EDUCATION

Attention, Students: Put Your Laptops Away

3:19

+ Queue

April 17, 2016 · 6:00 AM ET
Heard on Weekend Edition Sunday

https://www.npr.org/2016/04/17/474525392/attention-students-put-your-laptops-away

Disclaimer: I use these notes as a guide rather than a comprehensive coverage of the topic. They are neither a substitute for attending the lectures nor for reading the assigned material.

- "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)

Announcements

- Exam 1
- HW5 review
- HW6: BTrees

Trajectory update

- N-5: Behavior Trees
- N-4: Rule based systems
- N-3: Planning (HTN + A* Behavior planning)
- N-2: exam 1

- N-1: Guest lecture, PCG level generation
- N-0w: Decision making – Communication and Fuzzy logic
- N+1: Fuzzy logic

- Next week: spring break
- After: PCG

Sidebar – The Game Loop

The “**Game Loop**”

while game is running

process inputs

update game world

generate outputs

```
while( user doesn't exit )
  check for user input
  run AI
  move enemies
  resolve collisions
  draw graphics
  play sounds
end while
```

https://en.wikipedia.org/wiki/Game_programming

http://www.informit.com/articles/article.aspx?p=2167437&seqNum=3

What are production/rule systems?

- Based on the current game **state**, activate a set of **rules**, pick/**arbitrate** from those based on some heuristic (e.g. best matches conditions)
- What were these traditionally called?
- Use in Games Industry includes Environment-sensitive dialog generation (dynamic dialog gen)

- Pros:
  - Don’t need to specify response to every contingency
  - Can respond to novel conditions
- Cons:
  - Hard to author robust rule systems
  - Emergence vs. over-engineering
  - Hard to debug

What is Planning?

- Finding a **sequence of actions** to achieve a **goal** or **accomplish a task**
  - Problems requiring **deliberate** thought ahead of time and a sequence of actions
- Basic planning comes down to **search**:
  - "behavior planning using A*"
    - Given: state of the world, a goal, and models of actions
    - Find: sequence of actions (a plan) that achieves the goal
  - hierarchical task network planning
    - Given: state of world, a list of tasks, and models of actions and methods
    - Find: a sequence of actions (a plan) that achieves the tasks by recusively reducing them into appropriate sub-tasks supported by primitive actions

Decision Making: Reactions vs Anticipation

- Shallowness & Realism
  - All reactive techniques have the agent take **next** "best"/prescribed move, but **don't look further** into the future than the next state. Agents ought to be **motivated by goals** and able to consider the effects of actions
- Adaptability
  - Each technique is more general/adaptive than the last (Planning > Rule Systems > Behavior Trees > FSMs > Decision Tree). To **perform *well* in unanticipated** situations is challenging
- Design Burden
  - All decision making techniques impart an authoring burden on designers (FSMs: states and transitions, B trees: nodes and structure, Rules: all conditions and each individual rule, Planning: state rep and methods/operators).
  - Knowledge engineering is a "hot potato" (can't eat it, no free lunch)

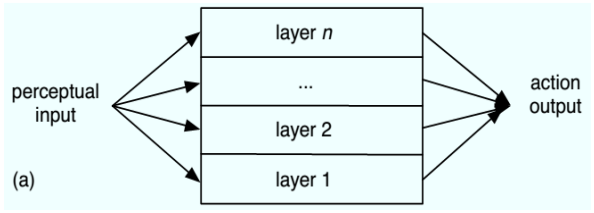
Planning Resources

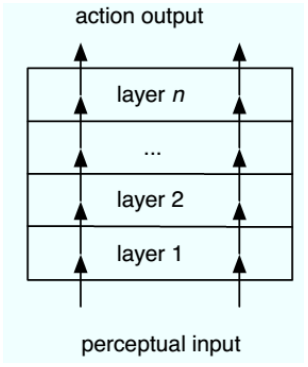
- F.E.A.R AI: https://www.youtube.com/watch?v=rf2T_j-FlDE
- Dana Nau HTN and games presentation
  - http://www.cs.umd.edu/~nau/papers/nau2013game.pdf
- Killzone 2 AI:
  - https://www.youtube.com/watch?v=7oWKCLdsGTE
  - http://www.ign.com/boards/threads/killzone-2-enemy-a-i-is-it-up-there-with-fear-as-1.177634641/

