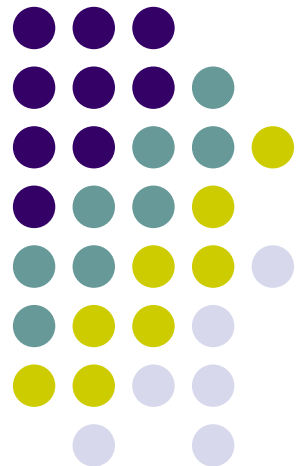
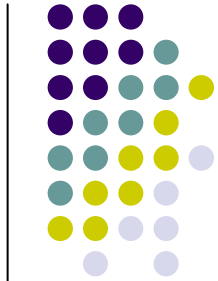
Haunt 2 Soar Bots

By Mike Parent

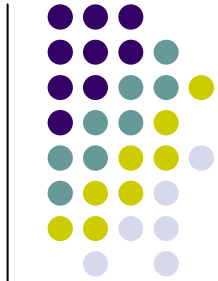


Overview



- Introduction
- Behaviors
- Implementation
- Wrap-up
- Future

Haunt 2 Motivation



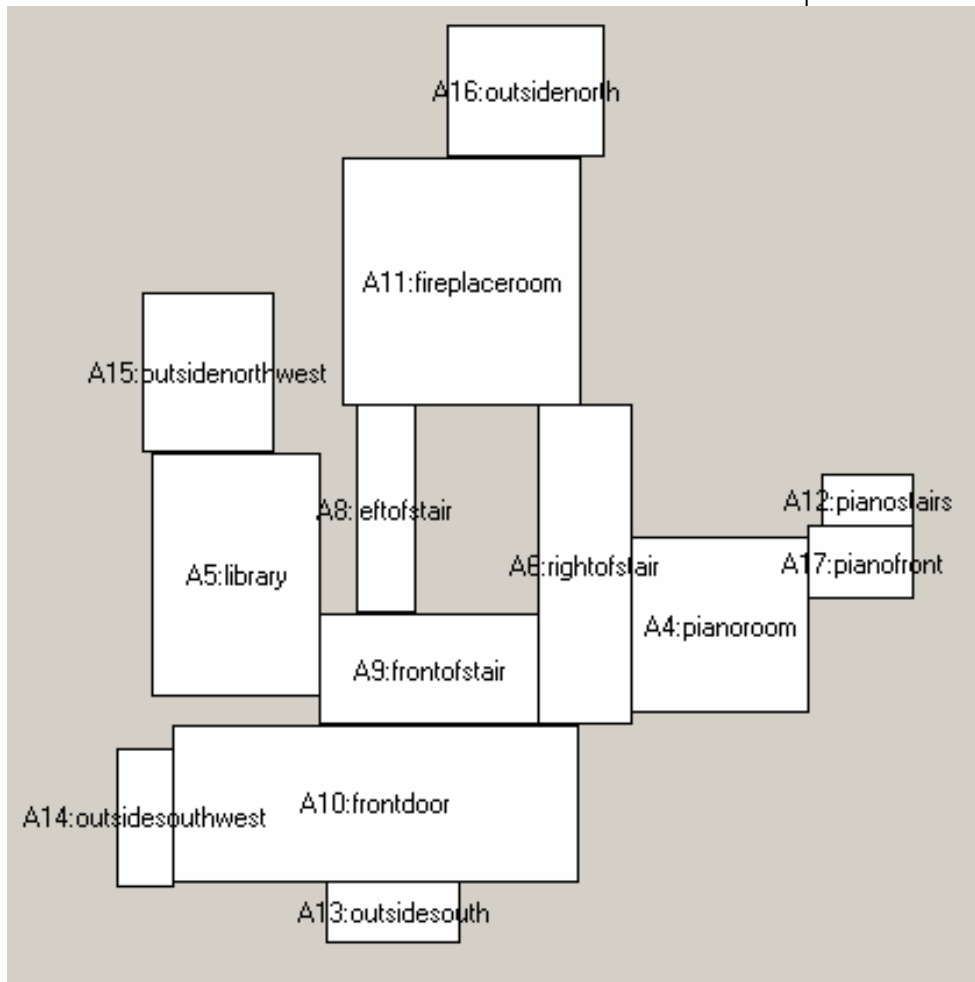
- Provide a complex environment in which to develop autonomous agents exhibiting human-like behavior

