DRC Accommodations

If you qualify for classroom accommodations because of a disability, please get an Accommodation Authorization from the Disability Resource Center (DRC) and submit it to the instructor in person outside of class (e.g., office hours) within the first two weeks of the quarter. Contact DRC at 459-2089 (voice), 459-4806 (TTY), or http://drc.ucsc.edu for more information on the requirements and/or process.
Readings

Readings so far

- First lecture: 19-35
- Decision Tree: 293-309
- State Machines: 309-333
- Behavior Trees: 334-371
Project 1 Updates

- Running against
  - Legacy2Reckoning
  - StarterPacMan
- Complete games
  - Maze rotation
  - Difficulty Increase (for Mrs. PacMan)
The AI Model

Execution Management

Group AI

Strategy

Character AI

Decision Making

Movement

World Interface

Animation

Physics

Content Creation

Scripting
Decision Making AI

Internal Knowledge

External Knowledge

Decision Maker

Action Request

Internal changes

External changes
Behavior Trees

Behavior trees organize tasks into a tree or, more generally, directed acyclic graph (DAG)

Three kinds of tasks, all of which can succeed or fail
- Conditions
- Actions
- Composites

The leaves of the tree will consist of conditions and actions (things that can be executed in a game engine)

The power of behavior trees comes from the execution policies associated with composites
Behavior Tree History

- In AI research, BTs come out of reactive planning

- In 2004 and 2005 two games introduced reactive planning to the game AI community
  - *Halo 2*
  - *Façade*

- Like with other decision making techniques, BTs can be implemented a number of ways
  - OO implementation (Millington and Funge)
  - A custom language (the ABL approach in Façade)

- This lecture will follow the Millington and Funge terminology
Selector

Class Selector (Task):
```python
def run():
    for c in children:
        if c.run():
            return true
    return false
```

- **Tries children until one succeeds**

```java
Attack
Taunt
Stare
```

expressiveintelligencestudio
Sequence

Class Sequence (Task):
```python
def run():
    for c in children:
        if not c.run():
            return false
    return true
```

- Executes all children sequentially, succeeding if all succeed

- Enemy visible?
- Turn away
- Run away
Behavior Tree Example

```
? →
| →
| └ Door open?
|    └ Move into room

? →
| →
| └ Move (to door)
|    └ Open door
|    └ Move into room
```
DIY: Code this sequence.

Door open?

Move into room

Move (to door)
Open door
Move into room
DIY: Write Pseudocode for this BT

If (IsOpen(door)) then
  MoveTo(room)
Else
  MoveTo(door)
  Open(Door)
  MoveTo(room)
Expanding the Behavior Tree

- Door open?
  - Move into room
  - Move (to door)
    - Door unlocked?
      - Open door
    - Barge door
    - Door open?
DIY: GetPowerPill Behavior Tree

Use Actions, Conditions and both Composites (Selectors and Sequencers).

Available actions and Conditions:
- PowerPillExists?
- MoveTo(Pill)
- LureAway(Ghosts)
- TakeAlternatePath
DIY: GetPowerPill Behavior Tree

1. PowerPill Exists?
2. MoveTo (Pill)
3. Alternate Path
4. LureGhosts Away
Primitive Tasks (Pseudo Code)

Class Task:
  children
def run()

Example Condition
Class EnemyNear (Task):
def run():
  if distanceToEnemy < 10:
    return true
  return false

Example Action
Class PlayAnimation (Task):
  animationID
  speed
def PlayAnimation(animID, loop=false, speed=1.0):
  this.animationID = animID
  this.speed = speed
def run():
  if animationEngine.ready():
    animationEngine.play(animation, speed)
  return true
  return false
Executing the Behavior Tree

- What is the problem with the implementation we have so far for the BT?
Executing the Behavior Tree

- What is the problem with the implementation we have so far for the BT?
- What if the task takes time?
- What if the NPC should be doing more than one thing at a time?
- What if NPCs need to share trees?
- What if it's handy to keep data local to the tree?
Concurrency

- What are some ways of fixing this problem?
  - Multi-threading
  - Have tree maintain execution state and rerun tree each decision cycle

