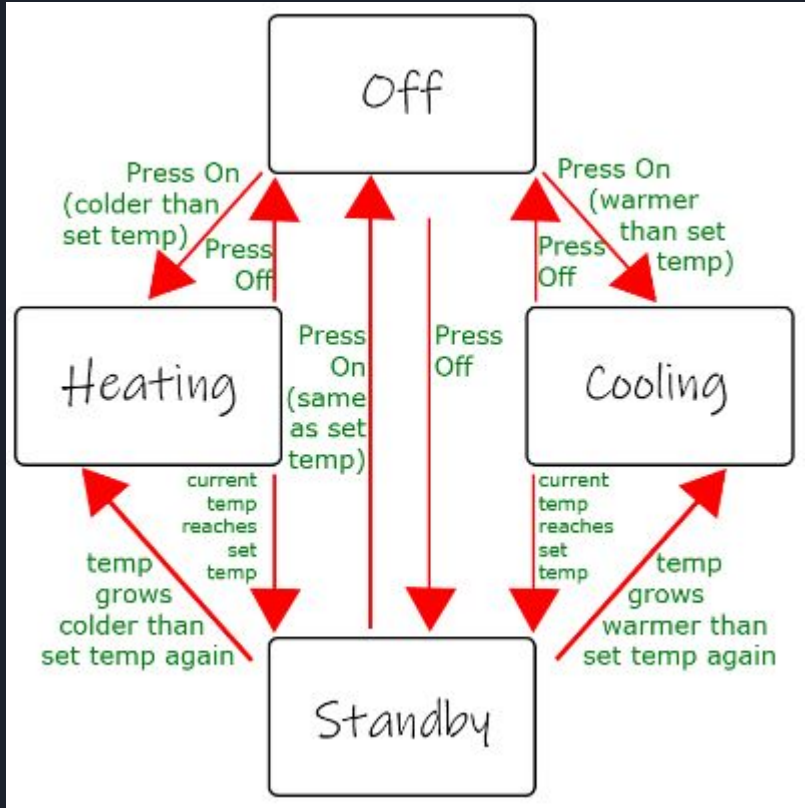
Decision-Making and Logic

Enabling Complex Behaviors

- As games, and player options, get more complicated, creating challenging enemy behavior can be more demanding
- If an enemy does one thing over and over, it is easy for players to identify the patterns used, and possibly an exploit that allows them to trivialize the challenge of the enemy
- Solution: Equip enemy agents with different behaviors for different identified scenarios
    - Allows for resolution of various scenarios in a variety of ways
    - Requires each behavior to be implemented in full
    - Also requires the logic for deciding/transitioning between these scenarios

