Decision Making: Behavior Trees

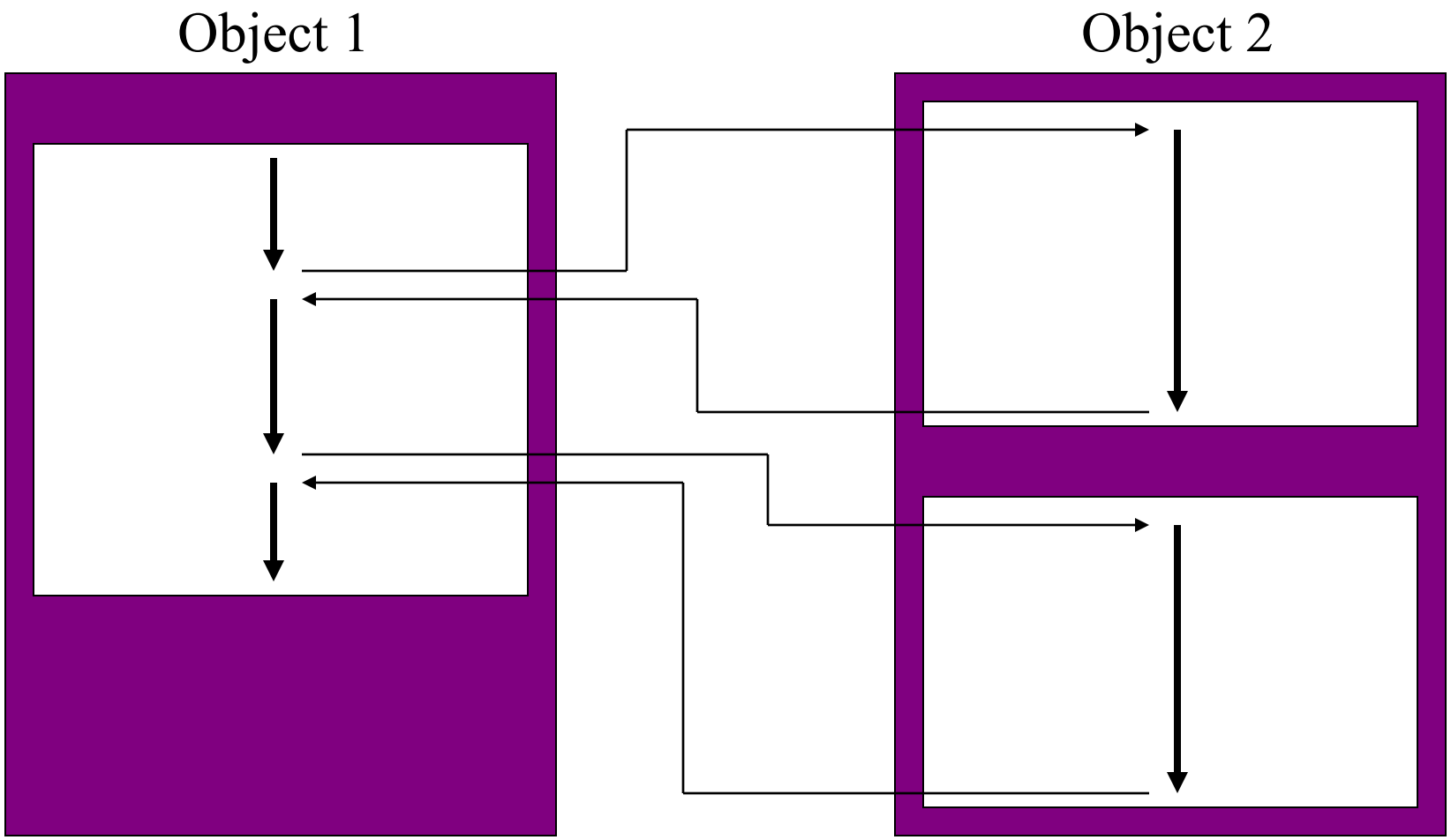
Matthew Guzdial

guzdial@ualberta.ca

Announcements

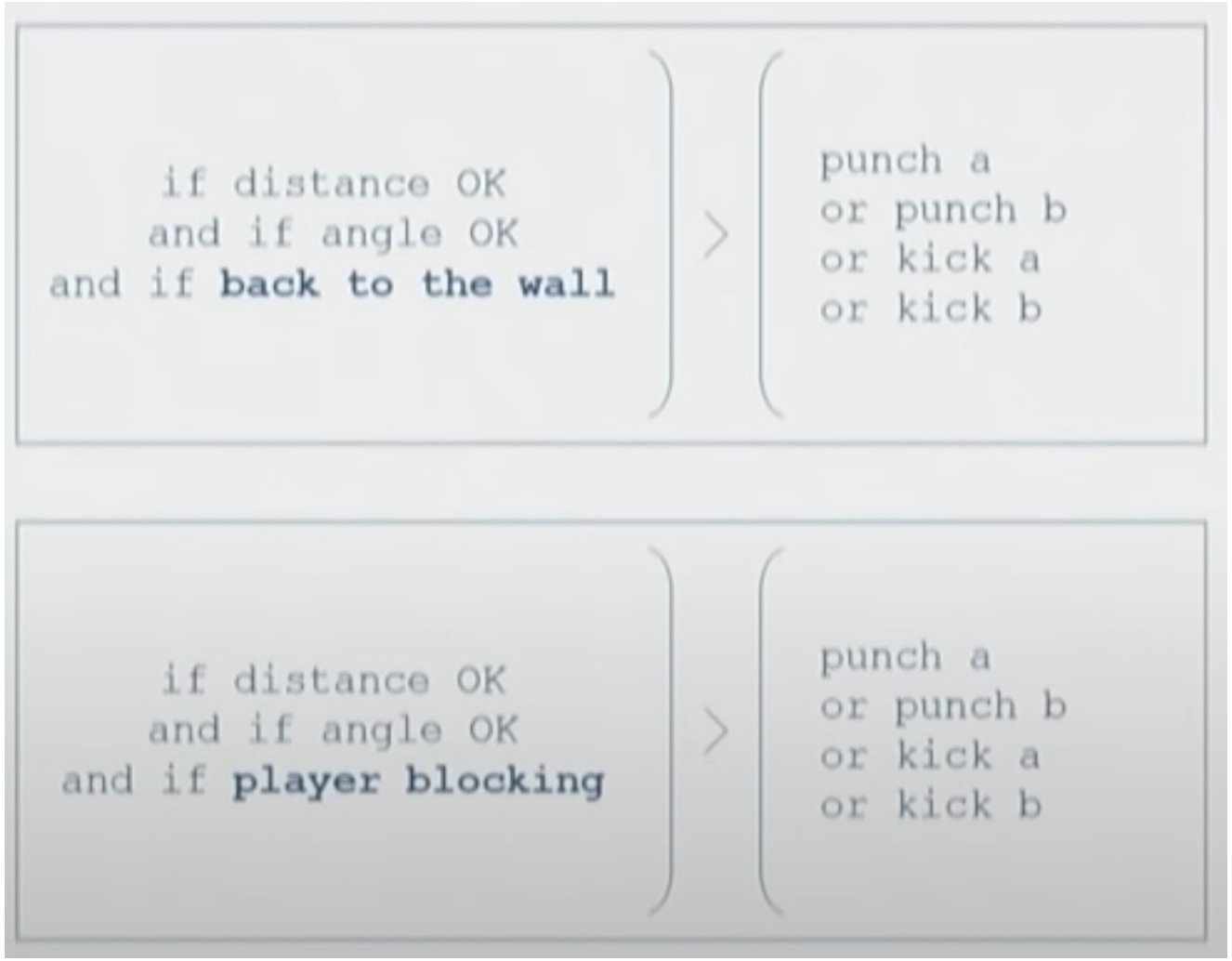
- HW2 was just due (75/87 submitted), HW3 released Monday, due in three weeks from Monday (Oct 25th)
- Quiz 2 on Friday (more on this at the end of the class)
- No class Monday (Thanksgiving)

Decision Making: What do I do for *this* update?



