CS425 Game Programming

Game AI

Announcements

- https://www.creativeheads.net/
- http://jobs.gamasutra.com/

So Far…

- Character animation
- Level loading
- Collision detection
- Physics
- Flocks and swarms and Crowd
- Pathfinding
- Now…

Artificial Intelligence

Game AI

- Do we need AI for all games?
  - What do we need Game AI for?
- What GAME AI (or general AI) techniques/technologies do you know about?
- Is the AI research by academics the same as AI used in the game industry?

Game AI

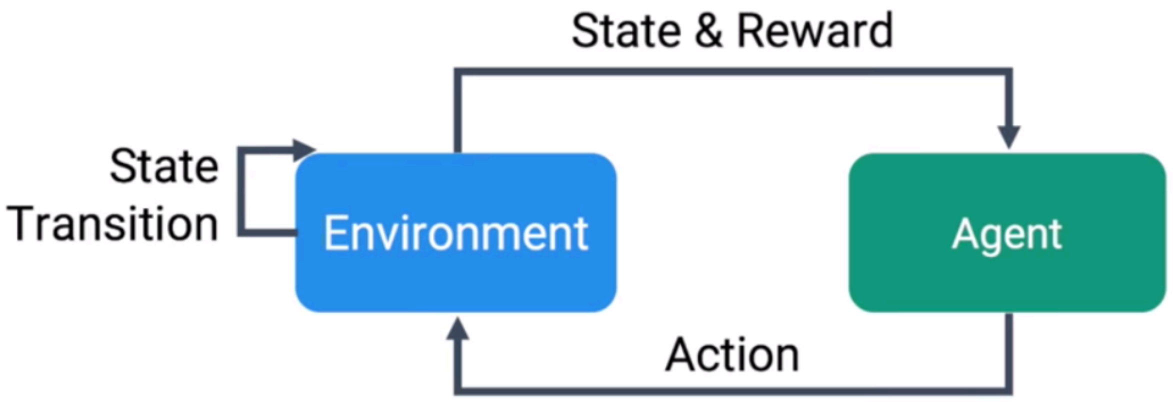
- Top 10 Video Games With The Worst AI
  - https://www.youtube.com/watch?v=mYhNvOg5yJ0

- Games with Best AI?

- What are the important properties in Game AI?

Game AI

- Scripting
- Finite State Machines
- Rule-based AI
- Fuzzy Logic
- Decision trees
- Behavior trees
- …



- What are the exiting libraries for Game AI?

Scripting

- What is scripting?
- Scripting Opponent Attributes:

```
CREATURE = 1;
If (LEVEL > 5)
  INTELLIGENCE = 20;
  SPEED = 50;
ELSE
  INTELLIGENCE = 5;
  SPEED = 25;
```

- Need to parse these files and link them to program variables.
- Why not just hardcode them?

Scripting Opponent Behavior

```
If (PlayerArmed == TRUE)
  DoFlee();
Else
  DoAttack();
```

- Patterns of movement
  - Explicit paths for characters to follow

Scripting Verbal Interactions

```
If (PlayerArmed == DAGGER)
  Say("What a cute little knife");
```

- Can load and playback WAV files.
- Of course these can be varied to take into account the nature of the characters involved.
- They can also convey information or respond to inquiries:

```
  If Ask("What is your name?")
     Say("I am Merlin.");
```

- More robust if keyword scripting is used.
  - Essentially search the string for a few keywords to match to.

Script Events

```
If (PlayerLocation(120, 76)
  Trigger(Explosion1);
If (PlayerLocation(10, 45)
  Trigger(TrapDoor7);
```

- Can we be a bit smarter about this?
  - Link to objects, not locations
  - Program/Implement a list of “effects” and reference them in a script
  - Package scripts with environment files

Decision Making

- Typically a small part of the effort needed to build a great game AI
- Most games use very simple decision making systems: state machines and decision or behavior trees
- Interest in more sophisticated decision making tools: fuzzy logic and neural networks (hard to get them working correctly) CONTROL!!!
- Input: knowledge
- Output: action request

Knowledge Representation

- External
  - Position of other characters
  - Layout of level
  - Whether a switch has been thrown
  - Direction that a noise is coming from

- Internal
  - Thought processes
  - Health
  - Goals
  - Past actions

Actions

- External
  - Throw a switch
  - Fire a weapon
  - Move to a room
- Internal
  - Change opinion
  - Change emotion state
  - Change goal