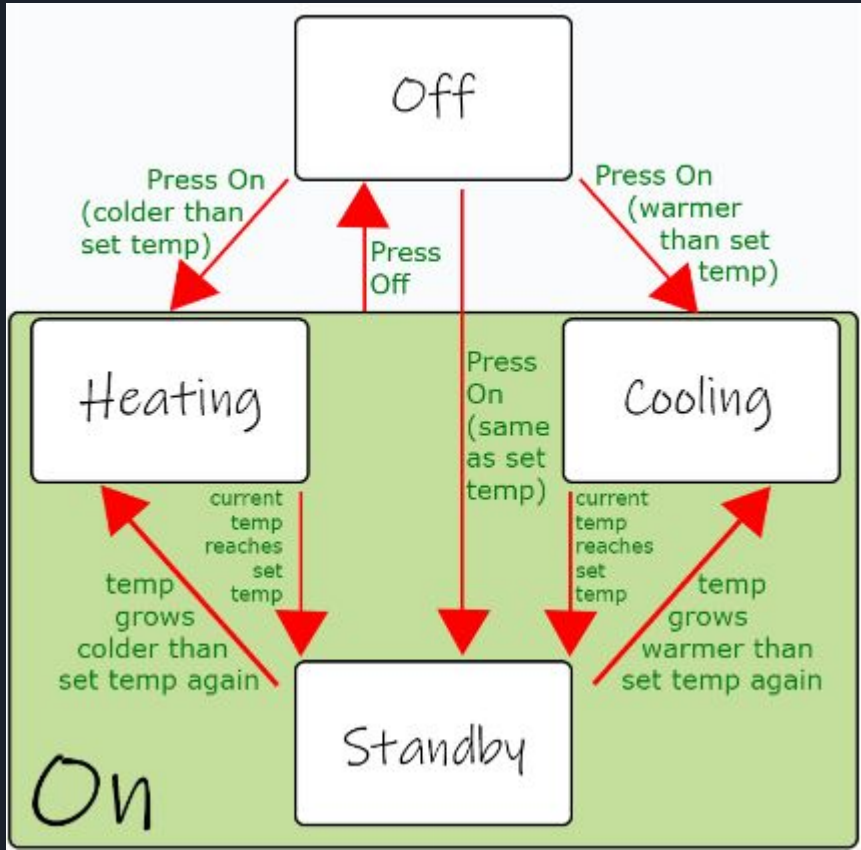
Finite State Machines

- States - Possible configurations something can exist in. Can influence current behavior, values held by variables, and states available to transition into
- Transitions - Allow for the object to move from one state to another under certain conditions
- Events - The “certain conditions” that will trigger a change in state
- Generally described by examining state, input, and output
- Some machines focus on doing the required work upon transition, while others focus on continuously doing the required work the whole time that state is held



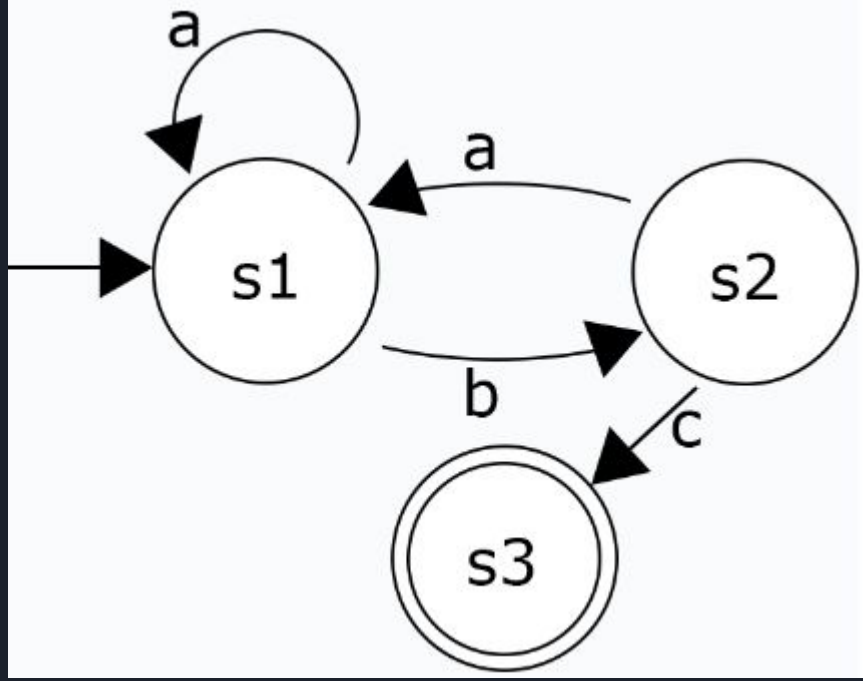
Finite State Machines

- Further FSM Elaborations:
  - Hierarchy allows us to group multiple states together (imagine embedding a smaller FSM into an existing FSM)
  - Parallelism allows multiple state machines to work in parallel while being displayed together (to aid comprehension)
  - Broadcasting enables communication between each other - ie state machine for thermostat and fans, uses thermostat to broadcast temperature so fans know when to engage or enter standby



Finite State Machines

- Given example starts in state s1
- s1 transitions back to itself when an input of a is received
- When a b is input to s1, we transition to state s2
- While in s2, we can accept another a, taking us to s1, or we can accept c, taking us to s3
- s3 here is indicated as an acceptance state; if we were validating input, we would indicate a value of “true” here



Satisfying Strings Include:

- abc
- aaaaaaaaaaaaabc
- abaaaababaaabc
- abababababc
- abaabaaabaaaabc

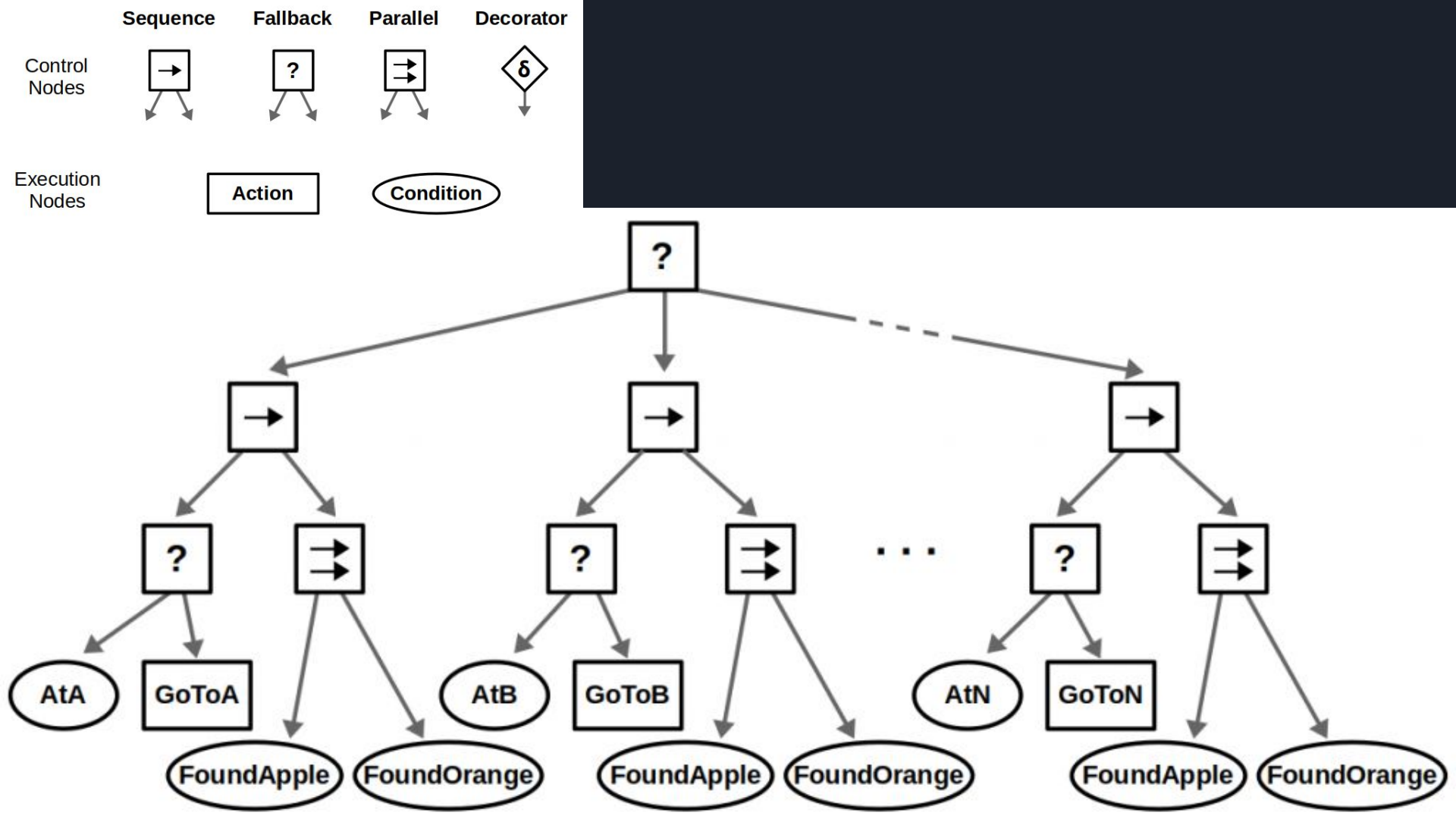

AI 101
FINITE STATE MACHINES
BRINGING AI TO (HALF) LIFE

Finite State Machines

- State machines are great for selecting between a known number of deterministic behaviors
- As there are more states with more transitions, more logic must be written to accommodate the logic to get from one given state to another
- If these states are not entirely independent, then the amount of context that the transitions must maintain and update can get out of hand
- Solution: Behavior trees!
    - Behavior Trees are more modular, but may be subject to weakness when circumstances are altered drastically.
    - Tree format allows for complicated sequences of behavior to be represented with modular nodes that are evaluated individually

