Gridworld Search and Rescue: A Project Framework for a Course in Artificial Intelligence

Eric Eaton
University of Maryland Baltimore County
Department of Computer Science and Electrical Engineering
ericeaton@umbc.edu

Abstract

This paper describes the Gridworld Search and Rescue simulator: freely available educational software that allows students to develop an intelligent agent for a search and rescue application in a partially observable gridworld. It permits students to focus on high-level AI issues for solving the problem rather than low-level robotic navigation. The complexity of the search and rescue problem supports a wide variety of solutions and AI techniques, including search, logical reasoning, planning, and machine learning, while the high-level GSAR simulator makes the complex problem manageable. The simulator represents a 2D disaster-stricken building for multiple rescue agents to explore and rescue autonomous injured victims. It was successfully used as the semester project for CMSC 471 (Artificial Intelligence) in Fall 2007 at UMBC.

Introduction

From the increasing nationwide concern with domestic disasters, there has been a surge in developing rescue mechanisms for victims of these disasters. One major recent effort includes developing autonomous robots that search a disaster area for victims. These autonomous robots can search areas that would be hazardous for rescue workers, aiding their search and possibly providing limited treatment to victims.

The search and rescue problem is important and highly relevant to current societal concerns. Additionally, it is relatively easy to create a simple solution (albeit a poorly performing solution) to this problem in the form of a random-walking rescue agent. The impact and real-world nature of the problem coupled with the ease of getting started makes the search and rescue problem ideal as a semester project for a general course in artificial intelligence.

The RoboCup-Rescue (Kitano *et al.* 1999) competitions are a popular forum for developing intelligent systems to solve this problem, based around either actual robots in the NIST Urban Search and Rescue test arenas (Jacoff, Messina, & Evans 2002) or virtual rescue agents in the RoboCup-Rescue simulation league. The simulation league includes building-level search and rescue with virtual robots using the USARSim simulator (Wang, Lewis, & Gennari 2003) and virtual NIST test arenas, or large-scale multiagent disaster coordination through the RoboCup-Rescue Agents Simulator. However, the current simulators offered by these groups focus on realistic simulation, which makes them unsuitable for short-term development projects. The large-scale Agents Simulator is beyond the scope of most basic AI courses, and the smaller-scale USARSim focuses much of the initial development of an intelligent agent on low-level robotic issues, such as driving in a straight line from one location to another. The realistic focus of these two simulators, while important for research purposes, make these simulators inappropriate for use in an introductory AI course.

This paper introduces the *Gridworld Search and Rescue simulator*[1]: a smaller-scale simulator suitable for educational use that allows students to develop an intelligent agent to solve the search and rescue problem from a higher-level perspective. The intelligent agent's task is to search a simulated disaster-stricken building, represented as a 2D gridworld, for victims and carry as many as possible to safety in a limited period of time. The simulator is designed to allow students to ignore low-level details and focus on applying AI techniques to the problem. Several gridworlds are included in the distribution, and the map editor facilitates the creation of new environments. Additionally, the distribution provides a manual client for user control of a rescue robot. The simulator supports multiple interacting agents and can be extended to support new features.

The intelligent agent controls a simulated robot with high-level sensors and effectors, including a long-range object recognition system, short-range medical diagnostic sensors, a positioning system, and accurate navigation. The agent's primary objective is to locate victims and to carry them to one of the building's exits. Agents are only credited with rescuing live victims, so the robot is equipped with short-range medical diagnostic sensors that provide information on the victim's vital signs. Students may use this sensor data with machine learning techniques to generate a model for triaging a patient's injuries and predicting how long they will survive. The agent might use this model to determine the priority order in which to rescue the victims.

The agent is preloaded with the structural plans for the building, but will need to explore the building for victims

[1]The Gridworld Search and Rescue simulator is available online at http://maple.cs.umbc.edu/∼ericeaton/searchandrescue/.

simultaneously with other competing agents. The structural plans of the building include only the floor plan, so the agent should be robust to the location of objects (such as desks and chairs) and other agents within the building that may complicate navigation. Additionally, the disaster may have blocked certain areas of the building, so the agent must be robust to such changes. The robot's long-range object recognition system and positioning system provide information on the agent's surroundings in the gridworld.