A single game loop calls Update() on all objects

Rules

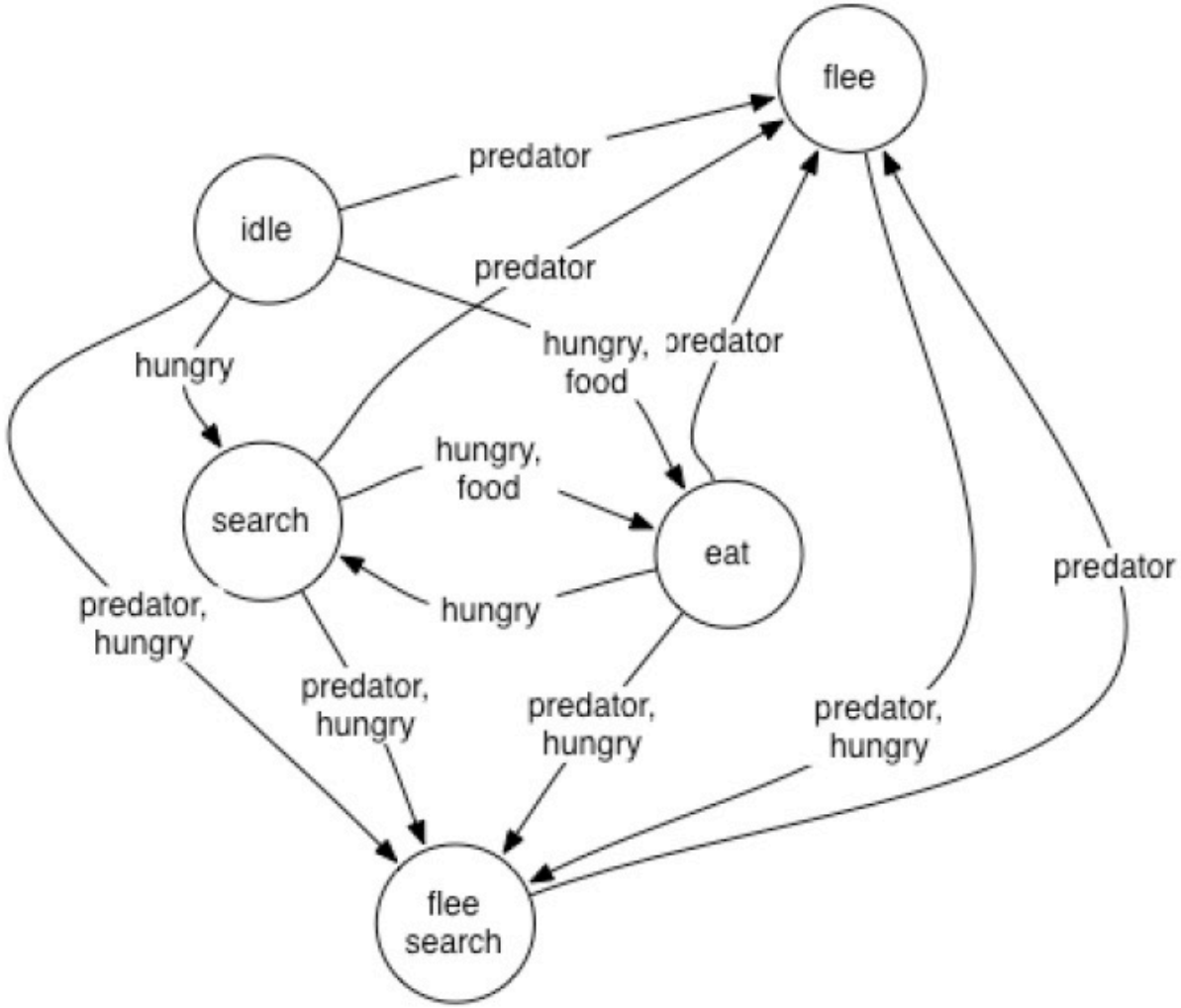


https://youtu.be/VoXSJBVqdek

Rules Usage

?

Finite State Machine



* Usually animations are linked to states, transitions, or both.