Planning Resources

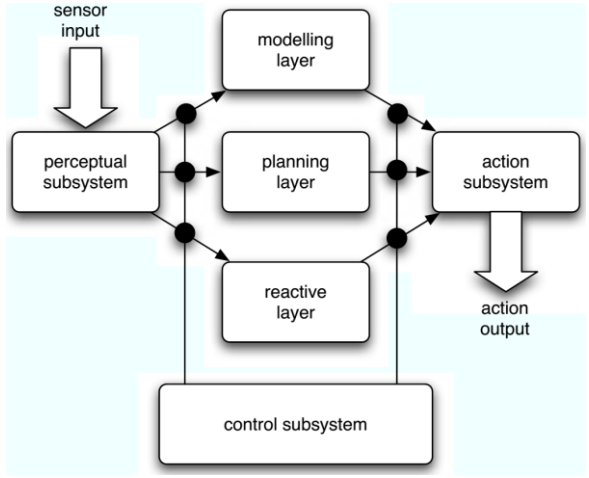
- Planning in modern games:
  - http://aigamedev.com/open/review/planning-in-games/
  - Nau HTN planning in Killzone: http://www.cs.umd.edu/~nau/papers/nau2013game.pdf
  - G.O.A.P: http://web.media.mit.edu/~jorkin/goap.html
  - Workshop at ICAPS 2013: http://icaps13.icaps-conference.org/technical-program/workshop-program/planning-in-games/
  - The AI of F.E.A.R.: http://alumni.media.mit.edu/~jorkin/gdc2006_orkin_jeff_fear.pdf
- SHOP, JSHOP, SHOP2, JSHOP2, Pyhop (HTN planners)
  - http://www.cs.umd.edu/projects/shop/
- Scala impl. of partial-order planning
  - https://github.com/boyangli/Scalpo
- Other planners:
  - http://www.cs.cmu.edu/~jcl/compileplan/compiling_planner.html
- Facing your F.E.A.R. lecture: https://www.youtube.com/watch?v=rf2T_j-FIDE

Decision Making: Architectures…

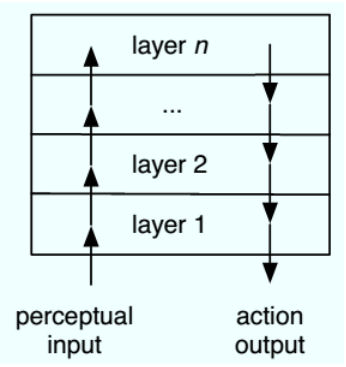




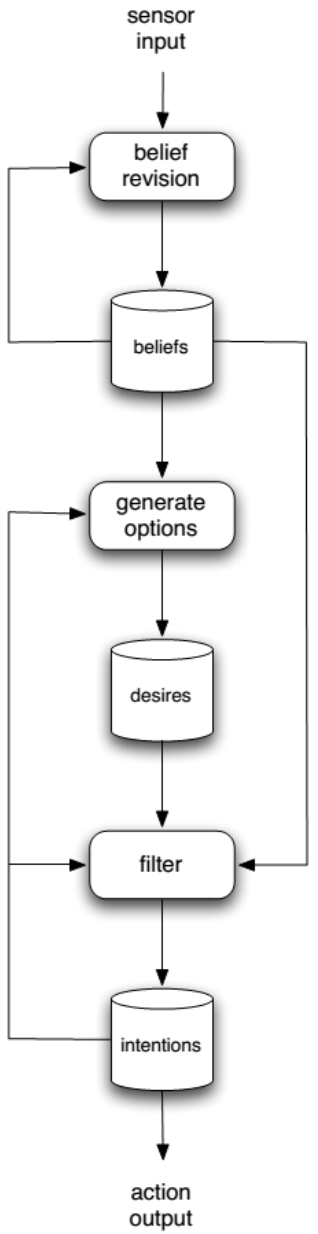
One-pass



Brooks Subsumption Arch.



Two-pass



BDI

See Multiagent Systems 2nd ed, Gerhard Weiss

Decision Making:
(Blackboard) Communication

2018-03-13

M&F 5.9

DM: Communication. Why?