The Gridworld Search and Rescue (GSAR) simulator uses a networked client-server framework to allow students to run their intelligent agent on local computers while interacting with the remote simulation server. The distribution also includes a visualization display client that connects to the remote server and provides a graphical display of the gridworld simulation. The simulator is written in Java[2] for system portability, and includes client interface libraries that facilitate intelligent agent development in either Java or Lisp.

The partial observability and uncertainty inherent in the GSAR problem make it sufficiently complex to challenge a variety of AI techniques, yet easy to understand. Students could apply a wide variety of AI techniques in their solutions to the GSAR problem, including search techniques, logical and probabilistic reasoning, planning, robotic navigation methods, and machine learning. The opportunity to triage victims' injuries provides a straightforward opportunity to apply machine learning techniques.

In Fall 2007, undergraduate students in the CMSC 471 (Artificial Intelligence) course at UMBC developed intelligent agents for the GSAR simulator as the course project. Students worked in small groups developing their agents, which incorporated A* search, state-space planning, reinforcement learning, and logical reasoning, to name a few techniques used in their designs. The GSAR project received high praise from students, who seemed to greatly enjoy it. At the end of the semester, the students participated in a competition to see which team's agent could rescue the most disaster victims.

Gridworld Search and Rescue Description

The Gridworld

The simulated building lies on a rectangular gridworld. Each cell in the grid can be occupied by only one object at a time (with the single exception of an agent carrying an object).

The gridworld has cardinal directions north, south, east and west and an inherent coordinate system, with the origin located at the southwest corner cell. A cell's coordinates remain fixed throughout the duration of the simulation, providing absolute locations for the agent's positioning system.

Each cell in the gridworld may have up to four walls, corresponding to each of the four directions. Objects cannot move through walls and sensors cannot penetrate walls. Walls cannot be demolished. For simplicity, the building does not contain any doors that require opening. Certain gridworld cells contain markers signifying some special nature of the locations; these markers are detected by the agent's long-distance sensors.

[2]The current GSAR implementation requires Java 1.5 or later.

Each agent starts at one of the building's entrances (assigned randomly) and must rescue victims by returning them to any of the building's entrances (for simulated pickup by rescue workers). The cells at the building's entry points are flagged with the marker "EXIT." In the provided gridworlds, the building's outer walls lie inside the gridworld's boundaries, forming a perimeter outside of the building for the agent to move between entrances, if necessary.

Initial Knowledge

The intelligent rescue agent has access to "blueprints" of the building, but no knowledge of its contents. At initialization, the agent is given the gridworld's dimensions, and the location of all walls and building entry points. The numbers and locations of victims, other agents, and objects within the building are unknown to the agent at this time. However, it is known that no victim is outside the building (otherwise, rescue workers would have assisted them already).

The agent's initial knowledge is provided automatically by the simulation server, offering the ability to keep the gridworld map secret until simulation time. The simulator also includes an option to disable the initial knowledge for a more difficult challenge, so that agents would need to discover all aspects of the building during the rescue operation.

Objects Within the Gridworld

While the agent knows the building's layout at initialization and can navigate roughly based on it, there may be objects within the gridworld that complicate the navigation paths. For example, a hallway may be blocked by unmovable debris, effectively acting as another wall in the building. The GSAR server supports a variety of user-customizable object classes, including movable, immovable, and autonomous objects. In the provided example gridworlds, the building contains an assortment of standard office furniture, such as tables, chairs, and bookshelves, some of which can be moved by the agent. Since only one object can occupy a gridworld cell, these objects also complicate navigation. Moreover, since multiple rescue agents are present in the environment simultaneously, another agent may move an object during the simulation, making the location of these objects slightly dynamic. Autonomous objects, such as victims and rescue agents, have the capability to move themselves.

Victims

Like victims in the real world, the simulated victims each behave differently. Victims are implemented as pseudo-random-walking autonomous objects in the simulator, with their probability of movement based on their current health status. Less-injured victims may move around quite a bit.