FSM Usage

http://www.gameaipro.com/GameAIPro2/GameAIPro2_Chapter33_Infected_AI_in_The_Last_of_Us.pdf

https://www.pcgamer.com/cdpr-calls-cyberpunk-2077s-current-npc-and-ai-problems-bugs/

?

Recommended Viewing

Game Maker's Toolkit:
https://www.youtube.com/watch?v=9bbhJi0NBkk&t=0s

Summary: People often assume "best" game AI means most difficult/challenging, which is almost never the case.

What makes good game AI?

- “Cheat” for the benefit of the player
- Predictable
- Transparent

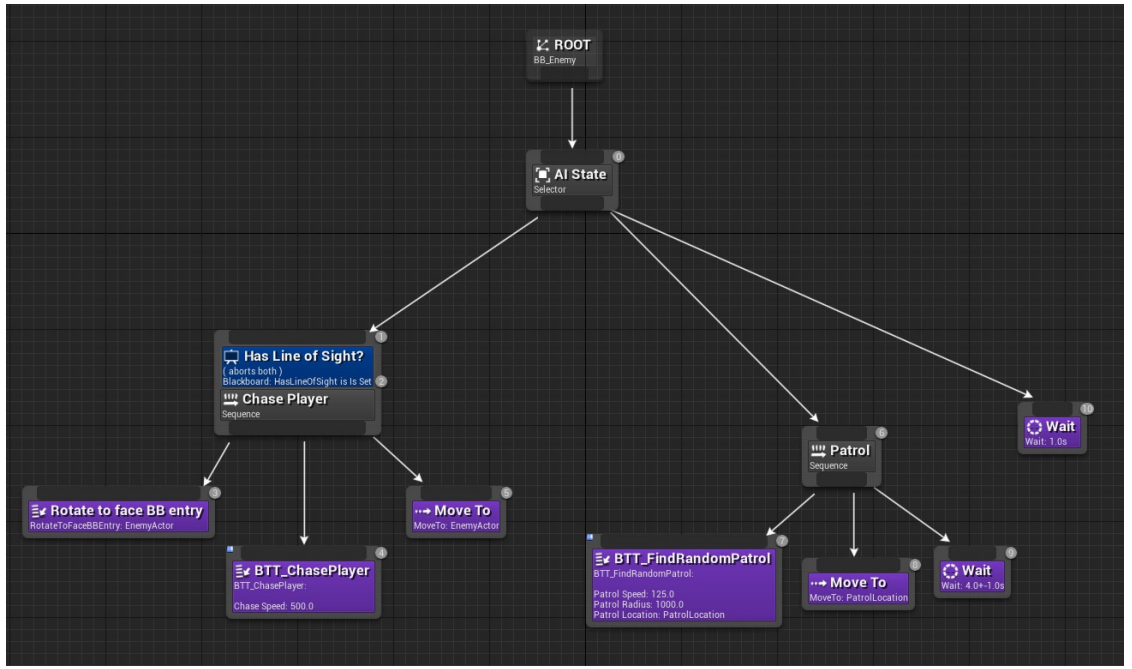
- Ultimately: Good Game AI supports the intended experience of a player.

Rules and FSMs

- Rules: Great for emergent behaviour, bad for control
- FSMs: Great for designer control, but brittle.

