COSC 442: Mobile Educational Game Development

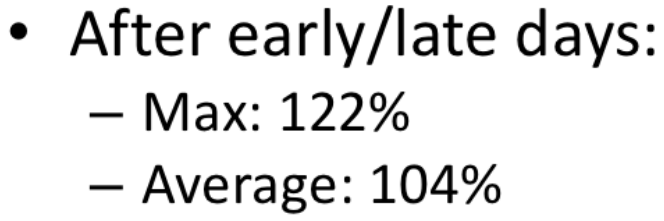
Dr. Bowen Hui

University of British Columbia Okanagan

A2 Feedback

- Generally very well done
  - See handout with submitted feedback on “Best part of the game” and “Most needed improvement”

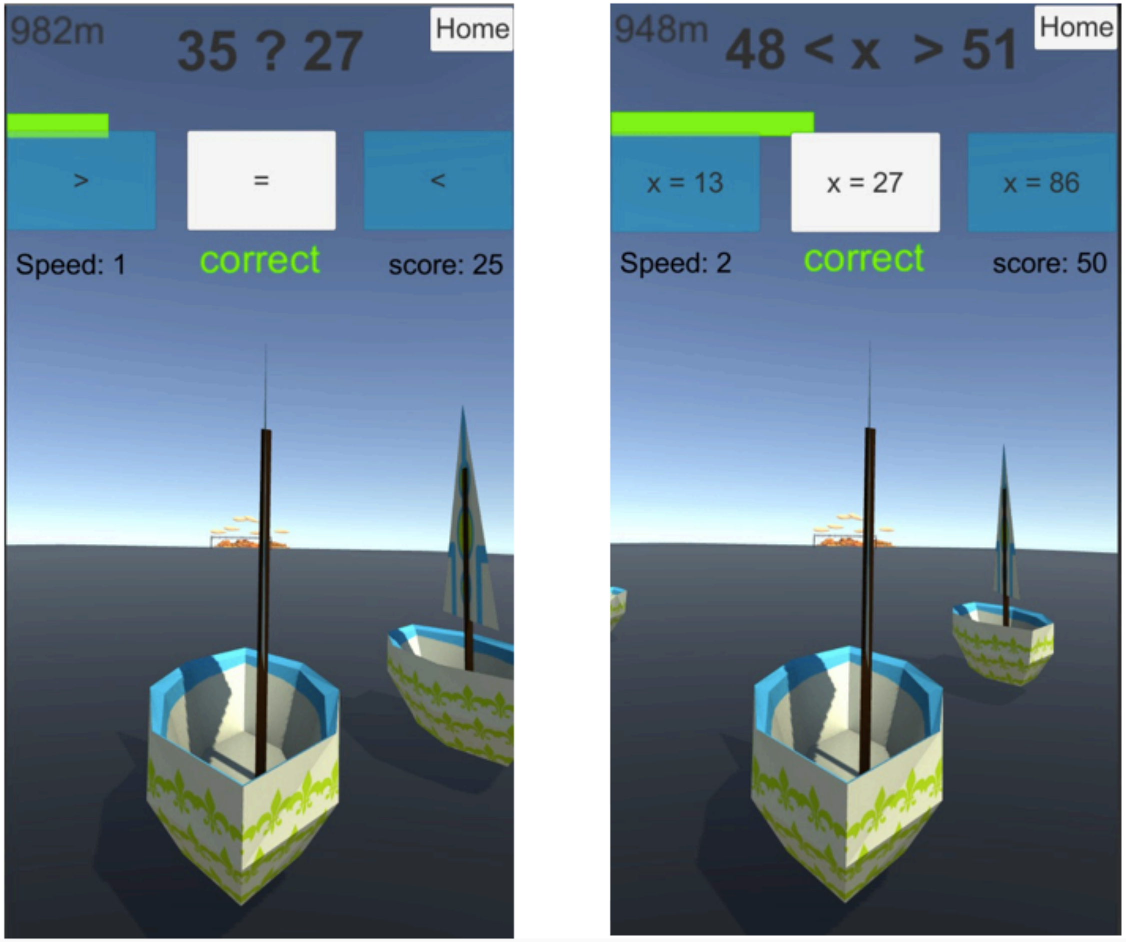
- Before early/late days:
  - Max: 100%
  - Average: 95%



- After early/late days:
  - Max: 122%
  - Average: 104%

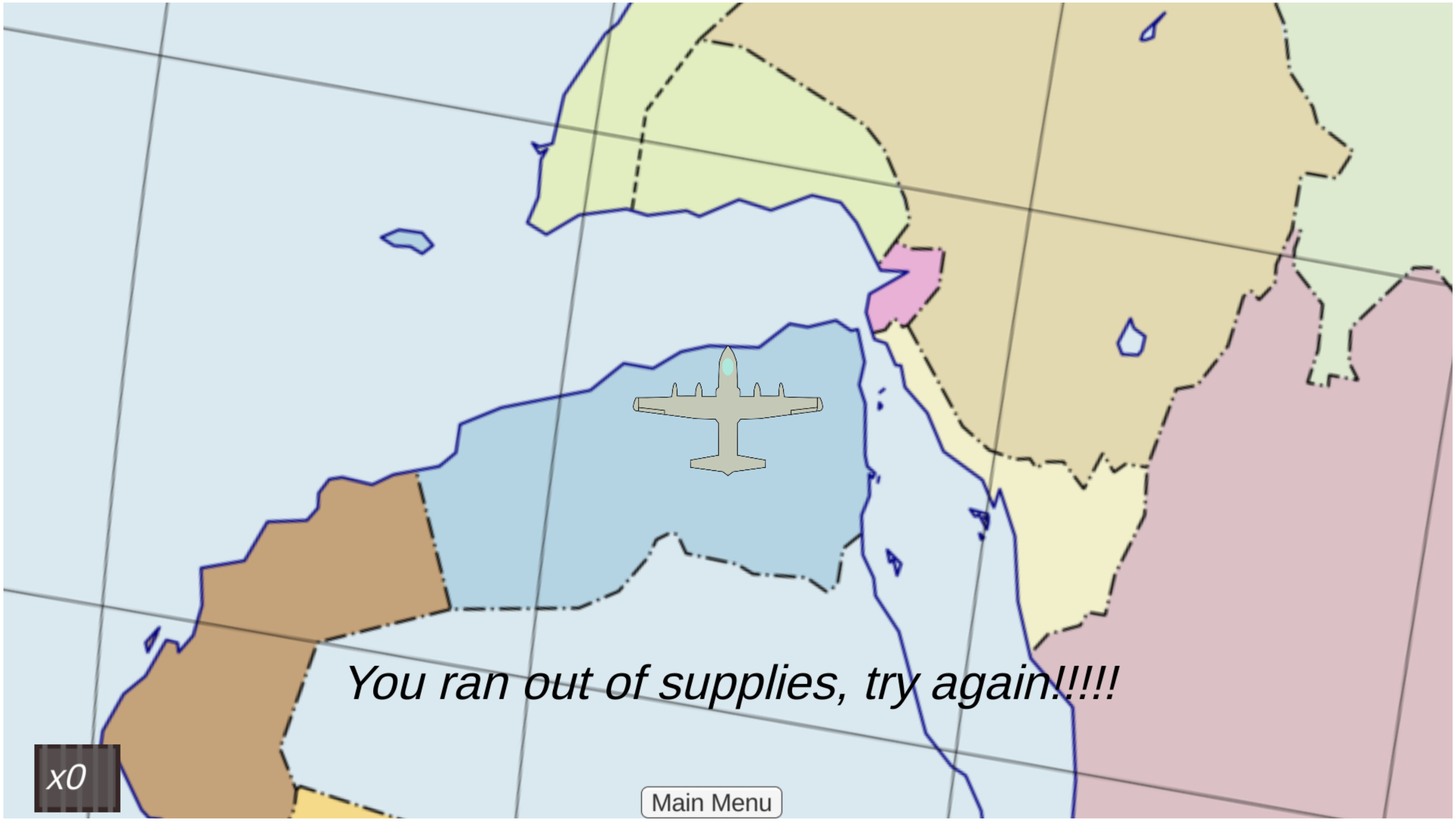
- Main comment:
  - Game metrics to implement for A3 collects objective data based on game events
  - Heuristic evaluation can collect additional subjective data via questionnaire

