University of Pennsylvania
ScholarlyCommons

Center for Human Modeling and Simulation | Department of Computer & Information Science

January 1995

Planning and Terrain Reasoning

Michael B. Moore
University of Pennsylvania

Barry D. Reich
University of Pennsylvania

Christopher W. Geib
University of Pennsylvania

Follow this and additional works at: http://repository.upenn.edu/hms

Recommended Citation
Moore, M. B., Reich, B. D., & Geib, C. W. (1995). Planning and Terrain Reasoning. Retrieved from http://repository.upenn.edu/hms/77

Presented at the *1995 AAAI symposium*.

This paper is posted at ScholarlyCommons. http://repository.upenn.edu/hms/77
For more information, please contact libraryrepository@pobox.upenn.edu.

Planning and Terrain Reasoning

Abstract
We describe the Zaroff system, a plan-based controller for the player who is "it" in a game of hide and seek. The system features visually realistic human figure animation including realistic human locomotion. We discuss the planner's interaction with a changing environment to which it has limited perceptual access.

Comments
Presented at the *1995 AAAI symposium*.

This conference paper is available at ScholarlyCommons: http://repository.upenn.edu/hms/77

Planning and Terrain Reasoning*

Michael B. Moore Christopher Geib Barry D. Reich

University of Pennsylvania
Department of Computer and Information Science
200 S. 33rd Street, Philadelphia, PA 19104-6389

E-mail: {mmoore,geib,reich}@graphics.cis.upenn.edu

Abstract

We describe the ZAROFF system, a plan-based controller for the player who is "it" in a game of hide and seek. The system features visually realistic human figure animation including realistic human locomotion. We discuss the planner's interaction with a changing environment to which it has limited perceptual access.

Introduction

The game of *hide and seek* challenges the ability of players to plan for acquiring information. The player who is "it" (hereafter called the *seeker*) must explore his environment attempting to locate other players. Those players must select hiding places which are difficult to discover while providing access for them to run safely to home base when the way is clear. The goal of this is to develop simulated agents that can play hide and seek (or more dangerous games). Due to the competitive nature of the game, reasoning must take place quickly in dynamically changing hostile surroundings.

This paper presents part of ZAROFF[1] a system for generating an *animated simulation* of humans playing hide and seek. (Figure 1 shows frames from one game.) We describe a planning system for the seeker and its vertical integration into a system that selects reactive behaviors to execute in an animated simulation. Operation of the planner is interleaved with execution of the reactive behaviors so that the agent may adapt to a dynamic environment. Adaptivity is also supported through least-commitment planning, as the planner only looks ahead one action at each level of its abstraction hierarchy.

The planner described in this system has been ported from an initial domain that combined search and manipulation tasks (Geib, Levison, & Moore 1994) to this new domain which requires locomotion. The software chosen for this work is *Jack*® (Badler, Phillips, & Webber 1993) running on Silicon Graphics workstations. *Jack* is a human modeling and simulation program developed at the Center for Human Modeling and Simulation at the University of Pennsylvania, that features visually realistic human locomotion based on both kinematic and dynamic techniques (Ko 1994). *Jack*'s LISP application programming interface (Becket 1994) is used to implement ZAROFF. This interface supports access to the environment (a database) and its behavioral simulation system.

In ZAROFF, a planner interacts with an animated simulation that provides a dynamically changing environment to which the planner has only limited perceptual access. These environmental characteristics motivate our system architecture.

Interacting with the environment

Our planner interacts with a multi-agent environment that consists of a database of graphical entities (e.g. geometric objects, human figures) and a behavioral simulation engine that moves objects in the database. The database records data on three-dimensional geometric figures with position and orientation. The planner's access to database information is restricted, to better simulate the limits of human perception. The planner's actions either directly manipulate the database, or indirectly affect the database by influencing the behavioral simulation engine.

Perception in ZAROFF

By limiting the planner's access to the database, the planner only has access to information about objects which the seeker can "see". For the planner to decide on an action, it must see all the objects involved in that action.

We implement a model of perception which restricts access to only those objects in the environment which are in a direct line of sight from the seeker. This is approximated using the graphics technique of ray-casting.

*This research is partially supported by ARO DAAL03-89-C-0031 including U.S. Army Research Laboratory; ARPA AASERT DAAH04-94-G-0362; DMSO DAAH04-94-G-0402; ARPA DAMD17-94-J-4486; U.S. Air Force DEPTH through Hughes Missile Systems F33615-91-C-0001; DMSO through the University of Iowa; and NSF CISE CDA88-22719.