- Lens: Multi-agent system
  - Collection of collaborative agents
  - Communicate & cooperate
  - Retain autonomy
  - Need for negotiation / mutually acceptable agreements (cooperative problem solving)
- Reasoning decomposition: distributed expertise
  - Problems too large for single / centralized agent
  - Reactive agents rarely communicate / collaborate
  - Problem independence, partial result sharing
- Hope: Sum greater than parts

An **agent** is a **computational entity** such as a software program or a robot that is **situated in some environment** and that to some extent is able to **act autonomously** in order to achieve its **design objectives**. As **interacting entities**, agents do not simply exchange data but are **actively engaged in cooperative and competitive scenarios**. They may **communicate** on the basis of semantically rich languages, and they **achieve agreements** and **make decisions** on the basis of processes such as negotiation, argumentation, voting, auctioning, and coalition formation. As intelligent entities, agents **act flexibly, that is, both reactively and deliberatively**, in a variety of environmental circumstances on the basis of processes such as planning, learning, and constraint satisfaction. As **autonomous entities**, agents have **far-reaching control over their behavior** within the frame of their objectives**, possess decision authority** in a wide variety of circumstances, and are **able to handle complex and unforeseen situations on their own** and without the intervention of humans or other systems. And as entities situated in some environment, **agents perceive their environment** at least partially and **act upon their environment without being in full control of it**.

See Multiagent Systems 2nd ed, Gerhard Weiss

Distributed Decision Making: A Recipe

1. Decompose the task
2. Allocate subtasks to “experts”
3. Await task accomplishment
4. Synthesize & Arbitrate results

Information sharing needed for most/all!

Communication Types

- Point to Point
  - Experts directly communicate w/eachother
  - Where have we seen this?
- Broadcast
  - Send information to group of experts
  - Talk about today.
- Mediated
  - Experts go through facilitator/arbitrator

Communication Mediums

- Firm software interfaces
- Databases
- Protocol layers (e.g.: TCP/IP + JSON)
- Hierarchies (hybrids)
- Pub/Sub services

BLACKBOARD ARCHITECTURES

Blackboards

- Isn’t a decision making algorithm
- Architecture / coord. mechanism / pattern
- Problem: Multiple decision making systems (experts). How to communicate (share data)?

On simplicity

- M&F: “Decision trees, state machines, and blackboard architectures have all been used to control steering behaviors. Blackboard architectures, in particular, **are suited to cooperating steering behaviors**; each behavior is an expert that can read (from the blackboard) what other behaviors would like to do before having its own say.”

