“A good decision is based on knowledge and not on numbers.” – Plato

“Once you make a decision, the universe conspires to make it happen.” – Ralph Waldo Emerson

“The quality of decision is like the well-timed swoop of a falcon which enables it to strike and destroy its victim.” – Sun Tzu

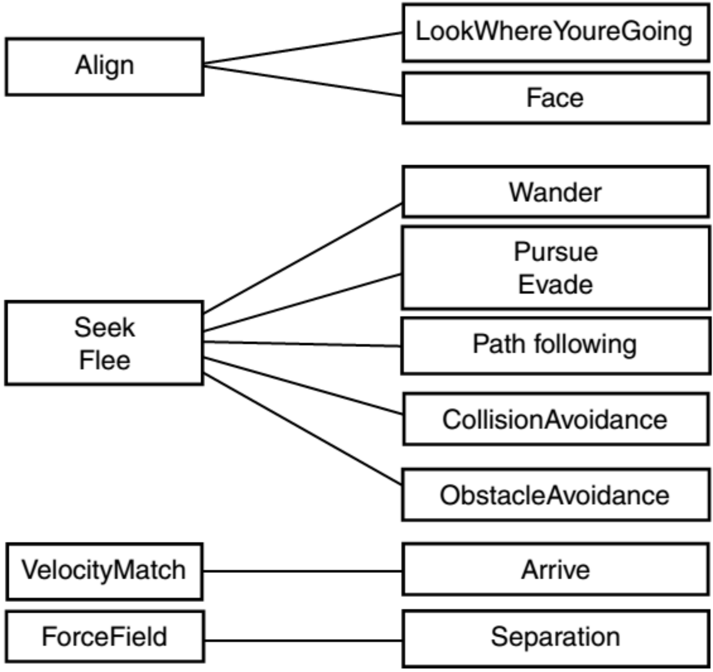
- Wikipedia: “Perfect is the enemy of good”
- “Everything should be as simple as possible, but not simpler.” – Einstein
- Occam (of Razor fame – parsimony, economy, succinctness in logic/problem-solving)
  - “Entities should not be multiplied more than necessary”
  - “Of two competing theories or explanations, all other things being equal, the simpler one is to be preferred.”
- “All that is complex is not useful. All that is useful is simple.” – Mikhail Kalashnikov (of AK-47 fame)

Class N-4

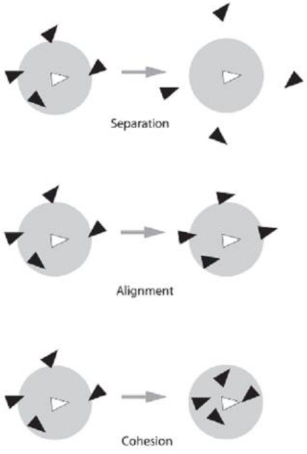
1. When might you precompute paths?
2. This is a single-source, multi-target shortest path algorithm for arbitrary directed graphs with non-negative weights. Question?
3. This is a all-pairs shortest path algorithm.
4. How can a designer allow static paths in a dynamic environment?
5. When will we typically use heuristic search?
6. What is an admissible heuristic?
7. When/Why might we use hierarchical pathing?
8. Does path smoothing work with hierarchical?
9. How might we combat fog-of-war?

Class N-3

1. Steering vs flocking?
2. Steering Family Tree
3. How might we combine behaviors?
4. What three steering mechanisms enable flocking?