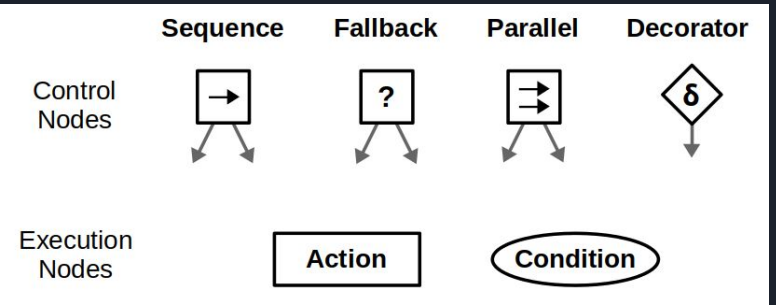
Behavior Trees

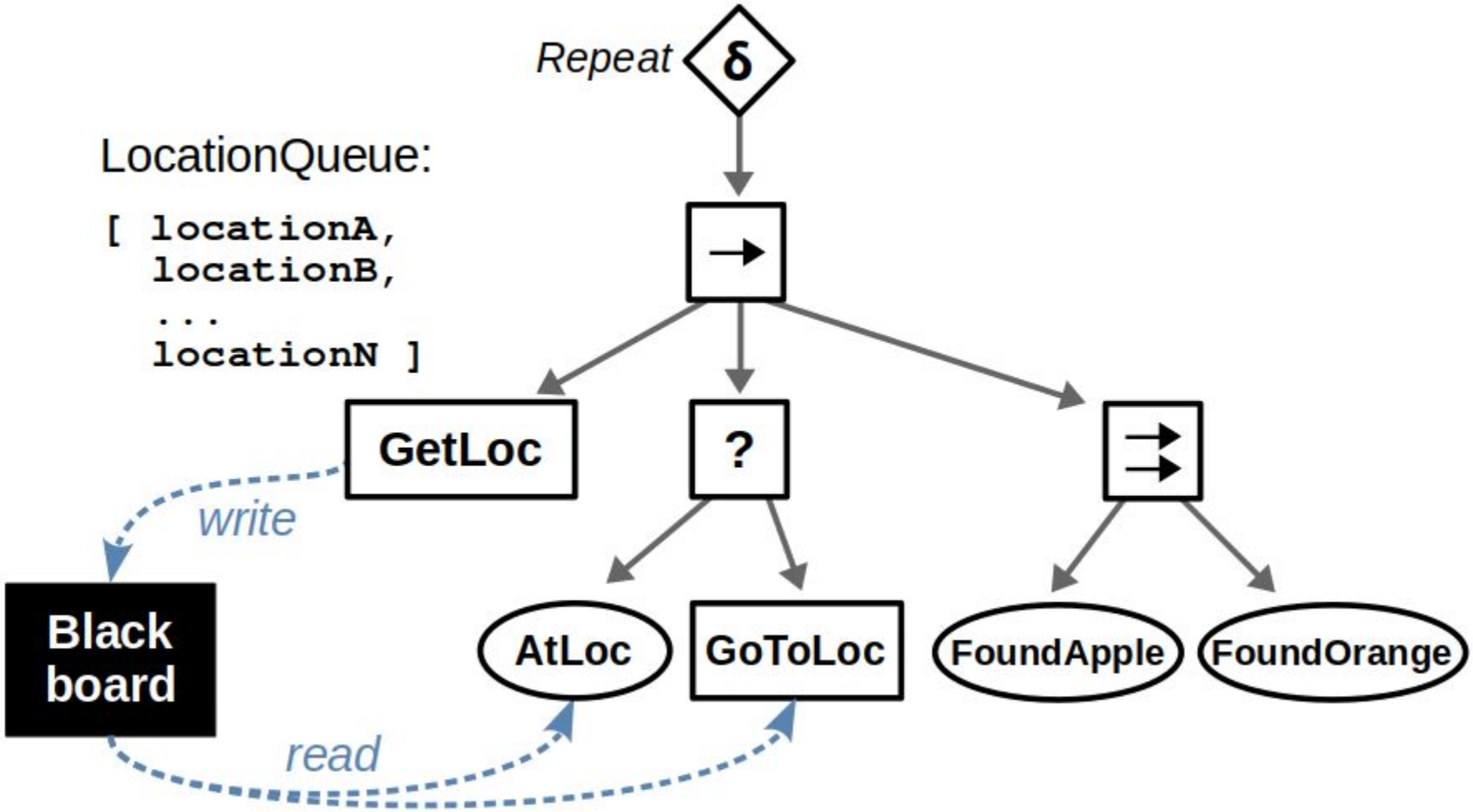
- Behavior trees are trees (acyclic directed graphs with only parent-child relationships)
- Leaf nodes are executable behaviors, from performing a check or performing a complicated and/or lengthy action
  - Leaf nodes return (Successful/Failure/Running)
- Internal nodes control traversal and logic, and return based on the status of their children
  - Sequence Nodes execute children in order until one child returns Failure or all children return Success
  - Fallback Nodes execute children in order until one of them returns Success or all children return Failure
  - Parallel nodes execute all of their children “in parallel” (each tick), returning success if some of their children do, or failure if all of their children return Failure
- Makes use of “Blackboard” to read and write variables for later reference (ie player last seen, speed, detection radius, etc