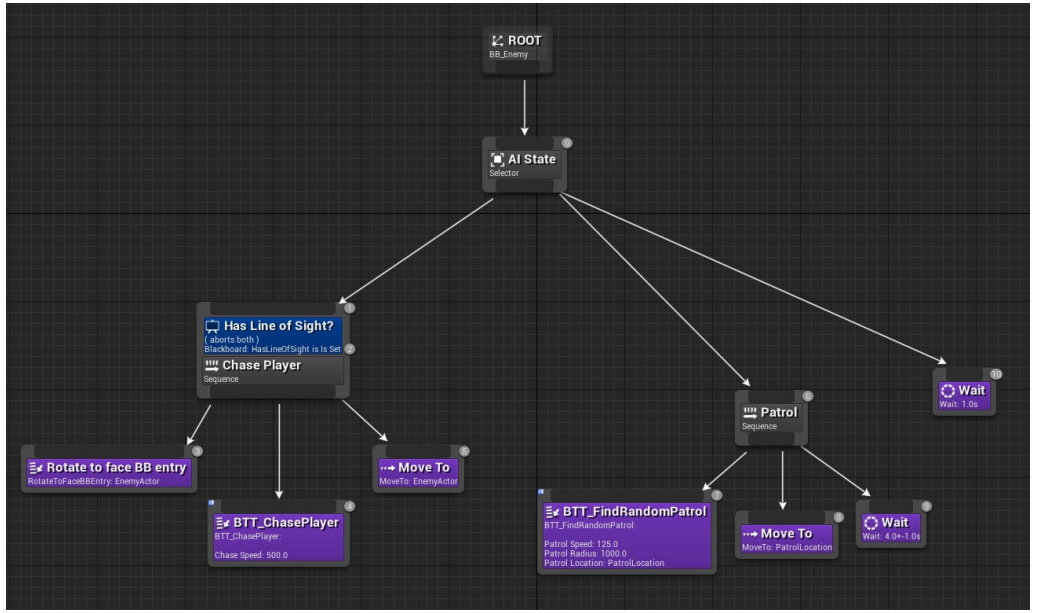
Behavior Trees

- Rather than a general graph, use a tree.
- Every iteration start at the root (“default state”)
- Walk to a leaf node (childless node), equivalent to states.
- Lots of cool new nodes that aren’t states to structure behaviours!



One node fires per tick

- Each node keeps track of current child to execute.
- At start of tick, walk the tree to find our current node.
  - If same as previous current node, continue to saved current child
- Nodes return True, False, or None



Behavior Tree Usages

https://docs.cryengine.com/display/CEPROG/Coordinating+Agents+with+Behavior+Trees