Introduction

- Haunt 2
  - MOD of Unreal Tournament
    - Alex Kerfoot – Unrealscript
  - Haunted mansion
  - Player controls a “ghost”
  - Independent Soar bots
  - Player will attempt to manipulate bots

Haunt 2 Environment

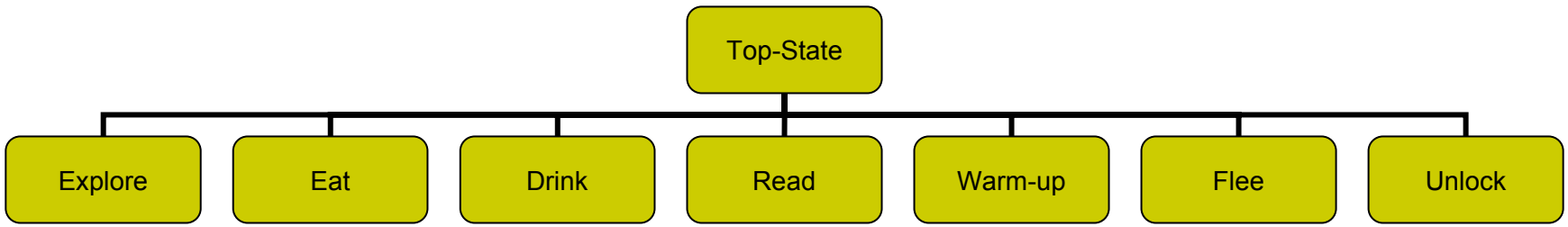
- Haunted Mansion
- 1 floor, 13 rooms
- Objects
  - Doors
  - Keys
  - Books
  - Food/Drinks
  - Boxes
  - Matches
  - Fireplaces



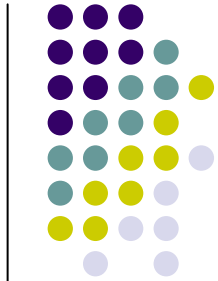
Soar Bots

- Input-link contains all pertinent world info
- Uses MOUT movement productions
- Desired Behaviors:
  - Fulfill Needs (hunger, thirst, cold)
  - Explore environment
  - React to player

Behavior Goal Hierarchy



Implementing Behaviors



1. Classify available objects
2. Determine status/needs
3. Propose & select actions
4. Break high-level task into sub-goals

Classification: Ontology

- Pre-existing world knowledge
- Used to classify objects

^ontology

^height-source

^class <WoodenBox>

^food-source

^class <Banana> ^value 10 ^must-cook no

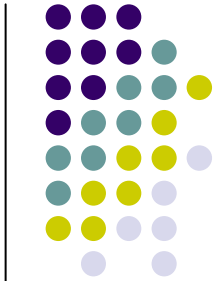
^food-source

^drink-source

^heat-source

…

Classification: Available Objects



- All world objects stored with o-support
- Available objects elaborated according to ontology and accessibility
- Used to propose behavior operators