Victims are characterized by a set of vital signs that are initialized based on their simulated level of injury and change over time. The vital signs include estimated Abbreviated Injury Scale (AIS) values (AAAM 1990) for major areas of the body, Glasgow Coma Scale (GCS) values (Teasdale & Jennett 1974), systolic and diastolic blood pressures, SpO_2 level, respiratory and pulse rates, body temperature, and estimated age. As a victim's health status changes, their vital signs fluctuate.

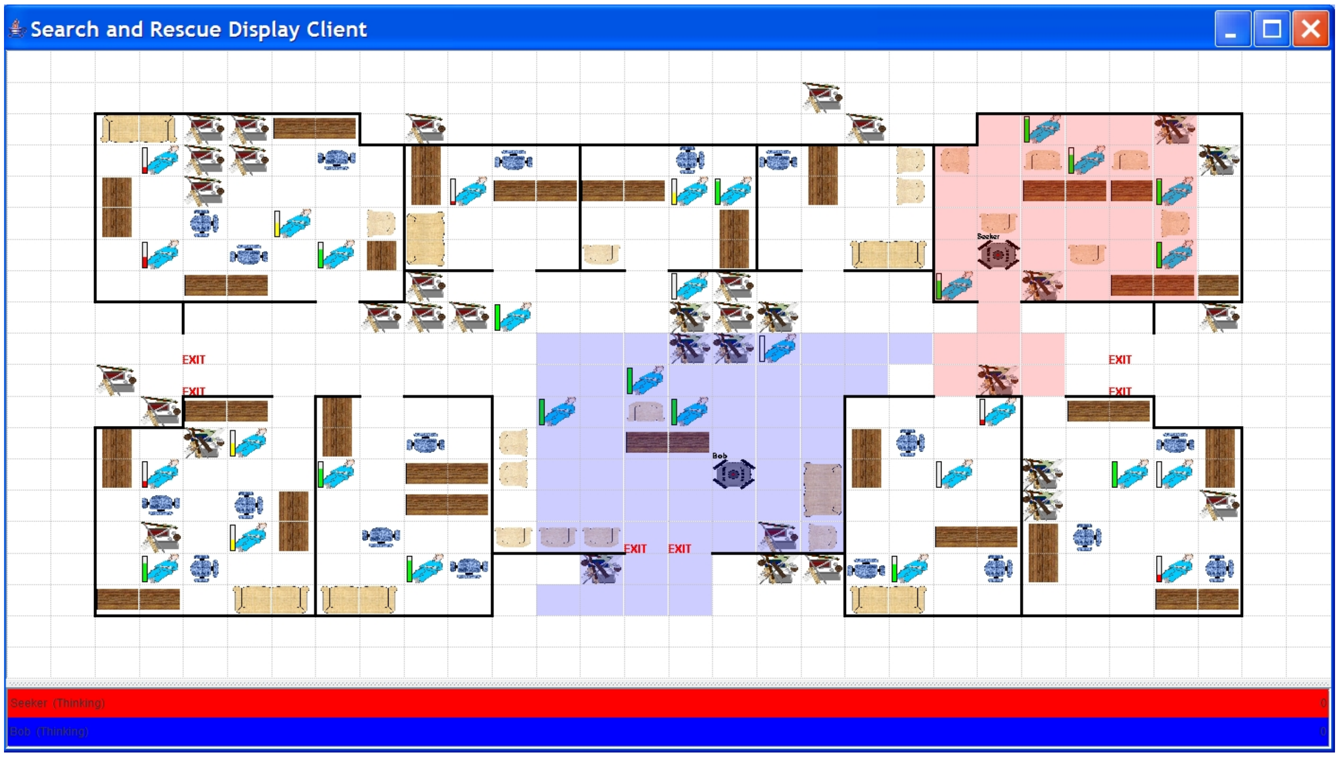

Figure 1: A screenshot from the display showing a GSAR simulation in progress with two rescue agents. The colored areas depict each robot's long-distance sensor range. Victims with varying degrees of injury are scattered about the gridworld; the bar graphs next to each victim show summaries of each victim's health status. The agents' goal is to rescue living victims by carrying them to one of the exits. The score bar beneath the display provides each agent's status and their current score.