Rule-Based AI

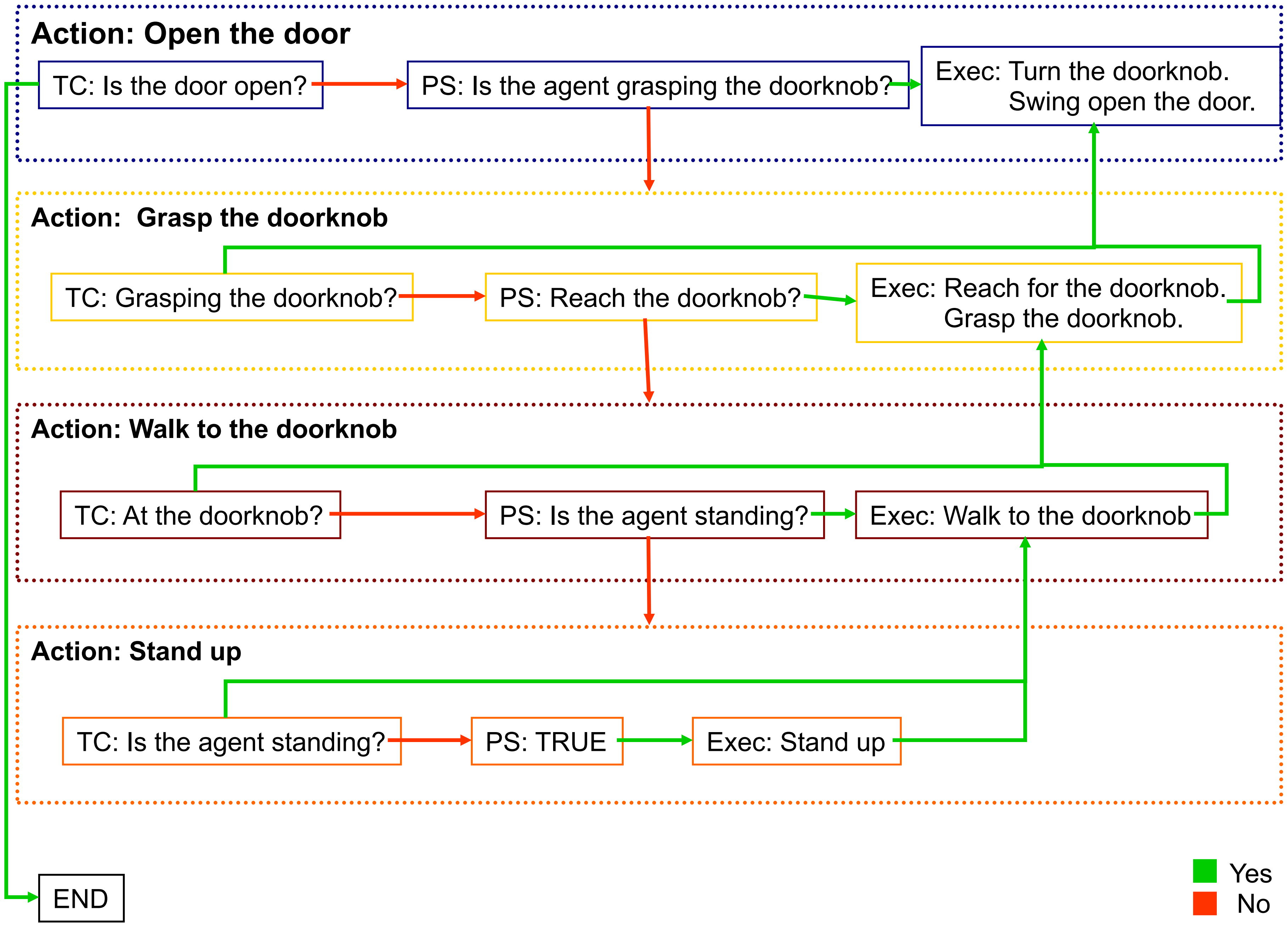
Rule-Based AI

- If-then style statements
- Expert systems
  - Medical diagnosis, engineering fault analysis, etc
- 2 components
  - Working memory: what is true about the world
  - Rules: what can be inferred from what is true

Inference

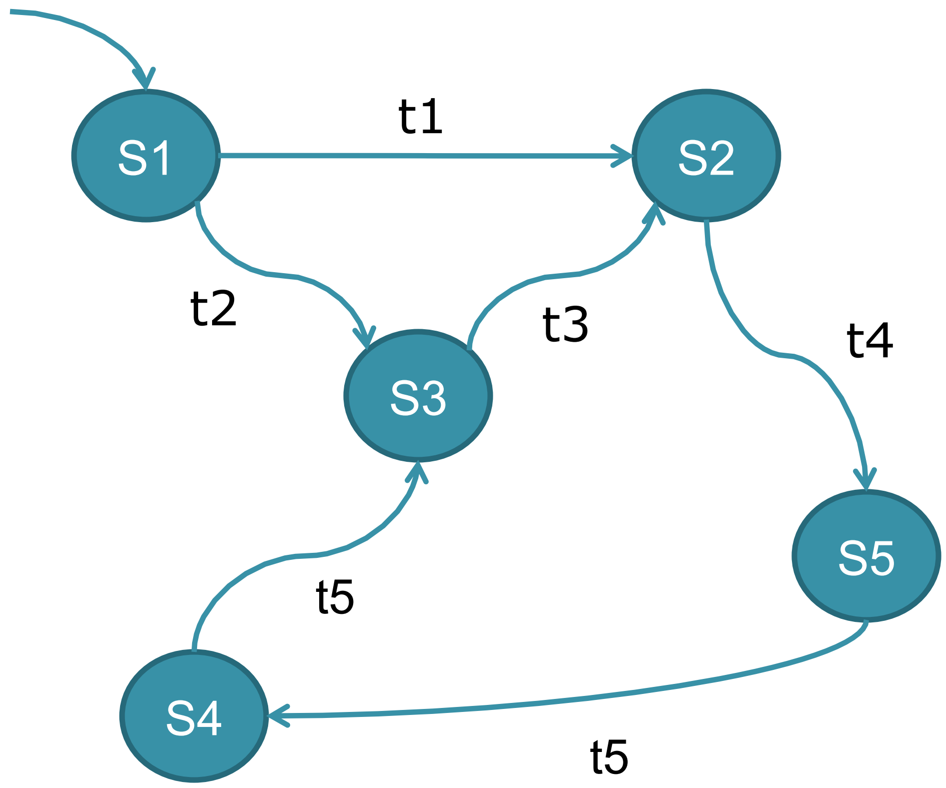
- Forward Chaining
  - Match rules to facts in working memory
  - Conflict resolution: determine what rule we want to fire
  - Fire rule: assert a new fact, trigger event, call function

- Backward Chaining
  - Have a goal and try to determine how to reach it
  - E.g. Open the door…