Align
LookWhereYoureGoing
Face
Seek
Flee
Wander
Pursue
Evade
Path following
CollisionAvoidance
ObstacleAvoidance
VelocityMatch
Arrive
ForceField
Separation

Millington Fig 3.29



Buckland Fig 3.16

Class N-2

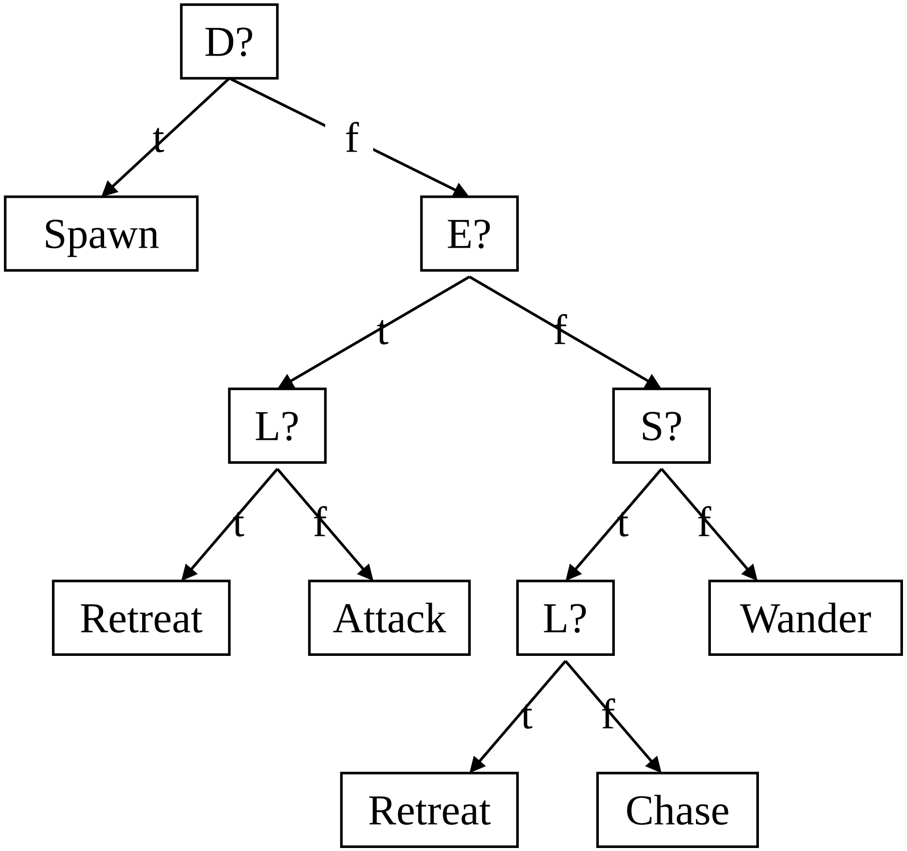
1. How can we describe decision making?
2. What makes FSMs so attractive?
3. What might make us not choose an FSM?
4. Two drawbacks of FSMs, and how to fix?
5. What are the performance dimensions we tend to assess?
6. What are two methods we discussed to learn about changes in the world state?

Class N-1

1. How many outcomes does a d-tree produce?
2. What are advantages of D-Trees?
3. Discuss the effects of tree balance.
4. Must d-trees be a tree?
5. Can d-trees translate into rules? If so how?
6. How can we use d-trees for prediction?
7. What is the notion of overfitting?

Quake D-Tree

- Attributes: E=<t,f> L=<t,f> S=<t,f> D=<t,f>
- Actions: Attack, Retreat, Chase, Spawn, Wander
- Could add additional trees:
  - If I’m attacking, which weapon should I use?
  - If I’m wandering, which way should I go?
  - Much like hierarchical FSMs