On reusability

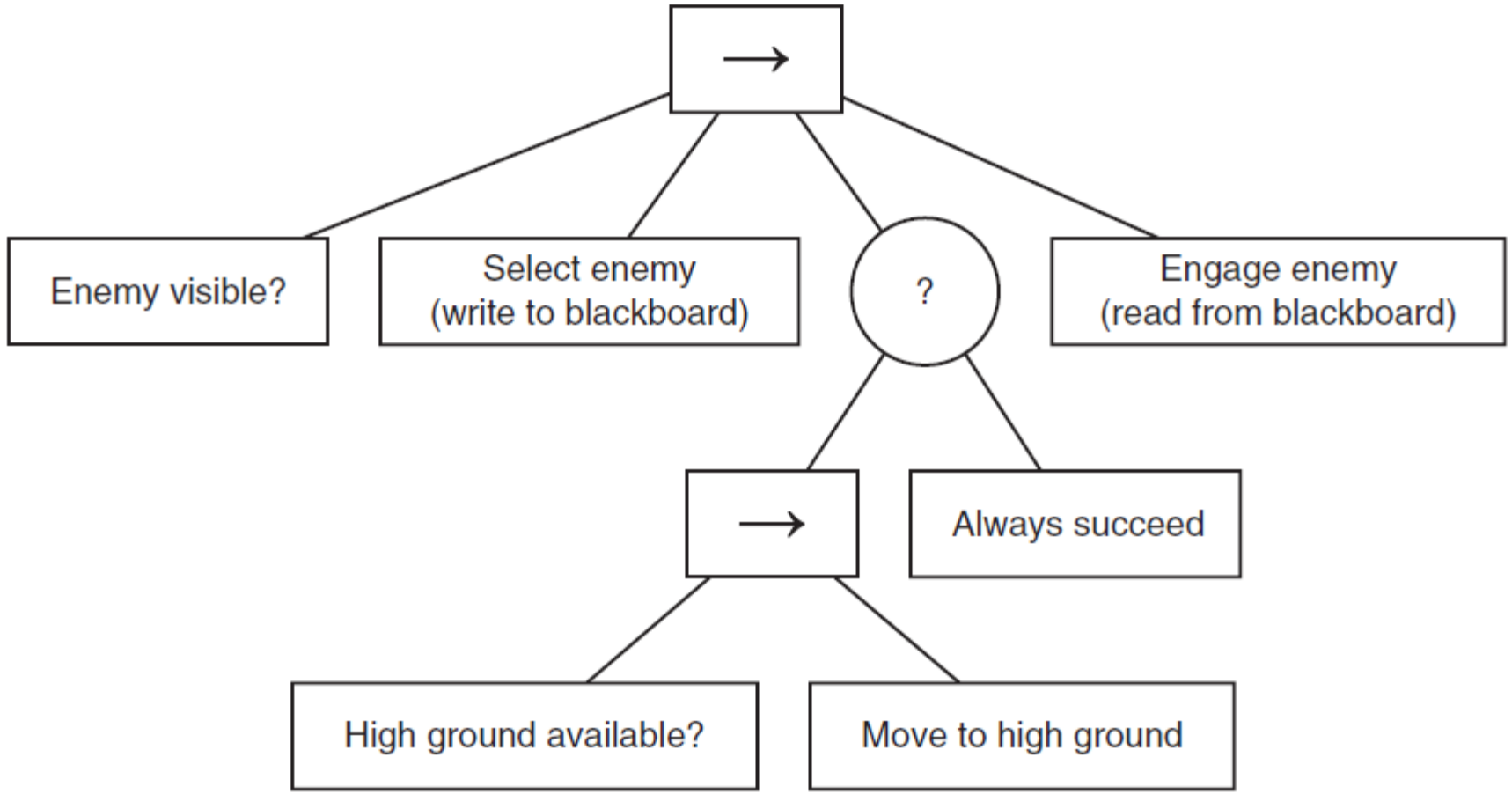
- M&F: “The most sensible approach is to **decouple the data** that behaviors need from the tasks themselves. We will do this by using an **external data store** for all the data that the behavior tree needs. We’ll call this data store a blackboard. […] For now it is simply important to know that **the blackboard can store any kind of data** and that **interested tasks can query it** for the data they need. Using this external blackboard, we can write tasks that are still independent of one another but can **communicate when needed**.”

Example

```
class MoveTo (Task):
        # The blackboard we're using
        blackboard

        def run():
                target = blackboard.get('target')
                if target:
                        character = blackboard.get('character')
                        steering.arrive(character, target)
                        return True
                else:
                        return False
```