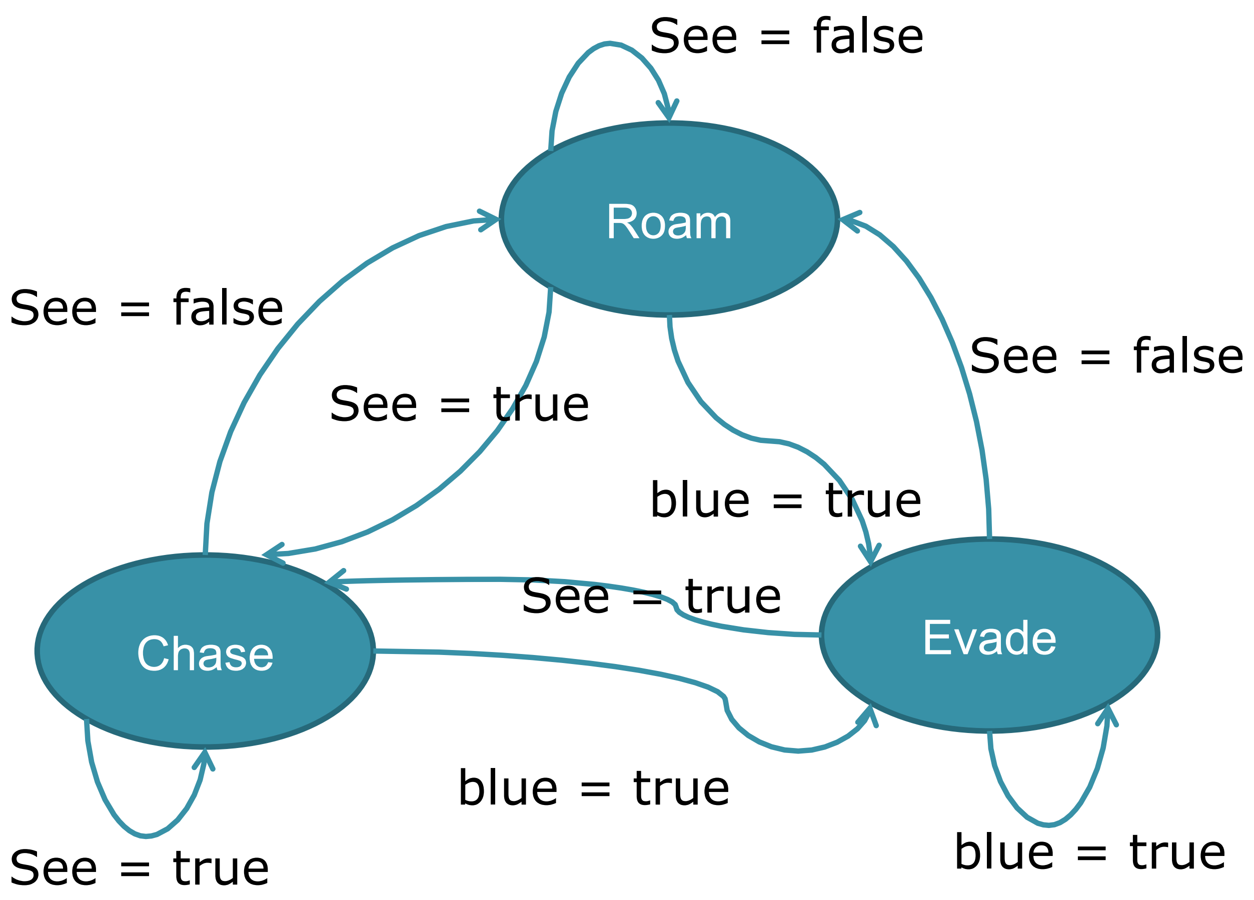

Action: Open the door
TC: Is the door open?
PS: Is the agent grasping the doorknob?
Exec: Turn the doorknob. Swing open the door.
Action: Grasp the doorknob
TC: Grasping the doorknob?
PS: Reach the doorknob?
Exec: Reach for the doorknob. Grasp the doorknob.
Action: Walk to the doorknob
TC: At the doorknob?
PS: Is the agent standing?
Exec: Walk to the doorknob
Action: Stand up
TC: Is the agent standing?
PS: TRUE
Exec: Stand up
END
🟩 Yes
🟥 No

Finite State Machines

Finite State Machines


$$\begin{tikzcd}
\arrow[r] & S1 \arrow[rr, "t1"] \arrow[dr, "t2"'] & & S2 \arrow[dr, "t4"] & \\
& & S3 \arrow[ur, "t3"'] & & S5 \arrow[dlll, "t5"] \\
& S4 \arrow[ur, "t5"] & & &
\end{tikzcd}$$

Ghost Controller



Implementation

```
Switch(currentState)
{
  Case Roam:
     If (See(Blue) == true) currentState = Evade;
     Else if (See(player) == true)
       currentState = Chase;
     Else if (See(player) == false)
       currentState = Roam;
  Case Chase:
     …
  Case Evade:
  …
}
```

How would this link to our current code setup?

Implementation 2

- We could also create or use a general finite state machine software package.
- Define states or nodes (WE LOVE NODES!).
- Define transitions
- Associate states with transitions
- Define a method that “advances” the state machine.
- Parallel state machines
- State machines can be nodes in other state machines.

State Machines and Rules

- What can Finite State Machines do that Rule-Based systems can’t?

- What can Rule-Based systems do that Finite State Machines can’t?

Let’s Design an AI

Game Design

- Characters: Frodo
- Actions:
  - Panic
  - Hold ring
  - Look for shade/water
  - Find spaceship

- Place: Tatooine
- Objects
  - Sand
  - Oasis
  - Camels
  - Raiders
- Goal
  - Stay with player
  - Keep player alive
  - Stay alive
  - Get to space ship

States?

Fuzzy Logic

Fuzzy Logic

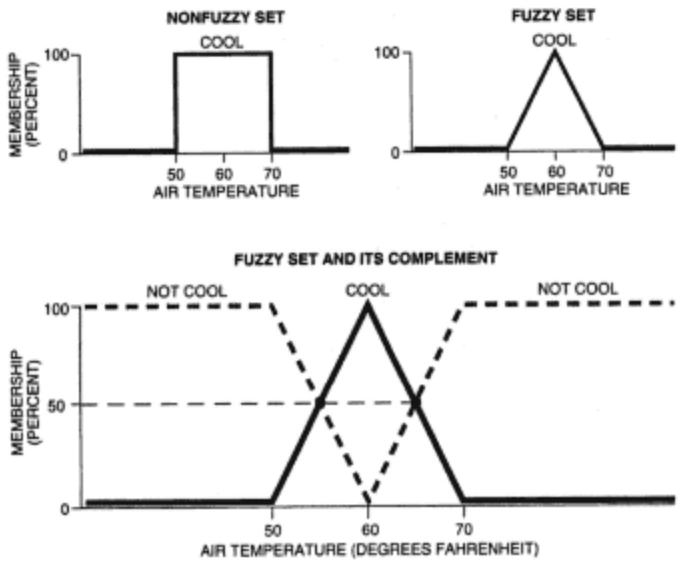
- What is it?
- “The essence of fuzzy logic is that everything is a matter of degree.”
- Imprecise
- Uncertainty
- “Very smart, but kind of slow” as opposed to intelligence = 170 and speed = 1.2mph
- Allows us to think more normally
- There are degrees of fuzziness (0 to 1)

