Isis World Simulator

The IsisWorld simulator is available to researchers for building and evaluating problem solving and language learning systems with everyday problems that take place in a kitchen. We aim to use IsisWorld to simulate everyday commonsense reasoning problems that span many realms, such as the social, visiual, kinesthetic, physical, spatial and mental.

What is a problem "realm"? Consider the problem of *hailing a taxi*. You could represent and reason about this problem in several different ways.

1. **Temporally**: Wait for a cab. Maybe if you stay put, a taxi will drive by
2. **Spatially**: Find a cab. You must eliminate the distance between yourself and a taxi.
3. **Socially**: Call a cab. Communicate your position to a distpatching agent and an available taxi will come to you.

It is this resourcefulness—having many ways to solve a problem—that allows human problem solvers to flexibily adapt to many problem solving situations. A system that lacks these abilities is *brittle*.

Further, we are looking for test-bed to study the problems of meta-reasoning: where a super-level planning system reasons about the world of a sub-planning system. Returning to a taxi example, we could consider the failure mode which causes a meta-level reasoner to step in and change the state of the planner. For example, it could ask the system to *elevate* the problem description to pursue the parent goal: *instead of "searching for a taxi" reconsider the problem as "traveling to your destination" and pursue other options: e.g., walking, train, asking a friend etc.*

More detailed arguments about using a simulator for studying AI and the choice to use kitchen problem domain is explain in this paper:

- An open source commonsense simulator for AI researchers. Dustin Smith and Bo Morgan. *Submitted to AAAI–10 Workshop on Metacognition.*

Use cases

The development of the simulator is focused on the following use cases.

1. Toast Making

Ralph is in the kitchen. Ralph has to "use" the knife to cut the bread, and then put the bread in the toaster, to make toast.

2. Knife sharing

Ralph and his mother Sue are in the kitchen. Sue is currently using the only knife. Ralph has to ask Sue to use the knife. If he grabs the knife from her hand, he will be cut.

3. Learning by observation

Sue is communicating a new sequence of actions. Ralph must identify Sue's plan and then recognize her goal as different, and then learn the new plan as some deviation of the existing plan (e.g., making toast with jelly)

4. Imprimer learning

Sue is teaching Ralph how *not* to use a kitchen. He must learn that the faucet must be turned off after being used, doors closed after they are opened, not to leave the refrigerator open for more time than necessary, etc. He must learn these how to represent and pursue these imagined goals and antigoals of his imprimer.

5. Language learning

Learning the labels of objects from example. Learning how to lable actions from descriptions at different temporal resolutions.

6. Carrying out instructions

Learning the proper sequence to carry out a sequence of actions from a linguistic description

7. Communicating instructions

Ralph describes his actions or Sue's actions as an English verb phrase.

Improvements to the Simulator

Over the summer of 2010, we plan to make many significant improvements to IsisWorld. These projects are:

Implementation of Physics

Angular and linear forces will be applied, and densities of objects will be represented by a Physical simulator. Because ODE integration in Panda3D is still preliminary [#1, #2], and commitment to open source principles has eliminated NVidia's PhysX platform as a viable option, our only current option is to use Panda3D's built-in physics support and collision detection. Some decent tutorials exist [#1 #2], and for ODE, when Panda3D's support becomes more robust [#1].

Design ideas

Changing physics engines should be as easy as switching the import statement in `simulator/physics.py`.

- Make the physics handling modular, accessible through the `IsisWorld.worldManager` variable
- Agents, Objects, and Environment items are initialized separately:
  - **Agents**: Are sub-classed instances, defined in `simulator/ralph.py` of `PhysicsCharacterController` class, which has an `updatePhysics` method. Geometrically, these are represented as capped cylinders, as TriMeshes are too computationally expensive.
  - **Objects**: In ODE, there are two types of objects: kinematic (non-self propelled) and dynamic (capable of self-motion). Most objects can be represented as a boundedBox, although some are best represented as cylinder or sphere.
    - Objects can be added through `worldManager.addObjectToPhysics`
  - **Environment**: The ground is represented as an infinite plane. Should represent walls of house as blocks, not TriMesh.
- Gravity is represented as a linear force downward

Initialization scripts for designing and loading environments

As mentioned in the position paper, we want environments to be *generated* from a space of possible dimensions. Most of the variable properties of the items (size, location, plurality, color, state) could be left to future work. Default locations can be specified in a configuration file corresponding to prepositions: - X on Y in Z: "on" and "in" are less descriptive than 3D (relative) coordinates

This will require a general module for describing and loading components, a large range of visual, physical, linguistic, spatial (default locations), and functions (properties, methods to modify properties) for each object.

- **Physical**: shape (for Physical collision mask), dynamic or kinesthetic
- **Visual**: visibility, color mask, transparency level, scale size (of model)
- **Spatial**: default orientation of model, default locations (as represented in abstract descriptions: "in kitchen on table")
- **Functional**: use a commonsense **type system** and inheritance structure for **attributes** and **values** and their **default value** (e.g., `{'is_on': {'domain': [True,False], 'defa` and **event listeners** to change the properties of these actions, some of which can affect the other kinds of item properties (physical), etc.

Design ideas

- Read `kitchen.isis` world in, creating a labeled graph structure: "X on Y" becomes: *graph.addEdge(x,y,label='on')*
- With root node "kitchen", topologically sort all items. For each item:
  - Visually, add models to their parent renderer `attachNodeWrtParent()`
  - Set default properties of the item
  - Physically, register item within physics handler

Extending corpus of models

Lots that can possibly be added. Stay focused on the kitchen and the use cases (make toast)

- Can add models from Google's 3D warehouse, export them to **egg** files. Instructions: #1
- Blender models can be exported using Chicken.
- Alice I saw a list of these somewhere that were already in the egg file format. Some of these already have animation methods!
- List of game models
- (not? immediately useful?) list of resources

Event Handling in simulated world

In addition to items being able to re-act to actions of the character, events can be initiated by states of the world. For example, when a toaster "is_on" and "contains_item", then after 5 minutes, it changes the state of the item inside to "burned", darkens the color, and turns itself off.

Design Ideas

Use the FSM of Panda3D to do this. Possible approach: using Honda's OMICS corpus

Kitchen use-cases

Several tasks for evaluating intelligent agents in IsisWorld.

Creating Toast in the Kitchen

Other Lower Priority Improvements

- Configuration parameters to disable non-essential, CPU intensive visual effects, like the clouds in the sky.
- Replace Ralph with nicer model(s)
- Obtain copyright information for all models
- Multi-client implementation
- Improving granularity of actions: kinesthetic grasp, move items between arms, damage to body depending on forces
- Adding "scale" parameter to existing perceptual controls
- Finer resolution perceptual/motor controls.
- Adding rotation-based animation methods to graphics

Resources for Developers

Here is a list of resources for developers that are getting started working with IsisWorld.

Panda3d.org has a really good manual, though it doesn't have 100% coverage of all the features in the API. Also the Panda3D forum is very valuable resource.

To learn about Git, Dustin recommends GitHub's videos.