^available

^food-source

^name

^location

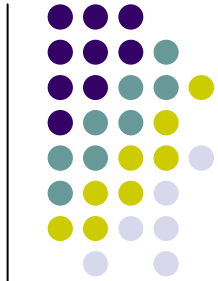
…

^food-source

^heat-source

…

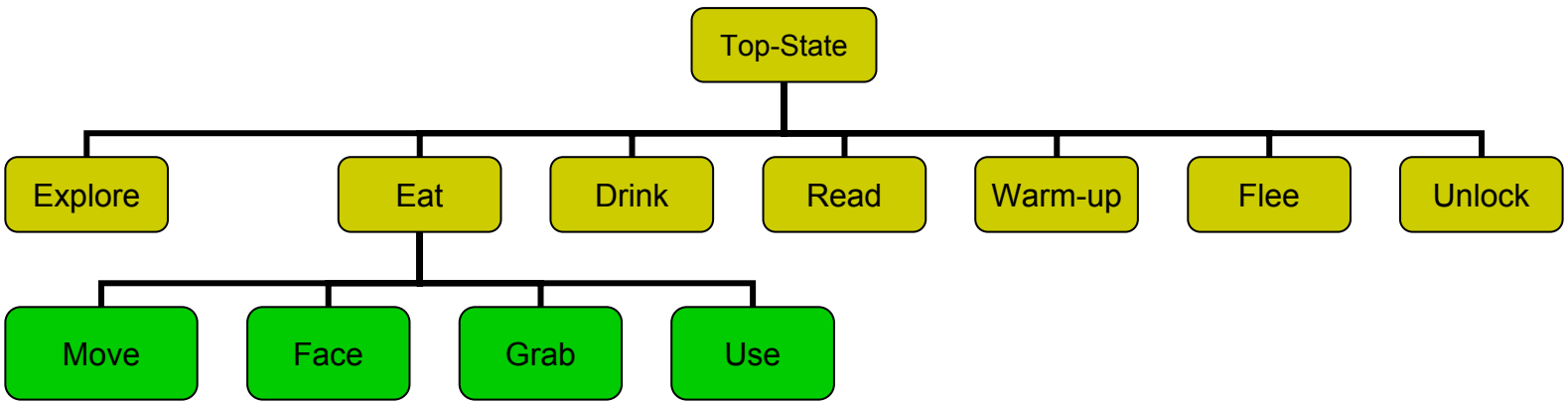
Status



- Determine needs of agent
  - Thirst
  - Hunger
  - Temperature
  - Fear
  - Fatigue (future)
  - Exertion (future)
- Group numeric values into priority categories
  - Example:
    - Starving: drop everything and find food
    - Mild hunger: eat if available

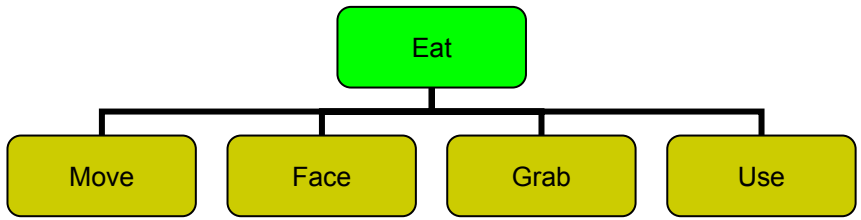
Proposal & Selection

- Select operators based on need & availability
- Preferences based on range, needs & action type
  - Higher: eating, drinking, fleeing
  - Lower: exploring, reading



Output-Link Actions

- High-level commands composed of primitive actions
  - Grab
  - Use / Use-With
  - Throw
  - Drop
  - Move (coordinate)
  - Face (direction)
  - Jump