Fuzzy Logic

$$\begin{tikzcd}
\boxed{\text{Crisp Input ---}\!\rightarrow\text{ Fuzzy Input}} \arrow[d] \\
\boxed{\text{Fuzzy Rules ---}\!\rightarrow\text{ Fuzzy Output}} \arrow[d] \\
\boxed{\text{Defuzzification---}\!\rightarrow\text{ Crisp Output}}
\end{tikzcd}$$

Fuzzification

- 185.3lb -> overweight
- 6'1" -> tall
- Membership functions
  - Map input variables to a degree of membership in a fuzzy set (between 0 and 1)
  - Equations
- Hedges
  - VERY(Truth(A)) = Truth(A)$^{0.5}$
  - NOT_VERY(Truth(A)) = Truth(A)^{2}



```
double FuzzyGrade(double value, double x0, double x1)
{
     double result = 0;
     double x = value;

     if(x <= x0)
           result = 0;
     else if(x >= x1)
           result = 1;
     else
           result = (x/(x1-x0))-(x0/(x1-x0));
     return result;
}
```

Fuzzy Rules

- Conjunction:
  - Truth(A AND B) = MIN(Truth(A), Truth(B))
- Disjunction:
  - Truth(A OR B) = MAX(Truth(A), Truth(B))
- Negation:
  - Truth(NOT A) = 1 – Truth(A)

Where Truth(A) means the degree of membership of A is some fuzzy set.

Example

- Overweight = 0.7, Tall = 0.3

- Overweight and tall = 0.3
- Overweight or tall = 0.7

- Not overweight = 0.3
- Not tall = 0.7
- Not very tall = ?

Fuzzy Rules

- All rules must be evaluated in parallel
- Output represents the degree of strength of each rule:
  - Attack to degree 0.3
  - Do nothing to degree 0.4
  - Flee to degree 0.7

Defuzzification

- Want a crisp number as output from a fuzzy system
- Aggregate the output strengths of the rules
- Define output membership functions
  - Composite membership function = each output set truncated to the output degree of membership for that set as determined by the strength of the rules.
  - All output sets are combined using disjunction

Defuzzification

- Example: Speeds to output actions
  - Flee = -10
  - Do nothing = 1
  - Attack = 10
  - Then multiply by the degree to which the output action is true
  - Calculate the weighted average
  - Output = (0.7*(-10) +0.4*1+0.3*10)/(0.7+0.4+0.3) = -2.5 (force)