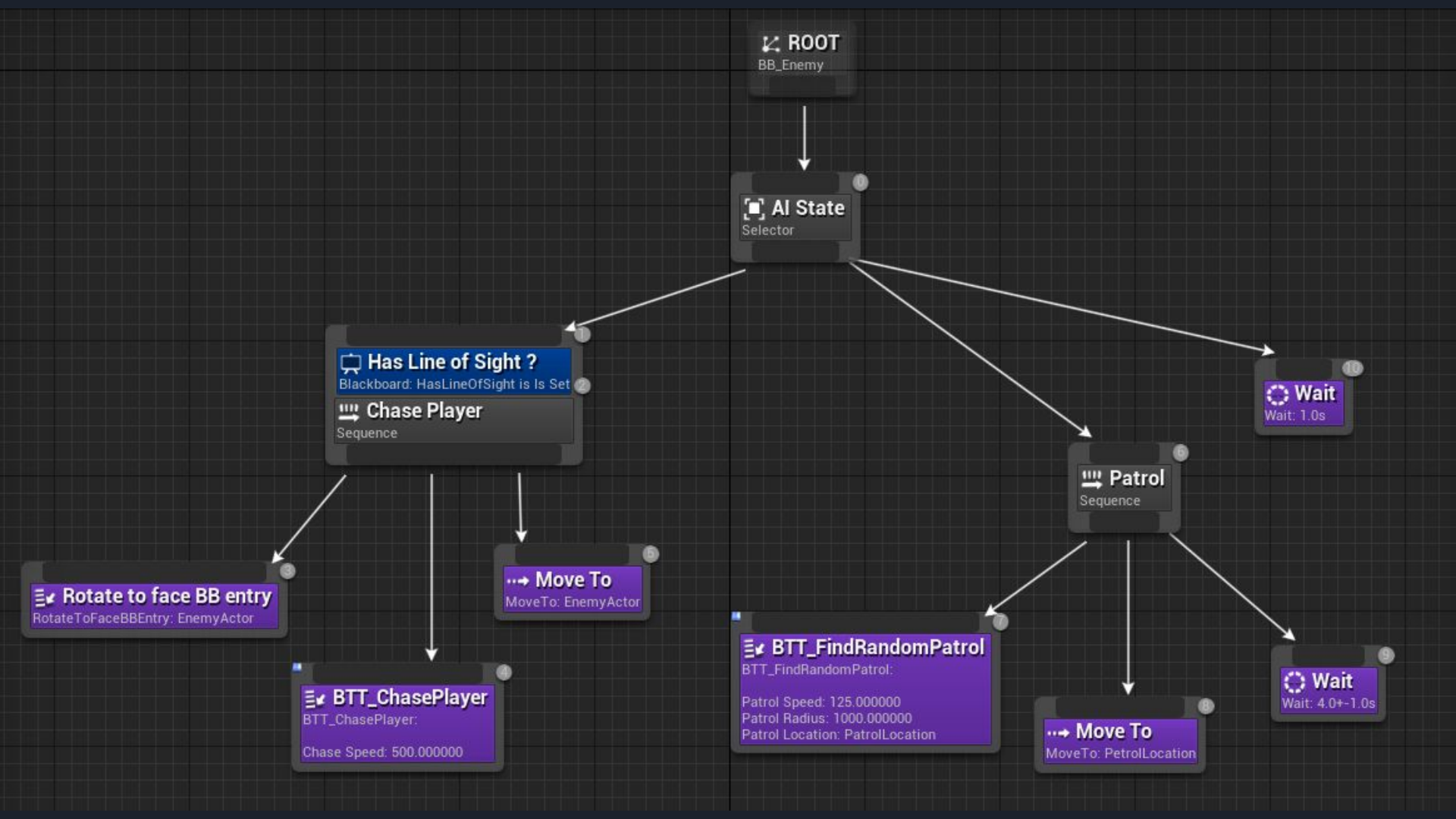

Sequence
Fallback
Parallel
Decorator
Control Nodes
?
δ
Execution Nodes
Action
Condition
?
?
?
?
?
AtA
GoToA
AtB
GoToB
AtN
GoToN
FoundApple
FoundOrange
FoundApple
FoundOrange
FoundApple
FoundOrange

Behavior Trees

- Decorator nodes modify children nodes for desired outcomes/behaviors
  - Common policies include: Invert, Repeat/Retry, Timeout, Force Failure, Success Is Failure, etc
- BTs receive ticks at regular intervals
  - In more primitive Behavior Trees, we reprocess the tree fully upon each tick, utilizing the return values to navigate to the appropriate spot
  - To optimize our application, we may alter this to only re-evaluate the tree when an event occurs (ie a leaf node succeeding/failing)
    - This may optimize resources and performance, but this can make our object behave strangely when a condition changes but is not being re-evaluated
      - The explicit definition of actions when transitioning from state to state are a strong point of state machines (more control at the cost of reusability and modularity)
      - Remedied within UE5 by “Observer Reports”, where nodes in our trees can check conditions so a task can abort if the conditions it requires become false




Repeat
δ
