Update on Pedagogical Agents

W. Lewis Johnson

Project participants: Richard Angros, Andrew Marshall, Jeff Rickel, Behnam Salemi, Erin Shaw, Marcus Thiebaux

Sponsors: ONR, AF Armstrong Lab, USC/ISI, USC Med Ctr.

http://www.isi.edu/isd/etg.html

Investigations

- Pedagogical agents as collaborative tutors
  - share virtual environment with students
  - act as guides, mentors, co-workers
  - exploit shared environment for learning
- Communication with embodied agents
  - mix verbal and nonverbal communication
- Authoring task knowledge
  - via demonstrations, experimentation, instructional dialog
- Team training
  - agents acting as team members
  - agents providing individual training in team context
- Combining intelligence + believability

Our Agents

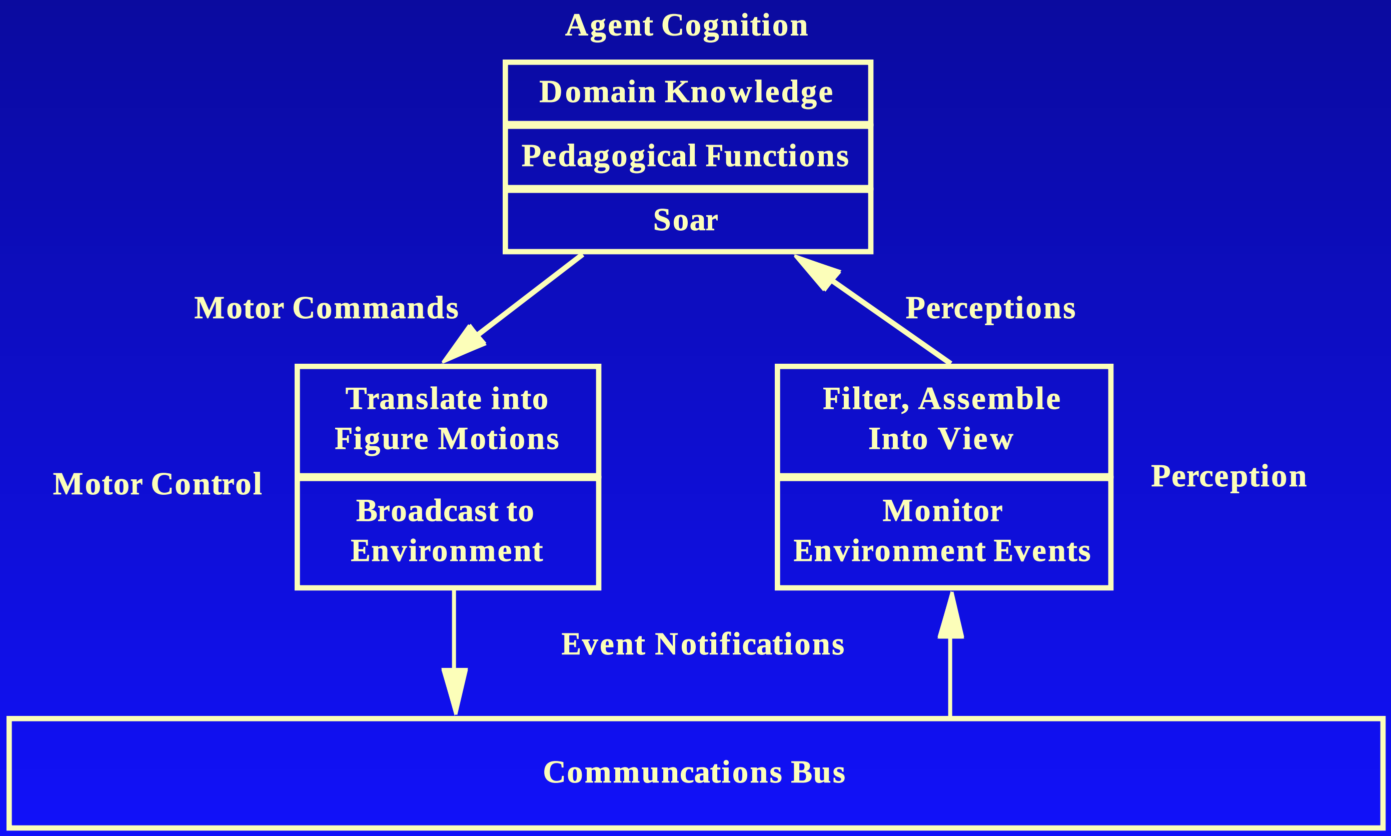
- **Steve (Soar Training Expert for Virtual Environments)**
  - research prototype
  - runs in networked virtual environments
  - application: navy equipment operations training
- **Adele (Agent for Distance Learning Environments)**
  - simplified, Java-based implementation
  - designed for Web-based execution
  - application: medicine
- **Coming: Steve II**
  - application: Air Force equipment operations