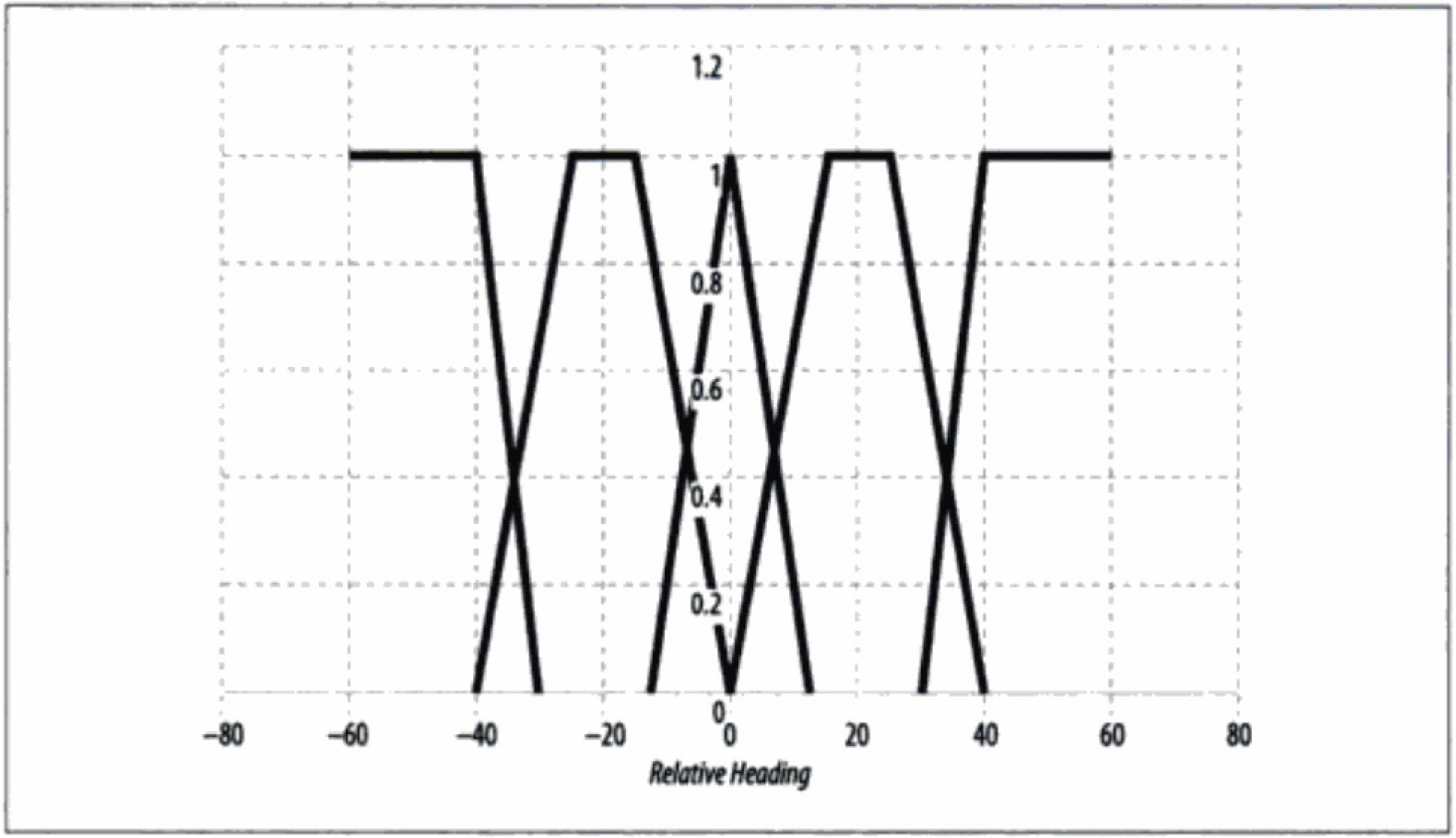
Fuzzy Logic in Games

- Control
- Threat Assessment
- Classification

Fuzzy Control

- Entity traveling toward a target (waypoint, enemy unit, treasure, home base, etc).
- Smoother path motions
- Instead of using exact coordinates use fuzzy concepts such as to the left, a little to the right, etc.
- Relate relative headings to *Far Left*, *Left*, *Ahead*, *Right*, and *Far Right*.

Relative Heading Fuzzy Sets



From AI for Game Developers

Membership Calculation Results for 33 deg

Fuzzy Set	Degree of Membership
Far Left	0.0
Left	0.0
Ahead	0.0
Right	0.47
Far Right	0.3

Right is a trapezoid membership function and Far right is a grade membership function.

NetForce = FarLeft * MFL + Left * MFL + Right * MFR + FarRight * MFR;
Where,
MFL = maximum force left = -100
MFR = maximum force right = 100

Steering force = 77!

http://www.red3d.com/cwr/steer/gdc99/

Fuzzy Threat Assessment

- Range = near, close, far, very far, etc
- Size = tiny, small, medium, large, massive, etc
- Results in: no threat, low threat, medium threat, high threat
- Models the computer as having less-than-perfect knowledge
- Allow, for example, the size of a defensive force to vary smoothly and less predictably.

Fuzzy Classification

- Rank characters according to combat prowess
  - Strength, weapon proficiency, number of hit point, armor class, etc.
  - Yield: wimpy, easy, moderate, tough, etc

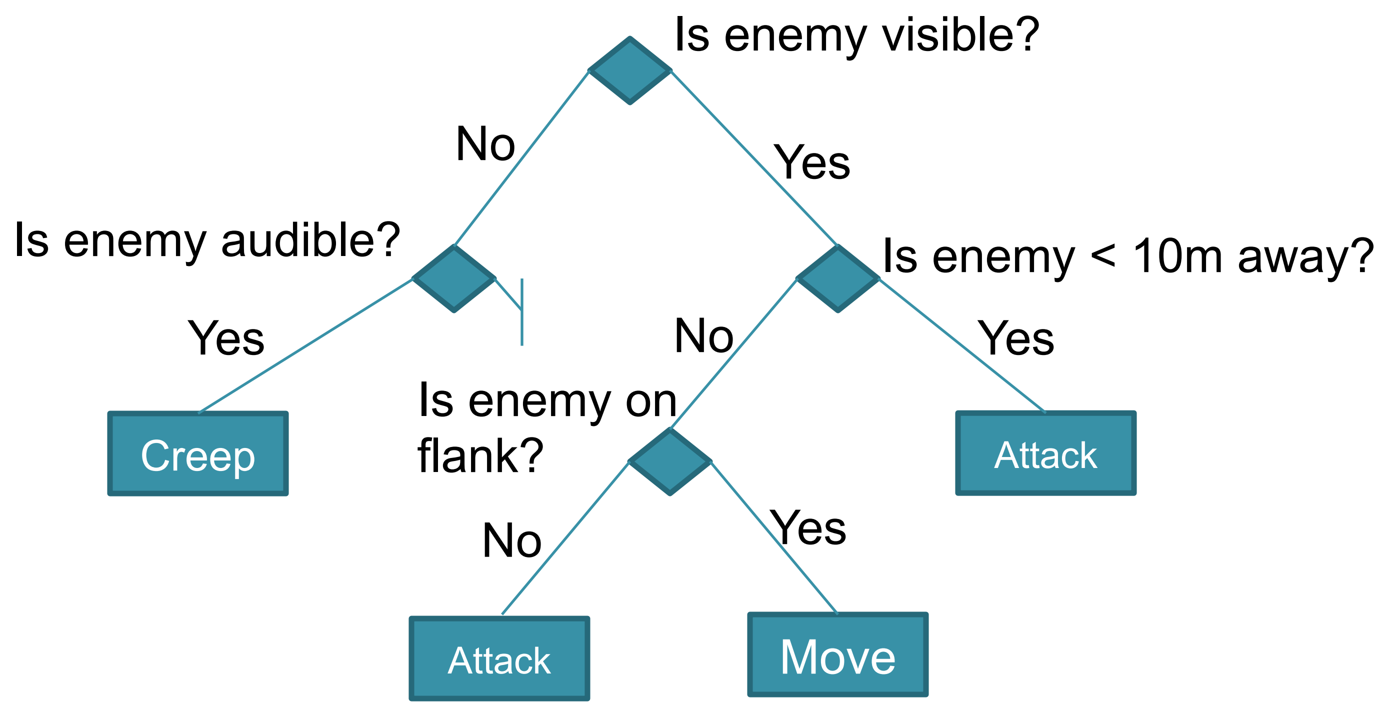
Decision Trees

Decision Trees

- Modular and easy to create
- From animation to complex strategic & tactical AI

- Problem: given a set of knowledge, generate a corresponding action from a set of possible actions
  - Mapping between input and output may be quite complex
  - Need to easily group lots of inputs together under one action, while allowing the input values that are significant to control the output