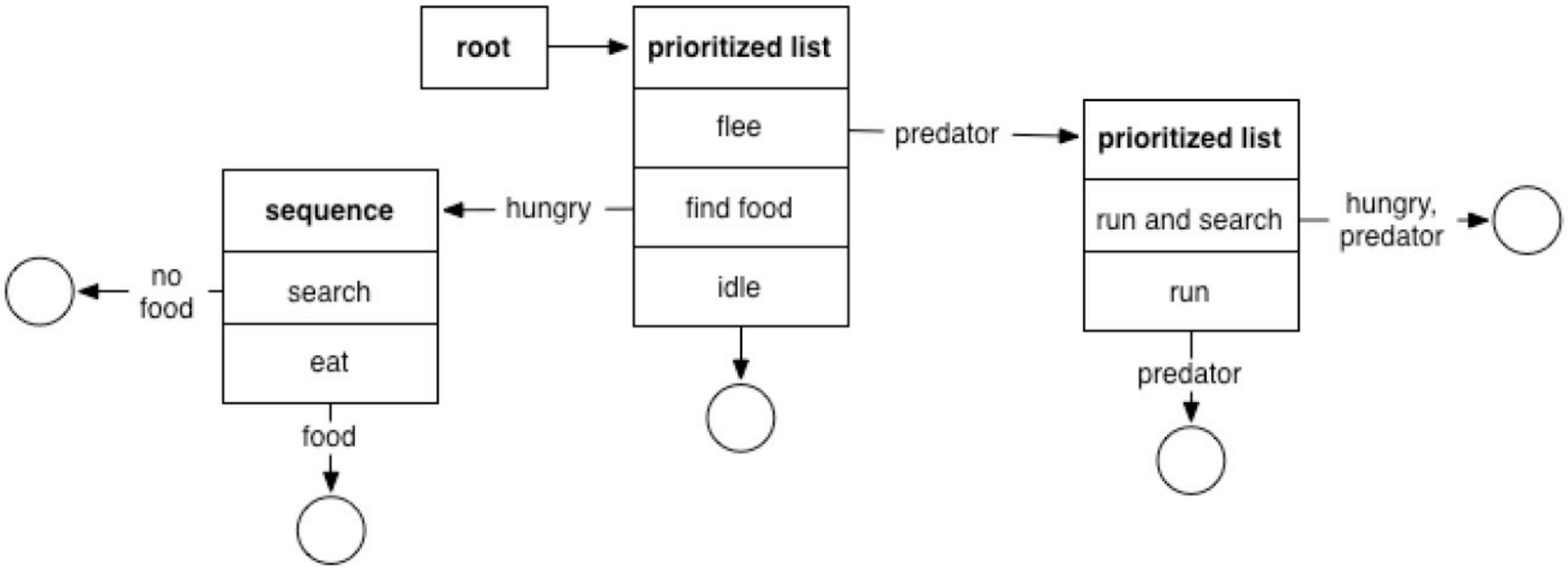
https://youtu.be/Qq_xX1JCreI

http://www.gameaipro.com/GameAIPro/GameAIPro_Chapter10_Building_Utility_Decisions_into_Your_Existing_Behavior_Tree.pdf

https://youtu.be/NU717sd8oUc

?

Behavior Trees: Example



Types of Nodes (Not Exhaustive)

- Actions: do something in the world (leaves)
- Selectors: make a decision based on world condition
- Prioritized list: Success if any child succeeds in order
- Sequence: Failure if any child fails in order
- Probabilistic: Choose probabilistically from set
- One-off (random or prioritized): Pick a single child randomly or with some priority

```
Class Action extends Node
{
    void run ()
    {
        if (execution conditions not met) do {
            return False
        }
        // Do whatever you need to do

        return True or False or None (if not done)
    }
}
```

```
Class PriorityList extends Node
{
    children = []
    currChild = 0

    void run ()
    {
        if currChild<children.length do {
            result = children[currChild].run()
            if result == True do {
                return True
            }
            else if result ==False do{
                currChild +=1
                return None
            }
            return None
        }
         return False
    }
}
```

```
Class Sequence extends Node
{
	children = []
	currChild = 0

	void run ()
	{
		result = children[currChild].run()
		if result== False do {
			return False
		}
		else if result==None do{
			return None
		}
		currChild +=1
		if (currChild==children.length){
			return True
		}
		return None
	}
}
```

Advanced hacks

- Interrupt daemons: jump from a node to an entirely different section of the tree based on external conditions changing

- Shortcuts: jump from within one child node to another directly

Advantages

- Appearance of goal-driven behavior
- Multi-step behavior
- Relatively Fast
  - Though slower on average than FSMs
- Recover from errors

Disadvantages

- Isn’t really thinking ahead about unique situations
- Only as good as the designer makes it (just follows the recipes)

