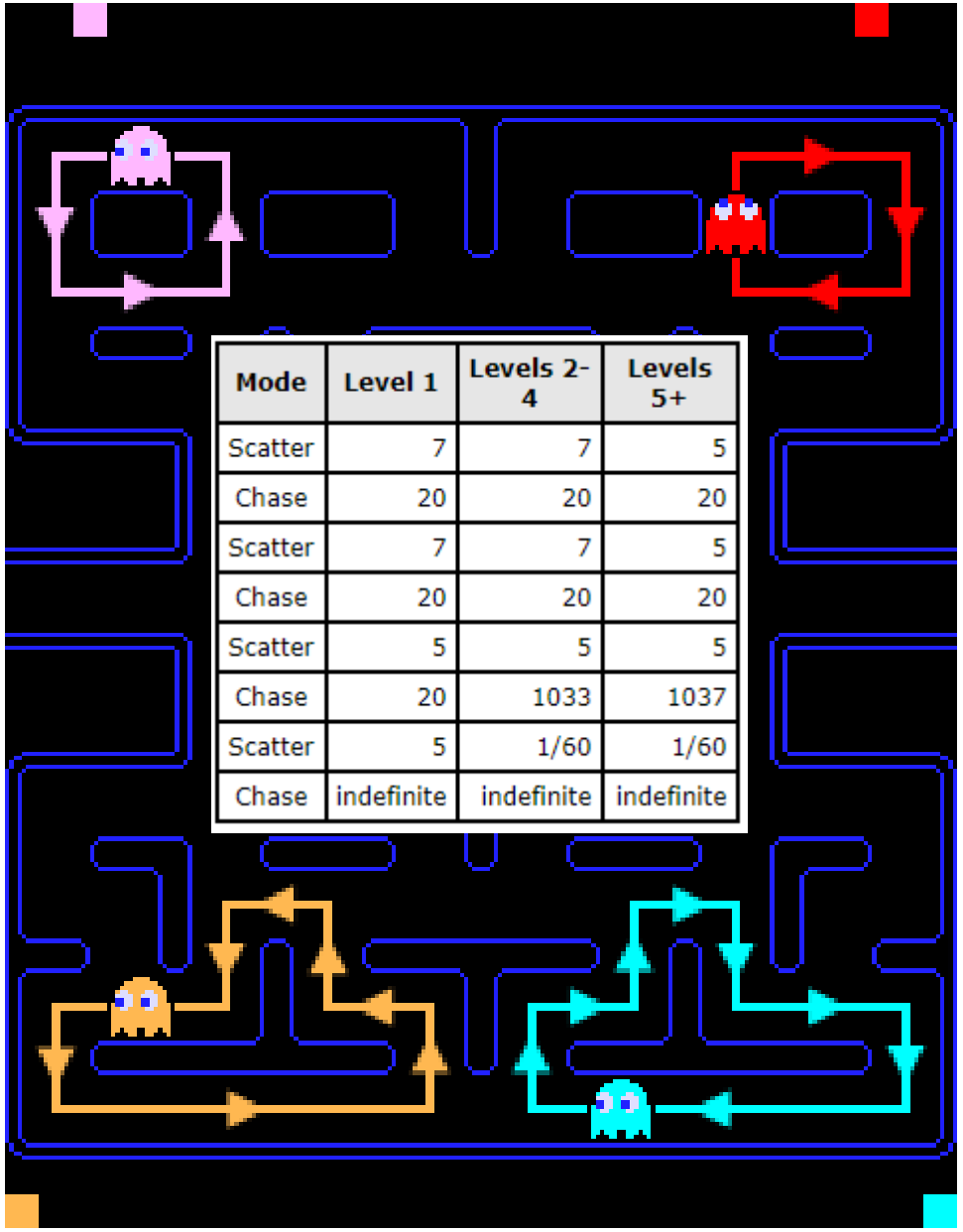


https://www.gamasutra.com/view/feature/3938/the_pacman_dossier.php?print=1

http://gameinternals.com/post/2072558330/understanding-pac-man-ghost-behavior

Mode: Scatter

- Not “scatter from player” but “scatter to the winds”
- In “scatter” mode, each ghost's target tile is changed to corner location outside the maze
  - can't reach the target tile, so circles until return to Chase
  - Blinky’s Cruise Elroy
- Designed to make attacks on player come in waves as opposed to endless assault



Mode	Level 1	Levels 2-4	Levels 5+
Scatter	7	7	5
Chase	20	20	20
Scatter	7	7	5
Chase	20	20	20
Scatter	5	5	5
Chase	20	1033	1037
Scatter	5	1/60	1/60
Chase	indefinite	indefinite	indefinite

https://www.gamasutra.com/view/feature/3938/the_pacman_dossier.php?print=1

Mode: Evade/Frightened

- Ghosts enter frightened mode when pman eats energizer
  - Ghosts all turn dark blue (early levels), become vulnerable, and aimlessly wander the maze for a few seconds
  - Ghosts flash moments before returning to their previous mode
- Movement in frightened mode
  - At intersection, choose random direction
  - Only mode w/ use of target tile
- PRNG: pseudo-random number generator
  - Gets reset with an identical seed value every new level and every new life

Trajectory Update

- HW4: A*
- To come: More decision making
  - Planning
  - Decision trees
  - Behavior trees
  - Rule based systems
  - Fuzzy Logic
  - Markov Systems