Vote Results: Best Learning Game



Boat Racer by Ravan Klar

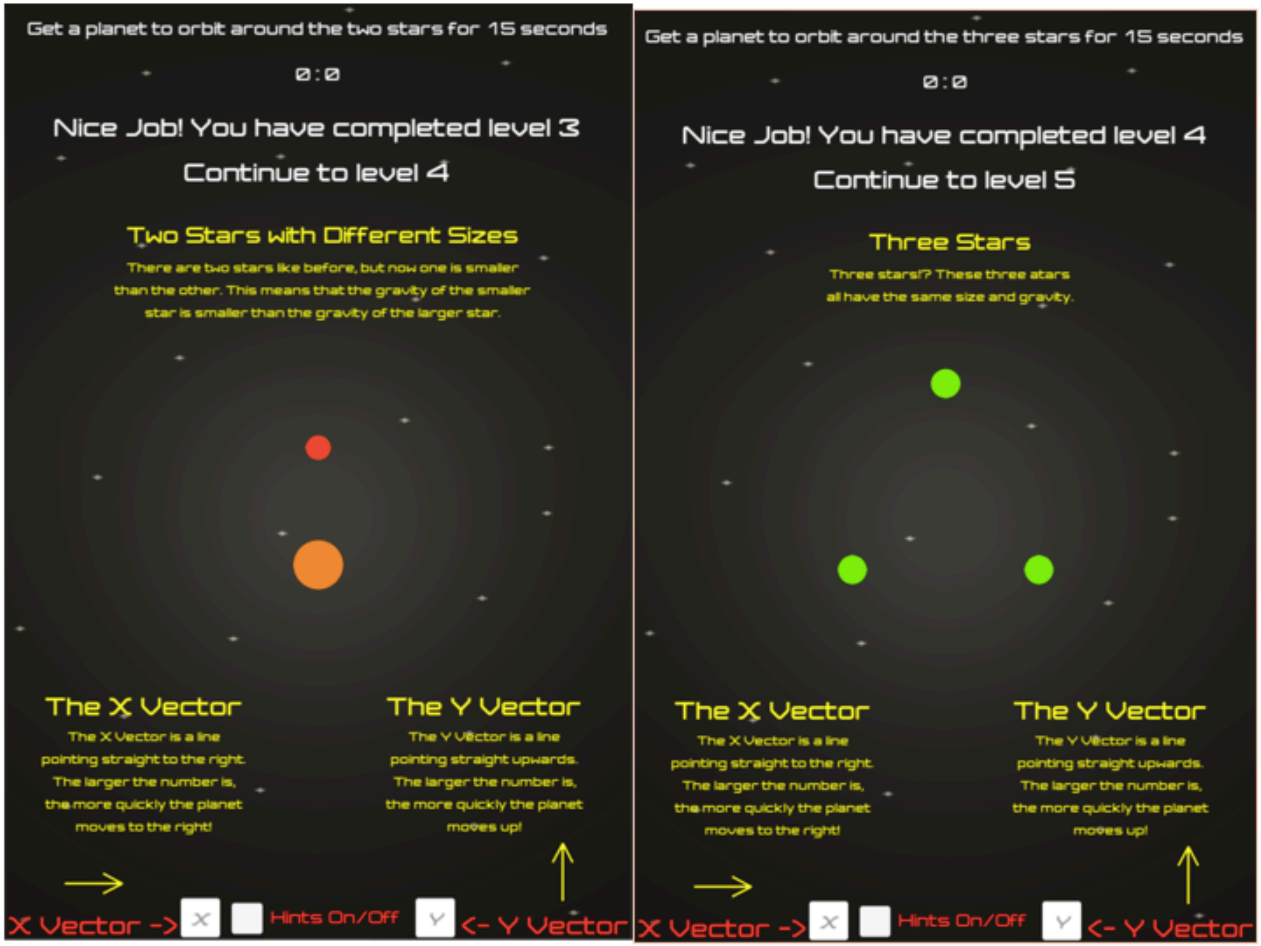
Vote Results: Most Fun Game



You ran out of supplies, try again!!!!!

Foreign Aid Fighter by Eric Nelson

Vote Results: Most Creative Game



Get a planet to orbit around the two stars for 15 seconds

0:0

Nice Job! You have completed level 3

Continue to level 4

Two Stars with Different Sizes

There are two stars like before, but now one is smaller than the other. This means that the gravity of the smaller star is smaller than the gravity of the larger star.

The X Vector

The X Vector is a line pointing straight to the right. The larger the number is, the more quickly the planet moves to the right!

The Y Vector

The Y Vector is a line pointing straight upwards. The larger the number is, the more quickly the planet moves up!

Get a planet to orbit around the three stars for 15 seconds

0:0

Nice Job! You have completed level 4

Continue to level 5

Three Stars

Three stars!? These three atars all have the same size and gravity.

The X Vector

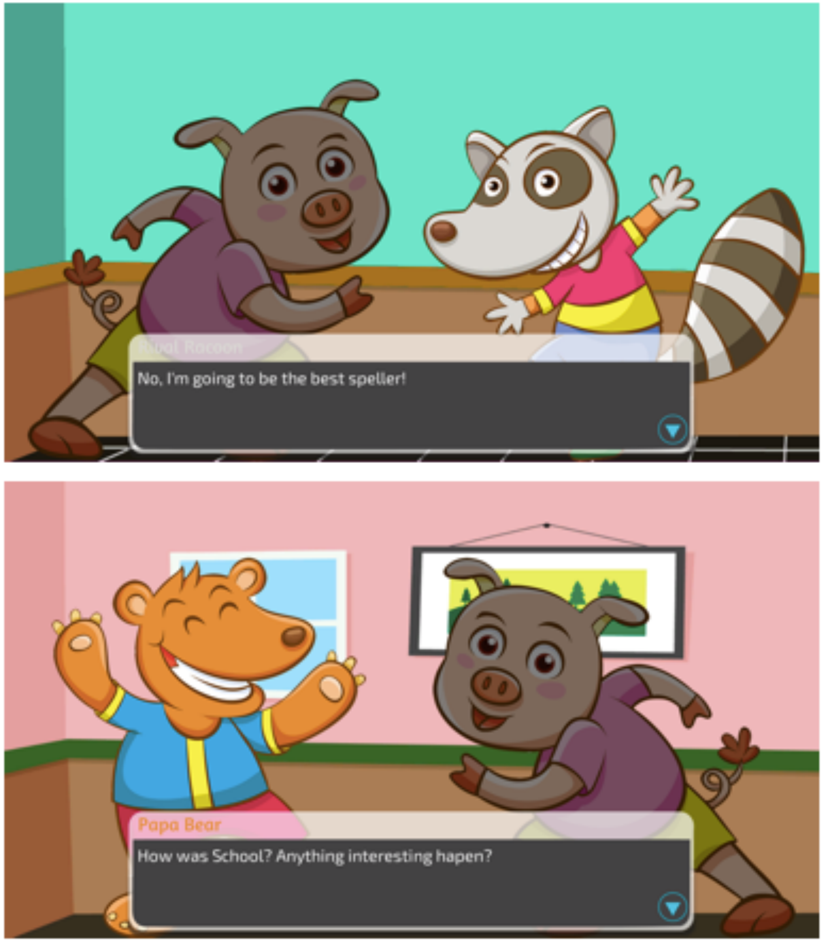
The X Vector is a line pointing straight to the right. The larger the number is, the more quickly the planet moves to the right!

The Y Vector

The Y Vector is a line pointing straight upwards. The larger the number is, the more quickly the planet moves up!