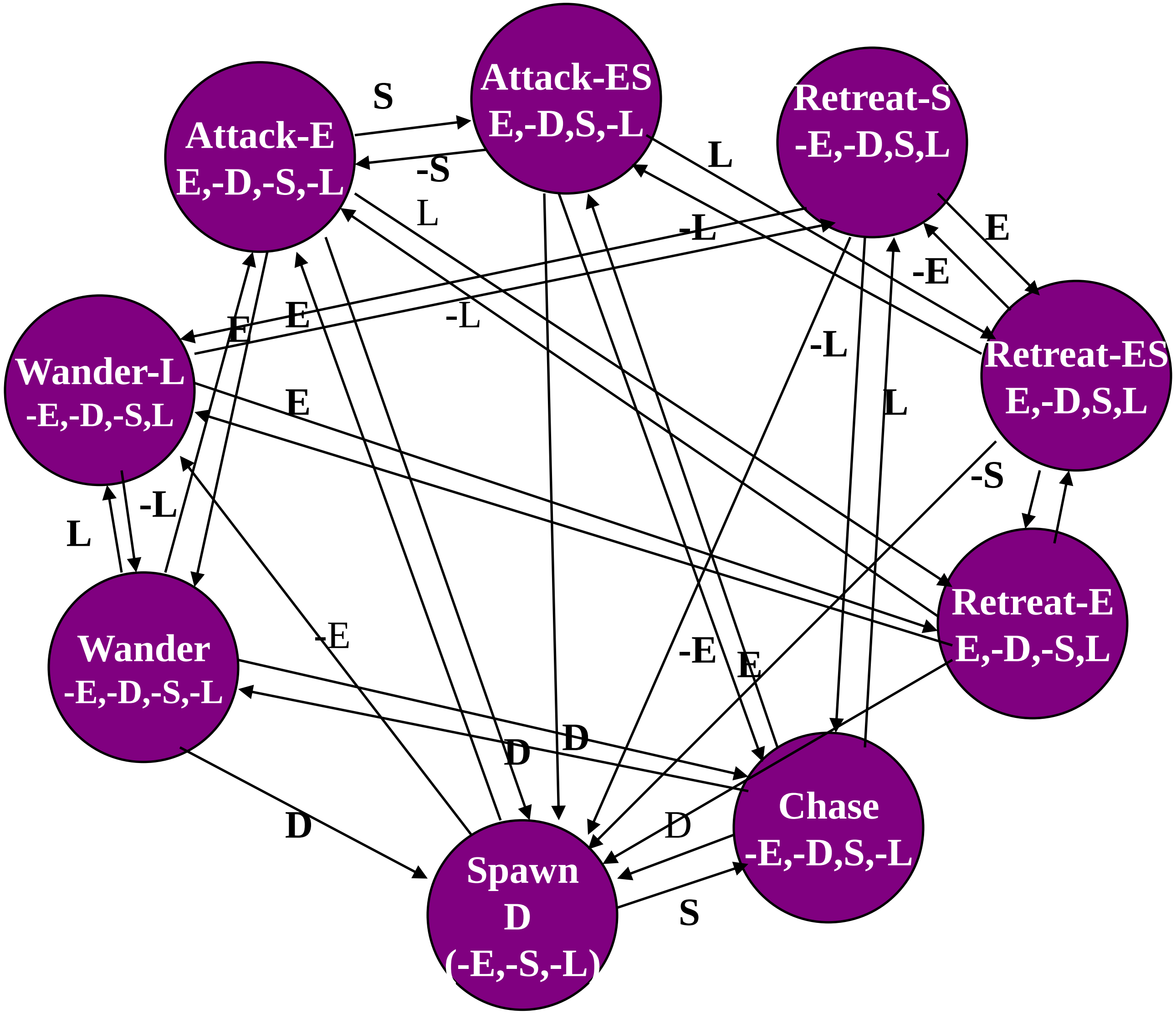


E: Enemy in sight; S: hear a sound;
D: dead; L: Low health

http://research.cs.wisc.edu/graphics/Courses/638-f2001/lectures/cs638-17.ppt.

http://web.eecs.umich.edu/~sugih/courses/eecs494/fall06/lectures/lecture13-gameai.pdf

Quake FSM



OOB

- Decision Making: f(knowledge) → action
  - N+2: Planning
  - N+1: Rule-based Agents, Fuzzy, Markov
  - N: Decision & Behavior Trees (M Ch5.2, 5.4)
  - N-1: Decision & Behavior Trees (M Ch5.2, 5.4)
  - N-2: FSMs
  - N-3: Steering
  - N-4: Graphs, Search, and Movement

