AI Bot Architectures

CS395 GAI
Spring 2005
Common Components

- World
- Sensory system
- Perceptual system
- Action selection
- Motor system
AI Bot Architectures

• Scripting
• Behavior-based
• Rule-based
• Goal-based
• Plan-based
• Layered
Types of AI Components

- Reactive
  - Responds directly to environmental factors
  - Ideal for twitch-type response

- Deliberative
  - Contains model of the game environment
  - Perform inference in decision-cycle

- Reflective
  - Learn from experience
Scripting

• Commonly called “AI” for marketing purposes
• Game environment is a stage, designer gives the actors directions

• Advantages
  – Tight, absolute control over agent behavior
  – Useful for scenario design

• Disadvantages
  – Static behaviors
    • Threat to replay-ability
  – Very specific to the particular game/map
Behavior-based AI Architectures

- Weak-AI technique
- Borrowed from the robotics community (Brooks)
- Improvement on the authoring of Finite State Machines
  - Design in terms of higher-level behaviors
- Believability vs. intelligence
- Externalizes environment models
  - No internal model of what the bot is actually doing and any given moment
Behavior-based AI Architectures

• Example: FlexBot behaviors for Half-Life
  – Written in the Generic Robot Language (Horswill)
  – Compiles down to FSM (C or Lisp code)

• Very efficient
  – Game engine is the bottleneck not the AI
Behavior-based AI Architectures

• Excerpt from a GRL control program

Sample code for a behavior in Groo

```plaintext
shoot? =
    (and facing-enemy?
        not-alt-fire?
        (or clip-not-empty?
            (= current-weapon
crowbar))
        (or (and enemy-long-range?
            long-range-weapon?)
            enemy-short-range?
            being-shot?))
```

```plaintext
Movement
Behaviors
```
```
• pitch behavior
• shoot behavior
• alt-fire behavior
• jump behavior
• switch-gun behavior
• reload behavior
• duck behavior
• use-item behavior
```

FlexBot DLL
Behavior-based AI Architectures

• Towards self-explanation in behavior-based control systems
Rule-based AI Architectures

- Production-rules explicitly define actions to be executed in response to certain conditions

- Advantages
  - Architecture is simple to build
  - Easy for non-programmers to develop rules

- Disadvantages
  - Difficult to organize
    - Lots of independent rules
  - Difficult to debug, identify conflicting rules
  - Sequences of actions must be defined using a series of stateful triggers
Rule-based AI Architectures

• Example: Age of Kings AI “scripting”
• Series of prioritized production rules

(defrule
  (building-type-count-total castle less-than 1)
  (can-build castle)
=>
  (build castle)
  (chat-local-to-self "castle")

• http://www.cs.uga.edu/~potter/aok/WDPsample.per
Goal-based Architectures

- Example: SOAR Quakebot (Laird)
  - Production rules suggest actions
  - Suggestions are evaluated vs. goals, and operator is chosen
  - Core implementation does no planning
  - Frequent re-evaluation of actions in the decision cycle

Figure 4. The Soar Decision Cycle
Plan-based Architectures

• Classical planners
  – Most frequently used for path planning
• HTN’s
Layered AI Architectures

(Sloman, Scheutz)
Useful Techniques

Techniques used to augment various architectures
Bayesian Networks

• Bayes’ Theorem
  – $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$
  – $P(A|B)$ “the probability of $A$ given that what I know is $B$”

• Example
  – “the probability that it rained yesterday, given that your lawn is wet”
  – $P(B|A) =$ the probability that the lawn would be wet if it actually rained yesterday
  – $P(A) =$ the probability of rain, all other things being equal
  – $P(B) =$ the probability of your lawn being wet, all other things being equal
Bayesian Networks

• Combination of prob. propositions in a graph structure called a “belief network” or a “Bayesian network”
• Model underlying cause-and-effect relationships between game phenomenon
• Dealing with uncertainty in the perceptual system
  – Infer likely facts about other players based on partial or incomplete observations
Bayesian Networks

- Example: using a Bayes network in a RTS to infer the existence or nonexistence of some technologies by the presence or absence of others

(Paul Tozour, *AI Game Programming Wisdom*)
Level-of-Detail for AI

• Path-planning
  – Waypoints
  – Voronoi diagrams
Learned Heuristics

- Initial frontier for reflective systems
- “Stench-of-death” tiles in RTS
- Influencing decision probabilities in Black & White