These vital signs are the observable variables of a probabilistic model for the victim's health status. The probability of a victim's survival is predicted using the Trauma Score - Injury Severity Score (TRISS) method (Boyd, Tolson, & Copes 1987), which is based on the Injury Severity Score (Baker *et al.* 1974) using the patient's AIS values, and the Revised Trauma Score (Champion *et al.* 1989) using the patient's GCS levels, blood pressure, and respiratory rate. O'Keefe and Jurkovich (2001) provide an overview of these trauma scoring systems. Further details on the probabilistic model may be found in the GSAR simulator source code.

Like actual disaster victims, the simulated victims may "die" during the simulation as determined by the probabilistic model, or they may be dead at the start of the simulation. Rescue agents are not credited for rescuing dead victims from the building. Although slightly morbid, this notion mirrors real-life rescue efforts and adds additional complexity to the GSAR problem. The intelligent agent has access to victim vital signs through its short-range sensors; this data might be useful in triaging the victim and predicting how long the victim will live to prioritize rescue efforts. The GSAR distribution provides labeled vital sign data that students can use for training machine learning models.

Simulator Architecture

The GSAR simulator is comprised of three primary components: a simulation server, a visualization tool that displays the gridworld during the simulation, and client libraries for interacting with the server. The simulator is designed as a client-server system to allow the instructor to control the server running on one machine, and each student team to have their own computer for running their intelligent agent. Currently, the simulator includes both Java and Lisp versions of the client library. All communication between the client and the server is in XML across network sockets. The server and display communicate via Java Remote Method Invocation (RMI). Figure 2 provides a graphical depiction of the GSAR simulator architecture and the interactions between the components.

The Rescue Agent

The intelligent agent controls a simulated robot with sophisticated high-level sensors and effectors. In keeping with a level of realism, the sensors cannot penetrate the walls, making the environment partially observable to the agent.

Sensors

The rescue robot is equipped with several sensors that provide the intelligent agent with high-level perception:

Long-range object recognition and localization sensors: The long-range sensors cover a rectangular area around the agent, forming a nine-by-nine square with the agent in the center. Any object within this range will be identified by

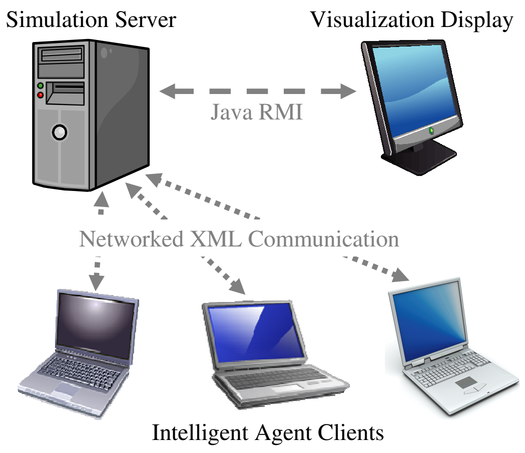

Figure 2: The GSAR simulator architecture.

name and precise coordinate location. The long-range sensors cannot penetrate walls, so the covered area may be reduced depending on the agent's location and will not always be a complete rectangle. These sensors are always active and provide information to the agent following every action.

Short-range medical diagnostic sensors: The short-range sensors must be activated by the agent to provide information on the victim in the specified adjacent cell. The *sense* action (described below) activates the short-range sensors and returns their results; therefore, the use of this sensor constitutes an action by the rescue robot. This sensor characterizes the victim's vital signs as a feature vector of real-valued data, as described in the section "Victims." The simulator distribution contains labeled training data that students can use to learn a model to triage the victims and predict how long they will survive. The exact specification of the feature vector is available in this data.

Self-feedback sensors: The self-feedback sensors provide information on the robot itself. This information includes the agent's current location, the current simulation time, the name of any object it is carrying, and the status of the last action the agent attempted to execute, such as whether the action succeeded. These sensors are always active and provide information to the agent following every action.