[1]Named for the hunter in Richard Connell's 1924 short story, *The Most Dangerous Game*

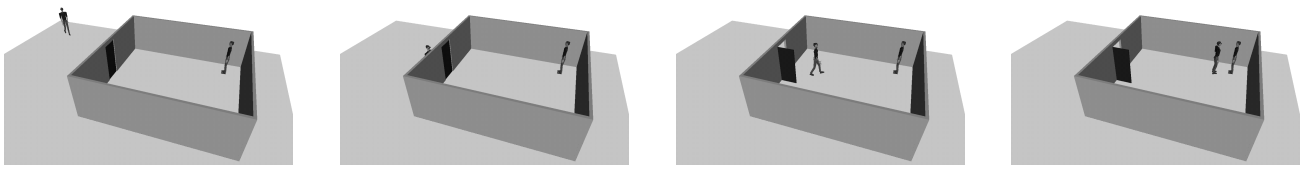

Figure 1: Frames from the conclusion of a game

Rays are cast from the seeker's eyes toward other figures. If any of these rays hit a figure before intersecting another object, that figure is perceivable. While "perception" is not synthetic vision, it satisfies the same role of forcing information-acquisition actions and motivates our use of a special purpose search planner.

This model of perception is less restrictive than that used in HOMER (Vere & Bickmore 1990), another animated simulation, which limited distance (only close objects can be seen) and angle of view (only objects in front can be seen) in a two-dimensional environment.

Action in ZAROFF

Actions chosen by the planner are carried out by an *action execution* module (see Figure 2). Both components are well matched to the dynamic environment in which ZAROFF acts: the planner quickly selects the next action to perform based on comparison between the perceived world state and an incomplete hierarchical plan that is regularly revised. The action execution module controls locomotion in a manner reactive to changes in the terrain and moving objects.

System Architecture

Our division of the control of a seeker between a *planning component* (the general purpose planner and special purpose search planner) and a *reactive behavior component* (the action execution module) reflects a distinction between deliberative and non-deliberative actions. Keeping track of where you are located in a complex environment and what hiding places have been checked requires deliberate effort, while walking from one place to another generally does not. Together, these two components create realistic animations of human decision-making and locomotion while playing hide and seek.

Figure 2 depicts information flow in ZAROFF. The system starts by initializing the plan with the input goal (finding a hiding human), populating the database with the initial locations of all the objects and human figures in the simulation, and creating a partial map from what the seeker can see around him. The planner and the Behavioral Simulation System start processing simultaneously. The planner queries the state of the database through the Filtered Perception module to decide how to elaborate the plan and select an action. If necessary, the Search Planner is consulted to assist in planning how to find things. When the planner decides on an action, it instructs Action Execution to carry it out. Further planning is suspended until the action has terminated (successfully or unsuccessfully).

In making decisions about what to do next, each component makes use of its own internal simulation, which differs from the graphical animation of the environment. The planner uses abstract descriptions of the effects of each action to choose one which will move closer to the specified goal. The search planner simulates the movements of an agent on its internal map of the environment. Action Execution simulates taking the next step in several alternate locations. At each level of decision making, an internal simulation is used *at an appropriate granularity.*

Action Execution

The Action Execution module is responsible for the control of all actions occurring in ZAROFF. Most actions such as opening and closing doors are performed directly by this module. Human locomotion is a special case which is performed by the Behavioral Simulation System (BSS) (Becket & Badler 1993; Badler, Phillips, & Webber 1993). Action Execution controls this locomotion indirectly.

Non-locomotion actions are performed directly by Action Execution manipulating the environment. For example, a door is opened by rotating it about its hinges. This rotation is done incrementally, a small amount each frame of animation.

Locomotion is performed indirectly by Action Execution creating *sensors* and binding them to human figures in the database. Since BSS is constantly monitoring the environment, this immediately initiates the appropriate agent locomotion.

Neither path-planning nor explicit instructions are used to drive locomotion; agent control and apparent complexity are the result of the interaction of a few relatively simple behaviors with a complex (and changing) environment. An agent is made aware of its environment through the use of a network of sensors. Based on the information gathered by these sensors the path through the terrain is incrementally computed. This allows the agent to react to unexpected events such as moving obstacles, changing terrain, or a moving goal (Reich *et al.* 1994).