Vector Space by Nick Borle

Vote Results: Best Graphics Game



Rival Racoon

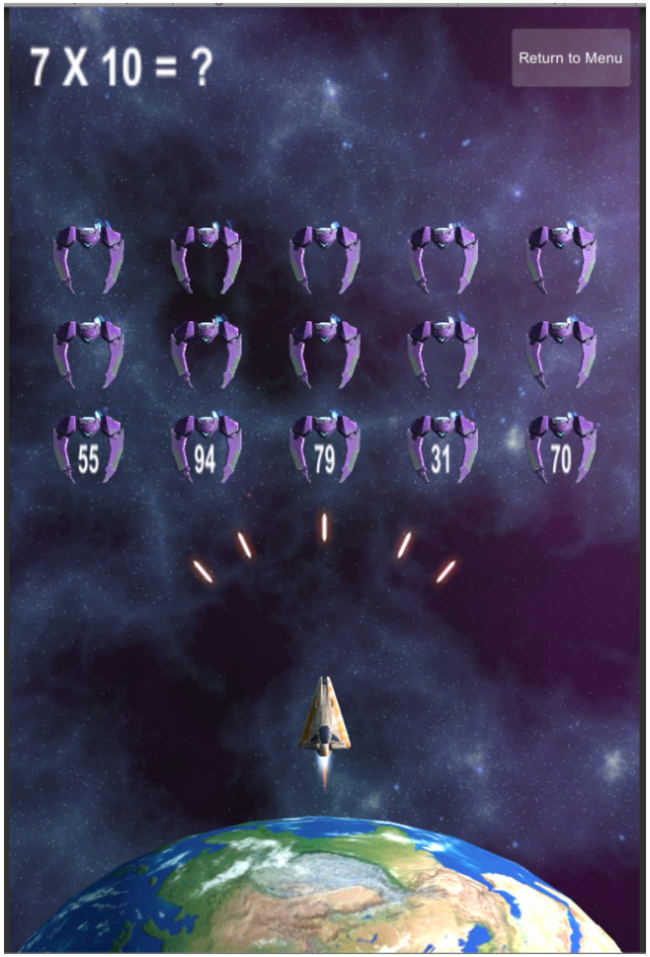
No, I'm going to be the best speller!

Papa Bear

How was School? Anything interesting hapen?

Simple Bee by Hayun Jin and Ileri Oyedele

Vote Results: Best Audio Game



Space Times by Julien Butler, Kayla Raine, and Alex Shaw

A3 Reminder

- Things to do before next week:
  - Ensure A2 prototype works (doesn’t crash)
  - Test your games thoroughly
  - Implement event logging (objective data)
  - Setup quantitative questionnaire (subjective data)

- Next week in class:
  - Run heuristic evaluation to collect data with each other
  - Schedule will be available at beginning of class
  - Participating in evaluating others’ games contribute to in-class exercise marks

Steps in Running Experiment

- Explain to the participant what your game is about
- Let them go through the tutorial or walk them through your game activity (show them the input controls)
- Answer any questions they might have
- Let participant play your game for ~10 minutes
  - Observe participant actions and log problems
  - Resolve problems if they arise
  - Watch the timer
- Ask participant to complete questionnaire
- Thank the participant

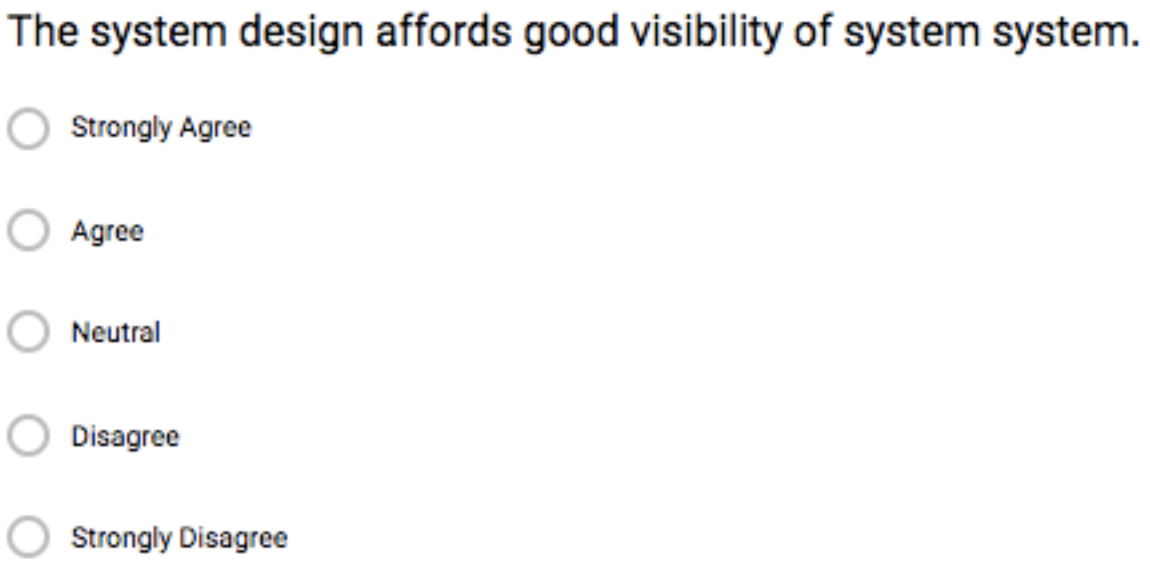
Logging Observations

- Remember each problem identified should have the following fields:
  - Issue identified with the game
  - Severity level
  - Heuristic violated
  - Description

- See previous slides for examples

A3 Questionnaire

- Define 5-point Likert scale questions for each of the heuristic
- Automate questionnaire (e.g. use Google forms)
- Example question for the heuristics “visibility of system status”: In the following, indicate how much you agree with each statement.



The system design affords good visibility of system system.

○ Strongly Agree

○ Agree

○ Neutral

○ Disagree

○ Strongly Disagree

Role of AI in Games

- Graphics and rendering is given highest priority in project scope and resources
  - AI typically done last
  - AI is very dependent on concrete details of game environment

- Goal of AI in games:
  - Imitate human-like characteristics
  - Make game fun and believable
  - ***Not*** to compute most optimal behaviour to win against player
  - Very different from AI in research/academia

- Too much unpredictability can be undesirable!
  - Game designers can't guarantee a fun game

Example Use of “AI” in Games

- Space Invaders (1978)
  - Using stored patterns to direct enemy movement
  - Incorporate random movement patterns
- Pacman (1980)
  - Four different ghosts with different personality behaviours
- Sims (2000)
  - Different objects affected character’s behaviours and relationships
  - Player defined characteristics that impact character choices in game
- Mortal Kombat Series
  - More realistic enemies
- Petz (2007)
  - Learns player habits and develops a deeper, more personal relationship with player
- Metal Gear Solid (2015)
  - NPC hunt players through disturbances in environment (e.g. footprints)
- Tuebor (2016)
  - Adapts difficulty to player’s behaviour to match their ability

AI Techniques in Games

- Most common is for controlling non-player characters (NPCs)

- Most common techniques used:
  - Finite state machine (FSM)
  - Search Tree

Why only simple AI techniques?

- Niche area: uses machine learning to adapt behaviour throughout game play
  - Good for developing relationship with player
  - E.g. Petz

AI Techniques in Games

- Most common is for controlling non-player characters (NPCs)
- Most common techniques used:
  - Finite state machine (FSM)
  - Search Tree

Simple AI techniques

- Relatively predictable behaviour
- Requires min. resources

- Niche area: uses machine learning to adapt behaviour throughout game play
  - Good for developing relationship with player
  - E.g. Petz

Background of FSM

- System uses FSM to implement pattern matching operations

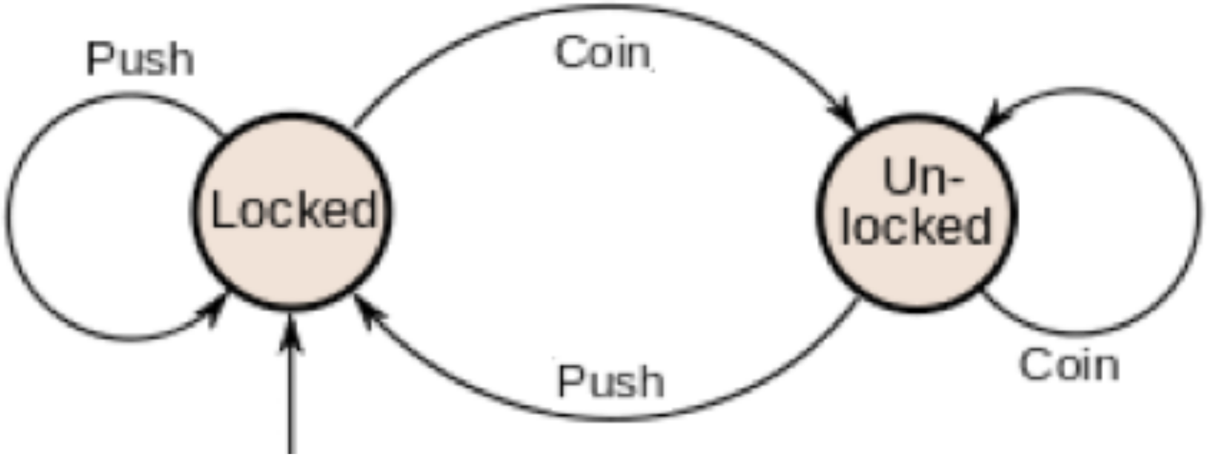
- Pattern matching example in commands:
  - E.g. ls *
  - E.g. ls *.cs