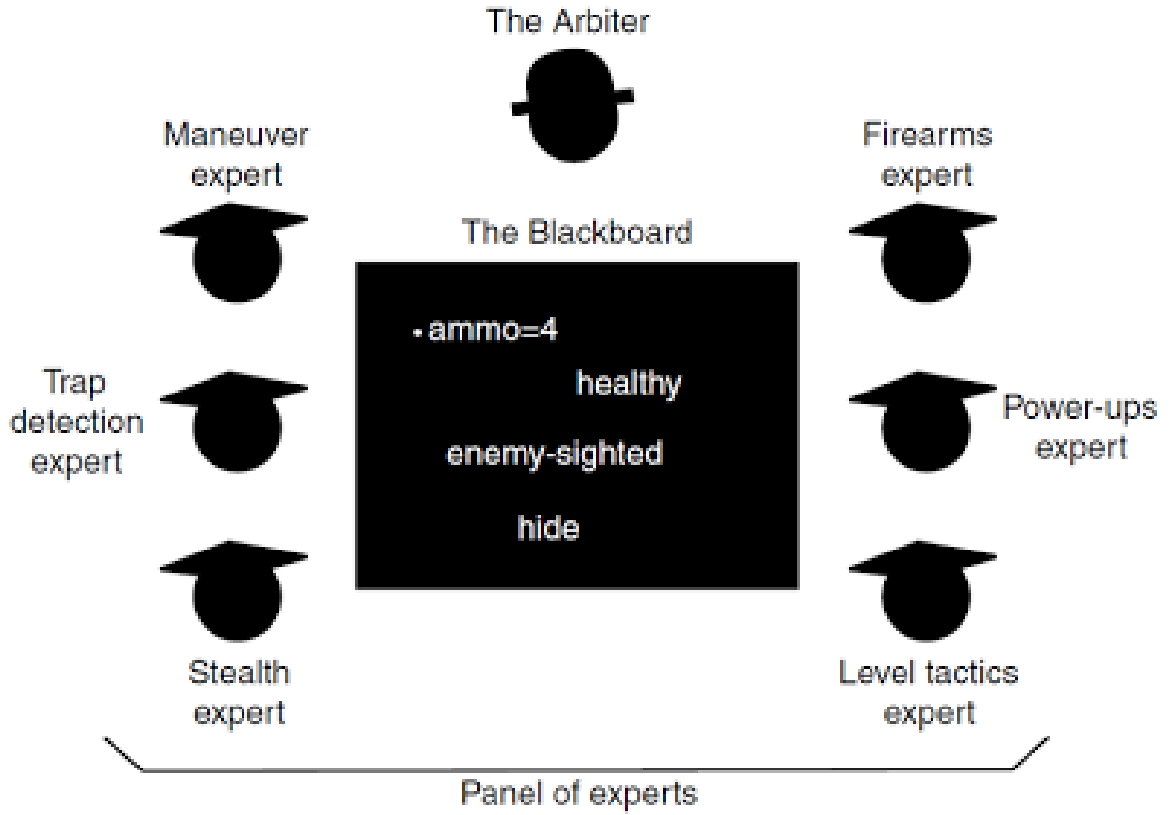
M&F 5.4


→
Enemy visible?
Select enemy
(write to blackboard)
?
Engage enemy
(read from blackboard)
→
Always succeed
High ground available?
Move to high ground

M&F Fig 5.36

Basic BB Architecture

- 3 main parts:
  - Experts
  - BB
  - Arbiter
- Other:
  - Action history
  - Scheduled Actions



Millington & Funge, Figure 5.54

BB Data Format

- Often uses application-specific organization
- Highly domain-dependent
  - 3D locations, maneuver (steering) info
  - FOL strings (flat, hierarchical)
  - Polymorphic data types
- Three typical features:
  - Value (e.g. 3)
  - Type (e.g. float)
  - Semantic Information (e.g. lives remaining)

Information on the BB

- Shared data
- Present task of each expert
- Current state of solution
- Intermediate results
- Next subproblems to be solved
- Requests for help
- Action scheduling

- E.g. Steering:
  - 3D locations
  - combinations of maneuvers
  - animations.
- E.g. Decision making:
  - game state
  - position of enemies or resources
  - internal state of a character
  - targets
  - strategies

BB Arbiter in Control

- Advertises next problems to be solved
- Checks on progress of experts
- Assign pending problems
- Monitor change
  - Polling vs Observer patterns
  - Can notify experts of relevant changes

BB Experts in Control

- On each decision making cycle
  - Experts look at BB, indicate interest (e.g. numeric insistence value)
  - Arbiter selects an expert to have control
  - Expert does work and/or modifies blackboard
  - Expert relinquishes control

BB Uses

- Conflict detection
  - Task level
  - (incompatible) solution level
  - Action level
    - Potential actions, along with a set of agreement flags
- Task sharing
- Result / information sharing
  - Includes both partial and complete results

Is a BB?

- RBS?
  - Experts: rules
  - BB: Facts DB
  - Arbiter: which rule(s) to fire
- FSMs?
  - Subset of RBS
  - Experts: transitions (rewrite state)
  - BB: current state + related info
  - Arbiter: which transition(s) to fire

BB Pros and Cons

- Pro:
  - Flexible, allowing for comm. + coop.; (*n* bb's)
  - Independent of cooperation strategy
  - Does not restrict internal structure of agent
- Con
  - Management code
  - Complicated data structures
  - Centralized structure (single point of failure)
  - System bottleneck
- Have a bad rep among game+academic AI. But they're used anyway, and "shall not be named"