Sensors

A sensor is a function which maps a human's position and orientation (his state) in the environment to

a stress value, where lower values represent more desirable states. The agent (here, the seeker) utilizes a set of sensors in interacting with its environment. Currently there are four classes of sensors:

Attraction: An attraction sensor (attractor) is used to draw the seeker toward a goal, either an object or a location. If a goal object moves, the point of attraction moves appropriately. The sensor output (stress value) of an attractor is high when the agent is far from the goal and decreases as the agent nears the goal.

Repulsion: A repulsion sensor (repulser) is used to avoid collisions between the seeker and objects. Repulsers have a sector-shaped region of sensitivity. If there are no objects in this region the sensor output is zero. Otherwise the output is proportional to the distance and size of the detected objects.

Field-of-View: A field-of-view sensor determines whether or not the agent is visible to any other agent. (It will be used to support the players who must hide.) The sensor output is proportional to the number of agents' fields-of-view it is in, and inversely proportional to the distances to these agents.

The Behavioral Simulation System

BSS provides general locomotion of objects in *Jack*, and is used in ZAROFF to generate human locomotion. The central control mechanism of BSS is a loop that includes perception, control, and action. During the perception phase the sensors are polled, during the control phase the next foot position is selected, and during the action phase the step is taken.

General purpose planning

The next two sections describe the system for planning the overall behavior of the seeker agent, which combines a hierarchical planner with a special purpose search planner.

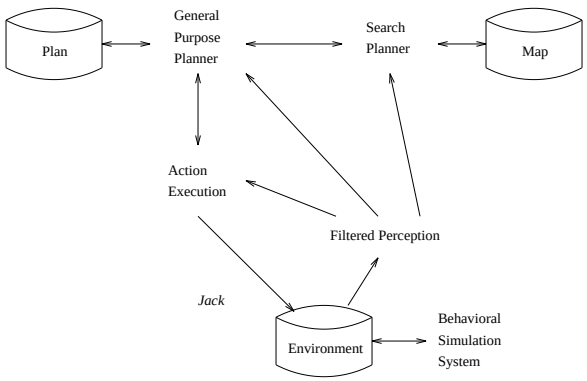

Figure 2: Information flow in ZAROFF

ITPLANS (Geib 1992) is a hierarchical planner, in which hierarchical expansion only takes place to the degree necessary to determine the next action to be carried out. It consists in an incremental expansion of the frontier of the plan structure to successively lower levels of abstraction. The incremental nature of the plan allows the system to make commitments at the appropriate level of detail for action while not committing the system to future actions that might be obviated by changes in the world. The close coupling of ITPLANS with the environment manifests itself in two ways:

First, the traversal and pruning process the planner follows at each interval relies on being able to determine the actual state of the world and compare that with its goals. During the expansion process ITPLANS examines the state of the world and its memory to determine if any of the goals within its plan have been satisfied. When a goal has been satisfied serendipitously, it can be pruned out of the plan structure, and the system can move on to consider its next goal.

Second, ITPLANS "leans on the world" (Agre 1988) when predicting the results of its actions. Rather than maintaining a complete model of the world and the state that results from executing the action, ITPLANS uses a simpler method based on associating conditional add and delete lists with each action. ITPLANS assumes that a given proposition is true in the state that results from the action if (1) the proposition is explicitly added by the add list or (2) the proposition is true *now* in the world and it is not contained on the delete list. By this method, ITPLANS can make predictions about the results of executing an action without modeling the entire world state.

Search planning

A consequence of limited perception is the occasional need to find objects. Our approach is to isolate this reasoning in a specialized module, a *search planner* that translates information acquisition goals to high-level physical goals to explore parts of the environment. As Haas points out (Haas 1993), any plan for acquiring information must rest on what the agent *knows* about the environment. That is, in order to search for an object, an agent must know (or discover during the search) the regions of space where the object might be.

Searches terminate successfully when a referent object is seen in the environment. They terminate unsuccessfully when there are no more regions to explore. A search may also be terminated if the environment changes in a way that obviates the search.