Effectors and actions

The simulated robot is equipped with omni-directional wheels, allowing immediate movement in any of the cardinal directions, and a lift capable of carrying an object. These effectors enable the agent to perform the following actions:

Move: The move action provides for navigation between adjacent cells along the cardinal directions. The action may fail if it attempts to violate the rules of the gridworld, such as trying to move through a wall or to an occupied cell.

Pickup: The pickup action allows the robot to pick up an object (such as a victim) in an adjacent cell and carry it along through the gridworld. The robot is not able to pick up all objects, and it can carry only one object at a time.

Dropoff: The dropoff action is the opposite of the pickup action: it drops the object the robot is carrying into the specified adjacent cell, which must be empty.

Sense: The sense action activates the short-range medical diagnostic sensors on an object in the specified adjacent cell, and returns the feature vector description of the object.

No-Op: Additionally, the agent can choose not to perform an action in a particular timestep by executing a No-Op. Invalid actions default to a No-Op.

Implementing the Intelligent Agent

The GSAR client API facilitates the creation of an intelligent agent to control a virtual rescue robot. The client library provides a number of data structures and utility functions for students to use in their implementation, facilitating the use of the agent's initial knowledge and the robot's sensors and effectors. The library also automatically handles all of the networked communication. The basic interface to the simulator is exceptionally simple and requires the student to provide definitions for very few functions (three in the current simulator version, two of which are optional). Students are also capable of creating their own avatars for their rescue robots that will appear in the simulation display.

Client interfaces are provided for both Java and Lisp; therefore, it would be easiest for students to implement their intelligent agent using one of these languages. All networked communication uses XML over network sockets, so it would be straightforward to translate the client library to other languages.[3]

One of the best aspects of the GSAR domain is the ease of getting started. It is very easy (requiring less than 30 minutes of implementation time, in the experience of the undergraduates in the course) to implement a simple solution in the form of a random-walking agent. Such an agent would move randomly until it found a victim to pick up, then move randomly until it found an exit, drop the victim off at the exit, and then return to searching for another victim. Although the random-walking agent does not use any AI techniques and does not perform very well, it demonstrates the ease with which students can begin developing a rescue agent.

Additionally, the simulator provides a manual client that students can use to manually explore the domain and test the robot's sensors and actions. Students can also use the manual client and the provided simple rescue agents to test the behavior of their intelligent agent with other GSAR agents.

The Simulation

The severity of injuries among the victims is controlled by the disaster-severity parameter of the simulation. The disaster-severity ranges from 0 for a very slight disaster which may result in minor injuries to 10 for a severe disaster that results in a huge number of injuries and casualties.

At the start of the simulation, the server provides each agent with their initial knowledge about the gridworld. All

[3]Please consider contributing any translations of the client libraries into other languages to the GSAR project.

agents are then given a specified amount of clock-time to process this information before the simulation begins.

The simulation runs for a specified number of timesteps and then totals the score for each agent. All agents operate in parallel, processing their current perceptions and returning an action each timestep. After initialization, at each simulation timestep, the agent will be presented with its current perception of the world. This perception will include data from the long-range sensors, self-feedback, and the short-range sensors, if they were activated the previous timestep.

Once the agent is presented with the perception, it will have a limited time (e.g. 10 real-world seconds; this is a variable parameter for the simulation) in which to respond with an action. The simulation proceeds as soon as all agents have responded with their actions for the current timestep to make the simulation as fast as possible. If an agent does not respond with an action within the limited time, that agent's action for the current timestep will default to a No-Op.

Multiple rescue agents could choose to execute actions that conflict. For example, two agents may try to move into the same cell at the same time. In such a case, one agent's action, chosen randomly, will succeed and the other conflicting actions will fail. Actions from a non-agent (such as a victim) never conflict with any action from an agent.

Each agent is credited with one point for each live victim it delivers to a building exit. Dead victims are not worth any points, nor are victims that rescue themselves by wandering to an exit. During the simulation, the rescue agents' current scores are shown on the GSAR display's scoreboard. At the end of the simulation, each agent is automatically informed of its score, which may be useful for reinforcement learning or other AI techniques which involve performance feedback.