Decision Tree Example



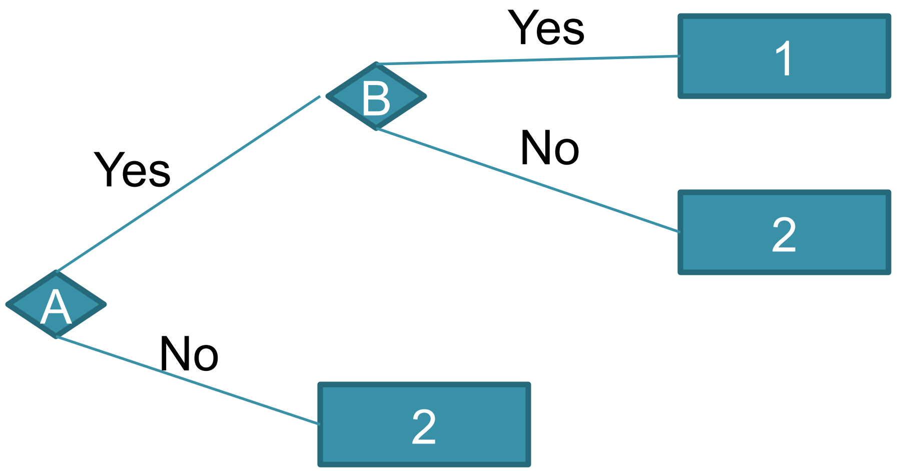
Decisions

- Simple
- Check a single value (no Boolean logic)

Data Type	Decisions
Boolean	Value is true
Enumeration	Matches one of a given set of values
Numeric value	Value within a given range
3D vector	Vector has a length within a given range (this can be used to check the distance between the character and an enemy, for example)

Combinations of Decisions

- Boolean operators represented in the tree structure
- If A and B then action 1, otherwise action 2



Decision Trees

- Binary trees aren’t required
- Balance is not required, but good
- You can do anything with a binary tree that you can do with a more complex tree
- Can include random decisions (e.g. flip a coin)

*f your students have any questions, please direct them to our University Relations webpage or **Blizzard Careers**' Twitter feed,*

New Grad Program

Our **New Grad Program** is back and we are on the hunt for graduating students interested in programming, engineering, IT, and business intelligence.

Please share with interested students and ask them to **apply NOW**!

*All positions are posted on our **Graduating Students** job board – check it out!*

Summer 2018 Internships

Blizzard's epic internship program returns in **2018** with opportunities in programming, design, art, business operations, and more.

Positions are posted **now until December 10th, 2017.**

We are hosting opportunities in our US locations – **Irvine** and **Austin**.

*All positions are posted on our **Internships** job board – check it out!*

University Relations Team Updates & Contact Information

We've updated our roster and look forward to working with you all this fall!

We added a new member to our team during this past year. Please join us in welcoming **Nick Araujo!**

Want to further connect with us? Connect with us on our Twitter and Facebook.

The WoW Student Art Contest

The 7th annual contest kicks off again this fall!

We'll be calling on **Environment Artists, Character Artists, FX Artists, and Character Animators** to show us their own unique take on Azeroth.

The contest is open now until January 28th, 2018.

Please check out the WoW Student Art contest - here!

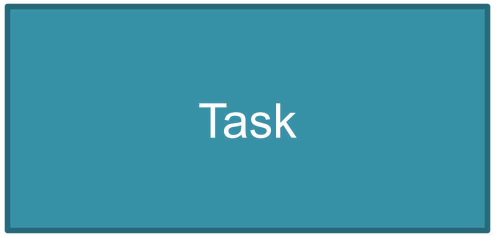
University Gaming Club Support

Our collegiate esports team, **Tespa**, has announce a new suite of student programs that support club members, esports teams, and leaders of registered gaming organizations on any campus in North America.
Please share with any relevant student orgs on your campus and ask them if they want to **get involved!**
*Read about the different student programs on **Tespa University** – check it out!*

Behavior Trees

Behavior Trees

- Halo 2, Unreal 4 AI
- Synthesis of a number of techniques
  - Hierarchical State Machines
  - Scheduling
  - Planning
  - Action Execution
- Main building block is a task
  - Look up value of variable in the game state
  - Execute an animation
  - Composed into sub-trees to present complex actions which are then composed into higher level behaviors
- Build into hierarchies without having to worry about the details of how each sub-task in the hierarchy is implemented



Types of Tasks

- All have the same basic structure: given some CPU time, do their thing, return status code
- Broken into smallest parts that can usefully be composed
- Coupled with GUI to edit trees
  - Technical artists
  - Level designers

- Conditions
- Actions
- Composites

Conditions

- Test some properties of the game

- Proximity
- Line of sight
- State of character (e.g. health, ammo)

- Each separate, parameterized tasks (reused)
- Returns true if condition is met, false otherwise

Actions

- Alter the state of the game
  - For animation
  - For movement
  - To change the character’s internal state
  - To play audio samples
  - To engage the player in dialog
  - To engage specialized AI (e.g. pathfinding)
- Each has its own implementation (and there can be many)

Tasks

- Single common interface

- Arbitrary conditions, actions and groups can be combined together without needing to know the rest of the tree
- Actions sit at leaf nodes
- Branches are composite nodes
  - Normally few (e.g. selector node and sequence node)
  - Run each of their child behaviors in turn
  - When a child behavior is complete and returns its status code, the composite decides whether to continue through its children or to stop and return a value.

