

Visual Soar

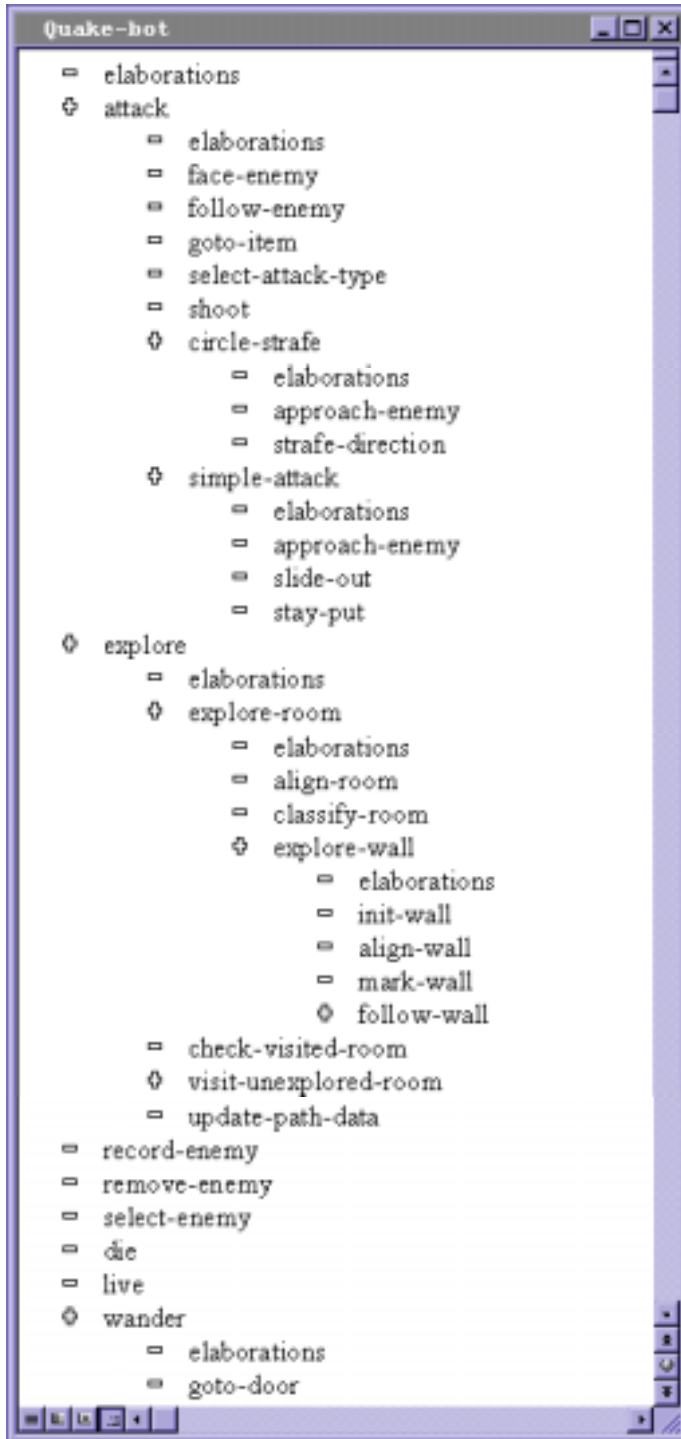
John E. Laird
University of Michigan

Motivation

- It is too easy to make mistakes writing Soar programs.
 - Spelling errors on attributes and values
 - Attributes on wrong objects
- It is too slow to write Soar programs
 - Creating operator hierarchies is cumbersome
- Need improved runtime debugging support
 - Put off for second version

Basic Design

- Editor inspired by Visual C++, TAQL, ViSoar
- Three editor windows
 - Operator Hierarchy Editor
 - Directly support task decomposition
 - State Map Editor [multiple]
 - Add “strong” typing during development
 - Rule Editor [multiple]
 - Semi-structured



Operator Map

- Displays hierarchical structure of operators
 - Automatically creates underlying folders & files
- Actions to operators:
 - Add, Move, Remove, Change Name
 - Create Alias
 - Select
 - Changes view in state and rule windows

```
Data Map: Align-room
- name explore-room
+ operator
  - name align-room
  - direction north south east west
+ superstate
+ io
  + input-link
    + item
      - classname [symbol]
      - range [float]
      + angle-off
      + sensor
        - visible [boolean]
        - infront [boolean]
      + origin
    + entity
      - name [symbol]
      - classname [symbol]
      - health [integer]
      + velocity
      - range [integer]
      + origin
  + output-link
    - thrust forward backward off
    - sidestep left off right
    - face [float]
    - speed on off
+ top-state
  + map
    - current-room <room>
    - current-door <door>
    + room <room>
      - explored yes
      - type hall room
      - initialized east-west north-south
      + wall
        - side north south east west
        - x [integer]
        - y [integer]
        + door <door>
  + path
  + item
```

State Map

- Displays structure of current state
 - Provides access to superstate and top-state
 - Supports semantic error checking
 - Supports fast rule creation.
- Actions to states:
 - Add, Move, Remove, Change
 - Set value type and range
 - Create pointer to existing structure

```
rule window align-window

sp {explore-room*propose*align-room
  (state <s> ^name explore-room)
  -->
  (<s> ^operator <o> + =)
  (<o> ^name align-room
    ^direction )}

sp {explore-room*apply*align-room
  (state <s> ^operator <o>)
  (<o> ^name align-room
    ^direction <dir>)
  -->
}
```

Rule Map

- Displays operator rules
- Full text editing
- Partially filled in templates using operator information
- Real-time syntax and semantic checks
- Point-and-click addition of state structure

Plans

- Develop prototype editor this summer using Java
 - Use JFC for many of the components
 - Jon Bauman, Brad Jones