- Automata theory: Study of formal languages and machines that accept/reject them

FSM Example

- This machine has:
  - States: Locked, Unlocked
  - Initial state: Locked
  - Actions: Push, Coin
  - Transitions: Defined by arcs
  - Final state: None

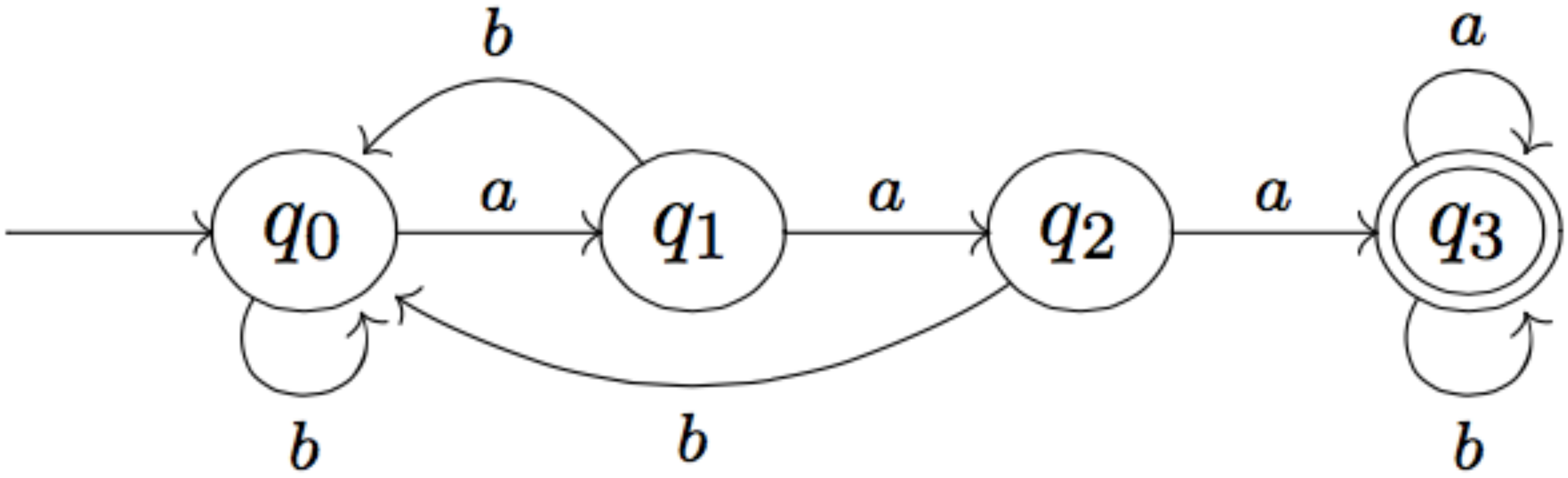
Image taken from wikipedia.org



A More Abstract Example

- This machine has:
  - States: {q0, q1, q2, q3}
  - Initial state: q0
  - Alphabet: {a, b}
  - Transitions: Defined by arcs
  - Final state: q3

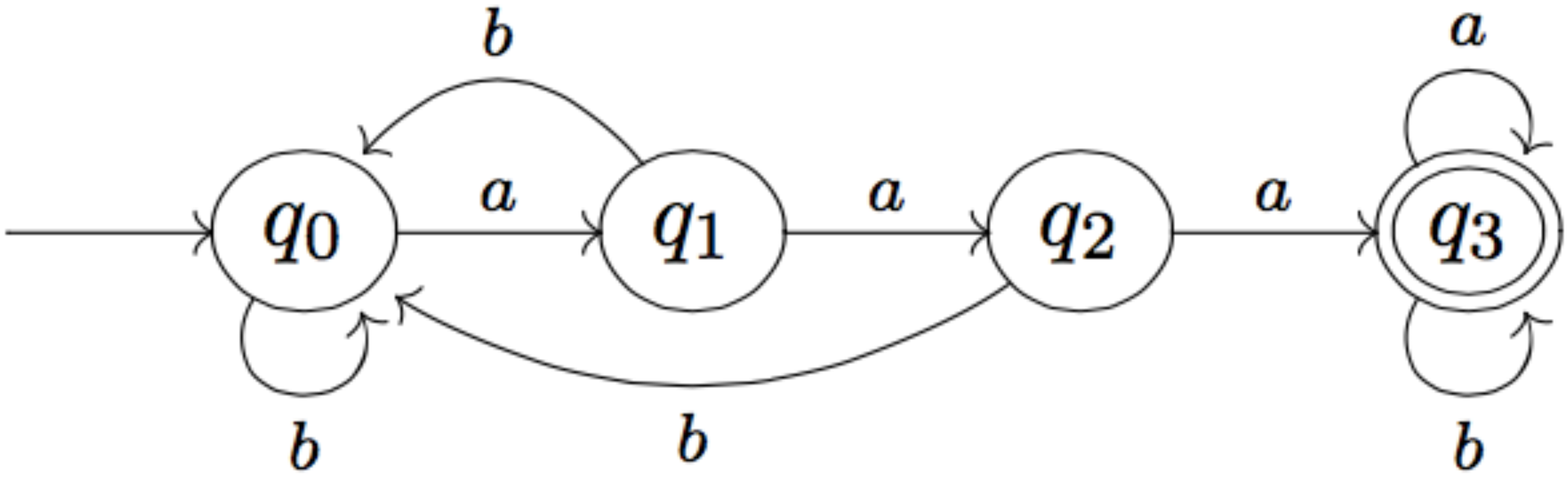
Which strings are accepted by this FSM?



A More Abstract Example

- This machine has:
  - States: {q0, q1, q2, q3}
  - Initial state: q0
  - Alphabet: {a, b}
  - Transitions: Defined by arcs
  - Final state: q3

E.g. Design FSM that accepts language defined by: ((b*) (ab)* (b*) (aab)* (b*))* aaa (a*b*)



What is a FSM

- Abstract notion of “machine”
  - Finite set of states
  - Transitions indicate when one state can change to another (don’t care how it’s done)
  - Alphabet (or symbols, or actions) define possible transitions
  - One initial state
  - Zero or more final states

- Commonly used to model elevators, traffic lights, combo locks, parsing text, game character behaviour

Examples of using FSMs to model game characters?

