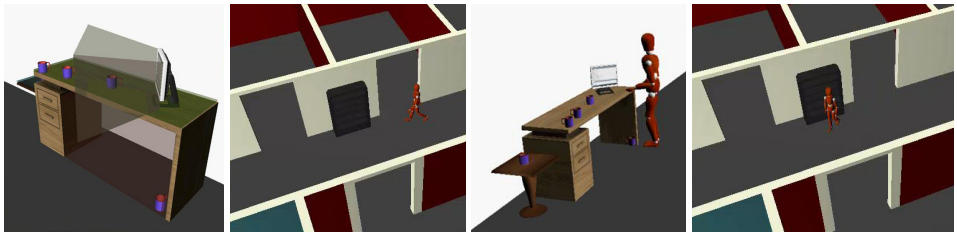
Modeling Agent Determination of Spatial Relationships

John Mooney*, Rohan Gandhi†, Jan M. Allbeck‡
Games and Intelligent Animation Laboratory
George Mason University



1 Introduction

There is an ongoing desire to make virtual humans a more accessible tool for use in entertainment, training, and evaluations. From the graphical level to the animation level to the intelligence level, complexities abound. As research progresses some of these complexities become hidden from the end user. Ultimately, we would like to treat agents as real humans and instruct them as you might another person. Here we present a framework, inspired by natural language constructs, that aims to obfuscate the complexities and allow users to control virtual humans through structured English input. Our focus is on object and environment interactions, particularly spatial relationships.

Natural language instructions contain a lot of implicit understandings that we take for granted. For example, knowing what pickup means, knowing what a table is, and knowing what is meant by the top of a table. For an instruction like *set the cup on the table*, top is understood to be the appropriate location. These implicit understandings do not exist in virtual humans and therefore must be constructed with representations and processes. Furthermore, virtual humans should follow instructions, by comparing the context outlined in the instructions to what they encounter in the virtual world. In addition to spatial relations, our instructional system utilizes object types, stored in a hierarchy, as well as object properties, such as color.

2 Spatial Data

For determining a large set of spatial relationships, we developed a 3D modeling tool for tagging objects with spatial data. Our Autodesk Maya plug-in allows a modeler to label a selected subset of faces with spatial data such as *top*, *under*, or *front*. Given a selection of faces, we automatically construct a tangent plane approximating the selection. The modeler is able to transform the constructed plane relative to the object, give the approximated region a unique name, and then export the data (points and normal) for use within our virtual world. Each object within the virtual world contains a set of spatial regions that are transformed with them.

We then determine a spatial relationship, such as *Cup on top of the table*, according to the following procedure:

1. Identify the spatial region by mapping the relation name (*Top*) from the direct object's (*Table*) set of spatial regions.
2. Perform a collision test between the base object's (*Cup*) bounding volume and the volume defined by the spatial region.
3. If the objects are in collision, then the relation is true. Else, the relation is false.

When a spatial command is issued as structured English imperative, it is parsed and a series of filters are applied. These filters determine a final set of object participants for the action. The structured English we have constructed presents four main components that we use to narrow the selection: *Object Bases*, *Qualifiers*, *Locators* (i.e. spatial relations), and *Quantifiers*. Each component acts as input into an object filter, where a filter receives a set of source objects and returns the resulting set matching a specific criteria.

3 Navigational Directives

Instructions that use spatial relationships to direct an agent through an environment to a goal location may include multiple segments where multiple objects and pathways could fit. For example, an instruction might say *leave the room, go through the door near the bookcase, go to the end of the hallway, into the room in front of you, go to the middle of the hallway, go through the door near the left bookcase, and stand near the crates* and there may be two doorways near the bookcase. In this case, the agent needs to attempt one option and see if the subsequent directives lead to the goal. If they do not, the agent has to backtrack and try the other option.

We implement this behavior by maintaining a depth first search tree of possible solutions to the instructions. At a particular node in the tree, the agent will evaluate the next instruction and add the solution nodes as children to the current node. The agent then proceeds to evaluate the first child of the current node with the next instruction in the instruction set. If the instruction does not fetch any solutions then the agent backtracks and evaluates the current node's sibling using the same instruction. The greatest possible height of the tree is bounded by the number of instructions. If a node has a height equal to the number of instructions then the goal has been reached. In other words, if we run out of instructions to process we have reached the goal.

*e-mail:jmooney3@gmu.edu
†email:rgandhi2@gmu.edu
‡jallbeck@gmu.edu