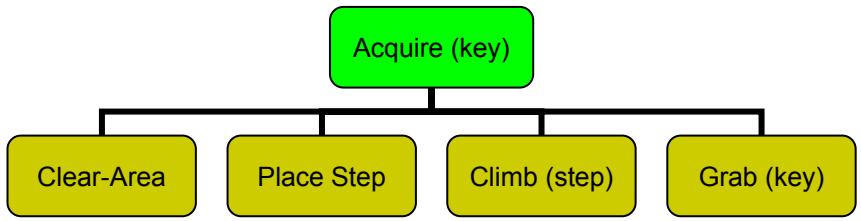


Example: Monkey & Bananas

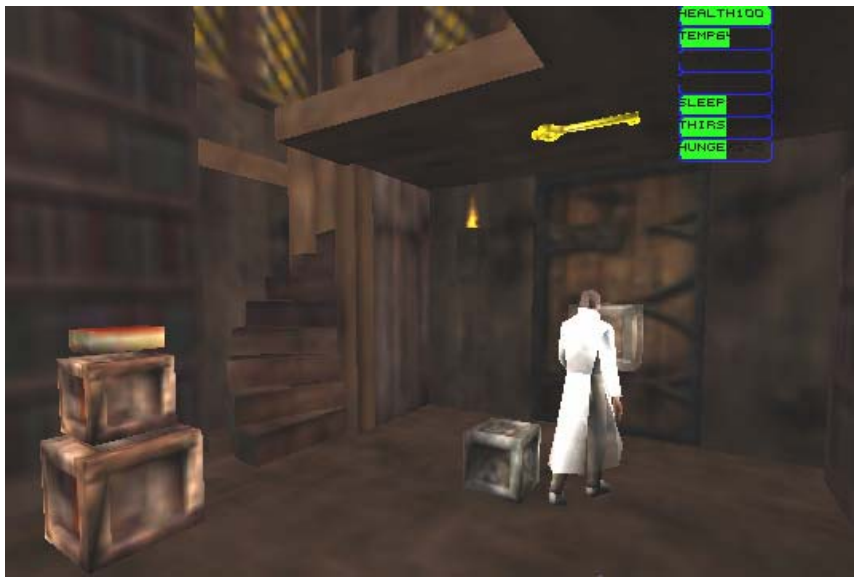
^available

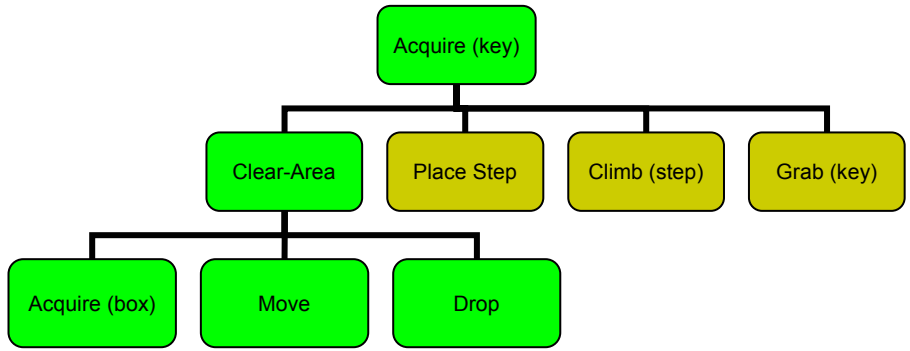
^unlock-tool <key>

^height-source <step>

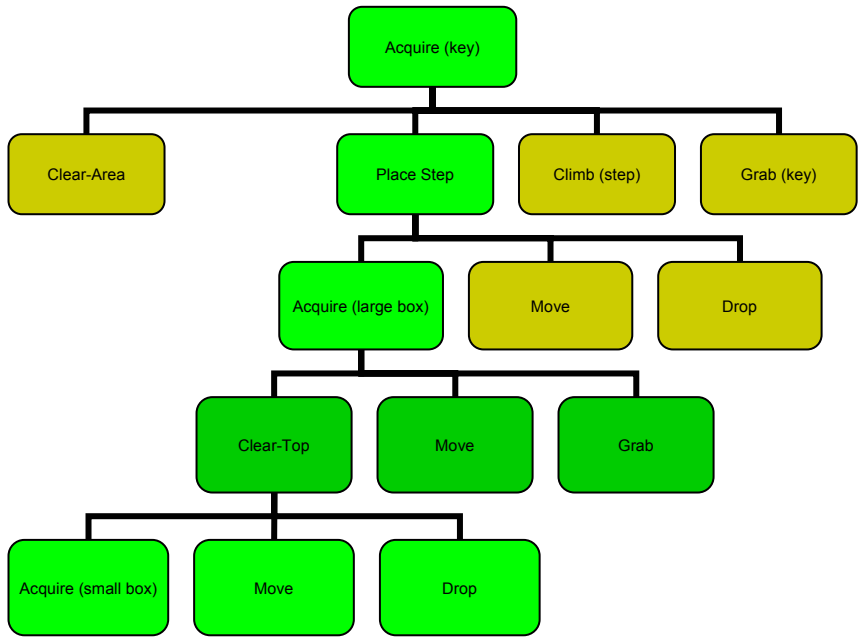


Step 1: Clear Area Below

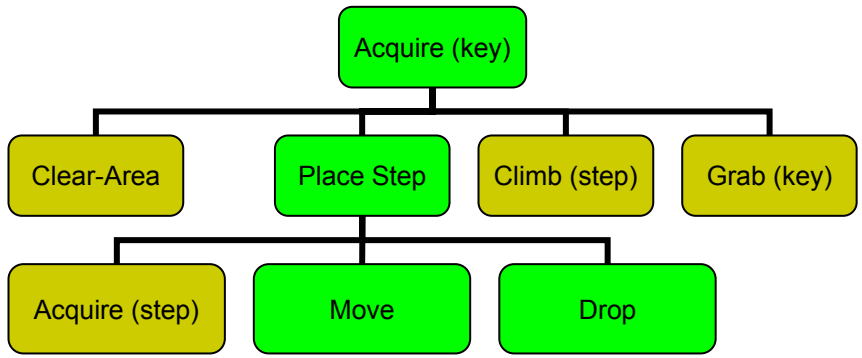




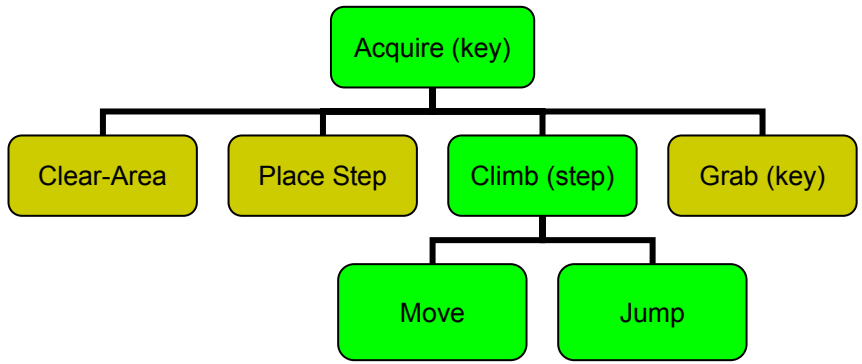
Step 2: Clear Box



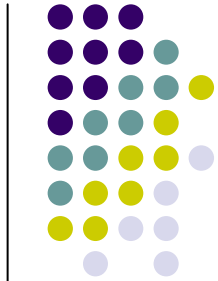
Step 3: Position Box



Step 4-5: Jump and Grab



Wrap-up



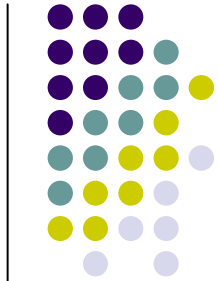
Nuggets

- Framework for implementing complex behaviors
- Easy to add new behaviors
- Behaviors are modular, and share sub-goal productions

Coal

- May require many calculations to decide between a large number of available behaviors
- More complex actions may require deep sub-goal trees

The Future



- Short-term
  - Bigger map, multiple floors
  - Possession
    - Player can directly influence a bot's behavior
    - "Scared" agents will reject possession
  - Reactions
    - Investigate sounds and rearranged objects
- Long-term
  - Interactive Fiction (Brian Magerko)
  - More realistic physiological and emotional models

Questions?