LocationQueue:
[locationA,
locationB,
...
locationN]
GetLoc
?
write
Black board
AtLoc
GoToLoc
FoundApple
FoundOrange
read


ROOT
BB_Enemy
AI State
Selector
Has Line of Sight ?
Blackboard: HasLineOfSight is Is Set
Chase Player
Sequence
Wait
Wait: 1.0s
Patrol
Sequence
Rotate to face BB entry
RotateToFaceBBEntry: EnemyActor
Move To
MoveTo: EnemyActor
BTT_FindRandomPatrol
BTT_FindRandomPatrol:
Patrol Speed: 125.000000
Patrol Radius: 1000.000000
Patrol Location: PatrolLocation
Wait
Wait: 4.0+-1.0s
BTT_ChasePlayer
BTT_ChasePlayer:
Chase Speed: 500.000000
Move To
MoveTo: PetrolLocation


AI 101
BEHAVIOUR
TREES
THE CORNERSTONE OF
MODERN GAME AI


UNSTRUCTURED
BEHAVIOR TREE
?
STATE MACHINE
GOAP

Parting Thoughts

- Do you have any enemy behaviors in your project?
- Do any enemies have multiple behaviors they conditionally choose from?
  - If so, how many, and how are they currently structured? How would you add in a new behavior?
  - If not, what other behavior would be suitable for the enemy to have access to?

Further Resources

- State Machines
  - Understanding State Machines
  - State Machines - Intro
  - MIT Open CourseWare - Computer Science Ch 4
- Behavior Trees
  - UnrealEngine - Behavior Tree Overview
  - Introduction to Behavior Trees