Related Work

Urban search and rescue (USAR) simulators tend to focus on either small-scale search and rescue in a building or large-scale disaster coordination. The virtual robot branch of RoboCup-Rescue uses the USARSim simulator (Wang, Lewis, & Gennari 2003) for building-level search and rescue. USARSim provides a realistic simulation platform for virtual robots and environments using the Unreal Tournament game engine. It includes virtual NIST USAR test arenas (Jacoff, Messina, & Evans 2002), a variety of other environments, and provides virtual versions of many common robot platforms with a variety of sensors and effectors.

There are a number of projects focused on the large-scale search and rescue problem, including the RoboCup-Rescue Agents Simulator,[4] the ALADDIN project[5] for disaster management, and the FireGrid project (Berry *et al.* 2005) for simulated fire emergencies. The RoboCup-Rescue Agents Simulator is the most widely used comprehensive large-scale USAR simulator, providing a multiagent platform where agents control teams of police, fire, and medical personnel in response to urban disasters, with complications due to traffic and civilian response.

[4]http://www.robocuprescue.org/agentsim.html

[5]http://www.aladdinproject.org/

Extensions and Future Work

The current version of the GSAR simulator could support multiagent techniques, with the agents communicating among themselves via network sockets. Teams of agents could then coordinate rescue efforts. This networked communication would need to be built into the agents' implementations; currently there is no support for communication via the simulator. However, there are future plans to include support for basic message passing between agents, possibly with uncertainty and limited transmission ranges. Including this feature would open the GSAR simulator for use in multiagent and team formation research. Also under consideration are several other cosmetic changes to the simulator, such as more support for custom objects, further configuration options, and modifying the display to provide a three-quarters overhead view of the gridworld.

Integration into the Curriculum

In Fall 2007, undergraduate students in the CMSC 471 (Artificial Intelligence) course at UMBC created intelligent agents for the GSAR domain in teams of two or three people as their semester project. Each group was required to incorporate two AI topics into their project. The project was assigned one-quarter of the way through the course, after we had covered search and constraint satisfaction.

Most general AI courses cover a diverse set of topics. One of the challenges of designing a semester-long project for such a course is balancing the students' desire to use topics that will be covered later in the semester, with their ability to learn about, apply and implement those techniques successfully. In class, we held a discussion on the various topics we would be covering later, how those techniques might be used in the GSAR project, and the challenges with selecting topics near the end of the syllabus.

To help avoid complications from students choosing topics without fully understanding them, the project assignment placed a strong emphasis on design. Each group was required to submit a project proposal after three weeks that demonstrated their understanding of their chosen techniques, and offered me a chance to give them early feedback on their proposed solutions. Although groups were not required to meet with me individually, most groups were eager to discuss their ideas before they designed their agents.

The final design was due three weeks after the proposal (approximately two-thirds of the way through the semester), leaving approximately one month for students to implement their agents. As part of earlier homework assignments, they had already implemented several algorithms in lisp, including A* search, and many groups ended up reusing that code in their agents. Our last major topic for the course was machine learning, so the victim triage component of the project was designed to provide students with a straightforward opportunity to include machine learning in their project. Instructors also have the option of allowing this component to be designed last, as it can easily plug into the agent.

At the end of the semester, the class held a competition between the teams' rescue agents. As part of the project description, aggressive behavior toward other rescue agents was explicitly prohibited, since it violated the spirit of the search and rescue problem. The competition was to be based solely on the agents' performance in rescuing live victims.

Several examples of successful project designs from the CMSC 471 class are given below:

- The most popular design for an agent used some pattern to search for victims (with a focus toward unexplored areas), a simple learned model (e.g. decision trees, SVMs) from Weka (Witten & Frank 2000) to triage victims, and informed search to determine the shortest path to an exit. Several groups proposed different variations on this design, and it was very successful in one case.
- One of the more creative and successful solutions used an algorithm based on binary space partitioning (de Berg *et al.* 2000) to subdivide the gridworld into logical rooms, which were used as subgoals in combination with A*.
- Another group implemented a production system for high-level control of their agent's actions, using A* for low-level path-finding.
- A successful design, although not one of the finalists, used conditional state-space planning of actions in combination with victim triage using a decision tree.
- Another agent wandered randomly to explore, predicted a victim's time-to-live using linear regression, and used q-learning to determine the shortest path to an exit.