Simplified FSM in Shooting Game

NPC starts in “wander” state

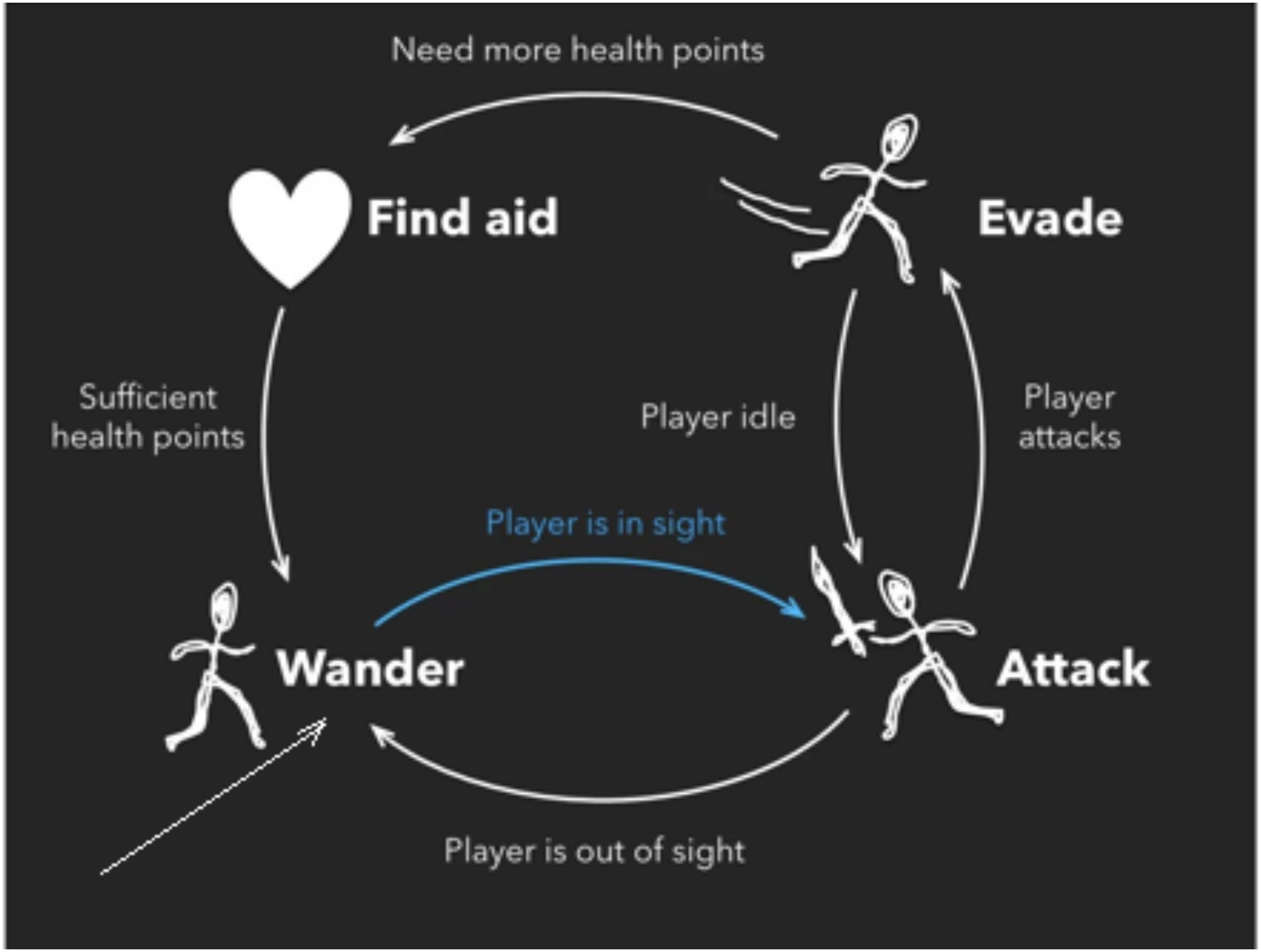
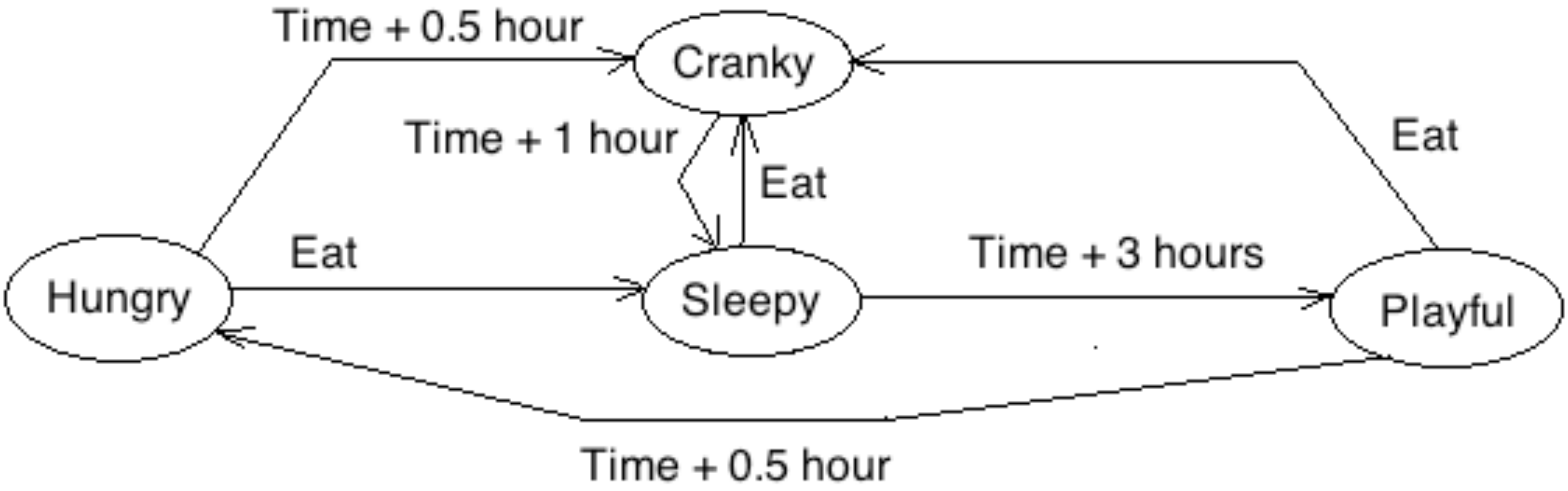


Image taken from http://sitn.hms.harvard.edu/flash/2017/ai-video-games-toward-intelligent-game/

Translating FSM into Code

- General setup:
  - Class defines possible states and transitions
  - Class manages initial state and current state
  - Every state is defined as a method
  - Repeat:
    - Update method takes input action, follows predefined transition, go to next state
    - Execute code in that state

Example for Little Kids



Code Structure for Child FSM

What does transition matrix look like?

```
Class Child
{
   // define possible states
   // define possible actions
   // encode possible transitions into matrix, one per action

   constructor method()
   {
      // initialize current state as initial state
   }

   hungry method() { ... }
   cranky method() { ... }
   sleepy method() { ... }
   playful method() { ... }

   updateState method() { ... }
}
```

Code Structure for Child FSM

```
Class Child
{
   // define possible states
   // define possible actions
   // encode possible transitions into matrix , one per action

   constructor method()
   {
      // initialize current state as initial state
   }

   hungry method() { ... }
   cranky method() { ... }
   sleepy method() { ... }
   playful method() { ... }

   updateState method() { ... }
}
```

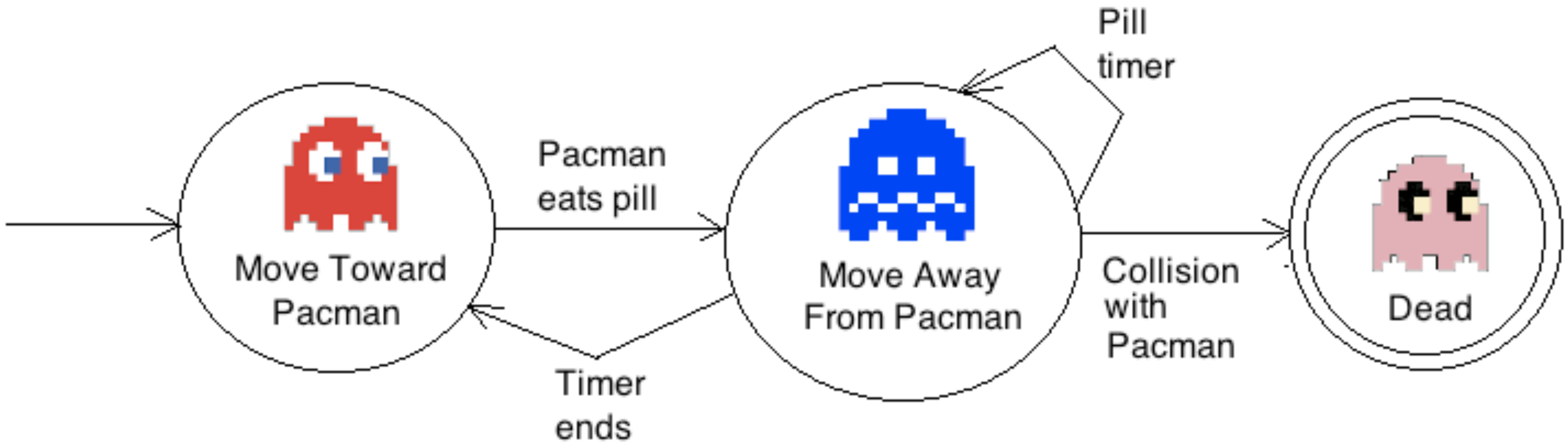
Alternate pattern:

- No transition matrix and updateState method
- Encode transition to next state as last line in each state method

Pacman Ghost FSM

- How to define an FSM for Pacman ghost behaviour?
  - Generally just moves around the maze
  - When Pacman in normal mode:
    - Ghost moves towards Pacman
  - When Pacman in pill mode:
    - Ghost moves away from Pacman
  - When Pacman in pill mode and collision occurs:
    - Ghost dies