Virtual Environments for Training

Points in Steve Video

- **Agent embodiment**
- **Attention to, reaction to the environment**
- **Communication using gaze, gestures, language**
- **Attention to dialog and task context**
- **Sensitivity to human’s placement, orientation**
- **Dynamic replanning as situation changes**
- **Ability to explain, guide actions**
- **Easy switching between pedagogical modes**
- **Ability to operate in team settings**

Steve’s Architecture



What Steve Perceives

- State of the world (objects and their attributes)
- Actions taken by students or agents
- Students’ position and line of sight
- Students’ field of view
- Time
- Human speech (work in progress)

Types of Motor Actions

- Point at an object
- Look at an object (change gaze)
- Manipulate an object
    - Press a button
    - Open/close a valve
    - Flip a switch
    - Pull out or insert a dipstick
- Speak

Task Representation

- Domain tasks are represented as hierarchical plans
- A plan consists of the following:
  - steps (primitive or complex)
  - causal links
  - ordering constraints
- Primitive actions are instantiated from a library
  - e.g., push a button

Task Representation (example)

- **Plan: functional-check**
- **Steps:**

  press-function-test

  check-alarm-light

  extinguish-alarm
- **Causal links:**

  press-function-test achieves test-mode for check-alarm-light

  check-alarm-light achieves know-whether-alarm-functional for end-task

  extinguish-alarm achieves alarm-off for end-task
- **Ordering:**

  check-alarm-light before extinguish-alarm

Task Representation (example)

- **Primitive plan step (manipulation):**

  name: press-function-test
  type: press-button
  button: function-test

- **Primitive plan step (sensing action):**

  name: check-alarm-light
  type: sense
  attribute: alarm-light ^state

- **Goal:**

  name: test-mode
  condition: (<p> ^hpac-mode test)

- **Goal:**

  name: know-whether-alarm-functional
  condition: (<m> ^check-alarm-light-result)

Demonstration: Approach

- **Plan Construction**
  - simple top-down task decomposition
- **Plan Execution (repeat the following until done)**
  - Choose next action
    - determine relevant subset of plan
    - choose next step in that subset
  - Execute action
    - explain step using speech, gestures, gaze
    - output action command and monitor results

Student Monitoring: Approach

- **Basic approach is the same as demonstration**
  - **construct plan**
  - **repeatedly determine relevant subset of plan**
  - **task is complete when end goals are satisfied**
  - **suggest or demonstrate next action when requested**
- **Main difference between monitoring and demonstration:**
  - **Steve only performs sensing actions**

Explanation: Why

- n **Supporting knowledge:**
  - causal links
  - relevant subset of plan
- n **Explanation strategy:**
  - step through relevant causal links using follow-up questions
- n **Answering questions after demonstration:**
  - Debrief maintains episodic memory
  - Debrief recalls situation in which action was performed
  - Steve generates explanation (as above)

Modeling Collaborative Dialogue

- **Cue phrases, e.g.**

  First, …

  Now we can …

  Oh, I need to …

- **Rhetorical pauses**
- **Use of gaze**
- **Focus stack (designed)**

Authoring by Demonstration

- **Author demonstrates task**
  - **perform actions**
  - **describe actions**
- **Agent experiments with task**
  - **try variants to uncover dependencies**
- **Agent discusses task with author**
  - **verify inductive hypotheses**
- **Current issues:**
  - **hierarchical procedures**
  - **sensing actions**
  - **examples in “curriculum sequence”**

Team Training

- **First objectives:**

  individual tutoring in teamcontext

  replacing team members with agents

- **Extensions to task representation**

  roles and responsibilities

- **Revise explanation and guidance in multi-agent context**

  explanations are sensitive to responsibilities

- **Communication among team members**

  request and inform speech acts

  nonverbal communication

New Agent Architecture Directions

- **Engineering goals:**
  - extensible and modular architecture, not monolithic
  - reduction in layers of interpretation
  - optimization for learning applications
- **Step I: Adele**
  - Java-based micro version of Steve
  - Soar-like capabilities: DC, WM, operators
  - designed to interact with simulation, user interface
- **Step II: Steve II (being designed)**
  - interface to external knowledge sources
  - simple authoring language
  - architectural choice: superset of Adele or subset of Soar

Nuggets

- **Golden nugget:**
  - **Progress in natural, situated multi-agent interaction**
  - **Quote from Ed Feigenbaum: “Wow, this is significant.”**
- **Hard lump:**
  - **Believability enhancing behaviors extremely important**