One of the challenges that many teams faced was designing a good heuristic function for using A* to find an exit. Manhattan distance is the most obvious heuristic for this domain, but it is very suboptimal due to walls and other obstructions.

Evaluation and Conclusion

While this paper can describe the simulator and the GSAR problem in detail, it cannot show you the true magic of a group AI project in this domain. As an instructor, I enjoyed the students' enthusiasm toward the GSAR problem in weeks before, and especially during the day of the competition. Each team competed in simulations against several other teams in two hours of cheers, smiles, laughter at the nonsensical actions of their own agents, and praise for other teams' work. Students ardently described their agents to each other, to me, and to several other professors and graduate students who visited the competition. Even students who tended to remain quiet in class chatted excitedly, becoming more engaged with the class than I had observed before.

Many of the students expressed their enjoyment of the GSAR project and the chance to apply the AI techniques they learned to a "real-world" problem. On end-of-course surveys, students commented, "Great project...we had a good time," and, "the project was a really good idea." Many of them listed it as their favorite part of the course.

This is the measure of success of the GSAR project—that it gets students excited about AI and proud of each other's accomplishments. It provides them a chance to apply a variety of AI techniques to a "real-world" problem in a simplified and manageable manner, promotes a bit of competition, and is fun.

Acknowledgments

This work was supported by NSF ITR #0325329 and the UMBC CSEE department. Thanks to Aaron Curtis, the CMSC 471 teaching assistant, for additional coding on the lisp client and the display, and development of the map-editor. Thanks also to the students of the Fall 2007 CMSC 471 course for their feedback and suggestions on the project.

References

Association for the Advancement of Automotive Medicine. 1990. *The Abbreviated Injury Scale*. AAAM.

Baker, S. P.; O'Neill, B.; Haddon, Jr., W.; and Long, W. B. 1974. The injury severity score: a method for describing patients with multiple injuries and evaluating emergency care. *Journal of Trauma* 14(3):187–196.

Berry, D.; Usmani, A.; Torero, J. L.; Tate, A.; McLaughlin, S.; Potter, S.; Trew, A.; Baxter, R.; Bull, M.; and Atkinson, M. 2005. FireGrid: Integrated emergency response and fire safety engineering for the future built environment. In *UK e-Science Programme All Hands Meeting (AHM-2005)*.

Boyd, C.; Tolson, M. A.; and Copes, W. S. 1987. Evaluating trauma care: The TRISS method. *Journal of Trauma* 27(4):370–378.

Champion, H. R.; Sacco, W. J.; Copes, W. S.; Gann, D. S.; Gennarelli, T. A.; and Flanagan, M. E. 1989. A revision of the trauma score. *Journal of Trauma* 29(5):623–629.

de Berg, M.; van Kreveld, M.; Overmars, M.; and Schwarzkopf, O. 2000. *Computational Geometry*. Springer-Verlag, second edition.

Jacoff, A.; Messina, E.; and Evans, J. 2002. Performance evaluation of autonomous mobile robots. *Industrial Robot: An International Journal* 29(3):259–267.

Kitano, H.; Tadokoro, S.; Noda, I.; Matsubara, H.; Takahashi, T.; Shinjou, A.; and Shimada, S. 1999. RoboCup-Rescue: Search and rescue for large scale disasters as a domain for multi-agent research. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 739–743.

O'Keefe, G., and Jurkovich, G. J. 2001. Measurement of injury severity and co-morbidity. In *Injury Control: A guide to research and program evaluation*. Cambridge University Press. 32–46.

Teasdale, G., and Jennett, B. 1974. Assessment of coma and impaired consciousness: a practical scale. *The Lancet* 2:81–84.

Wang, J.; Lewis, M.; and Gennari, J. S. 2003. Interactive simulation of the NIST USAR arenas. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 1350–1354.

Witten, I. H., and Frank, E. 2000. *Data Mining: Practical Machine Learning Tools with Java Implementations*. San Francisco, CA: Morgan Kaufmann.