Decision Making: Trees

2016-06-09

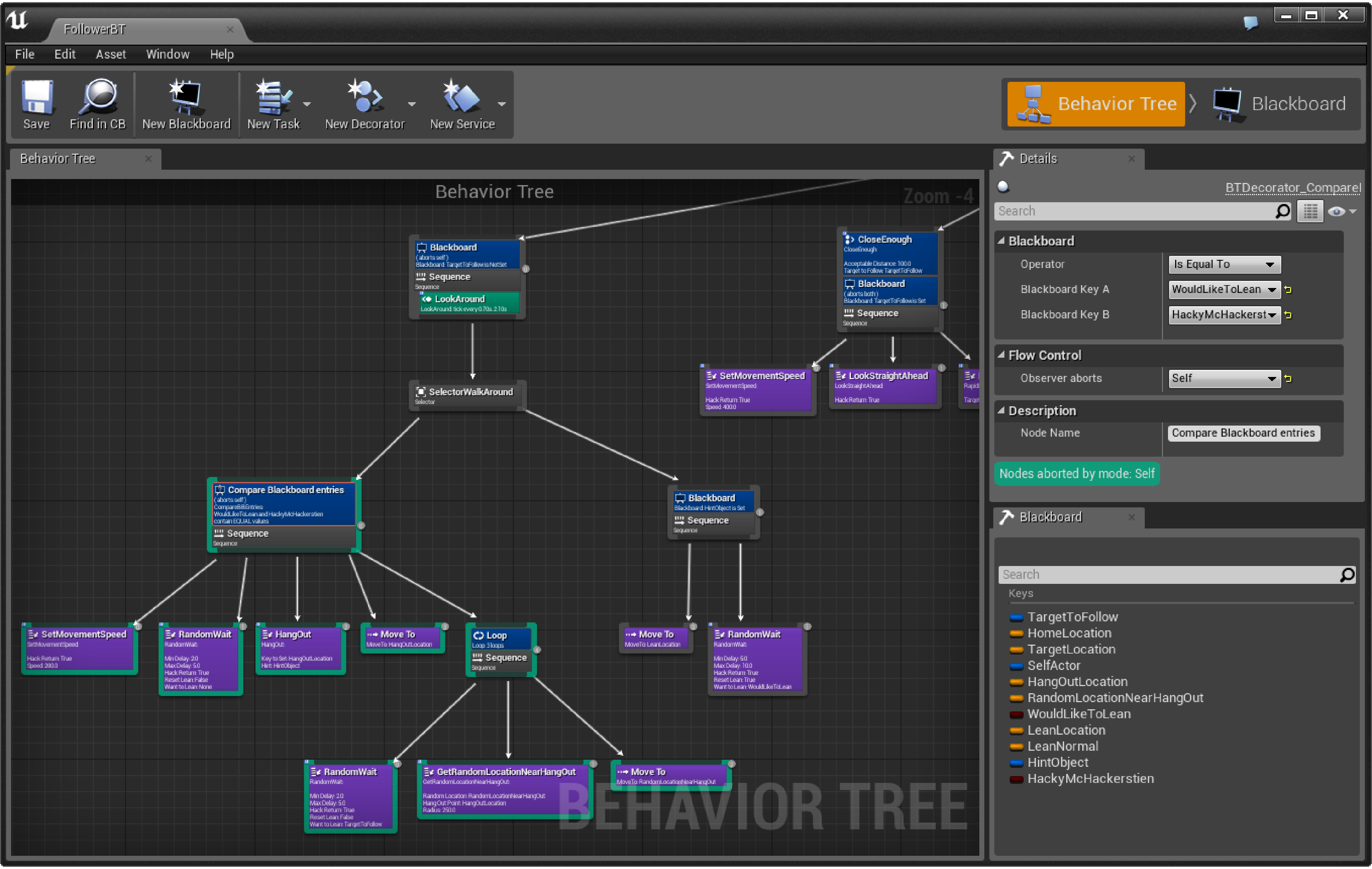
BEHAVIOR TREES (M CH. 5.4)