- We need a way to keep track of the BT execution state outside of the main thread
Primitive Tasks (Maintaining Exec State)

Class Task:
  children
  // Is the task currently active
  active = false
  // now returns, True, False or MoreTime
  def run()

Example Condition
Class EnemyNear (Task):
  def run(): // Doesn’t use active since conditions run quickly in main thread
    if distanceToEnemy < 10:
      return true
    return false
  return false
Class ActionThatTakesTime (Task):
    actionResult = none

    def callback(actionResult)
        this.actionResult = actionResult

    def performAction()

    def run():
        if active == false:
            active = true
            result = performAction()
        else if actionResult == none
            return MoreTime
        else
            active = false
            return actionResult

Class PlayAnimation (ActionThatTakesTime):
    animationID
    speed

    def PlayAnimation(animID, loop=false, speed=1.0)
        this.animationID = animID
        this.speed = speed

    def performAction()
        if animationEngine.ready():
            animationEngine.play(animation, speed, callback)
        else:
            actionResult = false
OneTaskFromGroup and Selector

Class OneTaskFromGroup (Task):
  currentChild = null
  childIndex = 0

  def selectFirstChildFromGroup():
    active = true
    childIndex = 0
    return children[childIndex]

  def selectNextChildInGroup():
    childIndex = childIndex + 1
    return children[childIndex]

  def hasMoreChildrenInGroup():
    return childIndex + 1 < children.length

Class Selector (OneTaskFromGroup):
  def run():
    if !active:
      currentChild = selectFirstChildFromGroup()
      result = currentChild.run()
      if result == MoreTime:
        return result
      else if result == True:
        active = false
        return True
      else:
        if hasMoreChildrenInGroup():
          currentChild = selectNextChildInGroup()
          return MoreTime
        else:
          active = false
          return False
Non-deterministic Composites

- Entering…
- Open…
- ~?
- Barge door…
- Get matches
- Get gasoline
- Douse door
- Ignite door

Entering… → Open… → ~? → Barge door… → Get matches → Get gasoline → Douse door → Ignite door
DIY: Orange Ghost
Decorators

- Decorators wrap a single task to modify the behavior in some way

Examples
- Limit
- UntilFail
- Inverter
- SemaphoreGuard
SemaphoreGuard to guard resources

- Technique 1: Hardcode test in action
- Technique 2: Condition plus sequence
  - Anim engine available?
  - Play animation
- Technique 3: Decorator
  - Guard
  - Play animation
Class Parallel (Task):

children
runningChildren
result

def run():
    result = undefined
    for child in children:
        thread = new Thread()
        thread.start(runChild, child)

    while result == undefined:
        sleep()

    return result

def runChild(child):
    runningChildren.add(child)
    returned = child.run()
    runningChildren.remove(child)

    if returned == false:
        terminate()
        result = false
    else if runningChildren.length == 0:
        result = true

def terminate():
    for child in runningChildren:
        child.terminate()
Using Parallel to Implement Group Behavior

Soldier 1: has ammo?
Soldier 1: attack...
Soldier 2: has ammo?
Soldier 2: in cover?
Soldier 2: snipe attack...
Soldier 3: has ammo?
Soldier 3: exit route?
Soldier 3: Guard exit...
Retreat...
Take cover...

~?
DIY: Red and Blue Ghosts

Whimsical

Pac-Man and Red ghost's positions.

2 tiles ahead then double Red's vector.
Behavior: NPC uses computer for awhile while player is standing in a particular part of the room

- Player in position?
- Use computers
- Until fail
- Player in position?
- Use computers
Interrupters

Player in position?

Interrupter

→

→

→

Use computers

Perform Interruption

Until fail
Sharing Data Across Nodes of the Tree

- Use a blackboard to share data

- The simplest implementation of a blackboard is a hashtable
  - Key is the variable name
  - Value is the value

- Where does the blackboard live?
  - Associate it with a BT in a wrapper class
  - Have run pass it all the way down
  - Have a decorator (scope hierarchy of blackboards)