Maintaining a Map

Our approach to search planning relies on maintaining information about the state of a heuristic search on an internal map. The heuristic search has finding a desired object as its goal. It uses distance from the agent to order regions for exploration. Two lists of regions

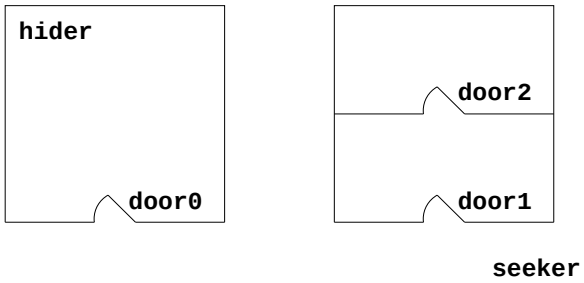

Figure 4: Plan view of example environment

are maintained by the search algorithm, an *open* list of regions yet to be explored and a *closed* list of regions which have been explored. Its internal map consists of nodes that correspond to bounded regions connected by links that correspond to doors.

In one iteration of the search, the closest region on the open list is selected to be explored. ItPlanS generates a plan for going to and exploring that region, opening any doors necessary along the way. After each action in this plan is executed, the resulting world is observed to determine if the desired object has been located. New doors and regions observed during the action are added to both the map and the open list.

Pemberton and Korf (Pemberton & Korf 1992) present optimal algorithms for heuristic search on graph spaces, where only a portion of the graph is available before the agent must commit to an action. We use their Incremental Best-First Search (IBFS) algorithm, which uses best-first search to find the closest known open node. Heuristic estimates for this known part of the graph are recalculated as necessary.

Example

Having given an overview of the system's components, we now illustrate Zaroff with an example drawn from a game of hide and seek. To illustrate the conduct of a search, we will use an example environment with two buildings, one of which has two internal rooms separated by a door (Figure 4). Consider the seeker's goal of finding a hiding player. This is specified as `goto(X)` with the added constraint that `type(X) = HUMAN`.

ItPlanS considers the action `goto(X)` to be primitive but underspecified since the variable `X` is not bound to a particular object of type `HUMAN`. In order to bind the variable, the search planner must be called to generate a plan for locating a `HUMAN`. To this end, ItPlanS adds to the plan a *find node* and calls the search planner to instantiate a search plan (Figure 3a).

The search planner reasons from this knowledge acquisition goal of locating a `HUMAN`, to the goal of exploring regions where a `HUMAN` might be. Satisfying this goal requires physically searching through possible regions.

ItPlanS asks the search planner to expand the find node. Each time a find node is expanded, the search planner first examines the *Jack* environment to determine if an object of the specified type is visible to the agent. If not, the search planner selects a region to explore next, generates a goal to explore that region, and adds it to the plan (Figure 3b). The initial map (Figure 5) of regions has two doors for the search planner to choose from; the closest, `door1`, is recommended for exploration. This goal is then further expanded by ItPlanS to go to `door1` and open it (Figure 3c). Since all of the arguments in the first action are bound to specific objects, it can be carried out. Action Execution performs this action indirectly by binding an attraction sensor to the seeker. When the seeker arrives, `door1` is opened directly by Action Execution.

After `door1` is opened, ItPlanS uses the search planner to evaluate the progress of the search by examining the world for objects having the property `HUMAN`. If one is located, the search is considered successful. If not, the search planner selects a new region for exploration and the searching process repeats until there are no more regions to explore.

In this case, opening `door1` does not reveal a `HUMAN`, but does permit the agent to see another door, `door2`, that is automatically added to the search planner's internal map. As `door2` is the closest unexplored space, on the next iteration the planner will plan to explore behind `door2`. Opening `door2` does not reveal the `HUMAN`, so the search proceeds to the next closest unexplored region – the other building.

Here we see the advantage of maintaining a map. Immediately after opening `door2`, the agent is inside one building and decides to go to the door of the other building. Since this destination region is known (from having seen it previously), we could simply take the action `goto(door0)`. This would result in the agent walking directly toward the door until stopped by the wall. To avoid getting caught in this local minimum, the search planner uses its internal map (Figure 6) to plan a path to the next region. The only known path to `door0` is to exit the current building through `door1`. The search planner returns this sequence of intentions