Behavior Trees

- Very popular/ubiquitous (Bungie’s Halo 2 – 2004)
- Synthesis of: HFSM, Scheduling, Planning
- Easy to understand
- Easy for non-programmers to create
- Aren’t good in all instances… (stay tuned)
- Instead of *state*, employ *tasks*
- Composable, self contained

Behavior Trees

- Simple reactive planning
- Tree of behaviors specify what an agent should do under all circumstances (manually provided)
- Actions: do something in the world (leaves)
- Selectors: make a decision
  - Prioritized list
  - Sequence
  - Sequential-looping
  - Probabilistic
  - One-off (random or prioritized)



Blackboard

Operator	Is Equal To
Blackboard Key A	WouldLikeToLean
Blackboard Key B	HackyMcHackerst

Flow Control

Observer aborts	Self

Description

Node Name	Compare Blackboard entries

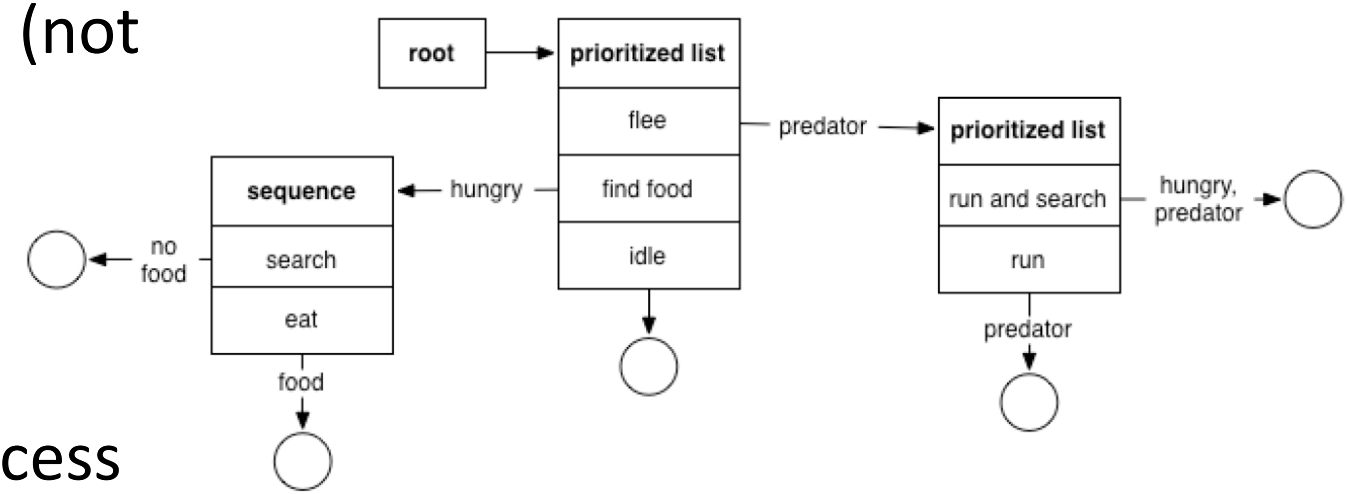
Nodes aborted by mode: Self

Keys
- TargetToFollow
- HomeLocation
- TargetLocation
- SelfActor
- HangOutLocation
- RandomLocationNearHangOut
- WouldLikeToLean
- LeanLocation
- LeanNormal
- HintObject
- HackyMcHackerstien

https://docs.unrealengine.com/latest/INT/Engine/AI/BehaviorTrees/index.html

Behavior Tree Structure

- Behavior tree made of connected tasks (not states!)
  - Conditions
  - Actions
  - Composites
- Tasks return success or failure
- Decomposition allows flexibility & easy GUI integration



```
Class Action extends Node
{
        children = []

        void run ()
        {
                if (execution conditions not met) do {
                        return False
                }
                // Do whatever you need to do
                return True or False
        }
}
```