Selector Node

- Returns a success status code as soon as one of its children runs successfully
- Keeps trying while children are failing
- If all children failure, returns failure status code
- Used to choose the first of a set of possible actions that is successful

- Example: Reach safety
  - Take cover **or** leave dangerous area **or** find backup
  - Try each in turn until safety is reached or no option left

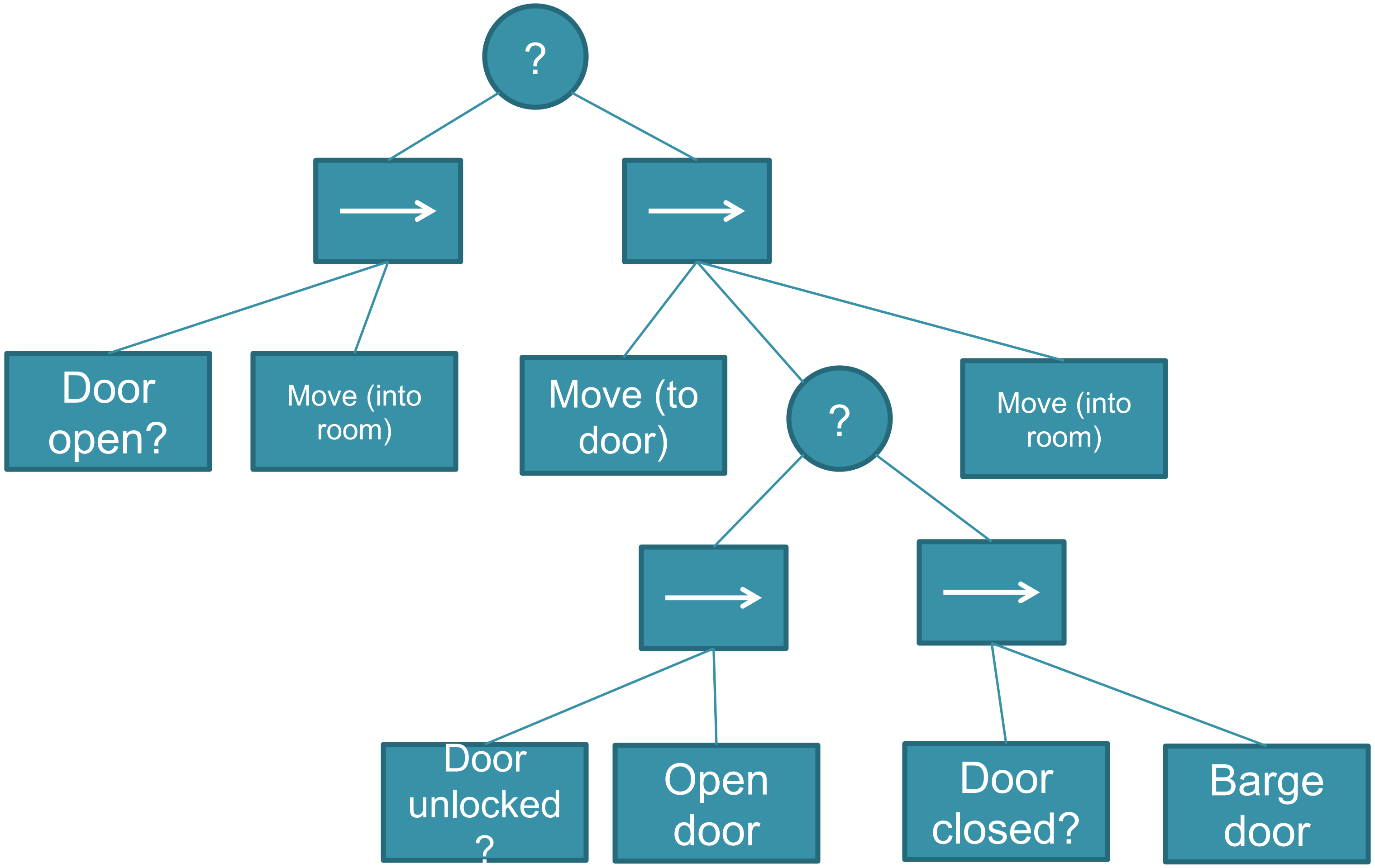
Sequence Node

- Returns failure code immediately if one of its children fails
- If it runs out of children, it will return success
- Represents a series of tasks that need to be undertaken

- Example: Find cover
  - Choose a cover point
  - Move to it
  - When in range, play roll animation to arrive behind it

- If one step fails the whole behavior fails

Behavior Tree Example



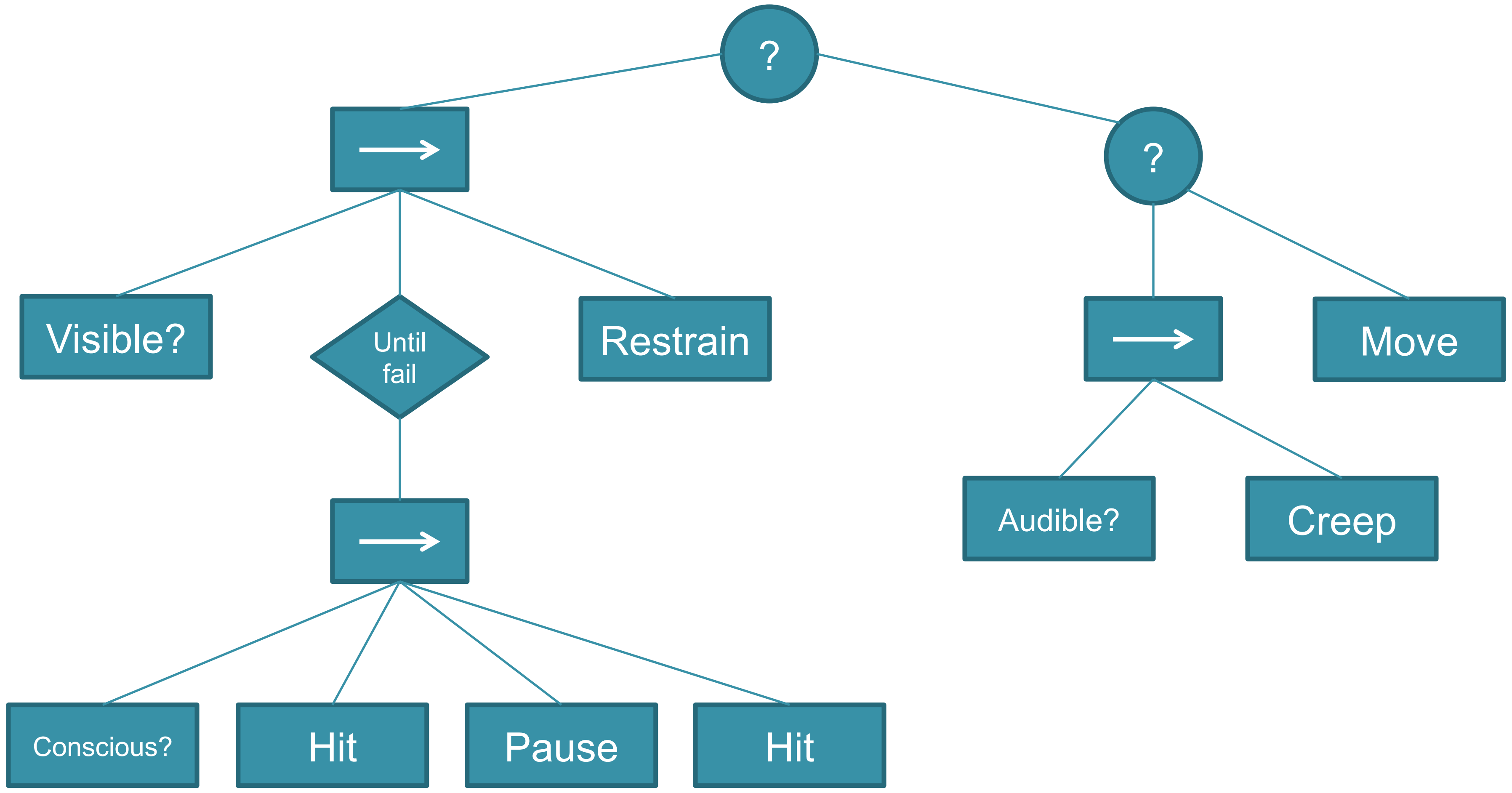
Implementation?