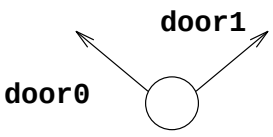

Figure 5: Initial map

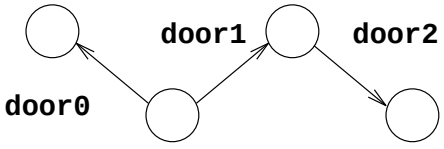

Figure 6: Final map

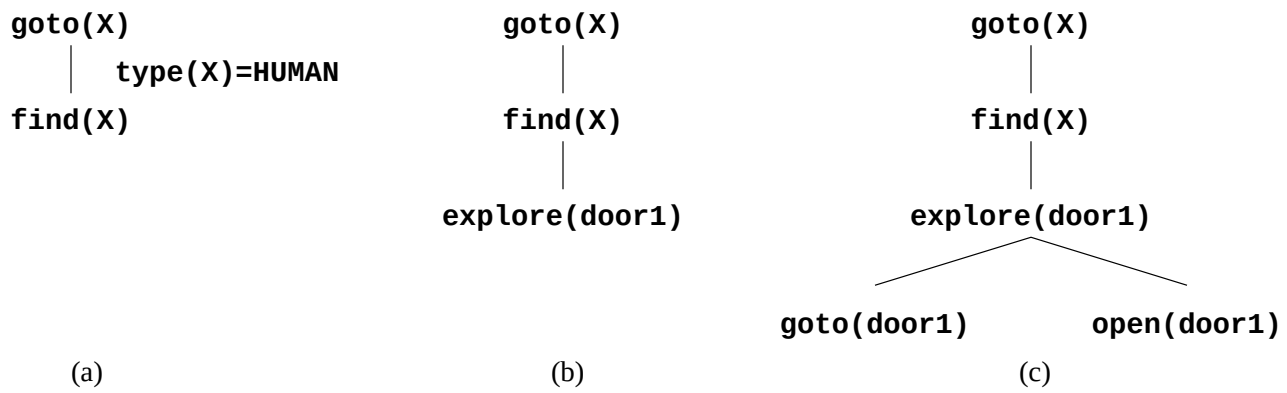

Figure 3: Evolution of the plan graph

to ItPlanS, which then invokes Action Execution to generate locomotion along this path. After opening `door0`, the seeker finally sees a `HUMAN` and can go to it.

Conclusion

We have implemented a plan-based controller for the *seeker* role in the game of hide and seek. Our agent dynamically reacts to changes in the environment, from the level of terrain up to changes in information about where the other players may be hiding. The implementation combines general purpose planning, special purpose reasoning about conducting a search, and reactive control of human behaviors.

Zaroff is an effective system for animating humans carrying out tasks that require locomotion. Limiting the human agent's awareness of its environment by simulated perception increases the realism of the behavior generated.

Acknowledgements

We wish to thank our advisors Norm Badler and Bonnie Webber for their support of this work. Thanks also to them, Welton Becket, Jonathan Crabtree, Brett Douville, and Jeff Nimeroff for commenting on drafts of this paper.

References

Agre, P. 1988. The dynamic structure of everyday life. Technical Report 1085, MIT Artificial Intelligence Laboratory.

Badler, N.; Phillips, C.; and Webber, B. 1993. *Simulating Humans: Computer Graphics, Animation and Control.* Oxford University Press.

Becket, W., and Badler, N. I. 1993. Integrated behavioral agent architecture. In *Proceedings of the Third Conference on Computer Generated Forces and Behavior Representation*, 57–68.

Becket, W. M. 1994. The Jack LISP API. Technical Report MS-CIS-94-01, University of Pennsylvania, Philadelphia, PA.

Geib, C. W.; Levison, L.; and Moore, M. B. 1994. SodaJack: An architecture for agents that search for and manipulate objects. Technical Report MS-CIS-94-16/LINC LAB 265, Department of Computer and Information Science, University of Pennsylvania.

Geib, C. 1992. Intentions in means-end planning. Technical Report MS-CIS-92-73, Department of Computer and Information Science, University of Pennsylvania.

Haas, A. 1993. Natural language and robot planning. Technical Report 9318, Department of Computer Science, SUNY Albany.

Ko, H. 1994. *Kinematic and Dynamic Techniques for Analyzing, Predicting, and Animating Human Locomotion.* Ph.D. Dissertation, University of Pennsylvania.

Pemberton, J. C., and Korf, R. E. 1992. Incremental path planning on graphs with cycles. In *Proceedings of AIPS 92*, 179–188.

Reich, B. D.; Ko, H.; Becket, W.; and Badler, N. I. 1994. Terrain reasoning for human locomotion. In *Proceedings of Computer Animation '94*, 996–1005. Geneva, Switzerland: IEEE Computer Society Press.

Vere, S., and Bickmore, T. 1990. A basic agent. *Computational Intelligence* 6:41–60.