A Knowledge-Based Agent for CAD systems

Olivier St-Cyr, Yves Lespérance, and Wolfgang Stuerzlinger

Department of Computer Science, York University
4700 Keele Street, Toronto, Ontario, Canada M3J 1P3
{olivier, lesperan, wolfgang}@cs.yorku.ca
http://www.cs.yorku.ca/~{olivier, lesperan, wolfgang}

Introduction

This poster describes an approach in improving the usability of computer-aided design (CAD) applications by adding an intelligent agent that assists the user in his/her interaction with the system. To implement the agent, I used ConGolog [GLL97, LLR99] – a very high-level programming language for developing knowledge-based agents that are embedded in complex environments. ConGolog supports the specification of a task-level model of a dynamic environment, the description of complex behaviours, and the synthesis of new plans at run-time. A prototype intelligent agent is being developed to work with an existing 3D CAD system [GS99]. This agent is intended to help the user in designing an office layout that satisfies his goals.

The CAD system [GS99] that our intelligent agent works with is built to allow the user to design a 3D virtual environment (office, kitchen, living room, etc.). The system's graphical user interface is quite simple. Interactions consist primarily of using the mouse to pick and place various types of objects (desk, chair, lamp, inkwell, etc.) in the room layout. In this, it resembles the Object Associations system [BS95]. The system handles the details of interactions based on a model of objects and the physical constraints that hold in the scene, for instance, an object being supported by a particular surface. But the system lacks a model of the user, of the task that he/she is trying to perform, and of the objectives that he/she is trying to achieve. It cannot really assist the user in quickly creating the desired room layout.

An early example for a system that attempts to aid the user in creating a room layout is CRACK [FM88]. This 2D system critiques the current design with text messages that explain the problem. The domain knowledge embedded in the critiquing system is not used to actively aid the user for placing objects.

It is believed that the use of intelligent agent technology can provide many benefits in the area of layout systems. An agent would maintain a high-level representation of the application domain, including object behaviour and user knowledge and goals. This could be used to enforce complex application specific constraints on the way objects are manipulated and on the layouts that are produced. Secondly, such a model could be used for disambiguation and consistency checking. Humans often communicate information about a task very inaccurately because they understand the context of the task and its goals from previous experience. An agent could use its domain knowledge to resolve ambiguities, as well as ask more meaningful questions when user input is required. Moreover, the user goal model could be exploited to detect inadvertent errors. Thirdly, the agent could also aid the user in constructing the virtual design using its knowledge of the domain and user goals. Because it is aware of the current state of the design, it can provide suggestions and advice to the user, guide him through the task, and respond to user's questions. All this would lead towards much more natural and intuitive interaction between the user and the CAD system.

The CAD system

Most of today's CAD systems are well suited in creating geometric objects. Nevertheless, users find common tasks, such as quickly furnishing a room, hard to accomplish, especially since placing objects is conceptually different from creating them. The IConS CAD system [GS99] used in this project is a recently developed 3D application that exploits knowledge about the behaviour of objects to provide simple and intuitive interaction techniques for object placement and manipulation. Objects are represented using polygonal models. The application uses a simple two-dimensional interface. The main interaction device is the mouse; keyboard commands are used only for switching modes and organizational tasks. Currently, three modes exist: constrained object movement, unconstrained object movement, and viewer navigation (see [GS99] for details).

A user of this system builds scenes based on a predefined library of objects. For each object, two sets of areas are defined: *offer areas* and *binding areas*. These areas are bound together by constraints and thus, limit the positioning of the constrained object during manipulation. Collision detection/avoidance is also used to ensure that no two objects occupy the same space. These principles (surface constraints and collision detection/avoidance) capture part of the natural behaviour of objects in the system.

General geometric constraints are already part of the IConS system. For example, *OnFloor* and *OnWall* are two of the system's constraints that can limit where an object can be placed. All the implemented constraints describe general placement guidelines for objects that can be used in any design; they are not specific to the application domain. This is where a knowledge-based agent could be useful; it could know for instance, that a computer should not be placed too close to a heat source.

The knowledge-based agent

Domain representation

In the office layout domain, there are many types of objects. These are organized into a hierarchy of classes, e.g. Desk, InkWell, etc. These object classes are themselves instances of a set of function-related metaclasses:

- WorkspaceAreaObjectClass, which includes Desk, Chair, SideTable, etc.
- MeetingAreaObjectClass, which includes MeetingTable, Chair, WhiteBoard, etc.
- StorageAreaObjectClass, which includes Bookshelf, Book, FilingCabinet, etc.
- OfficeEquipmentAreaObjectClass, which includes FaxMachine, Printer, Photocopier, etc.

Spatial relation types (e.g. OnTop, On Floor, OnWorkspace, etc) are also grouped in a hierarchy. There are primitive actions for creating and destroying an instance of an object or relation class.

Another set of fluents and primitive actions is used to represent interactions with the user or the CAD module. For example, the agent can perform the primitive action makeYesNoQuery(msg) and a possible response of the user is represented by the primitive action answerYesNoQueryYes. These declarations are used by ConGolog to initialize the agent's knowledge base, update it when actions occur, and check the legality of actions in a given state.

Agent behaviour specification

So far, the behaviours that have been scripted for the agent are rather simple. For example, the procedure, which interacts with the user to help him/her set up the workspace area of his/her office layout, goes as follows:

```
proc setUpWorkSpace(officeLayout) [
      addObj(Desk,officeLayout);
      addObj(Chair,officeLayout);
      resetYesNoQuery;
      makeYesNoQuery("Would you
      like to have one more chair
      added to your office?");
      yesNoQueryRespIn?;
      if yesNoQueryAnsYes then
      addObj(Chair,officeLayout));
endProc
```

Another area where ConGolog helps is in detecting errors and constraint violations. The agent can easily check the legality of the actions requested by the user before performing them. ConGolog's plan synthesis facilities could also be useful for dealing with unanticipated user requests, recovering from failures, or producing animations.

Acknowledgements

I would like to thank Yves Lespérance and Wolfgang Stuerzlinger for their thoughtful comments and discussions of the ideas discussed herein.

References

[BS95] Bukowski, R., & Sequin, C. (1995). *Object Associations: A Simple and Practical Approach to Virtual 3D Manipulation*. Proceedings of the ACM Symposium on Interactive 3D Graphics '95, 131-138. Monterey, CA.

[FM88] Fischer, G., & Morch, A. (1988). *CRACK: A Critiquing Approach to Cooperative Kitchen Design*. Proceedings of the International Conference on Intelligent Tutoring Systems, 176-185. Montreal, Canada.

[GLL97] De Giacomo, G., Lespérance, Y., & Levesque, H. J. (1997). *Reasoning about Concurrent Execution, Prioritized Interrupts, and Exogenous Actions in the Situation Calculus*. Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, 1221-1226. Nagoya, Japan.

[GS99] Goesele, M., & Stuerzlinger, W. (1999). *Semantic Constraints for Scene Manipulation*. Proceedings of the Spring Conference in Computer Graphics, 140-146. Budmerice, Slovak Republic.

[LLR99] Lespérance, Y., Levesque, H. J., & Reiter, R. (1999). *A Situation Calculus Approach to Modeling and Programming Agents*. In: Wooldridge, M., & Rao, A., editors, Foundations of Rational Agency, 275-299. Kluwer.