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

        void run ()
        {
                if (execution conditions not met) do {
                        return False
                }
                for child in children do {
                        if child.run() == True do {
                                return True
                        }
                }
                return False
        }
}
```

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

    void run ()
    {
        if (execution conditions not met) do {
            return False
        }
        for child in children do {
            if child.run() == False do {
                return False
            }
        }
        return True
    }
}
```

Node Types

- Conditions
- Actions
- Composites

Node Types

- Conditions
  - Test for some game property (e.g. proximity of player to NPC)
  - Each implemented as a task
- Actions
- Composites

Node Types

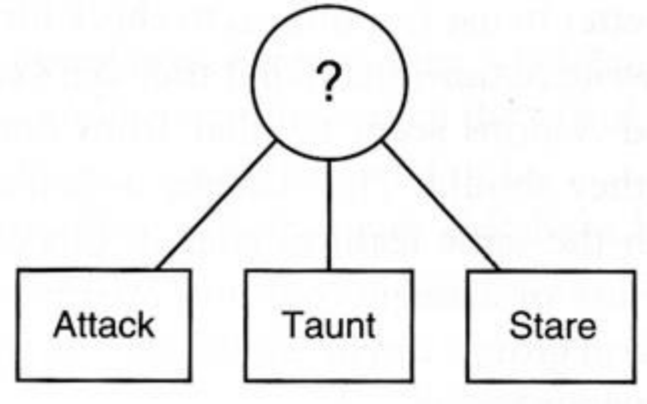
- Conditions
- Actions
  - Alter game state
    - (e.g. play animation, change character internal state, run AI code, play audio sample, etc.)
  - Each is a task
- Composites

Node Types

- Conditions
- Actions
- Composites
  - Differentiates BTs from decision trees
  - Allows for the combination of tasks without concern for what else is in the tree

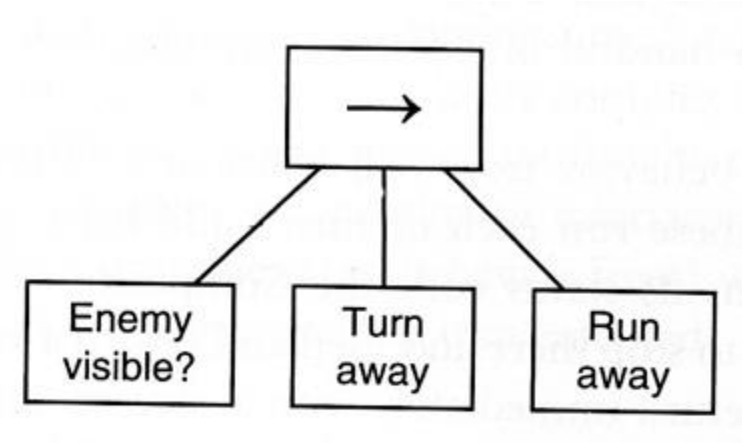
Composite Nodes: Selector

- Selector
  - – Run child tasks until one of them succeeds
  - – Return failure if all tasks fails



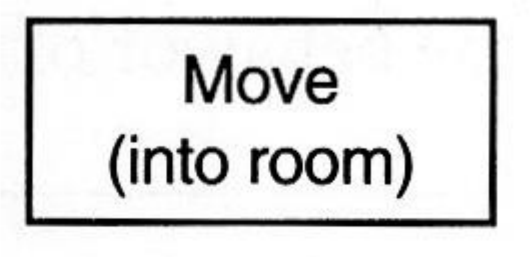
Composite Nodes: Sequence

- Selector
- Sequence
  - Series of tasks that all must succeed



Example

Enter room where player is standing.
Player may close the door.



Move
(into room)