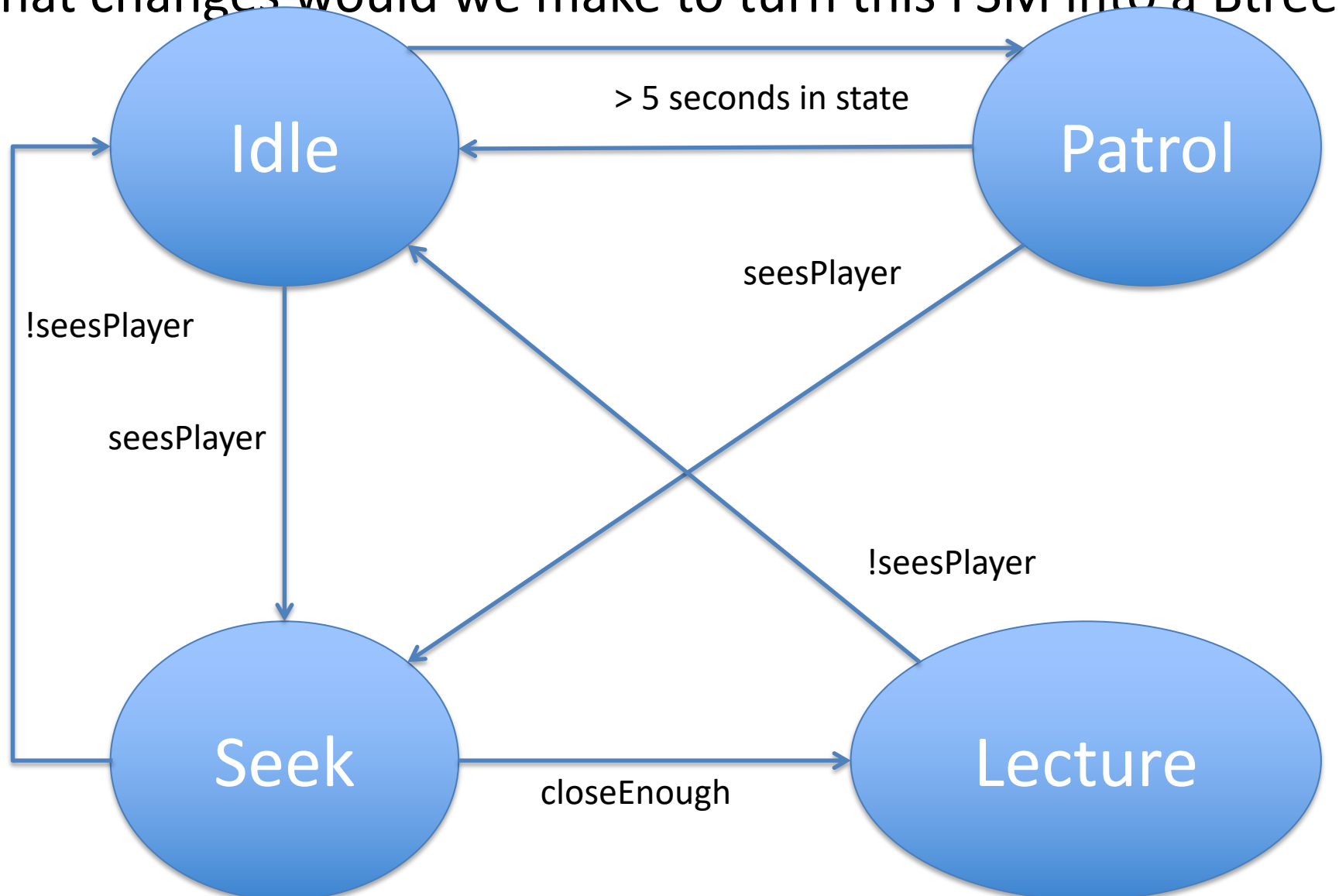
IMPORTANT NOTE

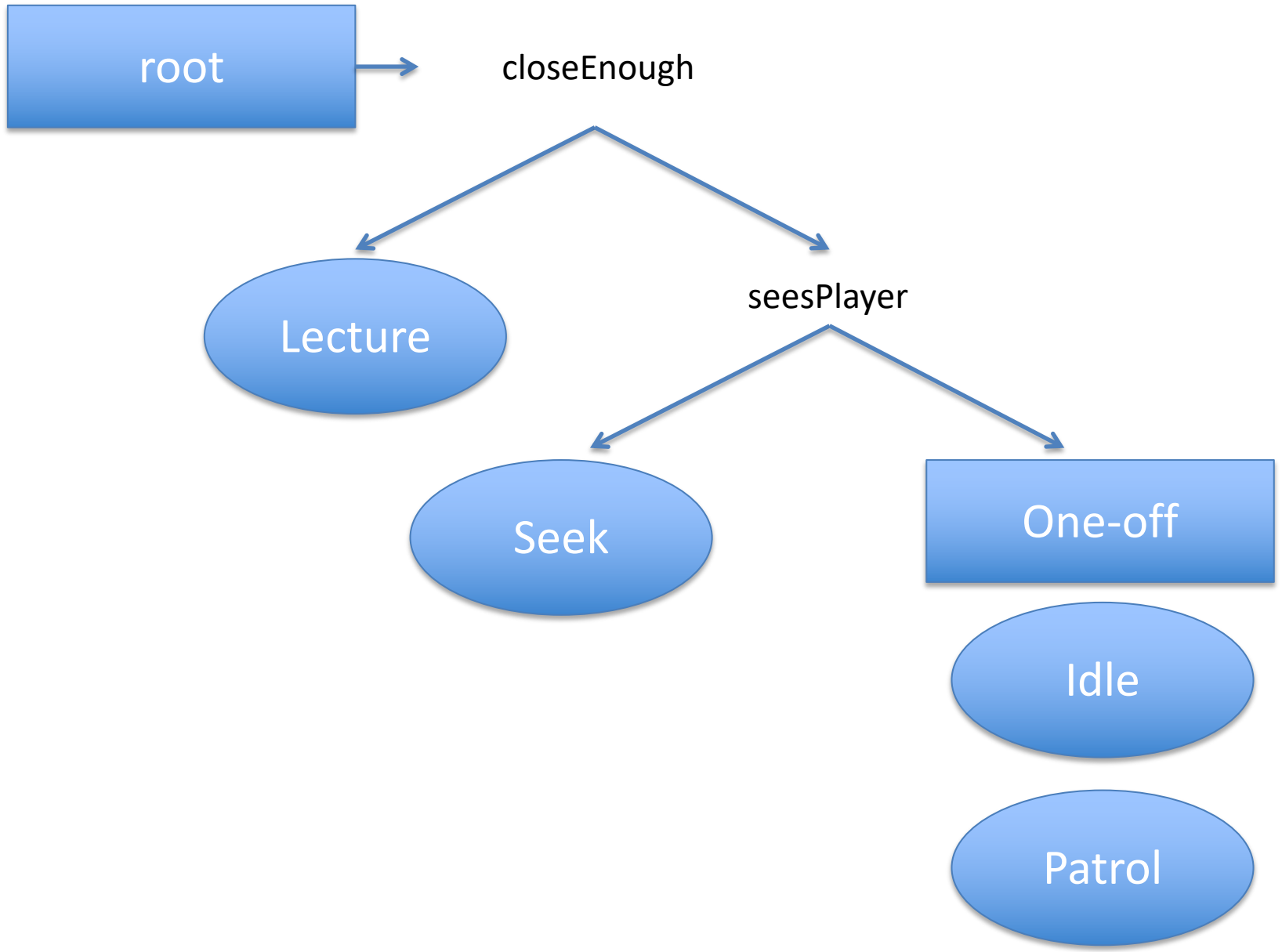
- Any desired set of behaviors that can be represented in a Behavior Tree can be represented in an FSM and vice versa.

- Differences:
  - **Sequences of Actions**: FSMs require extra variable tracking to handle sequences of states
  - **Error Recovery**: FSMs require many more linkages to do what Btrees do naturally

PQ1: https://forms.gle/wow5Cf1H54nLcCGj6
https://tinyurl.com/guz-pq15a

What changes would we make to turn this FSM into a Btree?




root
closeEnough
Lecture
seesPlayer
Seek
One-off
Idle
Patrol

When to use FSMs and when to use Btrees?

FSMs

- Simpler behavior.
- Distinct, unique behaviors.
- Easier for designers.
- Speedier (slightly)

Btrees

- Complex behavior.
- “Sequential” behaviors or behaviors that are reliant on one another.
- Industry Saturation

More Decision Making Options!

Example:
Combine FSMs and Btreess!

and more…

http://www.gameaipro.com/GameAIPro3/GameAIPro3_Chapter11_A_Character_Decision-Making_System_for_FINAL_FANTASY_XV_by_Combining_Behavior_Trees_and_State_Machines.pdf

Resources

- Behavior Trees from a development perspective in Horizon Zero Dawn https://youtu.be/Qq_xX1JCreI
- Behavior Trees from an AI perspective in Alien: Isolation (and more) https://youtu.be/6VBCXvfNlCM
  Wikipedia:
  https://en.wikipedia.org/wiki/Behavior_tree_(artificial_intelligence,_robotics_and_control)
- Other tree-like ways to model behaviour in complex systems:
  https://en.wikipedia.org/wiki/Activity_diagram
  https://en.wikipedia.org/wiki/Sequential_function_chart

Quiz 2 Review

Topics (be able to run): APSP, MCTS, Trees, Rule Systems, FSMs

Topics (concepts): Btrees, Graphs (spatial representations), Greedy and A* path planning