Pacman Ghost FSM



FSM Considerations

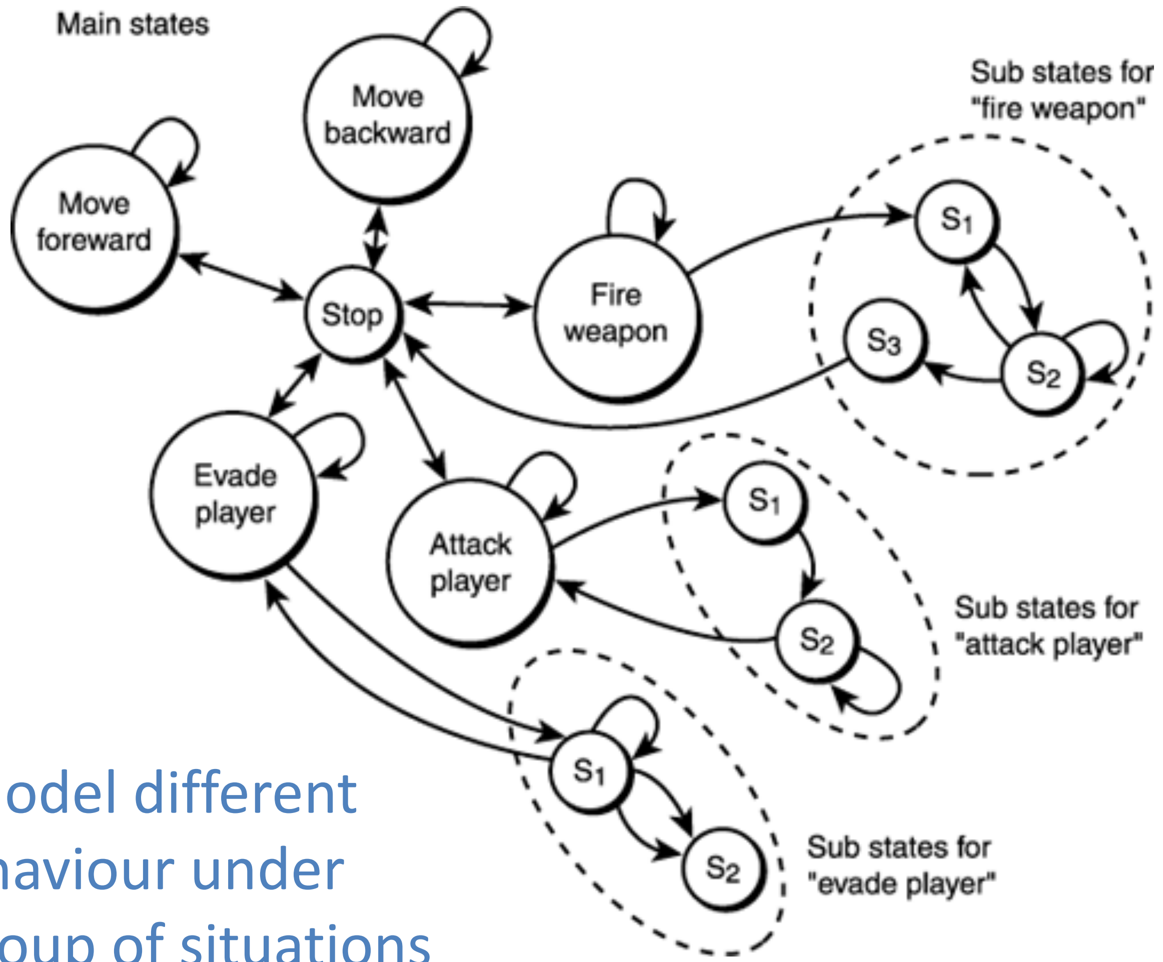
- Fairly simple to design
- Straightforward to translate into code
- Models actions by NPC and reactions to player
  - Actions and reactions only based on current state
  - No history or future actions are considered
- Simple FSMs can become very predictable

How might we make FSMs less predictable?

Modeling More Interesting Behaviour?

- How to define an FSM for a fighter behaviour?
  - Can move around, fire weapon, or evade player
  - When player not in sight:
    - Move around in search of player
  - When player is in sight:
    - Fire close range weapon if player within n distance
    - Fire long range weapon if player outside n distance
  - When player shoots at character:
    - Evade shot