Ordering issue?

Decorator Tasks

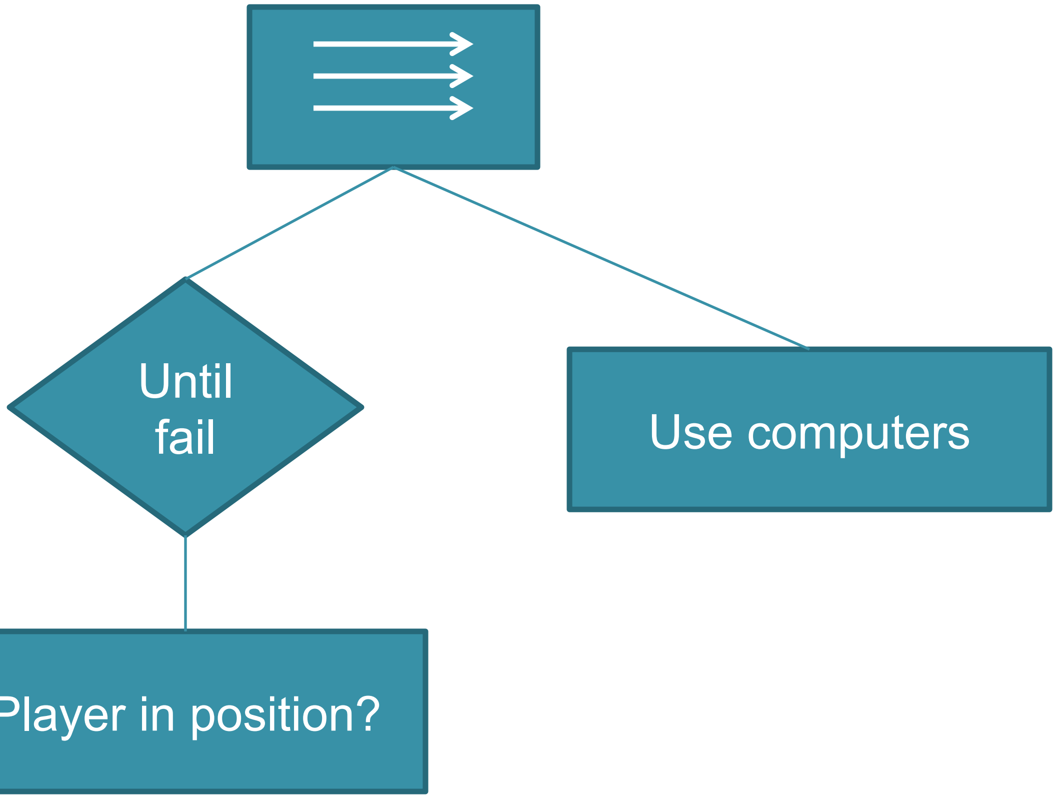
- Class that wraps another class, modifying its behavior
- Same interface as other tasks
- Has one single child task that it modifies in some way
- Example: Filters
  - Limit the number of times a task can be run
  - Keep running a task until it fails
  - Inverter (not)
  - Semaphore to protect resources

Behavior Tree Example



Parallel Task

- Similar to Sequence Task



Advanced Concepts

- Concurrency and Timing: actions can take a while
  - Threads
  - Waiting

Let’s Design an AI Using Behavior Trees

- Character and behaviors?

- Actions:
    -

More AI

- Map perceptions (sequences) to actions
  - Simple reflexes
  - Characters with memory
- Goal-based characters
- Utility-based characters
- Environment-based AI
  - Smart-objects
- Searching and search algorithms
  - A*, backgammon, chess, etc
- Learning
  - Neural and belief networks
  - Reinforcement learning
- Communication between characters

Model of Game AI

- Movement: get the character to where it needs to be intelligently
- Decision Making (action selection):
  - What to do next?
  - Set of behaviors to choose from
- Strategy:
  - Not needed in a lot of games
  - Team coordination (e.g. surround player)
- Intermixed
- Infrastructure:
  - Movement → action through animation or physics engine
  - Perception/world interface
  - Management for memory and processor time

Agent-Based AI

- Autonomous characters
- Take info from game data
- Determine what actions to take
- Carry out actions
- Contrast to traffic or pedestrian simulations