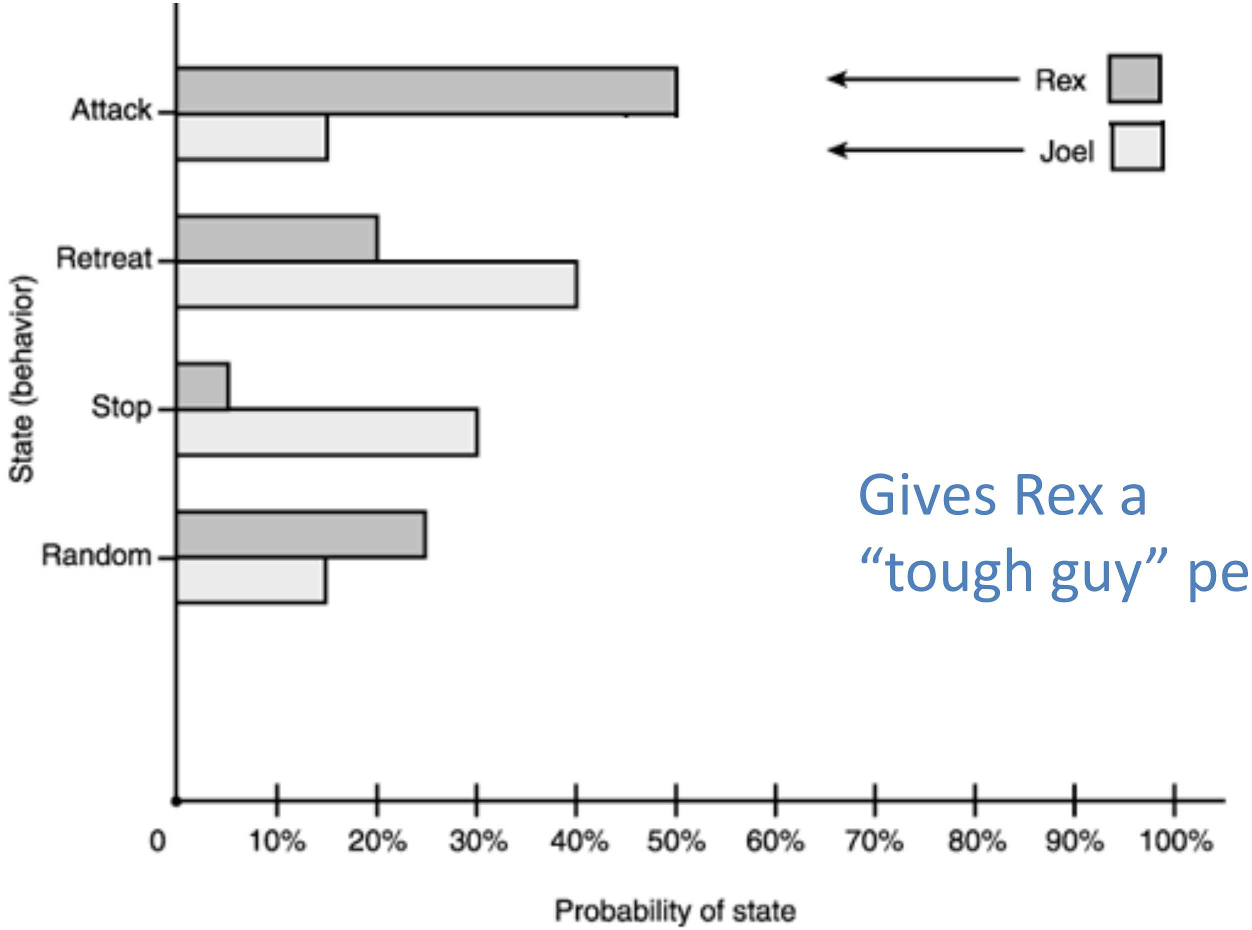
Master FSM Template for Fighter NPC



Substates model different types of behaviour under the same group of situations

Personality Distribution in FSMs

Image taken from yaldex.com

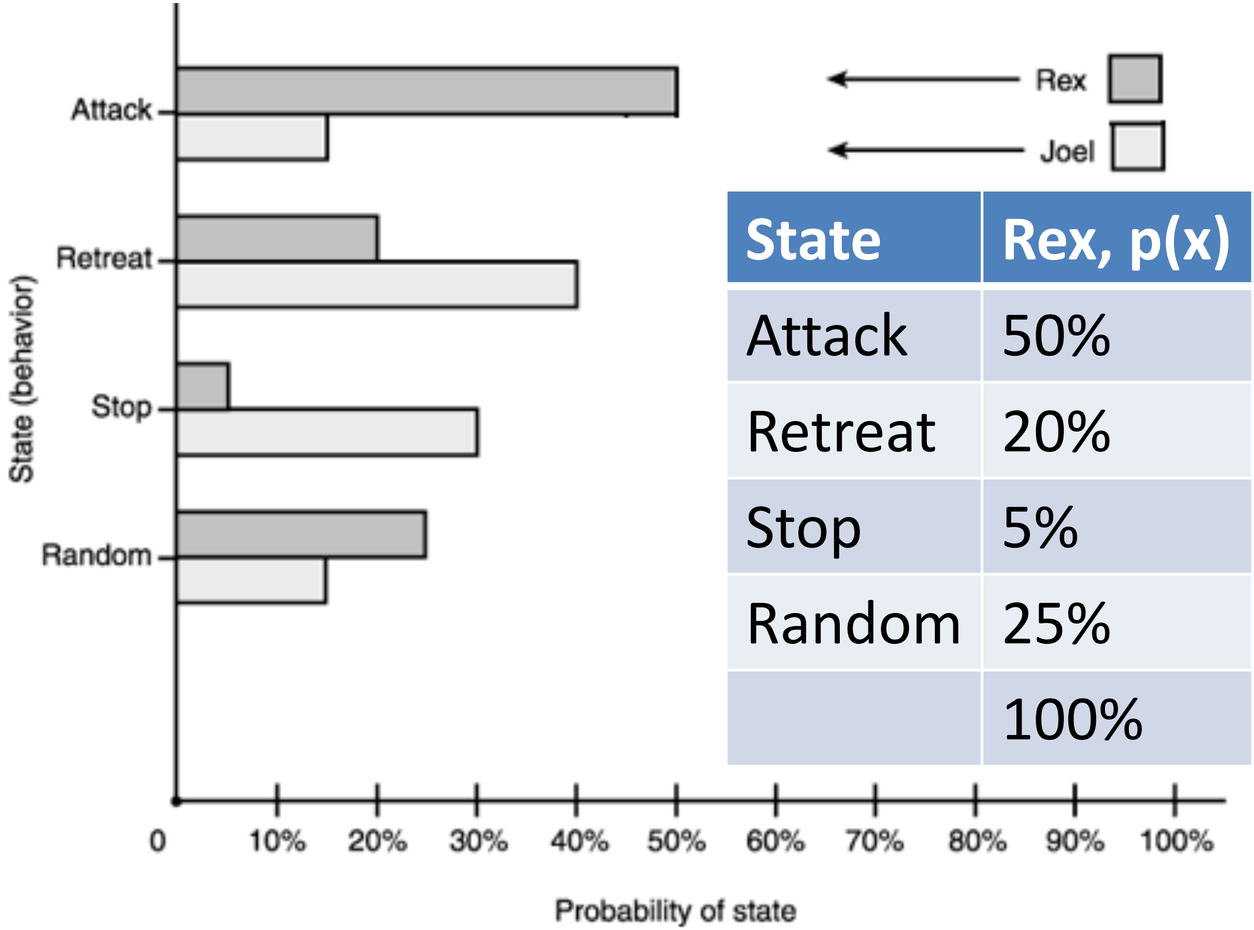


Gives Rex a "tough guy" personality

How to model using FSMs?

Personality Distribution in FSMs

Image taken from yaldex.com



State	Rex, p(x)	Joel, p(x)
Attack	50%	15%
Retreat	20%	40%
Stop	5%	30%
Random	25%	15%
	100%	100%

All behaviour percentages must sum up to 100% per character

Radii of Influence

- Character can switch behaviour based on a variable
  - E.g. distance from player
  - E.g. number of enemies present

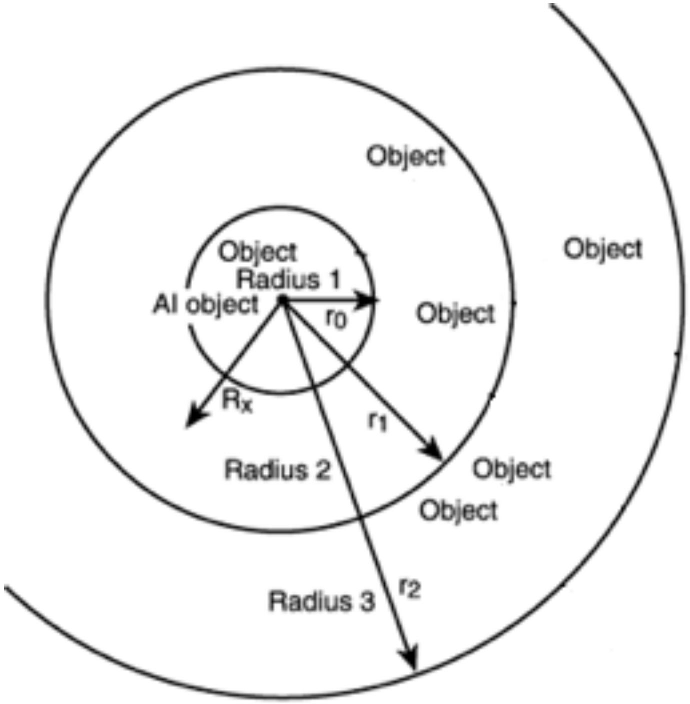


Image taken from yaldex.com

How do you implement different NPC behaviour to reflect radii of influence?

Radii of Influence

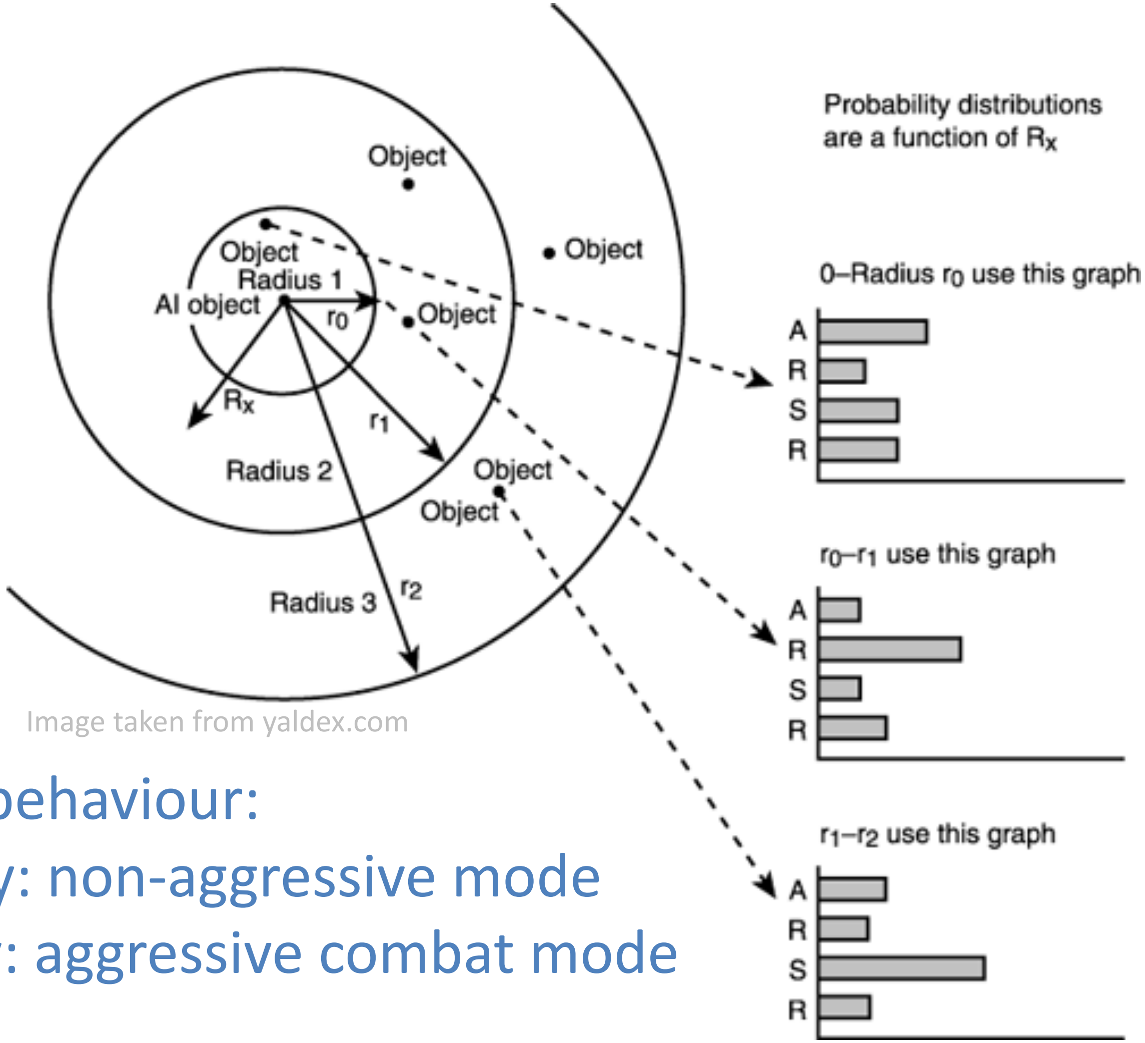


Image taken from yaldex.com

Character behaviour:

- Far away: non-aggressive mode
- Close by: aggressive combat mode

Implementation: use different probability tables per situation

Alternative Representations

- Decision trees (behavior trees)
  - No machine learning involved, just the output

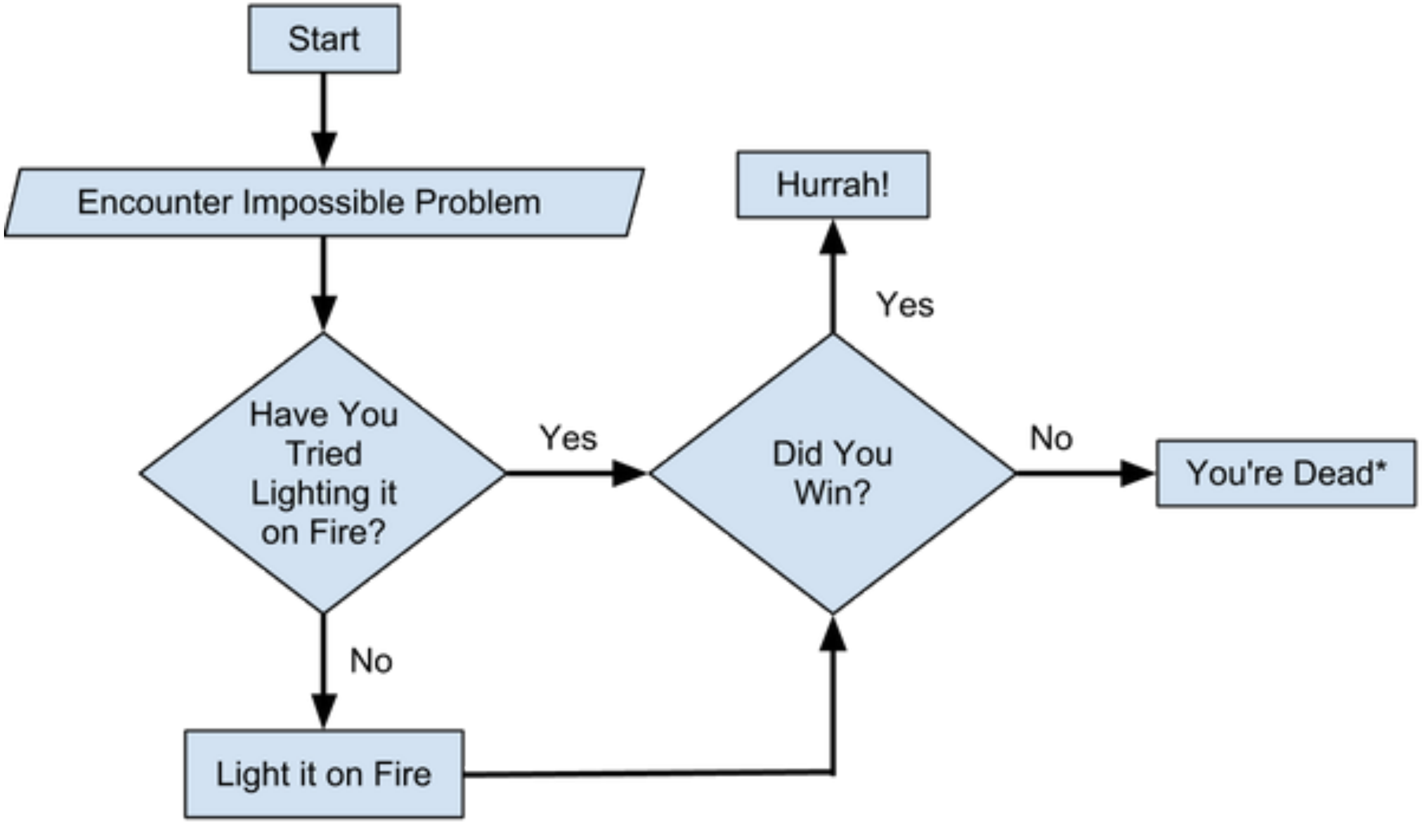

Start
Encounter Impossible Problem
Have You Tried Lighting it on Fire?
Yes
No
Light it on Fire
Did You Win?
Yes
Hurrah!
No
You're Dead*

Image taken from nerdy.quora.com

Alternative Representations

- Decision trees (behavior trees)
  - No machine learning involved, just the output
  - Can also be used in dialog generation

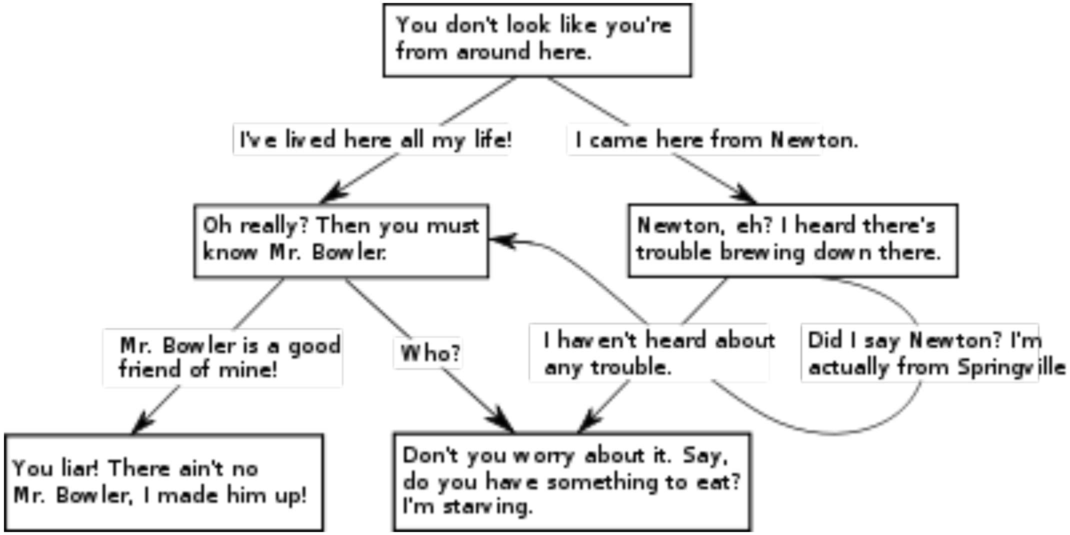

You don't look like you're from around here.
I've lived here all my life!
I came here from Newton.
Oh really? Then you must know Mr. Bowler.
Newton, eh? I heard there's trouble brewing down there.
Mr. Bowler is a good friend of mine!
Who?
I haven't heard about any trouble.
Did I say Newton? I'm actually from Springville
You liar! There ain't no Mr. Bowler, I made him up!
Don't you worry about it. Say, do you have something to eat? I'm starving.

Image taken from wikipedia.org

- Scripting with if-else statements
- Similar idea, different implementation
  - FSMs still most popular in game industry

Next Topic: Search Tree

- One of the first topics taught in typical intro AI course
  - E.g. COSC 322 applies this in AI project

- Used for modeling game state and searching for best strategy

- System strategy is not always the same – the best strategy depends on the player's current actions
  - Enhances personalized game experience (in contrast to the same behaviour given by FSMs)

Administration

- Next class:
  - Search trees

- TA office hours:
  - This Thursday 3:30pm
  - Fix bugs from A2
  - Incorporate feedback from A2
  - Implement event logging in A3 as prep for next week

- Next week:
  - Run heuristic evaluation with peers in class
  - Have your computers and questionnaires setup and ready to go